龙芯中科
LOONGSON TECHNOLOGY

LOONGSON

龙芯 2K3000/3B6000M 处理器

用户手册

V1.0

2026 年 08 月

龙芯中科技术股份有限公司

北京市海淀区稻香湖路中关村环保科技示范园龙芯产业园2号楼
Loogson Park #2, Zhongguancun Environment Protection Park,
Daoxianghu Rd, Haidian District, Beijing

www.loongson.cn

阅读指南

《龙芯 2K3000/3B6000M 处理器用户手册》主要介绍龙芯 2K3000/3B6000M 架构与寄存器描述，对芯片系统架构、主要模块的功能与配置、寄存器列表及位域进行详细说明。

除特别说明外，该文档中龙芯 2K3000 与龙芯 3B6000M 不做区分，内容完全一致。

目录

图目录

表目录

1 概述

龙芯 2K3000 处理器主要用于移动终端、云终端和工控领域。片内集成 8 个 LA364E 处理器核，内置 GPGPU、视频编解码模块（VPU）、DDR4/LPDDR4 控制器、GMAC 控制器、PCIe3.0 控制器以及众多系统上使用的 IO 接口。

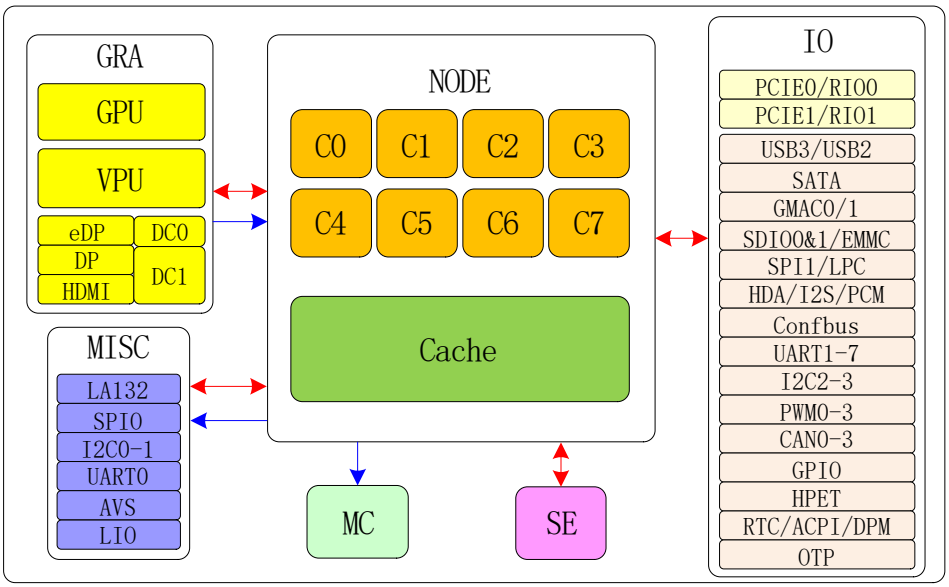


图 1-1 龙芯 2K3000 结构图

1.1 技术指标

- 片内集成 8 个 64 位三发射超标量 LA364E 处理器核，L1 Dcache/Icache 64KB
- 片内集成共享 6MB 二级 Cache
- 片内集成自主研发 GPGPU，支持图形加速、通用计算和 AI 加速
- 支持三路显示接口（HDMI+eDP+DP）
- 64 位 DDR4 接口，可配置成 2 个 32 位 LPDDR4
- 1 个 x4 PCIe 3.0 接口，可配置为 4 个 x1 PCIe 3.0 接口或者 1 个 x4 RapidIO 接口
- 1 个 x4 PCIe 3.0 接口，可配置为 2 个 x1 PCIe 3.0 接口或者 1 个 x4 RapidIO 接口
- 最多 4 个 USB 3.1 Gen1，最多 8 个 USB2.0
- 2 路网口（RGMII）
- 1 路 SATA3.0 接口（与 USB3.1 复用）
- HDA/I2S

- RTC/HPET 模块
- 8 路 UART 接口
- 4 路 CAN-FD 接口
- 4 路 I2C 接口
- 2 路 SPI 接口
- 1 路 LPC 接口
- 1 路 LIO 接口
- 1 路 eMMC 接口
- 2 路 SDIO 接口
- 1 路视频编解码器
- 4 路 PWM 接口
- 最多 58 路 GPIO，至少 4 路独立中断
- 支持动态调频调压
- ACPI 规范
- 安全模块，实现可信计算，支持 SM2/3/4 算法
- 内置温度传感器

1.2 功能及接口说明

1.2.1 CPU

- 8 个 LA364E
- 采用 LoongArch 指令系统（龙架构）
- 64KB 数据 Cache 和 64KB 指令 Cache
- 6MB 共享二级 CACHE
- 通过目录协议维护 IO DMA 访问的 Cache 一致性

1.2.2 GPU

- LG200
- 集成一路 DMA
- 集成 MMU
- 支持 4x MSAA
- 支持内存压缩
- 支持动态功耗管理
- 支持 OpenGL3.3、OpenGL ES3.0、OpenCL3.0

1.2.3 VPU

- 解码格式 HEVC/H264/VP8/VP7/AV1/MPEG4/JPEG
- 编码格式 HEVC/H264/JPEG

1.2.4 DDR4

- 支持 DDR4-3200/LPDDR4-3200
- 64 位 DDR4 接口，可配置成 2 个 32 位 LPDDR4
- DDR4 模式下最大内存容量 32GB
- LPDDR4 模式下最大内存容量 16GB
- 可配置为 64/32/16 位模式（DDR4）
- 支持 x8、x16 颗粒，不支持 x4 颗粒（DDR4）

1.2.5 PCIe

- 2 个 PCIe 接口，仅支持 RC
- 一个 PCIe 包含 4 个通道，可配置为 1x4/4x1，最高支持 PCIe3.0 速率；可配置为 x4 RapidIO 2.2 使用；
- 一个 PCIe 包含 4 个通道，可配置为 1x4/2x1，最高支持 PCIe3.0 速率；可配置为 x4 RapidIO 2.2 使用；
- 输出 4 路 PCIe 差分参考时钟
- 支持极性反转
- 支持线序反转

1.2.6 RapidIO

- 两路 RapidIO 接口，分别与 PCIe0/1 复用
- 兼容 RapidIO 2.2
- 使用串行接口，支持 1.25/2.5/3.125/5.0Gbps 速率
- 支持 RapidIO 协议 34/50 位地址的访问
- 支持 8/16 位 ID
- 支持 RapidIO 协议的 NREAD、NWRITE、 NWRITE_R、SWRITE、Maintenance Packet、Response Packet、Doorbell Packet、Message Packet
- x4/x2/x1 链路宽度自适应

1.2.7 显示

- HDMI/DP/eDP 各一路，分辨率均可达 4K@30Hz
- HDMI/DP 显示内容相同
- DP/eDP 不支持音频

- RGB444、RGB555、RGB565、RGB888 四种色深
- 输出抖动和伽马校正
- 支持线性显示缓冲，可切换的双路线性帧缓冲
- 中断和软复位
- 上电序列控制
- 低功耗管理

1.2.8 SATA

- 1 路 SATA 接口，与一路 USB3 接口复用
- 支持 SATA 1.5Gbps、SATA2 代 3Gbps 和 SATA3 代 6Gbps 的传输
- 兼容串行 ATA 2.6、AHCI 1.1 和 AHCI 1.3.1 规范

1.2.9 USB

- 4 路 USB3.1 Gen1 (5Gbps) 接口
- 8 路 USB2.0 接口（其中 4 路与 USB3.1 组成 USB3 接口）
- 兼容 XHCI Rev1.1 协议

1.2.10 GMAC

- 两路 10/100/1000Mbps 自适应以太网 MAC
- RGMII 接口
- 均兼容 IEEE 802.3
- 半双工/全双工自适应
- Timestamp 功能
- 半双工时，支持碰撞检测与重发 (CSMA/CD) 协议
- 支持 CRC 校验码的自动生成与校验，支持前置符生成与删除
- 两路均支持网络唤醒
- 支持 TSN
- 支持四通道 DMA 收发
- 支持硬件时间戳，支持 IEEE 1588/802.1AS 时间同步，支持 PPS 输出
- 支持 IEEE 802.1Qbv 协议，支持纳秒级精度控制，支持配置四通道门控状态，表项最大支持配置 256 项
- 支持 IEEE 802.1Qav 协议，支持对通道 1，2，3 进行信用值配置

1.2.11 eMMC

- 一路 eMMC 接口
- 8 位预分频逻辑（频率=系统时钟/ (p+1) ）

- DMA 数据传输模式
- 专用独立 DMA 通道
- 1 位/4 位/8 位的总线模式

1.2.12 SDIO

- 两路 SDIO 接口
- 8 字（32 字节）数据发送/接收 FIFO
- 扩展的 256 位 SD 卡状态寄存器
- 8 位预分频逻辑（频率=系统时钟/（p+1））
- DMA 数据传输模式
- 专用独立 DMA 通道
- 1 位/4 位（宽总线）的 SD 模式

1.2.13 HDA

- 支持 16、18 和 20 位采样精度，支持可变速率
- 最高采样频率 192KHz
- 7.1 频道环绕立体声输出
- 三路音频输入

1.2.14 I2S

- 支持 18、20、24、32 位的音频数据采样位宽。
- 支持 8、16、32 位的左右声道处理字宽。
- 包含两个缓存 FIFO，FIFO 的缓存容量为 8bytes。
- I2S 的中断处理模式可配，在 I2S 的发送和接收中断功能都使能后，当两个通道的缓存 fifo 为满仍要写以及为空仍要读时，则向 CPU 发出中断信号。
- I2S 可以为 codec 芯片提供系统时钟，时钟频率可配。
- 支持 master/slave 模式下 I2S 输入
- 支持 master/slave 模式下 I2S 输出
- 支持单声道和立体声道音频数据
- 支持（16、22.05、32、44.1、48）KHz 采样频率
- 支持 DMA 传输模式

1.2.15 SPI

- 2 路独立 SPI 控制器，均支持 4 线模式
- SPI0 支持系统启动
- 双缓冲接收器

- 极性和相位可编程的串行时钟
- 主模式支持
- 支持到 4 个的变长字节传输
- 支持标准读、连续地址读、快速读、双路 I/O 等 SPI Flash 读模式

1.2.16 LPC

- 1 路 LPC 接口
- 符合 LPC1.1 规范
- 支持 LPC 访问超时计数器
- 支持 Memory Read/write 访问类型
- 支持 Firmware Memory Read/Write 访问类型（单字节）
- 支持 I/O read/write 访问类型
- 支持 TPM I/O read/write 访问类型
- 支持 Memory 访问类型地址转换
- 支持 Serial IRQ 规范，支持 17 个中断源

1.2.17 UART

- 最多 8 路 2 线 UART，其中两路可作为 1 个 4 线使用
- 在寄存器与功能上兼容 NS16550A
- 两路全双工异步数据接收/发送
- 可编程的数据格式
- 16 位可编程时钟计数器
- 支持接收超时检测
- 带仲裁的多中断系统

1.2.18 I2C

- 4 路 I2C
- 一路可配置为 slave，读取芯片内部温度
- 兼容 SMBUS（100Kbps）
- 与 PHILIPS I2C 标准相兼容
- 履行双向同步串行协议
- 主从设备支持
- 能够支持多主设备的总线
- 总线的时钟频率可编程
- 可以产生开始/停止/应答等操作
- 能够对总线的状态进行探测

- 支持低速和快速模式
- 支持 7 位寻址和 10 位寻址
- 支持时钟延伸和等待状态

1.2.19 PWM

- 4 路 32 位可配置 PWM 定时器
- 支持定时器功能
- 支持计数器功能
- 支持防死区发生控制

1.2.20 HPET

- 支持 1 个周期性中断
- 支持 2 个非周期性中断

1.2.21 RTC

- 可产生 3 个定时中断
- 支持定时开关机功能

1.2.22 ACPI

- USB/GMAC 可 S3 唤醒
- 来电可自动启动
- 支持 S0,S3,S4,S5 状态

1.2.23 GPIO

- 4 个专用 GPIO 引脚，54 个复用 GPIO 引脚
- 输入中断功能
- 中断极性、触发类型可设置

1.2.24 CAN

- 4 路 CAN 接口
- 支持 CAN-FD

1.2.25 AVS

- 一路 AVS 接口

1.2.26 JTAG 接口

- JTAG 测试接口

1.2.27 中断控制器

- 支持软件设置中断
- 支持电平与边沿触发
- 支持中断屏蔽与使能
- 支持多种中断分发模式

2 时钟结构和时钟控制

2.1 芯片时钟结构

龙芯 2K3000 内部共有 5 个独立的 PLL，其中每个 PLL 最多可以提供 3 组频率上相互依赖的时钟输出。这 5 个 PLL 的用途分别为：

- ✓ 一个 PLL 产生 NODE、IO 和 I2S 时钟，NODE 时钟供 CPU 核、二级 Cache、NOC 网络以及 LA132 使用，各子模块可单独配置分频；
- ✓ 一个 PLL 产生 NODE_BAK、DDR、eMMC/SDIO 时钟，其中 NODE_BAK 时钟为 NODE 域备份时钟，可与 NODE 时钟互选；
- ✓ 一个 PLL 产生 GPU、VPU、250MHz 固定频率时钟，其中 250MHz 时钟分频后产生 USB3/USB2/SATA/GMAC 时钟；
- ✓ 一个 PLL 产生 DC、54MHz 固定频率、2.7GHz 固定频率时钟，这几路时钟提供给 DC 控制器使用；
- ✓ 一个 PLL 产生 48MHz 固定频率和 RIO 时钟，48MHz 时钟提供给 APB 模块使用，并经 2 分频后提供给 HDA/PCM 模块；

2.1.1 系统参考时钟

芯片的系统参考时钟有三个来源，分别为：

1. 单端输入时钟 SYS_SYSCLK，频率为 100MHz；
2. PCIe 参考时钟输入 PCIE_REFCLKP/N，频率为 100MHz；
3. USB 参考时钟输入 USB_REFCLKP/N，频率为 25MHz（根据引脚配置可选单端/差分输入）；

无论选择哪个时钟源，提供给以上 5 个 PLL 的参考时钟频率都是 100MHz。

2.1.2 内部 PLL 配置

PLL 的结构图如图 2- 2 所示，

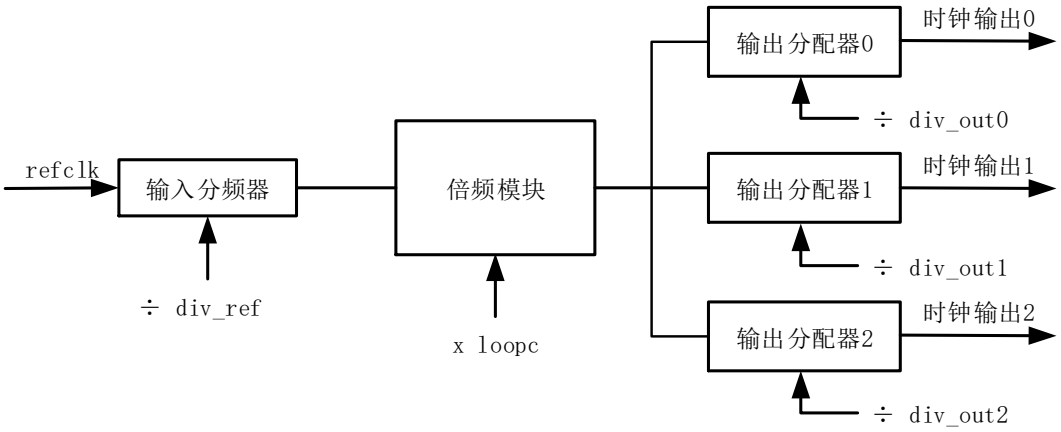


图 2-1 PLL 结构图

输出时钟频率的计算方式如下：

clock_out = refclk / div_ref * loopc / divoutN;

其中，需要保证输入分频器的输出（refclk / div_ref）在 25~ 100MHz 的范围内，倍频模块倍频后的频率（refclk / div_ref * loopc）在 3.75GHz ~ 6.4GHz 的范围内。

PLL相关的配置及说明见表 2- 1。这些配置通过 PLL0-4 配置寄存器进行控制。

表 2- 1 PLL 相关配置信号说明表

信号	位数	方向	说明
pll_div_out0	7	R/W	PLL 输出时钟 0 分频数
pll_div_out1	7	R/W	PLL 输出时钟 1 分频数
pll_div_out2	7	R/W	PLL 输出时钟 2 分频数
pll_loopc	9	R/W	PLL 倍频乘数
pll_div_ref	7	R/W	PLL 输入分频数
pll_locked	1	RO	PLL 锁定
sel_pll_out0	1	R/W	选择 PLL 输出时钟 0
sel_pll_out1	1	R/W	选择 PLL 输出时钟 1
sel_pll_out2	1	R/W	选择 PLL 输出时钟 2
set_pll_param	1	R/W	设置 PLL 配置参数
pll_bypass	1	R/W	PLL 内部 bypass
pll_pd	1	R/W	PLL powerdown

当 CHIP_CONFIG[3:2]为 10b 时表示可通过软件更改 PLL 的输出频率。这种配置下，芯片启动时默认的时钟频率为外部参考时钟频率，需要在启动过程中对芯片时钟进行软件配置。通过软件修改时钟配置的过程如下：

1. 将 sel_pll_out*设置为 0；
2. 将 pll_pd 信号设置为 1；
3. 将 set_pll_param 设置为 0，
4. 设置 pll_div_ref/pll_loopc/pll_div_out*的值；
5. 将 set_pll_param 设置为 1；

6. 将 pll_pd 信号设置为 0；
7. 等待 PLL 锁定信号 pll_locked 变为 1；
8. 设置 sel_pll_out*为 1。

2.1.3 RTC 时钟

RTC 时钟频率要求为 32.768KHz。可选择外接晶体或者晶振，芯片内部 RTC 模块可以自适应这两种时钟输入，无需特别控制。

2.1.4 PCIe PHY 参考时钟

2K3000 的 PCIe 有 2 个 PHY，它们共用内部参考时钟。可以从下面三个时钟源对参考时钟进行选择：

1、外部 100MHz 单端参考时钟 SYS_SYSCLK

2、外部 100MHz 差分输入 (PCIE_REFCLKP/N)

3、USB PHY 的参考时钟（如下文）

另外，PCIe 作为 RapidIO 使用时由外部提供 156.25MHz 的差分参考时钟。这种情况下，另一路 PCIe 仍可从以上三种参考时钟进行选择。

2.1.5 USB PHY 参考时钟

USB3 PHY 的参考时钟有以下两种选择方式，通过芯片引脚 USB_CLKSEL 进行选择：

1、外接 25MHz 差分输入（芯片引脚 USB_REFCLKP/N）；

2、外接 25MHz 单端输入（芯片引脚 USB_REFCLKP）；

当系统需要 USB 唤醒功能时，必须提供 USB PHY 参考时钟；

2.1.6 PHY 时钟之间关系

对于 USB3/2、PCIe PHY，在初始化之前需要保证其参考时钟已处于稳定状态，下表列出各参考时钟稳定的标志以及判断标准。

表 2- 2 PHY 时钟之间关系

模块	参考时钟来源	参考时钟稳定标志
USB3 PHY	25MHz 差分/单端输入	输入时钟稳定
USB2 PHY	USB3 PHY	参考 USB3 PHY
PRG	USB3 PHY	参考 USB3 PHY
	100MHz 差分输入	输入时钟稳定
	系统时钟	输入时钟稳定
PCIe	PRG	PRG 模块输出时钟稳定(PRG 模块配置寄存器 1[3]变为 1)

2.2 处理器核分频及使能控制

处理器核分频使用配置寄存器指令对私有的分频配置寄存器进行访问，每个处理器核可以分别控制。

私有的分频配置寄存器访问使能通过“其它功能设置寄存器”上的对应位进行控制。该寄存器基地址为 0x1FE00000，偏移地址 0x0420。

表 2- 3 其它功能设置寄存器

位域	字段名	访问	复位值	描述
22	freqscale_percore	RW	0x0	使能每个核私有的调频寄存器
23	clken_percore	RW	0x0	使能每个核私有的时钟使能

当 freqscale_percore 被设置为 1 时，使用私有的分频配置寄存器中的 freqscale 位对自己的时钟进行分频设置（包括了 freqscale_mode）；当 clken_percore 被设置为 1 时，使用私有的分频配置寄存器中的 clken 位对时钟使能进行控制。

该配置寄存器定义如下。偏移地址为 0x1050。

表 2- 4 处理器核私有分频寄存器

位域	字段名	访问	复位值	描述
5	freqscale[0]	RW	0x0	当前处理器核的分频设置
4	freqscale_mode	RW	0x0	当前处理器核的分频模式选择 0: (n+1) /16 1: 1/ (n+1)
3	clken	RW	0x0	当前处理器核的时钟使能
2:0	freqscale[3:1]	RW	0x0	当前处理器核的分频设置

2.3 Stable Counter 分频及使能控制

Stable Counter 的分频机制与其它的类似。使用其它功能设置寄存器中的对应位进行设置。其基地址为 0x1FE00000，偏移地址 0x0420。

表 2- 5 其它功能设置寄存器

位域	字段名	访问	复位值	描述
21	stable_reset	RW	0x0	稳定时钟复位控制 1：置为复位状态 0：解除软件复位
40	freqscale_mode_stable	RW	0x0	Stable clock 的调频模式选择 0: (n+1) /8 1: 1/ (n+1)
46:44	freqscale_stable	RW	0x0	Stable clock 调频寄存器
47	clken_stable	RW	0x0	Stable clock 时钟使能

3 电源管理

本章对电源管理进行简单介绍，具体寄存器及使用方法请参考第 22 章。

3.1 电源管理模块介绍

- 龙芯 2K3000 电源管理模块提供系统功耗管理实现机制。
- 支持 Advanced Configuration and Power Interface, Version 4.0a (ACPI)，提供相应的功耗管理功能。
- 系统休眠与唤醒，支持 ACPI S3（待机到内存），ACPI S4（待机到硬盘），ACPI S5（软关机），并且支持电源失效检测和自动系统恢复。支持多种唤醒方式（USB，GMAC，电源开关等）。
- 动态性能功耗控制，支持处理器核 DVFS 控制。
- 系统时钟控制，模块时钟门控，多种方式调节频率。
- 提供温度管理控制功能。支持 3 级报警机制。

3.2 电源级别

表 3- 1 显示了系统支持的 ACPI 状态及其相关说明。

表 3-1 ACPI 状态说明

状态	描述
G0/S0	全部工作，该模式下系统全部工作。
G1/S1	暂不支持
G1/S3	Suspend to RAM (STR)，上下文保存到内存
G1/S4	Suspend to Disk (STD)，保存到硬盘，除唤醒电路全部掉电
G2/S5	Soft off，只有唤醒电路上电
G3	Mechanical off，所有供电失效

4 芯片配置寄存器

龙芯 2K3000 有大量的配置寄存器，多数分布于各个功能模块中，本章介绍芯片级的配置寄存器。其中 4.1 至 4.14 节配置寄存器的基地址为 0x1fe00000，4.15 节至 4.60 节配置寄存器的基地址由固件进行配置（一般默认为 0x10010000）。

关于地址空间分配的寄存器将在第 5 章介绍，中断相关寄存器将在第 7 章和第 8 章介绍，温度传感器将在第 9 章介绍，这里不再赘述。下面详细介绍其他的配置寄存器。

4.1 版本寄存器

版本寄存器。

地址偏移：0000h

默认值：016h

表 4-1 版本寄存器

位域	字段名	访问	描述
7:0	Version	R	配置寄存器版本号

4.2 芯片特性寄存器

该寄存器标识了一些软件相关的处理器特性，供软件在使能特定功能前查看。

地址偏移：0008h

默认值：17ffh

表 4-2 芯片特性寄存器

位域	字段名	访问	复位值	描述
0	Centigrade	R	1'b1	为 1 时，表示 IOCSR[0x428]有效
1	Node counter	R	1'b1	为 1 时，表示 IOCSR[0x408]有效
2	MSI	R	1'b1	为 1 时，表示 MSI 可用
3	EXT_IOI	R	1'b1	为 1 时，表示 EXT_IOI 可用
4	IPI_percore	R	1'b1	为 1 时，表示通过 IOCSR 私有地址进行 IPI 发送
5	Freq_percore	R	1'b1	为 1 时，表示通过 IOCSR 私有地址调整频率
6	Freq_scale	R	1'b1	为 1 时，表示动态分频功能可用
7	DVFS_v1	R	1'b1	为 1 时，表示动态调频 v1 可用
8	Tsensor	R	1'b1	为 1 时，表示温度传感器可用
9	中断译码	R	1'b1	为 1 时，中断引脚译码模式可用
10	扁平模式	R	1'b1	为 1 时，表示支持扁平模式
11	Guest Mode	WR	1'b0	KVM 虚拟机模式
12	Freq_scale_16	R	1'b1	为 1 时，表示支持 16 分频模式
13	-	R	1'b0	保留
14	SE enabled	WR	1'b0	为 1 时，表示 SE 功能已使能

15	DMSI	R	1'b0	为 1 时，表示直接中断机制可用
16	RMSI	R	1'b0	为 1 时，表示重映射中断机制可用

4.3 厂商名称

该寄存器用于标识厂商名称。

地址偏移：0010h

默认值：6e6f_7367_6e6f_6f4ch

表 4-3 厂商名称寄存器

位域	字段名	访问	描述
63:0	Vendor	R	字符串“Loongson”

4.4 芯片名称

该寄存器用于标识芯片名称。

地址偏移：0020h

默认值：0000_3030_3033_4b32h

表 4-4 芯片名称寄存器

位域	字段名	访问	描述
63:0	ID	R	字符串“2K3000”

4.5 功能设置寄存器

地址偏移：0180h

默认值：4100_FFBB_BB00_3DE0h

表 4-5 功能设置寄存器

位域	字段名	访问	描述
3:0	-	RW	保留
4	MC0_disable_confspace	RW	是否禁用 MC0 DDR 配置空间
5	MC0_default_confspace	RW	将所有内存访问路由至配置空间
6	-	RW	保留
7	-	RW	保留
8	MC0_clken	RW	是否使能 MC0
43:9	-	RW	保留
63:56	-	R	保留

4.6 内存预取控制

地址偏移：0188h

默认值：0000_0000_1000_0000h

表 4-6 内存预取控制寄存器

位域	字段名	访问	描述
33	stb_win_flush_en	RW	DC 预取窗口刷新使能
32	stb_win_en	RW	DC 预取窗口使能
31:00	stb_win_retiretime	RW	DC 预取窗口退出使能

4.7 功能采样寄存器

地址偏移：0190h

默认值：-

表 4-7 功能采样寄存器

位域	字段名	访问	描述
39:32	CHIP_CONFIG	R	主板配置控制

4.8 路由设置寄存器

以下寄存器用于控制芯片内的部分路由设置。

地址偏移：0400h

默认值：2000_0000_4000_80f0h

表 4-8 芯片路由设置寄存器

位域	字段名	访问	描述
3:0	scid_sel	RW	共享缓存散列位控制
19:4	Resv	RW	保留
20	IO_throt_en	RW	IO 访问空闲压缩使能
21	MISC_throt_en	RW	MISC 访问空闲压缩使能
22	SE_throt_en	RW	SE 访问空闲压缩使能
23	GRA_throt_en	RW	GRA 访问空闲压缩使能
51:24	Resv	RW	保留
55:52	Se_dma_coherent	RW	SE DMA 访存一致性控制
60:56	Resv	RW	保留
61	enable_gather_spi	RW	SPI 访问空闲压缩使能

4.9 扩展路由设置寄存器

以下寄存器用于控制芯片内的部分路由设置。

地址偏移：0410h

默认值：0000_fefe_fcfe_0070h

表 4-9 扩展路由设置寄存器

位域	字段名	访问	描述
15:00	Resv	RW	保留
23:16	io_throttle	RW	
31:24	misc_throttle	RW	
39:32	se_throttle	RW	
47:40	gra_throttle	RW	

4.10 其它功能设置寄存器

以下寄存器用于控制芯片内部分功能使能。

地址偏移：0420h

默认值：0000_f200_f808_0000h

表 4-10 其它功能设置寄存器

位域	字段名	访问	描述
0	disable_jtag	RW	完全禁用 JTAG 接口
1	disable_ejtag	RW	完全禁用 JTAG 调试接口
2	disable_LA132	RW	完全禁用 LA132
3	disable_ejtag132	RW	完全禁用 LA132 JTAG 调试接口
4	Disable_antifuse0	RW	禁用 fuse
5	Disable_antifuse1	RW	禁用 fuse
6	Disable_ID	RW	禁用 ID 修改
7	disable_ejtaggpu	RW	完全禁用 GPU JTAG 调试接口
8	resetn_LA132	RW	LA132 复位控制
9	sleeping_LA132	R	LA132 进入睡眠状态
10	soft_int_LA132	RW	LA132 核间中断寄存器
15:12	core_int_en_LA132	RW	LA132 对应每个核的 IO 中断使能
18:16	freqscale_LA132	RW	LA132 分频控制
19	clken_LA132	RW	LA132 时钟使能
20	Reserved	RW	保留
21	Reserved	RW	保留
22	freqscale_percore	RW	使能每个核私有的调频寄存器
23	clken_percore	RW	使能每个核私有的时钟使能
27:24	confbus_timeout	RW	配置总线超时时间设置，实际时间为2的幂次方
29:28	Reserved	RW	保留
35:32	freqscale_mode_core	RW	每个核的调频模式选择 0: (n+1) /8 1: 1/ (n+1)
36	Reserved	RW	保留

37	freqscale_mode_LA132	RW	LA132 的调频模式选择 0: (n+1) /8 1: 1/ (n+1)
39:38	Reserved		
40	freqscale_mode_stable	RW	Stable clock 的调频模式选择 0: (n+1) /8 1: 1/ (n+1)
43:41	Reserved	RW	保留
46:44	freqscale_stable	RW	Stable clock 调频寄存器
47	clken_stable	RW	Stable clock 时钟使能
48	EXT_INT_en	RW	扩展 IO 中断使能
49	INT_encode	RW	使能中断引脚编码模式
53:52	Reserved	RW	保留
54	Reserved	RW	保留
57:56	thsensor_sel	RW	温度传感器选择，只能为 0
62:60	Auto_scale	R	自动调频当前值
63	Auto_scale_doing	R	自动调频正在生效标志

4.11 摄氏温度寄存器

以下寄存器用于观测芯片内部温度传感器数值。只有当芯片特性寄存器[0]有效时，该寄存器可用。

地址偏移：0428h

表 4-11 温度观测寄存器

位域	字段名	访问	描述
7:0	Centigrade temperature	RO	摄氏温度 最高位为符号位，0 代表正，1 代表负。当温度为负值时，按位取反后为真实温度。
63:8	Reserved	RW	保留

4.12 FUSE0 观测寄存器

以下寄存器用于观测部分软件可见的 Fuse0 数值。

地址偏移：0460h

表 4-12 FUSE0 观测寄存器

位域	字段名	访问	描述
127:0	Fuse_0	RO	

4.13 FUSE1 观测寄存器

以下寄存器用于观测部分软件可见的 Fuse1 数值。

地址偏移：0470h

表 4-13 FUSE1 观测寄存器

位域	字段名	访问	描述
127:0	Fuse_1	RO	

4.14 GPU 功耗控制寄存器

以下寄存器用于实现 GPU 的动态调频调压功能。

地址偏移：0528h

表 4-14 GPU 功耗控制寄存器

位域	字段名	访问	描述
63:32	gpu_lp_req	RW	GPU 低功耗配置请求
31:0	gpu_lp_ack	RW	GPU 低功耗配置响应

4.15 IO 拆包控制寄存器

IO 拆包控制寄存器，对 IO 设备的拆包进行配置。

地址偏移：0410h

默认值：0000_0000_0007_D7FFh

表 4-15 IO 拆包控制寄存器

位域	名称	访问	描述
63:32	r_order_ctrl	R/W	读序控制，每位对应一个 IO 设备： 0：强序 1：弱序
31:00	w_order_ctrl	R/W	写序控制，每位对应一个 IO 设备： 0：强序 1：弱序

位域	模块
0	PCIe0 PORT0/RIO0
1	PCIe0 PORT1
2	PCIe0 PORT2
3	PCIe0 PORT3
4	PCIe1 PORT0/RIO1
5	PCIe1 PORT1
8	USB3
9	USB2
10	SATA
11	GMAC0
12	GMAC1
13	GMAC-MSI
14	HDA/HDA1/I2S

15	MSI
16	DMA
17	eMMC
18	SDIO0
19	SDIO1
20	PCM

4.16 IO 路由控制寄存器

IO 路由控制寄存器，对 IO 设备的路由进行配置。

地址偏移：0418h

默认值：0000_0000_0000_003Fh

表 4-16 IO 路由控制寄存器

位域	名称	访问	描述
63	rcmd_auto	R/W	读请求 cache 分配方式控制： 0：软件配置，由该寄存器 bit62 控制 1：硬件自动设置
62	rcmd_alloc	R/W	读请求 cache 分配方式软件配置控制： 0：分配 1：不分配
18:00	split_bypass	R/W	拆包控制,每位对应一个 IO 设备（位域对应关系参考上一节）： 0：需要拆包 1：不需要拆包

4.17 IO 通用配置寄存器 0

通用配置寄存器 0，对 IO 设备进行配置。

地址偏移：0420h

默认值：2480_0000_CCCC_3CE0h

表 4-17 通用配置寄存器 0

位域	名称	访问	描述
63	Reserved	R/W	保留
62	gpu_lpreq_soft	R/W	GPU 低功耗软硬件控制选择 0：硬件控制 1：软件控制
61	venc_clken	R/W	使能 vpu enc 的时钟 0：关闭时钟 1：打开时钟
60	venc_soft_reset	R/W	vpu enc 的软件复位，高有效
59	vdec_soft_reset	R/W	vpu dec 的软件复位，高有效
58	vdec_clken	R/W	使能 vpu dec 的时钟 0：关闭时钟 1：打开时钟
57:46	Reserved	R/W	保留

位域	名称	访问	描述
45	pcie1_p1_clk_ok	RO	pcie1 端口 1 时钟 ready 0：没有时钟 1：时钟正常
44	pcie1_p0_clk_ok	RO	pcie1 端口 0 时钟 ready 0：没有时钟 1：时钟正常
43	pcie0_p3_clk_ok	RO	pcie0 端口 3 时钟 ready 0：没有时钟 1：时钟正常
42	pcie0_p2_clk_ok	RO	pcie0 端口 2 时钟 ready 0：没有时钟 1：时钟正常
41	pcie0_p1_clk_ok	RO	pcie0 端口 1 时钟 ready 0：没有时钟 1：时钟正常
40	pcie0_p0_clk_ok	RO	pcie0 端口 0 时钟 ready 0：没有时钟 1：时钟正常
39:20	Reserved	R/W	保留
19	pcie1_p1_clken	R/W	使能 pcie1 的 port1 时钟 0：关闭时钟 1：打开时钟
18	pcie1_p0_clken	R/W	使能 pcie1 的 port0 时钟 0：关闭时钟 1：打开时钟
17	pcie1_enable	R/W	使能 pcie1 控制器 0：禁止访问 1：允许访问
16	pcie1_soft_reset	R/W	pcie1 的软件复位 0：解除复位 1：保持复位
15	pcie1_refclk_sel	R/W	PCIe1 参考时钟源选择： 1：选择从 PAD 输入参考时钟； 0：选择内部参考时钟；
14	pcie1_rio_mode	R/W	RapidIO 设备使能模式。 0：不使用 RapidIO 设备,使用 pcie1 设备 1：使用 RapidIO 设备
13	pcie0_p3_clken	R/W	使能 pcie0 的 port3 时钟 0：关闭时钟 1：打开时钟
12	pcie0_p2_clken	R/W	使能 pcie0 的 port2 时钟 0：关闭时钟 1：打开时钟
11	pcie0_p1_clken	R/W	使能 pcie0 的 port1 时钟 0：关闭时钟 1：打开时钟
10	pcie0_p0_clken	R/W	使能 pcie0 的 port0 时钟 0：关闭时钟 1：打开时钟
9	pcie0_enable	R/W	使能 pcie0 控制器 0：禁止访问 1：允许访问
8	pcie0_soft_reset	R/W	pcie0 的软件复位 0：解除复位 1：保持复位

位域	名称	访问	描述
7	dc_clken	R/W	使能 dc 的时钟 0：关闭时钟 1：打开时钟
6	gpu_clken	R/W	使能 gpu 的时钟 0：关闭时钟 1：打开时钟
5	Reserved	R/W	保留
4	gpu_soft_reset	R/W	gpu 的软件复位，高有效
3	Reserved	R/W	保留
2	pcie0_refclk_sel	R/W	PCIe0 的时钟选择。 1：选择 PAD 输入时钟 0：选择内部参考时钟
1	pcie0_rio_mode	R/W	RapidIO 设备使能模式。 0：不使用 RapidIO 设备,使用 pcie1 设备 1：使用 RapidIO 设备
0	default_route_cfg0	R/W	保留 该位必须设置为 0。

4.18 IO 通用配置寄存器 1

本寄存器包含 USB、SATA、GMAC、HDA/I2S、LPC、SPI 相关的配置信息。

地址偏移：0430h

默认值：1000_0900_00C9_99F2h

表 4-18 通用配置寄存器 1

位域	名称	访问	描述
63	hda1_int_route	R/W	HDA1 中断引脚控制 0：使用 hda1_int 1：使用 hda0_int
62	lpc_ioaddr_new	R/W	需要设置成 1
61	rtc_access_speed	R/W	RTC 访问通路控制 0：低速访问通路，读取 RTC 时间延迟长 1：高速访问通路，读取 RTC 时间延迟短
60	hda_dma_64	R/W	使能 HDA64 位 DMA 地址模式 0：使用 32 位 DMA 地址模式 1：使用 64 位 DMA 地址模式
59	pwm_soft_reset	R/W	PWM 控制器软件复位 0：解除复位 1：保持复位
58	i2c_soft_reset	R/W	I2C 控制器软件复位 0：解除复位 1：保持复位
57	uart_soft_reset	R/W	UART 控制器软件复位 0：解除复位 1：保持复位

位域	名称	访问	描述
56	can_soft_reset	R/W	CAN 控制器软件复位 0：解除复位 1：保持复位
55	lpc_soft_reset	R/W	LPC 控制器软件复位 0：解除复位 1：保持复位
54	spi_soft_reset	R/W	SPI 控制器软件复位 0：解除复位 1：保持复位
53	pcm_soft_reset	R/W	PCM 控制器软件复位 0：解除复位 1：保持复位
52:48	Reserved	R/W	保留
47	sdio_soft_reset	R/W	SDIO 控制器软件复位 0：解除复位 1：保持复位
46	emmc_soft_reset	R/W	eMMC 控制器软件复位 0：解除复位 1：保持复位
45	hda1_soft_reset	R/W	HDA1 控制器软件复位 0：解除复位 1：保持复位
44	hda0_soft_reset	R/W	HDA0 控制器软件复位 0：解除复位 1：保持复位
43	sata_clken	R/W	SATA 时钟使能 0：没有时钟 1：时钟正常
42	sata_en	R/W	SATA 访问使能 0：禁止访问 1：允许访问
41	Reserved	R/W	保留
40	sata_cntl_soft_reset	R/W	SATA 控制器软件复位 0：解除复位 1：保持复位
39	gpio50_int_remap	R/W	GPIO50 引脚中断重映射 0：默认映射方式 1：将 GPIO50 中断映射到 GPIO3 对应的中断引脚
38	gpio15_int_remap	R/W	GPIO15 引脚中断重映射 0：默认映射方式 1：将 GPIO15 中断映射到 GPIO2 对应的中断引脚

位域	名称	访问	描述
37	gpio14_int_remap	R/W	GPIO14 引脚中断重映射 0：默认映射方式 1：将 GPIO14 中断映射到 GPIO1 对应的中断引脚
36	gpio13_int_remap	R/W	GPIO13 引脚中断重映射 0：默认映射方式 1：将 GPIO13 中断映射到 GPIO0 对应的中断引脚
35	lio_en	R/W	LIO 控制器使能 0：禁止访问 1：允许访问
34	pcm_en	R/W	PCM 控制器使能 0：禁止访问 1：允许访问
33	hpet_int_ctrl	R/W	hpet 中断分离控制 0：HPET 的中断全部分配到 hpet0_int 1：HPET 的中断分别分配到 hpet0/1/2_int
32	lpc_en	R/W	LPC 控制器使能 0：禁止访问 1：允许访问
31:22	Reserved	R/W	保留
21	hda1_en	R/W	HDA1 控制器访问使能 0：禁止访问 1：允许访问
20	hda0_i2s_en	R/W	HDA0/I2S 控制器使能 0：使能 I2S 控制器 1：使能 HDA0 控制器
19	usb3_clken	R/W	usb3 时钟使能 0：没有时钟 1：时钟正常
18	usb3_en	R/W	usb3 访问使能 0：禁止访问 1：允许访问
17	usb3_phy_soft_reset	R/W	usb3 PHY 软件复位 0：解除复位 1：保持复位
16	usb3_cntl_soft_reset	R/W	usb3 控制器软件复位 0：解除复位 1：保持复位
15:12	Reserved	R/W	保留
11	usb2_clken	R/W	usb2 时钟使能 0：没有时钟 1：时钟正常

位域	名称	访问	描述
10	usb2_en	R/W	usb2 访问使能 0：禁止访问 1：允许访问
9	usb2_phy_soft_reset	R/W	usb2 PHY 软件复位 0：解除复位 1：保持复位
8	usb2_cntl_soft_reset	R/W	usb2 控制器软件复位 0：解除复位 1：保持复位
7	gmac1_clken	R/W	gmac1 时钟使能 0：没有时钟 1：时钟正常
6	gmac1_sdb_flowctrl	R/W	gmac1 流控使能 0：流控关闭 1：流控打开
5	gmac0_clken	R/W	gmac0 时钟使能 0：没有时钟 1：时钟正常
4	gmac0_sdb_flowctrl	R/W	gmac0 流控使能 0：流控关闭 1：流控打开
3	Reserved	R/W	保留
2	sata_mode	R/W	USB3 PORT3 的模式选择 0：USB3 模式 1：SATA 模式
1:0	Reserved	R/W	保留，软件不可更改

4.19 引脚复用配置寄存器

地址偏移：0440h

默认值：0000_0006_1938_3C07h

本寄存器包含引脚复用的相关配置信息，注意当开启某个功能时，需要关闭与之复用的其他所有功能。

表 4-19 引脚复用配置寄存器

位域	名称	访问	描述
36:0	pad_sel	R/W	引脚复用选择控制，每位对应一个/组 PAD，具体请参考数据手册。 为 1 代表使能 需注意同一个 PAD 的对应的 pad_sel 只能有一位被使能，否则会导致功能异常。 [03:00]-PWM3/2/1/0

位域	名称	访问	描述
			[07:04]-CAN3/2/1/0 [09:08]-I2C3/2 [11:10]-USBOC1/0 [18:12]-UART7/6/5/4/3/2/1 [19]-SATA_LED [20]-SPI1 [21]-LPC [22]-RGMII1 [23]-LIO [24]-HDA [25]-I2S [26]-AVS [27]-eMMC [29:28]-SDIO1/0 [33]-THERMTRIP [34]-PROCHOT [35]-UART4WIRE

4.20 USB OC 选择配置

本寄存器用来选择 USB 控制器所对应的 overcurrent 检测引脚。每两位作为一组，共有 2 种配置，00 代表选择 USB_OC0 信号，01 代表选择 USB_OC1 信号。每组对应一个 USB 端口。

地址偏移：0448h

默认值：0000_0000h

表 4-20 USB OC 选择配置

位域	名称	访问	描述
23:22	usb2p3_ocsel	R/W	USB2 控制器 PORT3 端口对应 OC 信号选择
21:20	usb2p2_ocsel	R/W	USB2 控制器 PORT2 端口对应 OC 信号选择
19:18	usb2p1_ocsel	R/W	USB2 控制器 PORT1 端口对应 OC 信号选择
17:16	usb2p0_ocsel	R/W	USB2 控制器 PORT0 端口对应 OC 信号选择
15:14	usb3p3_ocsel	R/W	XHCI-USB3.0 控制器中 PORT3 端口对应 OC 信号选择
13:12	usb3p2_ocsel	R/W	XHCI-USB3.0 控制器中 PORT2 端口对应 OC 信号选择
11:10	usb3p1_ocsel	R/W	XHCI-USB3.0 控制器中 PORT1 端口对应 OC 信号选择
9:8	usb3p0_ocsel	R/W	XHCI-USB3.0 控制器中 PORT0 端口对应 OC 信号选择
7:6	usb3u2p3_ocsel	R/W	XHCI-USB3.0 控制器中 USB2 PORT3 端口对应 OC 信号选择
5:4	usb3u2p2_ocsel	R/W	XHCI-USB3.0 控制器中 USB2 PORT2 端口对应 OC 信号选择
3:2	usb3u2p1_ocsel	R/W	XHCI-USB3.0 控制器中 USB2 PORT1 端口对应 OC 信号选择

位域	名称	访问	描述
1:0	usb3u2p0_ocsel	R/W	XHCI-USB3.0 控制器中 USB2 PORT0 端口对应 OC 信号选择

4.21 LIO 控制寄存器

本寄存器包含 LIO 接口的配置信息。

地址偏移：0450h

默认值：C708_AF34_C708_8834h

表 4-21 LIO 控制寄存器

位域	名称	访问	描述
27:24	LIO cs cfg	RW	LIO cs 地址选择 0：只使用 CS0，地址空间为[31:0]； 1：CS 使用地址[23:22]，地址空间为[21:0]； 2：CS 使用地址[24:23]，地址空间为[22:0]； 3：CS 使用地址[25:24]，地址空间为[23:0]； 4：CS 使用地址[26:25]，地址空间为[24:0]； 5：CS 使用地址[27:26]，地址空间为[25:0]； 6：CS 使用地址[28:27]，地址空间为[26:0]； 7：CS 使用地址[29:28]，地址空间为[27:0]； 8：CS 使用地址[30:29]，地址空间为[28:0]； 其他：CS 使用地址[31:30]，地址空间为[29:0]；
23:20	LIO waitdata	RW	
19:16	LIO adlock_cfg	RW	ADLOCK 时间软件配置
15	LIO big_mem	RW	LIO big_mem 模式控制
14	LIO iopf_en	RW	保留
13	LIO io_width_16	RW	IO 数据位宽： 0：8 位 1：16 位
12:8	LIO io_count_init_i	RW	IO 数据读取延迟（boot 时钟周期数）
7	LIO rom_width16	RW	Rom 数据位宽： 0：8 位 1：16 位
6:2	LIO rom_count_init_i	RW	ROM 数据读取延迟（boot 时钟周期数）
1:0	LIO clock_period_i	RW	内部等待计数器步长设置： 0：步长为 1 1：步长为 4 2：步长为 2 3：步长为 1

4.22 固定地址配置

本寄存器用来配置中断控制器和 HPET 模块的固定访问地址（可由 BIOS 更改）。如果不使用该固定地址，操作系统可以通过配置访问获取配置寄存器或者低速 IO 设备块的基地址，然后加上固定偏移来获得中断控制器和 HPET 模块的访问地址。使用固定地址来访问中断控制器和 HPET 模块可以加快操作系统的中断处理过程。该固定地址由操作系统决定。如果操作系统使用了与本芯片默认设置的固定地址不同的地址，BIOS 需修改本寄存器来与操作系统保持一致。

地址偏移：0460h

默认值：5FFF_E004_5FFF_F004h

表 4-22 固定地址配置

位域	名称	访问	描述
63:44	hpet_fix_addr	R/W	固定地址的 31 到 12 位
43:35	Reserved	R/W	保留
34	hpet_fix_addr_en	R/W	使能该固定地址配置
33:32	Reserved	R/W	保留
31:12	apic_fix_addr	R/W	固定地址的 31 到 12 位
11:3	Reserved	R/W	保留
2	apic_fix_addr_en	R/W	使能该固定地址配置
1:0	Reserved	R/W	保留

4.23 DMA 一致性配置寄存器

本寄存器用来配置 DMA 设备访存是否维护 cache 一致性

地址偏移：0470h

默认值：0000_0000_FFFF_FFFFh

表 4-23 DMA 一致性配置寄存器

位域	名称	访问	描述
31:30	Reserved	R/W	保留
29:28	emmc_cc	R/W	eMMC 一致性控制
26:25	hda_cc	R/W	HDA 一致性控制
24	sata_cc	R/W	SATA 一致性控制
21:20	gmac_cc	R/W	GMAC 一致性控制
18:16	usb_cc	R/W	USB 一致性控制
14	vpu_cc	R/W	VPU 一致性控制
13	dc_cc	R/W	DC 一致性控制
12	gpu_cc	R/W	GPU 一致性控制
9:8	rio_cc	R/W	RIO 两个端口一致性控制
6	Reserved	R/W	保留
5:4	pcie1_cc	R/W	PCIe1 两个端口一致性控制

位域	名称	访问	描述
3:0	pcie0_cc	R/W	PCIe0 四个端口一致性控制

4.24 PLL0 配置寄存器

该寄存器用来设置 PLL0。

地址偏移：0x0480

默认值：bb10_0000_0000_0000h

表 4-24 PLL0 配置寄存器

位域	名称	访问	描述
63	aldo_en	R/W	
62:60	aldo_ctrl	R/W	
59	dldo_en	R/W	
58:56	dldo_ctrl	R/W	
55	aldo_bypass	R/W	
54	dldo_bypass	R/W	
53	sscclk_sel	R/W	展频时钟选择控制
52:48	Reserved	R/W	保留
47	node_clk_sel	R/W	NODE 时钟选择，为 1 选择 NODE_BAK 时钟
46	Reserved	R/W	保留
45	pll_pd	R/W	PLL powerdown
44	pll_bypass	R/W	PLL 内部 bypass
43	set_pll_param	R/W	设置 PLL 配置参数
42	sel_pll_out2	R/W	选择 PLL 输出时钟 2
41	sel_pll_out1	R/W	选择 PLL 输出时钟 1
40	sel_pll_out0	R/W	选择 PLL 输出时钟 0
39	pll_locked	RO	PLL 锁定
38:32	pll_div_ref	R/W	PLL 输入分频数
31:30	Reserved	R/W	保留
29:21	pll_loopc	R/W	PLL 倍频乘数
20:14	pll_div_out2	R/W	PLL 输出时钟 2 分频数
13:7	pll_div_out1	R/W	PLL 输出时钟 1 分频数
6:0	pll_div_out0	R/W	PLL 输出时钟 0 分频数

4.25 PLL1 配置寄存器

该寄存器用来设置 PLL1。

地址偏移：0x0490

默认值：bb17_0000_0000_0000h

表 4-25 PLL1 配置寄存器

位域	名称	访问	描述
63	aldo_en	R/W	
62:60	aldo_ctrl	R/W	
59	dldo_en	R/W	
58:56	dldo_ctrl	R/W	
55	aldo_bypass	R/W	
54	dldo_bypass	R/W	
53	sscclk_sel	R/W	展频时钟选择控制
52	mem_clken	R/W	内存时钟使能控制
51:50	soft_memdiv_mode	R/W	
49	soft_mc_resetn	R/W	
48	memdiv_resetn	R/W	
47:46	Reserved	R/W	保留
45	pll_pd	R/W	PLL powerdown
44	pll_bypass	R/W	PLL 内部 bypass
43	set_pll_param	R/W	设置 PLL 配置参数
42	sel_pll_out2	R/W	选择 PLL 输出时钟 2
41	sel_pll_out1	R/W	选择 PLL 输出时钟 1
40	sel_pll_out0	R/W	选择 PLL 输出时钟 0
39	pll_locked	RO	PLL 锁定
38:32	pll_div_ref	R/W	PLL 输入分频数
31:30	Reserved	R/W	保留
29:21	pll_loopc	R/W	PLL 倍频乘数
20:14	pll_div_out2	R/W	PLL 输出时钟 2 分频数
13:7	pll_div_out1	R/W	PLL 输出时钟 1 分频数
6:0	pll_div_out0	R/W	PLL 输出时钟 0 分频数

4.26 PLL2 配置寄存器

该寄存器用来设置 PLL2。

地址偏移：0x04a0

默认值：bb10_0000_0000_0000h

表 4-26 PLL2 配置寄存器

位域	名称	访问	描述
63	aldo_en	R/W	
62:60	aldo_ctrl	R/W	
59	dldo_en	R/W	
58:56	dldo_ctrl	R/W	
55	aldo_bypass	R/W	
54	dldo_bypass	R/W	

位域	名称	访问	描述
53	sscclk_sel	R/W	展频时钟选择控制
52:46	Reserved	R/W	保留
45	pll_pd	R/W	PLL powerdown
44	pll_bypass	R/W	PLL 内部 bypass
43	set_pll_param	R/W	设置 PLL 配置参数
42	sel_pll_out2	R/W	选择 PLL 输出时钟 2
41	sel_pll_out1	R/W	选择 PLL 输出时钟 1
40	sel_pll_out0	R/W	选择 PLL 输出时钟 0
39	pll_locked	RO	PLL 锁定
38:32	pll_div_ref	R/W	PLL 输入分频数
31:30	Reserved	R/W	保留
29:21	pll_loopc	R/W	PLL 倍频乘数
20:14	pll_div_out2	R/W	PLL 输出时钟 2 分频数
13:7	pll_div_out1	R/W	PLL 输出时钟 1 分频数
6:0	pll_div_out0	R/W	PLL 输出时钟 0 分频数

4.27 PLL3 配置寄存器

该寄存器用来设置 PLL3。

地址偏移：0x04b0

默认值：bb10_0000_0000_0000h

表 4-27 PLL3 配置寄存器

位域	名称	访问	描述
63	aldo_en	R/W	
62:60	aldo_ctrl	R/W	
59	dldo_en	R/W	
58:56	dldo_ctrl	R/W	
55	aldo_bypass	R/W	
54	dldo_bypass	R/W	
53	sscclk_sel	R/W	展频时钟选择控制
52:46	Reserved	R/W	保留
45	pll_pd	R/W	PLL powerdown
44	pll_bypass	R/W	PLL 内部 bypass
43	set_pll_param	R/W	设置 PLL 配置参数
42	sel_pll_out2	R/W	选择 PLL 输出时钟 2
41	sel_pll_out1	R/W	选择 PLL 输出时钟 1
40	sel_pll_out0	R/W	选择 PLL 输出时钟 0
39	pll_locked	RO	PLL 锁定
38:32	pll_div_ref	R/W	PLL 输入分频数
31:30	Reserved	R/W	保留

位域	名称	访问	描述
29:21	pll_loopc	R/W	PLL 倍频乘数
20:14	pll_div_out2	R/W	PLL 输出时钟 2 分频数
13:7	pll_div_out1	R/W	PLL 输出时钟 1 分频数
6:0	pll_div_out0	R/W	PLL 输出时钟 0 分频数

4.28 PLL4 配置寄存器

该寄存器用来设置 PLL4。

地址偏移：0x04c0

默认值：bb10_0000_0000_0000h

表 4-28 PLL4 配置寄存器

位域	名称	访问	描述
63	aldo_en	R/W	
62:60	aldo_ctrl	R/W	
59	dldo_en	R/W	
58:56	dldo_ctrl	R/W	
55	aldo_bypass	R/W	
54	dldo_bypass	R/W	
53	sscclk_sel	R/W	展频时钟选择控制
52:46	Reserved	R/W	保留
45	pll_pd	R/W	PLL powerdown
44	pll_bypass	R/W	PLL 内部 bypass
43	set_pll_param	R/W	设置 PLL 配置参数
42	sel_pll_out2	R/W	选择 PLL 输出时钟 2
41	sel_pll_out1	R/W	选择 PLL 输出时钟 1
40	sel_pll_out0	R/W	选择 PLL 输出时钟 0
39	pll_locked	RO	PLL 锁定
38:32	pll_div_ref	R/W	PLL 输入分频数
31:30	Reserved	R/W	保留
29:21	pll_loopc	R/W	PLL 倍频乘数
20:14	pll_div_out2	R/W	PLL 输出时钟 2 分频数
13:7	pll_div_out1	R/W	PLL 输出时钟 1 分频数
6:0	pll_div_out0	R/W	PLL 输出时钟 0 分频数

4.29 FREQSCALE 配置寄存器

该寄存器用于配置时钟的分频系数。

地址偏移：0x04d0

默认值：FFFF_FFFF_FFFF_FFF7h

表 4-29 FRESCALE 配置寄存器

位域	名称	访问	描述
63:52	Reserved	R/W	保留
51	pcie_clken	R/W	PCIe 模块时钟使能
50:48	pcie_freqscale	R/W	PCIe 时钟分频数
47	i2s_clken	R/W	I2S 时钟使能
46:44	i2s_freqscale	R/W	I2S 时钟分频数
43	gpu_clken	R/W	GPU 时钟使能
42:40	gpu_freqscale	R/W	GPU 时钟分频数
39	vpu_clken	R/W	VPU 时钟使能
38:36	vpu_freqscale	R/W	VPU 时钟分频数
35	io_clken	R/W	IO 网络时钟使能
34:32	io_freqscale	R/W	IO 网络时钟分频数
25:23	boot_freqscale	R/W	boot时钟分频数
22:20	apb_freqscale	R/W	apb时钟分频数
18:16	usb3_freqscale	R/W	USB3 控制器时钟分频数
14:12	sata_freqscale	R/W	SATA控制器时钟分频数
11:7	Reserved	R/W	保留
6:4	usb2_freqscale		USB2 控制器时钟分频数
3	node_freqscale_mode	R/W	NODE 网络时钟分频模式
2:0	node_freqscale	R/W	NODE 网络时钟分频数

Freqscale 分频计算公式为：

mode 为 0 时：fout= fin * (freqscale + 1) /8

mode 为 1 时：fout= fin * 1/ (freqscale + 1)

对于分频模式没有相应配置位的时钟分频数，其分频模式固定为 0.

4.30 GRA 拆包控制寄存器

GRA 拆包控制寄存器，对 GRA 设备的拆包进行配置。

地址偏移：0510h

默认值：0000_0000_0000_000Fh

表 4-30 GRA 拆包控制寄存器

位域	名称	访问	描述
63:32	r_order_ctrl	R/W	读序控制，每位对应一个 GRA 设备： 0：强序 1：弱序
31:00	w_order_ctrl	R/W	写序控制，每位对应一个 GRA 设备： 0：强序 1：弱序

位域	模块
0	GPU

1	DC
2	VDEC
3	VENC

4.31 GRA 路由控制寄存器

GRA 路由控制寄存器，对 GRA 设备的路由进行配置。

地址偏移：0518h

默认值：0000_0000_0000_001Fh

表 4-31 GRA 路由控制寄存器

位域	名称	访问	描述
63	rcmd_auto	R/W	读请求 cache 分配方式控制： 0：软件配置，由该寄存器 bit62 控制 1：硬件自动设置
62	rcmd_alloc	R/W	读请求 cache 分配方式软件配置控制： 0：分配 1：不分配
18:00	split_bypass	R/W	拆包控制，每位对应一个 GRA 设备 （位域对应关系参考上一节）： 0：需要拆包 1：不需要拆包

4.32 PCIe0 配置寄存器 0

本组寄存器包含对 PCIe0 的控制信号和状态采集信息。

地址偏移：0x0580

默认值：8000_0000_0f00_8191h

表 4-32 PCIe0 配置寄存器 0

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	header_clk_gate_en	R/W	PCIe 控制器 header 空间寄存器的自动时钟门控使能 0：不使能 1：使能
59:56	phy_aux_clk_div	R/W	供给 PHY 的辅助时钟分频系数 供给 PHY 的辅助时钟的频率为 (100/(phy_aux_clk_div+1))Mhz
55:52	maclk_idle_gate_en	R/w	从低位到高位分别对应 PCIe 端口 p0～p3 是否开启低 axi master 通道的空闲时关闭时钟的功能 0：不开启 1：开启
51:48	saclk_idle_gate_en	R/W	从低位到高位分别对应 PCIe 端口 p0～p3 是否开启低 axi slave 通道的空闲时关闭时钟的功能 0：不开启 1：开启
47:44	lp_cg_en	R/W	从低位到高位分别对应 PCIe 端口 p0～p3 是否开启低功耗状态下自动关闭时钟的功能 0：不开启低功耗状态下 PCIe 控制器的自动关闭时钟功能 1：开启低功耗状态下 PCIe 控制器的自动关闭时钟功能

43:40	force_pcie_cc	R/W	从低位到高位分别对应 PCIe 端口 p0～p3 的 axi master 是否使用软件设定的 cache 属性 0：由 PCIe 包的 no-snoop 属性决定 PCIe 控制器向内部总线发送请求的 cache 属性； 1：由 conf_pcie_cc 的配置值决定 PCIe 控制器向内部总线发送请求的 cache 属性；
39:36	conf_pcie_cc	R/w	从低位到高位分别对应 PCIe 端口 p0～p3 的软件设定的 axi master 发出请求的 cache 属性设定值 0：un-cache；1：cache
35:32	lane_shut	R/W	从低位到高位分别对应 F0 模块的 PCIe PHY 的 lane0～lane3 的关闭信号； 0：对应的 lane 处在正常工作模式；1：关闭对应的 lane；
31:28	port_rst_soft	R/W	从低位到高位分别对应 PCIe 端口 p0～p3 的软件复位 当需要对对应的 PCIe 端口进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
27:24	link_down_reset_en	R/W	从低位到高位分别对应 p0～p3 的链路断开的复位使能 0：链路断开时不产生复位；1：链路断开时产生复位；
23	p3_pipe_soft_rst	R/W	对 F0 的 PIPE 端口 3 进行软件复位 当需要对 p3 的 PIPE 进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
22	p2_pipe_soft_rst	R/W	对 F0 的 PIPE 端口 2 进行软件复位 当需要对 p2 的 PIPE 进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
21	p1_pipe_soft_rst	R/W	对 F0 的 PIPE 端口 1 进行软件复位 当需要对 p1 的 PIPE 进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
20	p0_pipe_soft_rst	R/W	对 F0 的 PIPE 端口 0 进行软件复位 当需要对 p0 的 PIPE 进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
19	aux_clk_en	R/W	0：关闭 F0 模块 PCIe PHY 的辅助时钟使能； 1：开启 F0 模块 PCIe PHY 的辅助时钟使能； 如果要让 F0 的 PCIe PHY 进入 L2/3 状态需要将此位置 1
18:17	Reserved	R/W	0：un-cache；1：cache
16	phy_soft_cfg_done	R/W	0：PHY 未完成软件配置；1：PHY 已完成软件配置 软件在完成 PHY 的参数配置后需将此位置 1
15	axi_protect_en	R/W	0：不使用 PCIe 控制器的总线保护功能；1：使用 PCIe 控制器的总线保护功能 此位的值需要在释放 PCIe0 模块的复位（对通用配置寄存器 0 进行操作）前进行修改
14:13	Reserved	R/W	保留
12	axi_awsplit_final	R/W	此位为 1 时启动对单拍的非连续字节的写的拆包功能
11:4	Reserved	R/W	保留
3	app_req_retry_en	R/W	保留
2	do_sleep	R/W	保留
1	power_fault	R/W	保留
0	slot_wake_n	R/W	保留

4.33 PCIe0 配置寄存器 1

本组寄存器包含对 PCIe0 的控制信号和状态采集信息。

地址偏移：0x0588

默认值：0006_0000h

表 4-33 PCIe0 配置寄存器 1

位域	名称	访问	描述
63	p3_link_up	RO	0：F0 的 p3 的链路未建立好； 1：F0 的 p3 的链路已建立好； 在链路建立好之前不允许通过 p3 访问外部设备，否则内部总线会卡死；
62:57	p3_link_state	RO	F0 的 p3 端口的 PCIe 链路状态机状态
56	p2_link_up	RO	0：F0 的 p2 的链路未建立好； 1：F0 的 p2 的链路已建立好； 在链路建立好之前不允许通过 p2 访问外部设备，否则内部总线会卡死；
55:50	p2_link_state	RO	F0 的 p2 端口的 PCIe 链路状态机状态
49	p1_link_up	RO	0：F0 的 p1 的链路未建立好； 1：F0 的 p1 的链路已建立好； 在链路建立好之前不允许通过 p1 访问外部设备，否则内部总线会卡死；
48:43	p1_link_state	RO	F0 的 p1 端口的 PCIe 链路状态机状态
42	p0_link_up	RO	0：F0 的 p0 的链路未建立好； 1：F0 的 p0 的链路已建立好； 在链路建立好之前不允许通过 p0 访问外部设备，否则内部总线会卡死；
41:36	p0_link_state	RO	F0 的 p0 端口的 PCIe 链路状态机状态
35:32	phy_ready	RO	从低位到高位分别对应 PCIe PHY 送给 p0～p3 端口的 phy_ready 的值
31:28	Reserved	R/W	保留
27	x4_mode_soft	R/W	当 x4_mode_soft 为 1 时，使用 x4_mode_val 的值来设置 PHY 是工作在 1X4 还是 4X1 模式 当 x4_mode_soft 为 0 时，PHY 工作在 1X4 模式
26	x4_mode_val	R/W	当 x4_mode_soft 为 1 时，此位为 1 表示 PHY 工作在 1X4 模式；此位为 0 表示 PHY 工作在 4X1 模式
25	rst_appli_n	R/W	对 RIO 功能使用 PHY 接口进行复位控制 0：对 RIO 的 PHY 接口进行复位 1：结束 RIO 的 PHY 接口的复位
24	phy_rst_soft	R/W	向此位写入 1 将置 PHY 的复位信号有效，再写入 0 则结束 PHY 的复位 用于让软件对 F0 模块的 PCIe PHY 进行总复位
23	Reserved	R/W	保留
22	warm_wait_bypass	R/W	此位为 1 时 PCIe 控制器的热复位控制跳过对 PHY 的 PLL 状态的等待
21	phy_state_bypass	R/W	此位为 1 时 PCIe 控制器的复位控制跳过对 PHY_READY 的状态观测；否则在等待 PHY_READY 为 1 时 PCIe 控制器的复位才能继续并结束
20	prg_state_bypass	R/W	此位为 1 时，PHY 与 PCIe 控制器的复位控制跳过对 prg_hfc_ready 状态的观察；否则要等待 prg_hfc_ready 为 1 后 PHY 与 PCIe 控制器复位才能继续并结束。 当 F0 使用外部参考时钟时，需要将此位配置为 1。
19	Reserved	R/W	保留
18	p3_warm_rst_clean	RO	1：F0 的 p3 端口在热复位时内部总线处在干净状态； 0：F0 的 p3 端口在热复位时内部总线有未完成请求，这些未完成请求由总线保护功能进行结束；
17	p2_warm_rst_clean	RO	1：F0 的 p2 端口在热复位时内部总线处在干净状态； 0：F0 的 p2 端口在热复位时内部总线有未完成请求，这些未完成请求由总线保护功能进行结束；
16	p1_warm_rst_clean	RO	1：F0 的 p1 端口在热复位时内部总线处在干净状态； 0：F0 的 p1 端口在热复位时内部总线有未完成请求，这些未完成请求由总线保护功能进行结束；

15	p0_warm_rst_clean	RO	1：F0 的 p0 端口在热复位时内部总线处在干净状态； 0：F0 的 p0 端口在热复位时内部总线有未完成请求，这些未完成请求由总线保护功能进行结束；
14	phy_status_p2	RO	F0 控制器中 port2 的 PIPE 接口看到的 phystatus
13	phy_status_p2	RO	F0 控制器中 port2 的 PIPE 接口看到的 phystatus
12	phy_status_p1	RO	F0 控制器中 port1 的 PIPE 接口看到的 phystatus
11	phy_status_p0	RO	F0 控制器中 port0 的 PIPE 接口 lane0 看到的 phystatus
10	phy_rstn	RO	此位为 0 表示 PHY 处在复位状态
9	prg_hfc_ready	RO	此位为 1 表示为 PHY 产生内部参考时钟的 PRG 的 PLL 完成时钟锁定
8	phy_hfc_ready	RO	此位为 1 表示 PHY 的 PLL 完成时钟锁定
7	phy_init_cfg_stop	RO	此位为 1 表示 PHY 的复位后预置初始化配置过程中出现错误
6	phy_init_cfg_busy	RO	此为 1 表示 PHY 正在进行其复位后的预置初始化配置
5:2	Reserved	RO	保留
1	phy_init_cfg_stop	RO	此位为 1 表示 PHY 的复位后预置初始化配置过程被停止
0	phy_init_cfg_done	RO	此为 1 表示 PHY 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 PHY 进行配置访问

4.34 PCIe0 配置寄存器 2

本组寄存器包含对 PCIe0 的状态采集信息。

地址偏移：0598h
默认值：0000_0000h

表 4-34 PCIe0 配置寄存器 2

位域	名称	访问	描述
63:62	Reserved	R/W	保留
61:58	lane0_tx_pset	RO	F0 中 PCIe PHY 的 lane0 的 TX 使用的 preset_index 值
57:40	lane0_tx_coef	RO	F0 中 PCIe PHY 的 lane0 的 TX 使用的预加重、去加重系数
39	p3_m_done	RO	F0 模块的 p3 的内部总线主端口状态 0：有未完成总线；1：无未完成总线操作；
38	p2_m_done	RO	F0 模块的 p2 的内部总线主端口状态 0：有未完成总线；1：无未完成总线操作；
37	p1_m_done	RO	F0 模块的 p1 的内部总线主端口状态 0：有未完成总线；1：无未完成总线操作；
36	p0_m_done	RO	F0 模块的 p0 的内部总线主端口状态 0：有未完成总线；1：无未完成总线操作；
35	p3_s_done	RO	F0 模块的 p3 的内部总线从端口状态 0：有未完成总线；1：无未完成总线操作；
34	p2_s_done	RO	F0 模块的 p2 的内部总线从端口状态 0：有未完成总线；1：无未完成总线操作；
33	p1_s_done	RO	F0 模块的 p1 的内部总线从端口状态 0：有未完成总线；1：无未完成总线操作；
32	p0_s_done	RO	F0 模块的 p0 的内部总线从端口状态 0：有未完成总线；1：无未完成总线操作；
31:0	Reserved	R/W	保留

4.35 PCIe1 配置寄存器 0

本组寄存器包含对 PCIe1 的控制信号和状态采集信息。

地址偏移：0x05a0

默认值：8000_0000_0f00_8191h

表 4-35 PCIe1 配置寄存器 0

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	header_clk_gate_en	R/W	PCIe 控制器 header 空间寄存器的自动时钟门控使能 0：不使能 1：使能
59:56	phy_aux_clk_div	R/W	供给 PHY 的辅助时钟分频系数 供给 PHY 的辅助时钟的频率为 (100/(phy_aux_clk_div+1)) Mhz
55:54	Reserved	R/W	保留
53:52	maclk_idle_gate_en	R/w	从低位到高位分别对应 PCIe 端口 p0～p2 是否开启低 axi master 通道的空闲时关闭时钟的功能 0：不开启 1：开启
51:50	Reserved	R/W	保留
49:48	saclk_idle_gate_en	R/W	从低位到高位分别对应 PCIe 端口 p0～p2 是否开启低 axi slave 通道的空闲时关闭时钟的功能 0：不开启 1：开启
47:46	Reserved	R/W	保留
45:44	lp_cg_en	R/W	从低位到高位分别对应 PCIe 端口 p0～p1 是否开启低功耗状态下自动关闭时钟的功能 0：不开启低功耗状态下 PCIe 控制器的自动关闭时钟功能 1：开启低功耗状态下 PCIe 控制器的自动关闭时钟功能
43:42	Reserved	R/W	保留
41:40	force_pcie_cc	R/W	从低位到高位分别对应 PCIe 端口 p0～p1 的 axi master 是否使用软件设定的 cache 属性 0：由 PCIe 包的 no-snoop 属性决定 PCIe 控制器向内部总线发送请求的 cache 属性； 1：由 conf_pcie_cc 的配置值决定 PCIe 控制器向内部总线发送请求的 cache 属性；
39:38	Reserved	R/W	保留
37:36	conf_pcie_cc	R/w	从低位到高位分别对应 PCIe 端口 p0～p1 的软件设定的 axi master 发出请求的 cache 属性设定值 0：un-cache；1：cache
35:32	lane_shut	R/W	从低位到高位分别对应模块的 PCIe PHY 的 lane0～lane3 的关闭信号； 0：对应的 lane 处在正常工作模式；1：关闭对应的 lane；
31:30	Reserved	R/W	保留
29:28	port_rst_soft	R/W	从低位到高位分别对应 PCIe 端口 p0～p1 的软件复位 当需要对对应的 PCIe 端口进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
27:26	Reserved	R/W	保留
25:24	link_down_reset_en	R/W	从低位到高位分别对应 p0～p1 的链路断开的复位使能 0：链路断开时不产生复位；1：链路断开时产生复位；
23:22	Reserved	R/W	保留
21	p1_pipe_soft_rst	R/W	对 PIPE 端口 1 进行软件复位 当需要对 p1 的 PIPE 进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
20	p0_pipe_soft_rst	R/W	对 PIPE 端口 0 进行软件复位 当需要对 p0 的 PIPE 进行软件复位时，需先写入 1 置复位，再写入 0 撤销复位
19	aux_clk_en	R/W	0：关闭模块 PCIe PHY 的辅助时钟使能； 1：开启模块 PCIe PHY 的辅助时钟使能； 如果要让 F1 的 PCIe PHY 进入 L2/3 状态需要将此位置 1
18:17	Reserved	R/W	保留
16	phy_soft_cfg_done	R/W	0：PHY 未完成软件配置；1：PHY 已完成软件配置 软件在完成 PHY 的参数配置后需将此位置 1

15	axi_protect_en	R/W	0：不使用 PCIe 控制器的总线保护功能；1：使用 PCIe 控制器的总线保护功能 此位的值需要在释放 PCIe1 模块的复位（对通用配置寄存器 0 进行操作）前进行修改
14:13	Reserved	R/W	保留
12	axi_awsplit_final	R/W	此位为 1 时启动对单拍的非连续字节的写的拆包功能
11:4	Reserved	R/W	保留
3	app_req_retry_en	R/W	保留
2	do_sleep	R/W	保留
1	power_fault	R/W	保留
0	slot_wake_n	R/W	保留

4.36 PCIe1 配置寄存器 1

本组寄存器包含对 PCIe1 的控制信号和状态采集信息。

地址偏移：0x05a8

默认值：0006_0000h

表 4-36 PCIe1 配置寄存器 1

位域	名称	访问	描述
63:50	Reserved	RO	保留
49	p1_link_up	RO	0：F1 的 p1 的链路未建立好； 1：F1 的 p1 的链路已建立好； 在链路建立好之前不允许通过 p1 访问外部设备，否则内部总线会卡死；
48:43	p1_link_state	RO	F1 的 p1 端口的 PCIe 链路状态机状态
42	p0_link_up	RO	0：F1 的 p0 的链路未建立好； 1：F1 的 p0 的链路已建立好； 在链路建立好之前不允许通过 p0 访问外部设备，否则内部总线会卡死；
41:36	p0_link_state	RO	F1 的 p0 端口的 PCIe 链路状态机状态
35:34	Reserved	RO	保留
33:32	phy_ready	RO	从低位到高位分别对应 PCIe PHY 送给 p0～p1 端口的 phy_ready 的值
31:28	Reserved	R/W	保留
27	x4_mode_soft	R/W	当 x4_mode_soft 为 1 时，使用 x4_mode_val 的值来设置 PHY 是工作在 1X4 还是 4X1 模式 当 x4_mode_soft 为 0 时，PHY 工作在 1X4 模式
26	x4_mode_val	R/W	当 x4_mode_soft 为 1 时，此位为 1 表示 PHY 工作在 1X4 模式；此位为 0 表示 PHY 工作在 4X1 模式
25	rst_appli_n	R/W	对 RIO 功能使用 PHY 接口进行复位控制 0：对 RIO 的 PHY 接口进行复位 1：结束 RIO 的 PHY 接口的复位
24	phy_rst_soft	R/W	向此位写入 1 将置 PHY 的复位信号有效，再写入 0 则结束 PHY 的复位 用于让软件对 F1 模块的 PCIe PHY 进行总复位
23	Reserved	R/W	保留
22	warm_wait_bypass	R/W	此位为 1 时 PCIe 控制器的热复位控制跳过对 PHY 的 PLL 状态的等待
21	phy_state_bypass	R/W	此位为 1 时 PCIe 控制器的复位控制跳过对 PHY_READY 的状态观测；否则在等待 PHY_READY 为 1 时 PCIe 控制器的复位才能继续并结束。
20	prg_state_bypass	R/W	此位为 1 时，PHY 与 PCIe 控制器的复位控制跳过对 prg_hfc_ready 状态的观察；否则要等待 prg_hfc_ready 为 1 后 PHY 与 PCIe 控制器复位才能继续

			并结束。
19	Reserved	R/W	保留
18:17	Reserved	RO	保留
16	p1_warm_rst_clean	RO	1：F1 的 p1 端口在热复位时内部总线处在干净状态； 0：F1 的 p1 端口在热复位时内部总线有未完成请求，这些未完成请求由总线保护功能进行结束；
15	p0_warm_rst_clean	RO	1：F1 的 p0 端口在热复位时内部总线处在干净状态； 0：F1 的 p0 端口在热复位时内部总线有未完成请求，这些未完成请求由总线保护功能进行结束；
14:13	Reserved	RO	保留
12	phy_status_p1	RO	F1 控制器中 port1 的 PIPE 接口看到的 phystatus
11	phy_status_p0	RO	F1 控制器中 port0 的 PIPE 接口 lane0 看到的 phystatus
10	phy_rstn	RO	此位为 0 表示 PHY 处在复位状态
9	prg_hfc_ready	RO	此位为 1 表示为 PHY 产生内部参考时钟的 PRG 的 PLL 完成时钟锁定
8	phy_hfc_ready	RO	此位为 1 表示 PHY 的 PLL 完成时钟锁定
7	phy_init_cfg_stop	RO	此位为 1 表示 PHY 的复位后预置初始化配置过程中出现错误
6	phy_init_cfg_busy	RO	此为 1 表示 PHY 正在进行其复位后的预置初始化配置
5:2	Reserved	RO	保留
1	phy_init_cfg_stop	RO	此位为 1 表示 PHY 的复位后预置初始化配置过程被停止
0	phy_init_cfg_done	RO	此为 1 表示 PHY 在其复位后完成了预置的初始化配置 此位为 1 后才允许对软件对 PHY 进行配置访问

4.37 PCIe1 配置寄存器 2

本组寄存器包含对 PCIe1 的状态采集信息。

地址偏移：05b8h
默认值：0000_0000h

表 4-37 PCIe1 配置寄存器 2

位域	名称	访问	描述
63:62	Reserved	R/W	保留
61：58	lane0_tx_pset	RO	F0 中 PCIe PHY 的 lane0 的 TX 使用的 preset_index 值
57：40	lane0_tx_coef	RO	F0 中 PCIe PHY 的 lane0 的 TX 使用的预加重、去加重系数
39：38	Reserved	RO	保留
37	p1_m_done	RO	F0 模块的 p1 的内部总线主端口状态 0：有未完成总线；1：无未完成总线操作；
36	p0_m_done	RO	F0 模块的 p0 的内部总线主端口状态 0：有未完成总线；1：无未完成总线操作；
35：34	Reserved	RO	保留
33	p1_s_done	RO	F0 模块的 p1 的内部总线从端口状态 0：有未完成总线；1：无未完成总线操作；
32	p0_s_done	RO	F0 模块的 p0 的内部总线从端口状态 0：有未完成总线；1：无未完成总线操作；
31:0	Reserved	R/W	保留

4.38 PCIe 配置访问路由控制寄存器

本寄存器用来控制 PCIe 配置访问的路由。当配置访问命中在 PCIe 控制器二级总线（secondary bus）上，但设备号非 0 时，可通过配置该寄存器禁止该配置访问被转发到二级总线上。每个 PCIe 端口分别有一个配置位，配置为 0 代表禁止。

地址偏移：0x0638

默认值：0000_0000h

表 4-38 PCIe0 PHY 配置寄存器

位域	名称	访问	描述
31：14	-	R/W	保留
13:0	pcie_route	R/W	每个 PCIe 控制器有一个控制位，对应关系如下： 03:00 对应 PCIe0 的 port3-0 05:04 对应 PCIe1 的 port1-0 其他位，保留

4.39 PRG 模块配置寄存器 0

本寄存器用来对 PRG 模块进行配置。

地址偏移：0x0640

默认值：0000_09d4_06c0_01cbh

表 4-39 PRG 配置访问寄存器

位域	名称	访问	描述
55:32	PRG_SSCSTEP	R/W	PLL SSC 展频步长控制
31:29	Reserved	R/W	保留
28	PRG_VCO_START	R/W	VCO 辅助启动控制位 1：开启 0：关闭
27	PRG_SSCSPRD	R/W	PLL SSC 展频模式控制 1：central spread 0：down spread
26	PRG_SIN2SQUARE_SEL	R/W	从 A/B 两种结构放大器选择一种，将 BUMP 输入的差分时钟转化为方波时钟 1：选择放大器 A 0：选择放大器 B
25	PRG_RSTN	R/W	PLL 复位控制 1：正常工作 0：复位 pll
24	SYSCLK_SEL	R/W	SYSCLK 选择 仅在 PRG_REFCLK_SEL 为 0 时生效 1：选择系统时钟为参考时钟 0：选择片外 BUMP 输入时钟为参考时钟
23	PRG_PU	R/W	PLL 上电控制 1：正常工作 0：power down
22	PRG_LDO_PDN	R/W	LDO Power Down 控制信号 0：Power Down
21	PRG_LDO_BY	R/W	LDO Bypass 控制 1：有效

位域	名称	访问	描述
20	PRG_DSMCLK_SEL	R/W	not used
19	PRG_CP_SEL	R/W	PLL charge pump 电流选择控制 1：IP from ISRC_MASTER 0：IP from ISRC_CP
18	PRG_BYP_REG	R/W	PLL 内 REG 模块 BYPASS 控制 1：bypass reg 0：正常工作
17	PRG_BYP	R/W	1：不经过 PLL 直接输出输入时钟 0：输入时钟经过 PLL 后输出
16	DRIVER_CLK_SEL	R/W	输出差分时钟展频控制信号 1：差分时钟无 SSC 功能 0：差分时钟有 SSC 功能
15:7	Reserved	R/W	保留
6	PRG_SSC_OUT_ENABLE	R/W	输出单端 SSC 时钟使能信号 1：使能
5	PRG_SSCEN	R/W	PLL 展频功能控制 1：使能 SSC 功能 0：禁用 SSC 功能
4	PRG_SIN2SQR_ENABLEB	R/W	Sin2sqare 差分放大器 B 使能 1：使能
3	PRG_SIN2SQR_ENABLEA	R/W	Sin2sqare 差分放大器 A 使能 1：使能
2	PRG_FRACEN	R/W	PLL 小数分频功能控制 1：小数分频 0：整数分频 该位必须设置为 0
1	PRG_DRIVER_OUT_ENABLE	R/W	输出差分时钟使能信号 1：使能
0	PRG_CKOUT_EN	R/W	PLL 输出分频器功能控制 1：正常工作 0：复位

4.40 PRG 模块配置寄存器 1

本寄存器用来对 PRG 模块进行配置。

地址偏移：0x0648

默认值：0103_3333_7730_1900h

表 4-40 PRG 模块配置寄存器 0

位域	名称	访问	描述
63:59	Reserved	R/W	保留

位域	名称	访问	描述
58:56	PRG_CKOUT_DIVN	R/W	PLL 输出分频器分频数设置 000：0 分频 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101/110/111：32 分频
55:32	PRG_SSCOFFSET	R/W	PLL SSC 展频偏置控制
31:28	PRG_SSCMA	R/W	PLL SSC 展频幅度
27:24	PRG_ICPSEL	R/W	PLL charge pump 电流选择
23:20	PRG_FVCO_TUNE_ABS	R/W	PLL 内部配置
19	Reserved	R/W	保留
18:16	PRG_FDBLY_SEL	R/W	PLL 内部 fbclk 延时设置
15:4	PRG_SSCSTPSUM	R/W	PLL SSC 展频三角波频率控制
3	PRG_PLL_LOCK	RO	PLL 锁定检测信号 1：PLL 完成锁定 0：PLL 未完成锁定
2:0	Reserved	R/W	保留

4.41 PRG 模块配置寄存器 2

本寄存器用来对 PRG 模块进行配置。

地址偏移：0x0650

默认值：35f0_0310_0320_0000h

表 4-41 PRG 模块配置寄存器 1

位域	名称	访问	描述
63	Reserved	R/W	保留
62:60	DRIVER_LDO_ACODE	R/W	LDO 输出电压配置
59	Reserved	R/W	保留
58	DRIVER_LDO_PDN	R/W	LDO Power Down 控制信号
57	DRIVER_LDO_BY	R/W	LDO BYPASS 控制信号
56	CLKIN_EN	R/W	Driver 输入时钟使能信号
55	DRIVER4_EN	R/W	Driver4 使能信号
54	DRIVER3_EN	R/W	Driver3 使能信号
53	DRIVER2_EN	R/W	Driver2 使能信号
52	DRIVER1_EN	R/W	Driver1 使能信号
51:43	Reserved	R/W	保留
42:40	PRG_LDO_ACODE	R/W	LDO 输出电压配置信号
39:36	PRG_CKIN_DIVN	R/W	PLL 输入分频器分频数设置
35	Reserved	R/W	保留
34:32	PRG_REFDLY_SEL	R/W	REFCLKIN 延时设置

位域	名称	访问	描述
31:30	Reserved	R/W	保留
29:20	PRG_DIV_M	R/W	PLL 整数部分分频数设置 (32-1023)
19:0	PRG_DIV_N	R/W	PLL 小数部分分频数设置

4.42 PAD 驱动配置寄存器

本寄存器用来对 PAD 的驱动能力进行配置。

地址偏移：0x06e0

默认值：0000_0006_0000_0048h

表 4-42 PAD 驱动配置寄存器

位域	名称	访问	描述
37:36	pad_ctrl_acpi	R/W	ACPI PAD 驱动能力控制
35:32	pad_ctrl_rgmii	R/W	RGMII PAD 驱动能力控制
31:30	pad_ctrl_cfg	R/W	CFG PAD 驱动能力控制
29:28	pad_ctrl_pcie	R/W	PCIe PAD 驱动能力控制
27:26	pad_ctrl_sata	R/W	SATA PAD 驱动能力控制
25:24	pad_ctrl_hda	R/W	HDA PAD 驱动能力控制
23:22	pad_ctrl_lio	R/W	LIO PAD 驱动能力控制
21:20	pad_ctrl_lpc	R/W	LPC PAD 驱动能力控制
19:18	pad_ctrl_spi	R/W	SPI PAD 驱动能力控制
17:16	pad_ctrl_jtag	R/W	JTAG PAD 驱动能力控制
15:14	pad_ctrl_uart	R/W	UART PAD 驱动能力控制
13:12	pad_ctrl_pwm	R/W	PWM PAD 驱动能力控制
11:10	pad_ctrl_i2c	R/W	I2C PAD 驱动能力控制
9:8	pad_ctrl_gpio	R/W	GPIO PAD 驱动能力控制
7:4	pad_ctrl_sdio	R/W	SDIO PAD 驱动能力控制
3:0	pad_ctrl_emmc	R/W	eMMC PAD 驱动能力控制

4.43 GRA 访存优先级配置寄存器

本寄存器用来对 GPU、DC 以及 VPU 的优先级进行配置。

地址偏移：0x0780

默认值：0011_1110h

表 4-43 GRA 访存优先级配置寄存器

位域	名称	访问	描述
23:20	prior_dc1	R/W	DC1 优先级配置，低 2 位控制读请求
19:16	prior_dc0	R/W	DC0 优先级配置，低 2 位控制读请求
15:12	prior_venc	R/W	VENC 优先级配置，低 2 位控制读请求，高 2 位控制写请求

位域	名称	访问	描述
11:8	prior_vdec	R/W	VDEC 优先级配置，低 2 位控制读请求，高 2 位控制写请求
7:4	prior_gpu	R/W	GPU 优先级配置，低 2 位控制读请求，高 2 位控制写请求
0	dc_prior_fix	R/W	DC 优先级固定配置使能 0-由硬件自动控制 DC 优先级 1-由 prior_dc0/1 控制 DC 优先级

4.44 DC 配置寄存器

本寄存器用来对 DC 控制器进行配置。

地址偏移：0x0788

默认值：0000_1212h

表 4-44 DC 配置寄存器

位域	名称	访问	描述
63	multi_axi_id_en	R/W	DC 多 ID 访问使能
19:16	prior_dc0	R/W	DC0 优先级配置，低 2 位控制读请求
15:12	prior_venc	R/W	VENC 优先级配置，低 2 位控制读请求，高 2 位控制写请求
34:16	dc_spram_ctrl	R/W	DC 内 SPRAM 控制位
15:8	dc1_prior_level	R/W	DC1 优先级水平控制 [7:4]-优先级 1 水位 [3:0]-优先级 2 水位
7:0	dc0_prior_level	R/W	DC0 优先级水平控制 [7:4]-优先级 1 水位 [3:0]-优先级 2 水位

4.45 USB3.0 控制器配置寄存器 0

地址偏移：0840h

默认值：0000_0000_0100_4400h

位域	名称	访问	描述
53:49	usb3_debug	RO	usb3_debug 的[4:0]位，具体含义需查阅相关资料
48:45	disU3rxdetect_ack	RO	disU3rxdetect 的确认信号，与端口数量对应
44	host_system_err	RO	系统报错，反映在 USBSTS.HSE 域
43:32	host_current_belt	RO	指示所有接收到的 BELT 的最小值
26	disU3rxdetect	R/W	这个信号用来停止控制器发送命令，且释放 pipe 总线

25	U3rxdetect	R/W	当信号被设置时，开始做 rx detect
24	port_power_control_present	R/W	指示端口是否有功耗切换 1:有功耗切换 0:没有功耗切换
23:20	host_u3_port_disable	R/W	USB3.0 端口关闭 1:端口关闭 0:端口使能
19:16	host_u2_port_disable	R/W	USB2.0 端口关闭 1:端口关闭 0:端口使能
15:12	host_num_u3_port	R/W	使能的 USB3.0 的端口数量
11:7	host_num_u2_port	R/W	使能的 USB2.0 的端口数量
7:0	hub_port_perm_attach	R/W	指示设备是否永久在连接状态，对应接口数量的高 4 位是 USB3.0 的端口，对应接口数量的低 4 位是 USB2.0 的端口 1:永久连接 0:非永久连接

4.46 USB2.0 配置寄存器 0

地址偏移：08b8h

默认值：03FC_00FF_0000_1040h

位域	名称	访问	描述
59	u3_pme_en		
58	u2_pme_en		
57:54	phy1_pll_enable	R/W	
53:50	phy0_pll_enable	R/W	
49:41	Resv	R/W	保留
40:32	Resv	R/W	保留
12	port_power_control_present	R/W	指示端口是否有功耗切换 1:有功耗切换 0:没有功耗切换
11:8	host_u2_port_disable	R/W	USB2.0 端口关闭 1:端口关闭 0:端口使能
7:4	host_num_u2_port	R/W	使能的 USB2.0 的端口数量
3:0	hub_port_perm_attach	R/W	指示设备是否永久在连接状态，对应接口数量是 USB2.0 的端口 1:永久连接 0:非永久连接

4.47 USB3.0 电源控制寄存器

地址偏移：08d8h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:62	Reserved	R/W	保留
61:32	pm_pmu_config_strap		
12	pmu_disable	R/W	禁用 usb 休眠唤醒功能 0：使能 1：禁用
10	vcc_soft_restore		
9:8	power_state_request	R/W	电源状态请求 00：U0 或 L0 模式 11：U3 或 L2 模式
7:6	power_state_u2pmu	RO	USB2 端口功耗状态
5	pme_gen_u2pmu	RO	USB2 产生唤醒事件
4	connect_state_u2pmu	RO	USB2 任一端口低功耗状态指示
3:2	power_state_u3pmu	RO	USB3 端口功耗状态
1	pme_gen_u3pmu	RO	USB3 产生唤醒事件
0	connect_state_u3pmu	RO	USB3 任一端口低功耗状态指示

4.48 USB2.0 电源控制寄存器

地址偏移：08e0h

默认值：0000_0000_0000_000ch

位域	名称	访问	描述
63:13	Reserved	R/W	保留
12	pmu_disable	R/W	禁用 usb 休眠唤醒功能 0：使能 1：禁用
9:8	power_state_request	R/W	电源状态请求 00：L0 模式 11：L2 模式
7:6	power_state_u2pmu	RO	USB2 端口功耗状态
5	pme_gen_u2pmu	RO	USB2 产生唤醒事件
4	connect_state_u2pmu	RO	USB2 任一端口低功耗状态指示
3:0	Reserved	RO	保留

4.49 GPU 窗口 0 基址寄存器

地址偏移：0900h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	gpu_win0_base	R/W	GPU 窗口 0 基址

4.50 GPU 窗口 0 大小寄存器

地址偏移：0908h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	gpu_win0_mask	R/W	GPU 窗口 0 大小。将高位设置为 0，低位连续 1 的个数代表有效地址宽度

4.51 GPU 窗口 0 重映射寄存器

地址偏移：0910h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:1	gpu_win0_mmap	R/W	GPU 窗口 0 重映射设置
0	gpu_win0_enable	R/W	窗口使能控制，高有效

4.52 GPU 窗口 1 基址寄存器

地址偏移：0920h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	gpu_win1_base	R/W	GPU 窗口 1 基址

4.53 GPU 窗口 1 大小寄存器

地址偏移：0928h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	gpu_win1_mak	R/W	GPU 窗口 1 大小。将高位设置为 0，低位连续 1 的个数代表有效地址宽度

4.54 GPU 窗口 1 重映射寄存器

地址偏移：0930h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:1	gpu_win1_mmap	R/W	GPU 窗口 1 重映射设置
0	gpu_win1_enable	R/W	窗口使能控制，高有效

4.55 VPU 窗口 0 基址寄存器

地址偏移：0940h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	vpu_win0_base	R/W	VPU 窗口 0 基址

4.56 VPU 窗口 0 大小寄存器

地址偏移：0948h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	vpu_win0_mask	R/W	VPU 窗口 0 大小。将高位设置为 0，低位连续 1 的个数代表有效地址宽度

4.57 VPU 窗口 0 重映射寄存器

地址偏移：0950h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:1	vpu_win0_mmap	R/W	VPU 窗口 0 重映射设置
0	vpu_win0_enable	R/W	窗口使能控制，高有效

4.58 VPU 窗口 1 基址寄存器

地址偏移：0960h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	vpu_win1_base	R/W	VPU 窗口 1 基址

4.59 VPU 窗口 1 大小寄存器

地址偏移：0968h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:0	vpu_win1_mak	R/W	VPU 窗口 1 大小。将高位设置为 0，低位连续 1 的个数代表有效地址宽度

4.60 VPU 窗口 1 重映射寄存器

地址偏移：0970h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:48	Reserved	R/W	保留
47:1	vpu_win1_mmap	R/W	VPU 窗口 1 重映射设置
0	vpu_win1_enable	R/W	窗口使能控制，高有效

5 地址空间分配

龙芯 2K3000 物理地址位宽为 40 位。

地址路由主要通过系统网络以及 IO 子网络组成。系统网络可以对每个 Master 端口接收到的请求进行路由配置；IO 子网络采用 PCI 总线的拓扑结构，通过软件扫描设备并配置寄存器基地址的方式来访问，因此每个设备都有一个标准的 PCI 配置头。

全芯片的地址空间划分如下表所示，除以下标记的地址空间和属性外，其他的地址和属性组合为非法访问。

表 5-1 芯片地址空间划分

起始地址	结束地址	大小	属性	说明
0x0	0x0FFF_FFFF	256MB	-	内存空间
0x1000_0000	0x17FF_FFFF	128MB	Uncache	PCI Memory 0
0x1800_0000	0x19FF_FFFF	32MB	Uncache	PCI IO
0x1A00_0000	0x1BFF_FFFF	32MB	Uncache	PCI CFG
0x1C00_0000	0x1CFF_FFFF	16MB	-	BOOT Flash
0x1D00_0000	0x1DFF_FFFF	16MB	-	LIO[a]
0x1FE0_0000	0x1FFF_FFFF	2MB	Uncache	配置空间
0x2000_0000	0x7FFF_FFFF	1.5GB	Uncache	PCI Memory 1
0x9000_0000	0xFFFF_FFFF	1.75GB	-	内存空间
0x1_0000_0000	0x3F_FFFF_FFFF	248GB	-	内存空间
0x40_0000_0000	0x7F_FFFF_FFFF	256GB	Uncache	PCI MEM
0xC0_0000_0000	0xCF_FFFF_FFFF	64GB	Uncache	SE 地址空间
0xFD_FC00_0000	0xFD_FDFF_FFFF	32MB	Uncache	PCI IO
0xFD_FE00_0000	0xFD_FFFF_FFFF	32MB	Uncache	PCI HEADER
0xFE_0000_0000	0xFE_00FF_FFFF	16MB	Uncache	PCI CFG TYPE0
0xFE_1000_0000	0xFE_10FF_FFFF	16MB	Uncache	PCI CFG TYPE1
0xFE_2000_0000	0xFE_2FFF_FFFF	256MB	Uncache	PCI CFG TYPE0
0xFE_3000_0000	0xFE_3FFF_FFFF	256MB	Uncache	PCI CFG TYPE1

注 a：当 LIO 使能时，该地址空间为 LIO 地址空间；当 LIO 未被使能时，该地址空间为 BOOT 地址空间。

5.1 SCACHE 地址散列

共享 Cache 的交叉寻址方式根据地址散列来确定具体访问目标，并可以通过软件在使用 Cache 之前配置修改。

系统中设置了名为 SCID_SEL 的配置寄存器来确定地址选择位，如下表所示。SCID_SEL 位于路由设置寄存器[3:0]。

表 5-2 SCID_SEL 地址位设置

SCID_SEL	地址位选择
4'b0000	全地址散列
4'b0001	根据[9:6]选择目标位置

5.2 地址映射配置

默认情况下，龙芯 2K3000 的地址路由已按表 5- 1 进行设置。但软件可以根据需要对每个主端口接收到的请求进行路由配置，每个主端口都拥有 8 个地址窗口，可以完成 8 个地址窗口的目标路由选择。

每个地址窗口由 BASE、MASK 和 MMAP 三个 64 位寄存器组成，BASE 以 1MB 对齐；MASK 采用类似网络掩码高位为 1 的格式；MMAP 的低四位表示对应目标设备端口的编号，MMAP[4]表示允许取指，MMAP[5]表示允许块读，MMAP[6]表示允许交错访问使能，MMAP[7]表示窗口使能。

在 2K3000 中，对于超过 4 个的相同模块，将 4 个为一组，作为一个内部结点。例如，8 个处理器核，以 4 个为一组，共分为 2 个内部结点；8 个 SCache，以 4 个为一组，共分为 2 个内部结点。具体的对应关系如表 5-5 所示。

在 2K3000 中，因为内部结点的引入，窗口配置时，需要增加内部结点号，放在 MMAP 寄存器的 bit[8]位上，同时将第[10]位供结点交错使能位使用。

这里要特别注意的是，2K3000 中地址窗口映射空间的最小单位为 1MB。

表 5-3 MMAP 寄存器位域说明

[63:40]	[39：20]	[19:11]	[10：4]	[3：0]
保留	转换后地址	保留	窗口使能配置	从设备号

其中“窗口使能配置”中的具体定义见下表。

表 5-4 MMAP 字段对应的空间访问属性

[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3:0]
内部结点交错使能	保留	内部结点号	窗口使能	允许对 Scache 进行交错访问，当从设备号为 0 时有效，按照对应的“交错选择位”配置进行路由。	允许块读	允许取指	设备号

对于设备的 ID 号，定义如下：

表 5-5 内部结点号和设备号

模块	设备号	内部结点号
Core	-	0 - 1
Scache	0 - 3	0 - 1
MC	4	0
SE	8	0
MISC	9	0
IO	E	0
GRA	F	0

其中，Core/Scache 都有 2 个对应的内部结点。其他模块只有内部结点 0。

对于共享 Cache 的访问，不同于地址窗口的映射关系，龙芯 2K3000 可以根据实际应用的访问行为，来决定共享Cache的交叉寻址方式。共享Cache模块所对应的地址空间根据地址位的特定位确定，并可以通过软件在使用 Cache 前动态配置修改。系统中设置了名为 SCID_SEL 的配置寄存器来确定地址选择位，如 5.1 节所述。

SCACHE 交错访问配置使能后，Slave 号仅为 0 有效。为 0 表示路由到 SCACHE，并由 SCID_SEL 决定如何在 4 个 SCACHE 进行交错访问。

每个结点的地址窗口转换寄存器如下表所示。除 Core/Scache 外，其他模块只在结点 0 上有效。

例如：内部结点 0 寄存器基地址为 0x1FE0_0000；内部结点 1 寄存器基地址为 0x1FE1_0000。

表 5-6 地址窗口寄存器表

地址偏移	寄存器	地址偏移	寄存器
0x2000	CORE0_WIN0_BASE	0x2100	CORE1_WIN0_BASE
0x2008	CORE0_WIN1_BASE	0x2108	CORE1_WIN1_BASE
0x2010	CORE0_WIN2_BASE	0x2110	CORE1_WIN2_BASE
0x2018	CORE0_WIN3_BASE	0x2118	CORE1_WIN3_BASE
0x2020	CORE0_WIN4_BASE	0x2120	CORE1_WIN4_BASE
0x2028	CORE0_WIN5_BASE	0x2128	CORE1_WIN5_BASE
0x2030	CORE0_WIN6_BASE	0x2130	CORE1_WIN6_BASE
0x2038	CORE0_WIN7_BASE	0x2138	CORE1_WIN7_BASE
0x2040	CORE0_WIN0_MASK	0x2140	CORE1_WIN0_MASK
0x2048	CORE0_WIN1_MASK	0x2148	CORE1_WIN1_MASK
0x2050	CORE0_WIN2_MASK	0x2150	CORE1_WIN2_MASK
0x2058	CORE0_WIN3_MASK	0x2158	CORE1_WIN3_MASK
0x2060	CORE0_WIN4_MASK	0x2160	CORE1_WIN4_MASK
0x2068	CORE0_WIN5_MASK	0x2168	CORE1_WIN5_MASK
0x2070	CORE0_WIN6_MASK	0x2170	CORE1_WIN6_MASK
0x2078	CORE0_WIN7_MASK	0x2178	CORE1_WIN7_MASK
0x2080	CORE0_WIN0_MMAP	0x2180	CORE1_WIN0_MMAP
0x2088	CORE0_WIN1_MMAP	0x2188	CORE1_WIN1_MMAP
0x2090	CORE0_WIN2_MMAP	0x2190	CORE1_WIN2_MMAP
0x2098	CORE0_WIN3_MMAP	0x2198	CORE1_WIN3_MMAP
0x20a0	CORE0_WIN4_MMAP	0x21a0	CORE1_WIN4_MMAP
0x20a8	CORE0_WIN5_MMAP	0x21a8	CORE1_WIN5_MMAP
0x20b0	CORE0_WIN6_MMAP	0x21b0	CORE1_WIN6_MMAP
0x20b8	CORE0_WIN7_MMAP	0x21b8	CORE1_WIN7_MMAP
0x2200	CORE2_WIN0_BASE	0x2300	CORE3_WIN0_BASE
0x2208	CORE2_WIN1_BASE	0x2308	CORE3_WIN1_BASE

0x2210	CORE2_WIN2_BASE	0x2310	CORE3_WIN2_BASE
0x2218	CORE2_WIN3_BASE	0x2318	CORE3_WIN3_BASE
0x2220	CORE2_WIN4_BASE	0x2320	CORE3_WIN4_BASE
0x2228	CORE2_WIN5_BASE	0x2328	CORE3_WIN5_BASE
0x2230	CORE2_WIN6_BASE	0x2330	CORE3_WIN6_BASE
0x2238	CORE2_WIN7_BASE	0x2338	CORE3_WIN7_BASE
0x2240	CORE2_WIN0_MASK	0x2340	CORE3_WIN0_MASK
0x2248	CORE2_WIN1_MASK	0x2348	CORE3_WIN1_MASK
0x2250	CORE2_WIN2_MASK	0x2350	CORE3_WIN2_MASK
0x2258	CORE2_WIN3_MASK	0x2358	CORE3_WIN3_MASK
0x2260	CORE2_WIN4_MASK	0x2360	CORE3_WIN4_MASK
0x2268	CORE2_WIN5_MASK	0x2368	CORE3_WIN5_MASK
0x2270	CORE2_WIN6_MASK	0x2370	CORE3_WIN6_MASK
0x2278	CORE2_WIN7_MASK	0x2378	CORE3_WIN7_MASK
0x2280	CORE2_WIN0_MMAP	0x2380	CORE3_WIN0_MMAP
0x2288	CORE2_WIN1_MMAP	0x2388	CORE3_WIN1_MMAP
0x2290	CORE2_WIN2_MMAP	0x2390	CORE3_WIN2_MMAP
0x2298	CORE2_WIN3_MMAP	0x2398	CORE3_WIN3_MMAP
0x22a0	CORE2_WIN4_MMAP	0x23a0	CORE3_WIN4_MMAP
0x22a8	CORE2_WIN5_MMAP	0x23a8	CORE3_WIN5_MMAP
0x22b0	CORE2_WIN6_MMAP	0x23b0	CORE3_WIN6_MMAP
0x22b8	CORE2_WIN7_MMAP	0x23b8	CORE3_WIN7_MMAP
0x2400	SCACHE0_WIN0_BASE	0x2500	SCACHE1_WIN0_BASE
0x2408	SCACHE0_WIN1_BASE	0x2508	SCACHE1_WIN1_BASE
0x2410	SCACHE0_WIN2_BASE	0x2510	SCACHE1_WIN2_BASE
0x2418	SCACHE0_WIN3_BASE	0x2518	SCACHE1_WIN3_BASE
0x2420	SCACHE0_WIN4_BASE	0x2520	SCACHE1_WIN4_BASE
0x2428	SCACHE0_WIN5_BASE	0x2528	SCACHE1_WIN5_BASE
0x2430	SCACHE0_WIN6_BASE	0x2530	SCACHE1_WIN6_BASE
0x2438	SCACHE0_WIN7_BASE	0x2538	SCACHE1_WIN7_BASE
0x2440	SCACHE0_WIN0_MASK	0x2540	SCACHE1_WIN0_MASK
0x2448	SCACHE0_WIN1_MASK	0x2548	SCACHE1_WIN1_MASK
0x2450	SCACHE0_WIN2_MASK	0x2550	SCACHE1_WIN2_MASK
0x2458	SCACHE0_WIN3_MASK	0x2558	SCACHE1_WIN3_MASK
0x2460	SCACHE0_WIN4_MASK	0x2560	SCACHE1_WIN4_MASK
0x2468	SCACHE0_WIN5_MASK	0x2568	SCACHE1_WIN5_MASK
0x2470	SCACHE0_WIN6_MASK	0x2570	SCACHE1_WIN6_MASK
0x2478	SCACHE0_WIN7_MASK	0x2578	SCACHE1_WIN7_MASK
0x2480	SCACHE0_WIN0_MMAP	0x2580	SCACHE1_WIN0_MMAP
0x2488	SCACHE0_WIN1_MMAP	0x2588	SCACHE1_WIN1_MMAP
0x2490	SCACHE0_WIN2_MMAP	0x2590	SCACHE1_WIN2_MMAP

0x2498	SCACHE0_WIN3_MMAP	0x2598	SCACHE1_WIN3_MMAP
0x24a0	SCACHE0_WIN4_MMAP	0x25a0	SCACHE1_WIN4_MMAP
0x24a8	SCACHE0_WIN5_MMAP	0x25a8	SCACHE1_WIN5_MMAP
0x24b0	SCACHE0_WIN6_MMAP	0x25b0	SCACHE1_WIN6_MMAP
0x24b8	SCACHE0_WIN7_MMAP	0x25b8	SCACHE1_WIN7_MMAP
0x2600	SCACHE2_WIN0_BASE	0x2700	SCACHE3_WIN0_BASE
0x2608	SCACHE2_WIN1_BASE	0x2708	SCACHE3_WIN1_BASE
0x2610	SCACHE2_WIN2_BASE	0x2710	SCACHE3_WIN2_BASE
0x2618	SCACHE2_WIN3_BASE	0x2718	SCACHE3_WIN3_BASE
0x2620	SCACHE2_WIN4_BASE	0x2720	SCACHE3_WIN4_BASE
0x2628	SCACHE2_WIN5_BASE	0x2728	SCACHE3_WIN5_BASE
0x2630	SCACHE2_WIN6_BASE	0x2730	SCACHE3_WIN6_BASE
0x2638	SCACHE2_WIN7_BASE	0x2738	SCACHE3_WIN7_BASE
0x2640	SCACHE2_WIN0_MASK	0x2740	SCACHE3_WIN0_MASK
0x2648	SCACHE2_WIN1_MASK	0x2748	SCACHE3_WIN1_MASK
0x2650	SCACHE2_WIN2_MASK	0x2750	SCACHE3_WIN2_MASK
0x2658	SCACHE2_WIN3_MASK	0x2758	SCACHE3_WIN3_MASK
0x2660	SCACHE2_WIN4_MASK	0x2760	SCACHE3_WIN4_MASK
0x2668	SCACHE2_WIN5_MASK	0x2768	SCACHE3_WIN5_MASK
0x2670	SCACHE2_WIN6_MASK	0x2770	SCACHE3_WIN6_MASK
0x2678	SCACHE2_WIN7_MASK	0x2778	SCACHE3_WIN7_MASK
0x2680	SCACHE2_WIN0_MMAP	0x2780	SCACHE3_WIN0_MMAP
0x2688	SCACHE2_WIN1_MMAP	0x2788	SCACHE3_WIN1_MMAP
0x2690	SCACHE2_WIN2_MMAP	0x2790	SCACHE3_WIN2_MMAP
0x2698	SCACHE2_WIN3_MMAP	0x2798	SCACHE3_WIN3_MMAP
0x26a0	SCACHE2_WIN4_MMAP	0x27a0	SCACHE3_WIN4_MMAP
0x26a8	SCACHE2_WIN5_MMAP	0x27a8	SCACHE3_WIN5_MMAP
0x26b0	SCACHE2_WIN6_MMAP	0x27b0	SCACHE3_WIN6_MMAP
0x26b8	SCACHE2_WIN7_MMAP	0x27b8	SCACHE3_WIN7_MMAP
0x2800	SE_WIN0_BASE	0x2900	MISC_WIN0_BASE
0x2808	SE_WIN1_BASE	0x2908	MISC_WIN1_BASE
0x2810	SE_WIN2_BASE	0x2910	MISC_WIN2_BASE
0x2818	SE_WIN3_BASE	0x2918	MISC_WIN3_BASE
0x2820	SE_WIN4_BASE	0x2920	MISC_WIN4_BASE
0x2828	SE_WIN5_BASE	0x2928	MISC_WIN5_BASE
0x2830	SE_WIN6_BASE	0x2930	MISC_WIN6_BASE
0x2838	SE_WIN7_BASE	0x2938	MISC_WIN7_BASE
0x2840	SE_WIN0_MASK	0x2940	MISC_WIN0_MASK
0x2848	SE_WIN1_MASK	0x2948	MISC_WIN1_MASK
0x2850	SE_WIN2_MASK	0x2950	MISC_WIN2_MASK

0x2858	SE_WIN3_MASK	0x2958	MISC_WIN3_MASK
0x2860	SE_WIN4_MASK	0x2960	MISC_WIN4_MASK
0x2868	SE_WIN5_MASK	0x2968	MISC_WIN5_MASK
0x2870	SE_WIN6_MASK	0x2970	MISC_WIN6_MASK
0x2878	SE_WIN7_MASK	0x2978	MISC_WIN7_MASK
0x2880	SE_WIN0_MMAP	0x2980	MISC_WIN0_MMAP
0x2888	SE_WIN1_MMAP	0x2988	MISC_WIN1_MMAP
0x2890	SE_WIN2_MMAP	0x2990	MISC_WIN2_MMAP
0x2898	SE_WIN3_MMAP	0x2998	MISC_WIN3_MMAP
0x28a0	SE_WIN4_MMAP	0x29a0	MISC_WIN4_MMAP
0x28a8	SE_WIN5_MMAP	0x29a8	MISC_WIN5_MMAP
0x28b0	SE_WIN6_MMAP	0x29b0	MISC_WIN6_MMAP
0x28b8	SE_WIN7_MMAP	0x29b8	MISC_WIN7_MMAP
0x2c00		0x2d00	GRA_UC_WIN0_BASE
0x2c08		0x2d08	GRA_UC_WIN1_BASE
0x2c10		0x2d10	GRA_UC_WIN2_BASE
0x2c18		0x2d18	GRA_UC_WIN3_BASE
0x2c20		0x2d20	GRA_UC_WIN4_BASE
0x2c28		0x2d28	GRA_UC_WIN5_BASE
0x2c30		0x2d30	GRA_UC_WIN6_BASE
0x2c38		0x2d38	GRA_UC_WIN7_BASE
0x2c40		0x2d40	GRA_UC_WIN0_MASK
0x2c48		0x2d48	GRA_UC_WIN1_MASK
0x2c50		0x2d50	GRA_UC_WIN2_MASK
0x2c58		0x2d58	GRA_UC_WIN3_MASK
0x2c60		0x2d60	GRA_UC_WIN4_MASK
0x2c68		0x2d68	GRA_UC_WIN5_MASK
0x2c70		0x2d70	GRA_UC_WIN6_MASK
0x2c78		0x2d78	GRA_UC_WIN7_MASK
0x2c80		0x2d80	GRA_UC_WIN0_MMAP
0x2c88		0x2d88	GRA_UC_WIN1_MMAP
0x2c90		0x2d90	GRA_UC_WIN2_MMAP
0x2c98		0x2d98	GRA_UC_WIN3_MMAP
0x2ca0		0x2da0	GRA_UC_WIN4_MMAP
0x2ca8		0x2da8	GRA_UC_WIN5_MMAP
0x2cb0		0x2db0	GRA_UC_WIN6_MMAP
0x2cb8		0x2db8	GRA_UC_WIN7_MMAP
0x2e00	IO_WIN0_BASE	0x2f00	GRA_C_WIN0_BASE
0x2e08	IO_WIN1_BASE	0x2f08	GRA_C_WIN1_BASE
0x2e10	IO_WIN2_BASE	0x2f10	GRA_C_WIN2_BASE

0x2e18	IO_WIN3_BASE	0x2f18	GRA_C_WIN3_BASE
0x2e20	IO_WIN4_BASE	0x2f20	GRA_C_WIN4_BASE
0x2e28	IO_WIN5_BASE	0x2f28	GRA_C_WIN5_BASE
0x2e30	IO_WIN6_BASE	0x2f30	GRA_C_WIN6_BASE
0x2e38	IO_WIN7_BASE	0x2f38	GRA_C_WIN7_BASE
0x2e40	IO_WIN0_MASK	0x2f40	GRA_C_WIN0_MASK
0x2e48	IO_WIN1_MASK	0x2f48	GRA_C_WIN1_MASK
0x2e50	IO_WIN2_MASK	0x2f50	GRA_C_WIN2_MASK
0x2e58	IO_WIN3_MASK	0x2f58	GRA_C_WIN3_MASK
0x2e60	IO_WIN4_MASK	0x2f60	GRA_C_WIN4_MASK
0x2e68	IO_WIN5_MASK	0x2f68	GRA_C_WIN5_MASK
0x2e70	IO_WIN6_MASK	0x2f70	GRA_C_WIN6_MASK
0x2e78	IO_WIN7_MASK	0x2f78	GRA_C_WIN7_MASK
0x2e80	IO_WIN0_MMAP	0x2f80	GRA_C_WIN0_MMAP
0x2e88	IO_WIN1_MMAP	0x2f88	GRA_C_WIN1_MMAP
0x2e90	IO_WIN2_MMAP	0x2f90	GRA_C_WIN2_MMAP
0x2e98	IO_WIN3_MMAP	0x2f98	GRA_C_WIN3_MMAP
0x2ea0	IO_WIN4_MMAP	0x2fa0	GRA_C_WIN4_MMAP
0x2ea8	IO_WIN5_MMAP	0x2fa8	GRA_C_WIN5_MMAP
0x2eb0	IO_WIN6_MMAP	0x2fb0	GRA_C_WIN6_MMAP
0x2eb8	IO_WIN7_MMAP	0x2fb8	GRA_C_WIN7_MMAP

需要注意的是，窗口配置不能对 Cache 一致性的请求进行地址转换，否则在 SCache 处的地址会与处理器一级 Cache 处的地址不一致，导致 Cache 一致性的维护错误。

窗口命中公式:　　(IN_ADDR & MASK)　== BASE

新地址换算公式:　OUT_ADDR = (IN_ADDR & ~MASK) | {MMAP[63:20],20'h0}

根据缺省的路由，芯片启动后，能够自己去对应的 SPI Flash 取指，并能正确访问芯片上的 IO 设备地址。当需要访问其它地址空间时，软件需要通过修改相应的配置寄存器实现新的地址空间路由和转换。

此外，当出现由于 CPU 猜测执行引起对非法地址的读访问时，8 个地址窗口都不命中，将返回随机数据，以防止 CPU 死等。

5.3　IO 地址空间

龙芯 2K3000 的 IO 地址空间与 PCI 定义的地址空间形式相同，主要分为三类：

1. 配置空间：该地址空间用于访问各个 IO 设备的 PCI 配置头，其地址组成符合 PCI 配置访问的地址组织形式；
2. IO 空间：该地址空间用于访问 PCI 协议定义的 IO 地址空间。在 2K3000 中只有 PCIe 有这段地址空间，用于通过 IO 类型的请求访问 PCIe 控制器的下游设备。

3. MEM 空间：除了以上两种地址空间之外的所有地址空间为 MEM 空间。

5.3.1 PCI 设备的配置空间

这里需要先区分配置头空间和设备寄存器空间两个概念。配置头空间指的是每个 IO 设备的配置头所在的地址空间，而设备寄存器空间指的是配置头中的 BAR 所指定的设备配置寄存器的地址空间。访问 I/O 设备的设备寄存器空间需要先读取其配置头空间中的 BAR 地址作为其配置空间的起始地址，然后再对相应设备寄存器进行读写。

首先介绍 64 位模式下访问配置头空间的地址格式，龙芯 2K3000 中定义地址段 0xFE_0000_0000 - 0xFE_1FFF_FFFF 是 CPU 的 64 位配置地址空间，CPU 通过这段地址访问各个设备的 PCI 配置头。由地址的[63:28]决定配置头类型（0xFE0 是 Type0，0xFE1 是 Type1，其他保留）；[23:16]在 Type1 类型配置头访问时表示总线号（Bus Number），Type0 时保留；[15:11]表示设备号（Device Number）；[10:8]表示功能号（Function Number）；[27:24]和[7:0]组合起来表示偏移（offset），大小为 4K。下图是 CPU 访问 PCI 配置头空间的 64 位地址格式。

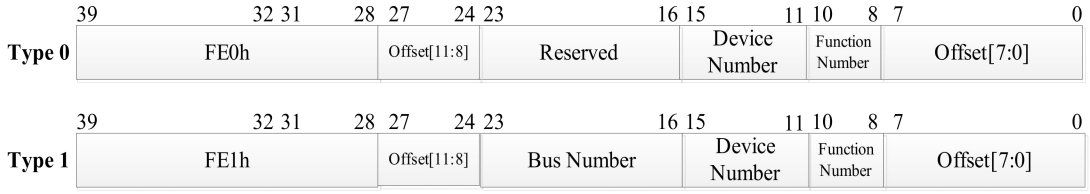


图 5-1 64 位配置访问地址格式

为了保证系统的兼容性，还提供了 32 位地址模式下的配置地址格式，如下图所示。该模式下，地址偏移只有 8 位（即 256B）。其中，地址的[31:24]决定配置头类型（0x1A 是 Type0，0x1B 是 Type1）；[23:16]在 Type1 类型配置头访问时表示总线号（Bus Number），Type0 时保留；[15:11]表示设备号（Device Number）；[10:8]表示功能号（Function Number）；[7:0]表示偏移（offset），大小为 256B。

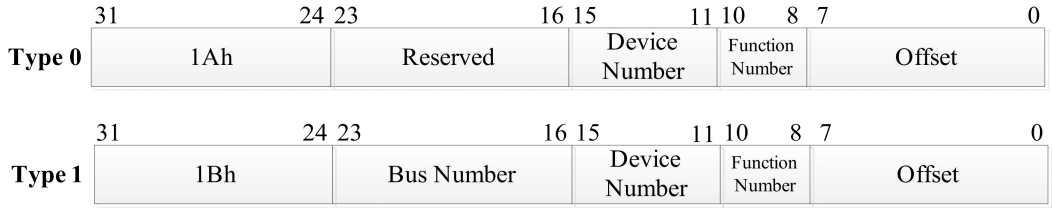


图 5-2 32 位配置访问地址格式

5.3.2 PCI 设备和功能

龙芯 2K3000 芯片中，具备配置头空间的设备如表 5- 7 所示。其中所有 IO 低速设备作为一个整体进行管理，由地址的[19:16]位进行译码，第 14 章将会做详细说明。各个设备的设备号、功能号以及地址空间如下表所示（该表只包括各个控制器本身的配置空间，不关

注 PCIe 控制器的 Type1 访问等）。除了 PCIe 和 GMAC 控制器之外，其他设备的配置头空间大小都只有 256B。

访问类型中，B 表示字节访问（1byte），H 表示半字访问（2byte），W 表示字访问（4byte），D 表示双字访问（8byte），Q 表示 4 字访问（16byte），C 表示 cacheline 访问。

表 5-7 各个设备的配置头访问对应关系

Device	Device Number、Function Number	Vendor ID	Device ID	Size	Access type（BAR 空间）
APB	2, 0	0x0014	0x7A42	512K	参考第 14 章
GMAC0	3, 0	0x0014	0x7A23	32K	W
GMAC1	3, 1	0x0014	0x7A23	32K	W
USB2	4,0	0x0014	0x7A44	1M	W
GPU	6, 0	0x0014	0x7A35	256 (REG) 256M（GMEM,可软件配置）	W (REG) BHWDQC (GMEM)
DC	6, 1	0x0014	0x7A46	64K	W
HDA	6, 2	0x0014	0x7A37	64K	BHW
VPU-DEC	6, 3	0x0014	0x7A56	16K	W
VPU-ENC	6, 4	0x0014	0x7A66	16K	W
HDA	7, 0	0x0014	0x7A07	64K	W
I2S	7, 0（二选一）	0x0014	0x7A27	64K	W
PCM	7,1	0x0014	0x7A47	64K	W
SATA	8, 0	0x0014	0x7A18	1K	W
PCIe0-Port0	9, 0	0x0014	0x7A99	8K	BHW
PCIe0-Port1	10, 0	0x0014	0x7A89	8K	BHW
PCIe0-Port2	11, 0	0x0014	0x7A89	8K	BHW
PCIe0-Port3	12, 0	0x0014	0x7A89	8K	BHW
PCIe1-Port0	13, 0	0x0014	0x7A99	8K	BHW
PCIe1-Port1	14, 0	0x0014	0x7A89	8K	BHW
SPI1	22, 0	0x0014	0x7a1b	4K (REG) 16M (MEM)	B (REG) BHWDQ (MEM)
RapidIO0	24, 0	0x0014	0x7a1d	8K (REG) 1G (MEM)	W
RapidIO1	27, 0	0x0014	0x7a1d	8K (REG) 1G (MEM)	W
USB3	25,0	0x0014	0x7a34	1Ms	W
eMMC	28, 0	0x0014	0x7A88	4K	W
SDIO0	28, 1	0x0014	0x7A48	4K	W
SDIO1	28,2	0x0014	0x7A48	4K	W

上述设备中，除了 PCIe IO/MEM、RapidIO MEM 和 Graphic Memory 外，其它设备的地址空间大小是固定不变的，软件可以改变地址空间的起始地址。PCIe IO/MEM 地址空间的大小需要根据所接设备来决定。

Graphic Memory 的大小根据所接内存的容量来决定。BIOS 需要通过访问桥片通用配置寄存器来修改 GPU 配置头的 BAR 寄存器 2/3 的 MASK 值来配置 Graphic Memory 的大小，然后软件再通过 PCI 扫描的方式来获得显存的大小。

在使用 PCIe 外接独立显卡的情况下，独立显卡自带的独立显存空间位于 PCIe MEM 地址空间内，作为 PCIe 设备统一管理。

除 PCIe 外，所有的配置头空间都遵循 PCI Type0 类型的格式，如下表所示，但是每个设备的缺省值不尽相同，后文中将展开介绍。

5.3.3 设备地址空间分配示例

对 PCI 设备的访问主要通过 PCI MEM 空间来完成。软件可以在该地址段内任意分配各个设备的访问地址。这些设备（除 LPC 外）的访问地址可以由软件动态分配。一种分配方式如下：通过扫描 PCI 总线，读取各个设备（PCI 配置访问）的配置空间，获取各个设备使用的 MEM 空间和 I/O 空间大小，系统软件从 0x40000,0000~0x7fff,ffff 这个地址内分配合适大小的 MEM 空间，从 0x1800,0000~0x19ff,ffff 这个地址内分配合适大小的 I/O 空间（PCIe 设备）。

除了这些 PCI 类型的设备外，还包含一些使用固定地址访问的设备，比如：中断控制器、HPET 控制器、confbus 配置寄存器、IO 低速设备块。

表 5- 12 给出了固定地址设备和 PCI 设备的一种地址分配示例，以及它们的地址空间大小和支持的访问类型。访问类型中，B 表示字节访问（1byte），H 表示半字访问（2byte），W 表示字访问（4byte），D 表示双字访问（8byte），Q 表示 4 字访问（16byte），C 表示 cacheline 访问。

表 5-8 固定地址设备地址空间

模块	地址空间	地址空间大小	访问类型
INT	0x1000,0000~0x1000,0fff	4K	BHW
HPET	0x1000,1000~0x1000,1fff	4K	BW
CONF REG	0x1001,0000~0x1001,ffff	64K	BHW
MISC	0x1008,0000~0x100f,ffff	512K	BW
LPC REG	0x1000,2000~0x1000,2fff	4K	W
LPC MEM	0x1200,0000~0x13ff,ffff	32M	BHWDQC
LPC I/O	0x1800,0000~0x1800,ffff	64K	B
LPC TPM	0x1801,0000~0x1801,ffff	64K	B

6 共享 Cache （SCache）

SCache 模块是龙芯 2K3000 处理器内部所有处理器核所共享的二级 Cache。SCache 模块的主要特征包括：

- 16 项 Cache 访问队列。
- 关键字优先。
- 通过目录支持 Cache 一致性协议。
- 采用 12 路组相联结构。
- 支持 ECC 校验。
- 支持 DMA 一致性读写和预取读。
- 支持多种共享 Cache 散列方式。
- 支持按窗口锁共享 Cache。
- 保证读数据返回原子性。

共享 Cache 模块包括共享 Cache 管理模块 scachemanage 及共享 Cache 访问模块 scacheaccess。Scachemanage 模块负责处理器来自处理器和 DMA 的访问请求，而共享 Cache 的 TAG、目录和数据等信息存放在 scacheaccess 模块中。为降低功耗，共享 Cache 的 TAG、目录和数据可以分开访问，共享 Cache 状态位、w 位与 TAG 一起存储，TAG 存放在 TAG RAM 中，目录存放在 DIR RAM 中，数据存放在 DATA RAM 中。失效请求访问共享 Cache，同时读出所有路的 TAG、目录，并根据 TAG 来选出目录，并根据命中情况读取数据。替换请求、重填请求和写回请求只操作一路的 TAG、目录和数据。

为提高一些特定计算任务的性能，共享 Cache 增加了锁机制。落在被锁区域中的共享 Cache 块会被锁住，因而不会被替换出共享 Cache。通过芯片配置寄存器空间可以对共享 Cache 模块内部的四组锁窗口寄存器进行动态配置，锁 Cache 时连续区域大小需小于 5.5MB。

配置窗口基地址为 0x1fe0_0000，或者通过 IOCSR 指令访问。

表 6-1 共享 Cache 锁窗口寄存器配置

名称	地址	位域	描述
Slock0_valid	0x3ff00200	[63:63]	0 号锁窗口有效位
Slock0_addr	0x3ff00200	[39:0]	0 号锁窗口锁地址
Slock0_mask	0x3ff00240	[39:0]	0 号锁窗口掩码
Slock1_valid	0x3ff00208	[63:63]	1 号锁窗口有效位
Slock1_addr	0x3ff00208	[39:0]	1 号锁窗口锁地址
Slock1_mask	0x3ff00248	[39:0]	1 号锁窗口掩码
Slock2_valid	0x3ff00210	[63:63]	2 号锁窗口有效位
Slock2_addr	0x3ff00210	[39:0]	2 号锁窗口锁地址

Slock2_mask	0x3ff00250	[39:0]	2 号锁窗口掩码
Slock3_valid	0x3ff00218	[63:63]	3 号锁窗口有效位
Slock3_addr	0x3ff00218	[39:0]	3 号锁窗口锁地址
Slock3_mask	0x3ff00258	[39:0]	3 号锁窗口掩码

举例来说，当一个地址 addr 使得 slock0_valid && ((addr & slock0_mask) == (slock0_addr & slock0_mask)) 为 1 时，这个地址就被锁窗口 0 锁住了。

7 处理器核间中断与通信

龙芯 2K3000 为每个处理器核都实现了 8 个核间中断寄存器（IPI）以支持多核 BIOS 启动和操作系统运行时在处理器核之间进行中断和通信。

龙芯 2K3000 中支持两种不同的访问方式，一种是地址访问模式，另一种是 IOCSR 指令访问模式。推荐使用处理器 IOCSR 指令访问模式。

7.1 按地址访问模式

对于龙芯 2K3000，下列寄存器可以使用基地址 0x1FE0_0000 进行访问。

由于龙芯 2K3000 中包括 8 个物理核，被划分为 2 个内部结点。

表 7-1 处理器核间中断相关的寄存器及其功能描述

名称	读写权限	描述
IPI_Status	R	32 位状态寄存器，任何一位有被置 1 且对应位使能情况下，处理器核 INT4 中断线被置位。
IPI_Enable	RW	32 位使能寄存器，控制对应中断位是否有效
IPI_Set	W	32 位置位寄存器，往对应的位写 1，则对应的 STATUS 寄存器位被置 1
IPI_Clear	W	32 位清除寄存器，往对应的位写 1，则对应的 STATUS 寄存器位被清 0
MailBox0	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。
MailBox01	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。
MailBox02	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。
MailBox03	RW	缓存寄存器，供启动时传递参数使用，按 64 或者 32 位的 uncache 方式进行访问。

与处理器核间中断相关的寄存器及其功能描述如下。

表 7-2 0 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core0_IPI_Status	0x1000	R	0 号处理器核的 IPI_Status 寄存器
Core0_IPI_Enalbe	0x1004	RW	0 号处理器核的 IPI_Enalbe 寄存器
Core0_IPI_Set	0x1008	W	0 号处理器核的 IPI_Set 寄存器
Core0_IPI_Clear	0x100c	W	0 号处理器核的 IPI_Clear 寄存器
Core0_MailBox0	0x1020	RW	0 号处理器核的 IPI_MailBox0 寄存器
Core0_MailBox1	0x1028	RW	0 号处理器核的 IPI_MailBox1 寄存器
Core0_MailBox2	0x1030	RW	0 号处理器核的 IPI_MailBox2 寄存器
Core0_MailBox3	0x1038	RW	0 号处理器核的 IPI_MailBox3 寄存器

表 7-3 1 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core1_IPI_Status	0x1100	R	1 号处理器核的 IPI_Status 寄存器
Core1_IPI_Enalbe	0x1104	RW	1 号处理器核的 IPI_Enalbe 寄存器
Core1_IPI_Set	0x1108	W	1 号处理器核的 IPI_Set 寄存器
Core1_IPI_Clear	0x110c	W	1 号处理器核的 IPI_Clear 寄存器
Core1_MailBox0	0x1120	R	1 号处理器核的 IPI_MailBox0 寄存器
Core1_MailBox1	0x1128	RW	1 号处理器核的 IPI_MailBox1 寄存器
Core1_MailBox2	0x1130	W	1 号处理器核的 IPI_MailBox2 寄存器
Core1_MailBox3	0x1138	W	1 号处理器核的 IPI_MailBox3 寄存器

表 7-4 2 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core2_IPI_Status	0x1200	R	2 号处理器核的 IPI_Status 寄存器
Core2_IPI_Enalbe	0x1204	RW	2 号处理器核的 IPI_Enalbe 寄存器
Core2_IPI_Set	0x1208	W	2 号处理器核的 IPI_Set 寄存器
Core2_IPI_Clear	0x120c	W	2 号处理器核的 IPI_Clear 寄存器
Core2_MailBox0	0x1220	R	2 号处理器核的 IPI_MailBox0 寄存器
Core2_MailBox1	0x1228	RW	2 号处理器核的 IPI_MailBox1 寄存器
Core2_MailBox2	0x1230	W	2 号处理器核的 IPI_MailBox2 寄存器
Core2_MailBox3	0x1238	W	2 号处理器核的 IPI_MailBox3 寄存器

表 7-5 3 号处理器核的核间中断与通信寄存器列表

名称	偏移地址	权限	描述
Core3_IPI_Status	0x1300	R	3 号处理器核的 IPI_Status 寄存器
Core3_IPI_Enalbe	0x1304	RW	3 号处理器核的 IPI_Enalbe 寄存器
Core3_IPI_Set	0x1308	W	3 号处理器核的 IPI_Set 寄存器
Core3_IPI_Clear	0x130c	W	3 号处理器核的 IPI_Clear 寄存器
Core3_MailBox0	0x1320	R	3 号处理器核的 IPI_MailBox0 寄存器
Core3_MailBox1	0x1328	RW	3 号处理器核的 IPI_MailBox1 寄存器
Core3_MailBox2	0x1330	W	3 号处理器核的 IPI_MailBox2 寄存器
Core3_MailBox3	0x1338	W	3 号处理器核的 IPI_MailBox3 寄存器

上面列出的是龙芯 2K3000 芯片所组成的多处理器系统逻辑核 0 至逻辑核 3 的核间中断相关寄存器列表。由于每个龙芯 2K3000 中包括 8 个物理核，被划分为 2 个结点。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]上加上内部结点号。例如，核 4 至核 7 的基地址为 0x1FE10000，具体的寄存器对应于该基址上核 0 至核 3 的偏移。

不同内部结点如下表所示：

表 7-6 不同内部结点的基地址

内部结点号	基地址
0	0x1FE0_0000
1	0x1FE1_0000

7.2 配置寄存器指令访问

使用处理器核 IOCSR 指令，可以通过私有空间对配置寄存器进行访问。

表 7-7 当前处理器核核间中断与通信寄存器列表

名称	偏移地址	权限	描述
perCore_IPI_Status	0x1000	R	当前处理器核的 IPI_Status 寄存器
perCore_IPI_Enalbe	0x1004	RW	当前处理器核的 IPI_Enalbe 寄存器
perCore_IPI_Set	0x1008	W	当前处理器核的 IPI_Set 寄存器
perCore_IPI_Clear	0x100c	W	当前处理器核的 IPI_Clear 寄存器
perCore_MailBox0	0x1020	RW	当前处理器核的 IPI_MailBox0 寄存器
perCore_MailBox1	0x1028	RW	当前处理器核的 IPI_MailBox1 寄存器
perCore_MailBox2	0x1030	RW	当前处理器核的 IPI_MailBox2 寄存器
perCore_MailBox3	0x1038	RW	当前处理器核的 IPI_MailBox3 寄存器

为了向其它核发送核间中断请求及 MailBox 通信，通过以下寄存器进行访问。

表 7-8 处理器核核间通信寄存器

名称	偏移地址	权限	描述
IPI_Send	0x1040	WO	32 位中断分发寄存器 [31] 等待完成标志，置 1 时会等待中断生效 [30:26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 中断向量号，对应 IPI_Status 中的向量
Mail_Send	0x1048	WO	64 位 MailBox 缓存寄存器 [63:32] MailBox 数据 [31] 等待完成标志，置 1 时会等待写入生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:2] MailBox 号 0 - MailBox0 低 32 位 1 - MailBox0 高 32 位 2 - MailBox1 低 32 位 3 - MailBox1 高 32 位 4 - MailBox2 低 32 位 5 - MailBox2 高 32 位 6 - MailBox3 低 32 位 7 - MailBox3 高 32 位 [1:0] 保留

FREQ_Send	0x1058	WO	32 位频率使能寄存器 [31] 等待完成标志，置 1 时会等待设置生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 处理器核号 [15:5] 保留 [4:0] 写入对应的处理器核私有频率配置寄存器。IOCSR[0x1050]

需要注意的是，由于 Mail_Send 寄存器一次只可以发送 32 位的数据，当发送 64 位数据时必须拆分为两次发送。因此，目标核在等待 Mail_Box 内容时，需要通过其它的软件手段来确保传输的完整性。例如，发送完 Mail_Box 数据之后，通过核间中断来表示已经发送完成。

7.3 配置寄存器指令调试支持

除了前一节提到的 IPI_Send、Mail Send、Freq Send，还有一个 Any Send 寄存器可供使用，其地址如下。

表 7-9 处理器配置访问寄存器

名称	偏移地址	权限	描述
ANY_Send	0x1158	WO	64 位寄存器访问寄存器 [63:32] 写入数据 [31] 等待完成标志，置 1 时会等待中断生效 [30:27] 写入数据的 mask，每一位表示 32 位写数据对应的字节不会真正写入目标地址，如 1000b 表示写入第 0-2 字节，0000b 则 0-3 字节全部写入 [26] 保留 [25:16] 目标处理器核号 [15:0] 写入的寄存器偏移地址

8 I/O 中断

龙芯 2K3000 芯片中断源分为两部分，一部分来自 NODE 节点的模块，另一部分来自 IO 设备，两部分中断采用层次化管理。NODE 支持两种不同的中断方式。第一种为传统中断方式；第二种为扩展 IO 中断方式。IO 设备中断通过 INT0/1 路由到 NODE 上，作为两个中断源与 NODE 上的其他中断源进行统一管理，任意一个 IO 中断源可以被配置为是否使能、触发的方式、以及被路由的目标处理器核中断脚。

NODE 模块中断引脚定义如下。

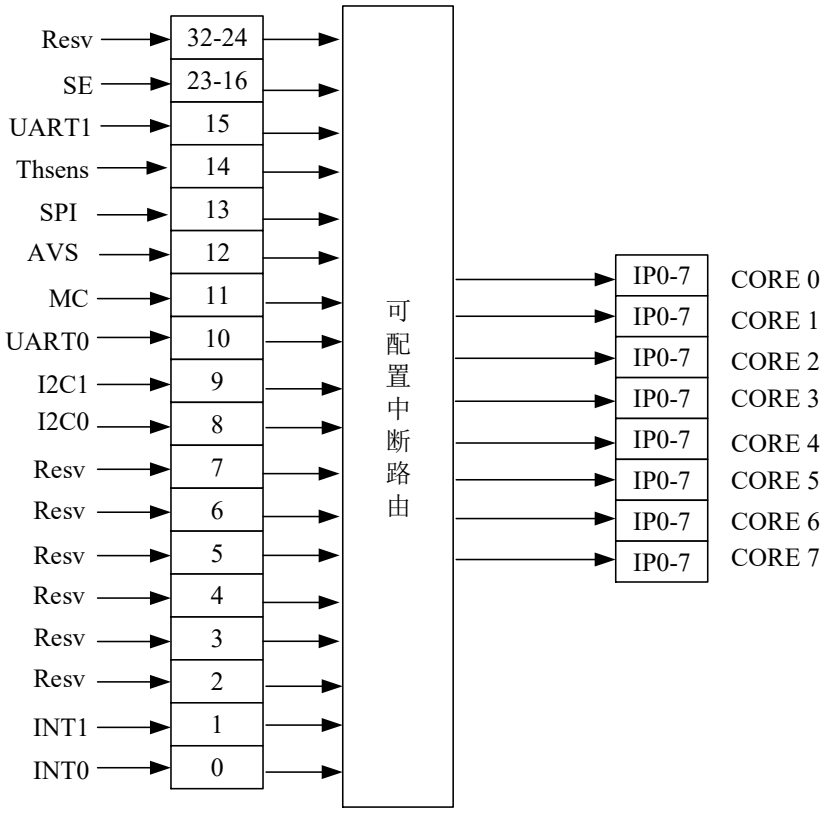


图 8-1 龙芯 2K3000 NODE 中断示意图

IO 模块的中断引脚定义如下。

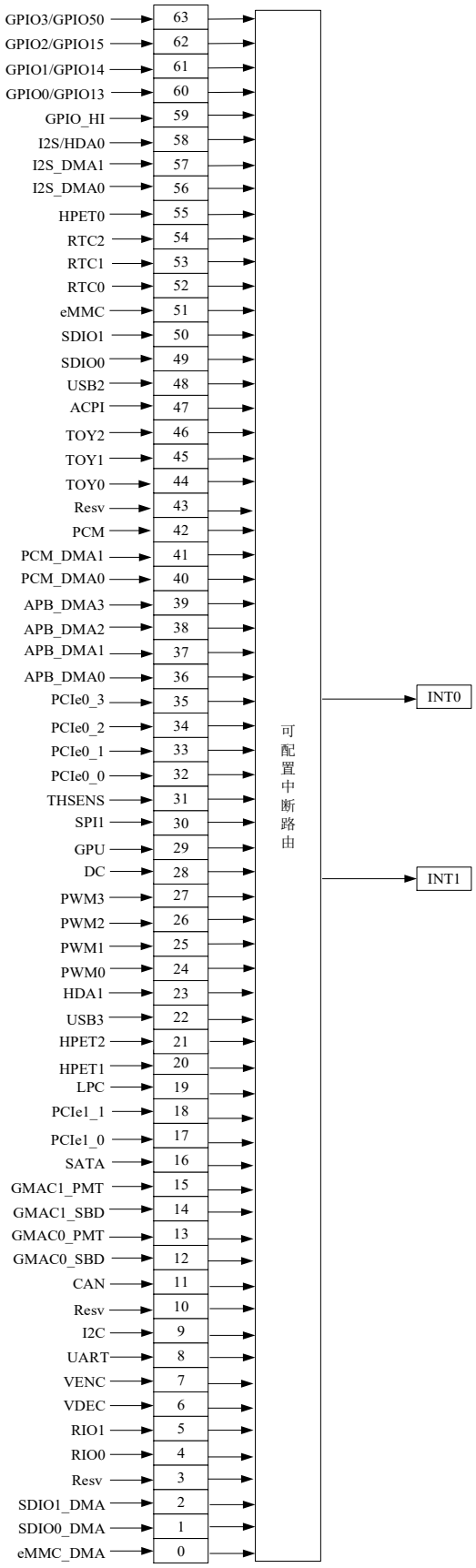


图 8-2 龙芯 2K3000 IO 中断示意图

IO 设备中断通过 INT0/1 路由到 NODE 上，需要使能“其他功能设置寄存器”中的对应位。该寄存器基地址为 0x1fe00000，偏移地址 0x0420。

表 8-1 其它功能设置寄存器

位域	字段名	访问	复位值	描述
48	BRIDGE_INT_en	RW	0x0	南北桥 IO 中断使能

8.1 IO 设备中断控制

8.1.1 中断相关寄存器描述

IO 中断控制器针对每一个中断源，有一套控制和状态寄存器。

寄存器名	位宽	访问类型	说明	默认值
INT_MASK	1	R/W	中断屏蔽寄存器。 0：使能该中断； 1：屏蔽该中断。	1
MSI_EN	1	R/W	消息包中断使能寄存器。 0：关闭消息包方式； 1：使能消息包方式。	0
INTEDGE	1	R/W	触发方式设置寄存器。 0：电平触发中断； 1：边沿触发中断。	0
INTCLR	1	WO	脉冲触发中断清除寄存器。 写 1 清除该中断，写 0 无效。	N/A
SOFT_INT	1	R/W	软件触发中断寄存器。 使用软件来模拟设备中断，等价于设备的中断输出。	0
AUTO_CTRL0	1	R/W	中断分发模式控制寄存器（与 AUTO_CTRL1 合用）。 {AUTO_CTRL1，AUTO_CTRL0}： 00b：固定分发模式； 01b：轮转分发模式； 10b：空闲分发模式； 11b：忙碌分发模式。	0
AUTO_CTRL1	1	R/W	中断分发模式控制寄存器（与 AUTO_CTRL0 合用）。 {AUTO_CTRL1，AUTO_CTRL0}： 00b：固定分发模式； 01b：轮转分发模式； 10b：空闲分发模式； 11b：忙碌分发模式。	0
ROUTE_ENTRY	8	R/W	中断路由寄存器。 用来配置将该中断路由给哪个处理器，该寄存器按照位图的形式组织。 bit0：路由给 INTn0； bit1：路由给 INTn1。 bit7:2：保留。	0
MSI_VECTOR	8	R/W	消息包中断向量寄存器。	见下文
INTISR_CHIP0	1	RO	路由到 INTn0 的中断状态（在服务）寄存器。 0：无中断； 1：有中断。	0

INTISR_CHIP1	1	RO	路由到 INTn1 的中断状态（在服务）寄存器。 0：无中断； 1：有中断。	0
INTIRR	1	RO	中断请求寄存器。 0：没有中断请求； 1：有中断请求。	0
INTISR	1	RO	中断状态（在服务）寄存器。 0：没有中断被接收； 1：有中断被接收。	0
INT_POLARITY	1	R/W	中断电平触发极性选择寄存器。 对于电平触发类型： 0：高电平触发； 1：低电平触发。	0

IO 中断控制器的访问地址是固定的，由固件进行分配，地址空间大小为 4KB。中断控制器相关寄存器的地址分布见表 5-2。

表 8-2 中断寄存器地址分布

寄存器名	地址偏移	访问	说明
INT_APIC_ID	0x000	RO	中断控制器标识寄存器
INT_MASK	0x020	R/W	中断屏蔽寄存器
MSI_EN	0x040	R/W	消息包中断使能寄存器
INTEDGE	0x060	R/W	触发方式设置寄存器
INTCLR	0x080	WO	脉冲触发中断清除寄存器
SOFT_INT	0x0a0	R/W	软件触发中断寄存器
AUTO_CTRL0	0x0c0	R/W	中断分发模式控制寄存器 0
AUTO_CTRL1	0x0e0	R/W	中断分发模式控制寄存器 1
ROUTE_ENTRY_0	0x100	R/W	中断路由寄存器[7- 0]
ROUTE_ENTRY_8	0x108	R/W	中断路由寄存器[15- 8]
ROUTE_ENTRY_16	0x110	R/W	中断路由寄存器[23-16]
ROUTE_ENTRY_24	0x118	R/W	中断路由寄存器[31-24]
ROUTE_ENTRY_32	0x120	R/W	中断路由寄存器[39-32]
ROUTE_ENTRY_40	0x128	R/W	中断路由寄存器[47-40]
ROUTE_ENTRY_48	0x130	R/W	中断路由寄存器[55-48]
ROUTE_ENTRY_56	0x138	R/W	中断路由寄存器[63-56]
MSI_VECTOR0	0x200	R/W	MSI 中断向量寄存器[7- 0]
MSI_VECTOR8	0x208	R/W	MSI 中断向量寄存器[15- 8]
MSI_VECTOR16	0x210	R/W	MSI 中断向量寄存器[23-16]
MSI_VECTOR24	0x218	R/W	MSI 中断向量寄存器[31-24]
MSI_VECTOR32	0x220	R/W	MSI 中断向量寄存器[39-32]
MSI_VECTOR40	0x228	R/W	MSI 中断向量寄存器[47-40]
MSI_VECTOR48	0x230	R/W	MSI 中断向量寄存器[55-48]
MSI_VECTOR56	0x238	R/W	MSI 中断向量寄存器[63-56]
INTISR_0	0x300	RO	路由到 INTn0 的中断状态（在服务）寄存器
INTISR_1	0x320	RO	路由到 INTn1 的中断状态（在服务）寄存器
INTIRR	0x380	RO	中断请求寄存器
INTISR	0x3a0	RO	中断状态（在服务）寄存器
INT_POLARITY	0x3e0	R/W	中断触发电平选择寄存器

中断控制器标识寄存器

地址偏移：000-003h　　属性：RO

默认值：07000000h　　大小：32 位

位域	名称	访问	描述
31:24	apic_id	RO	中断控制器 ID
23:0	Reserved	RO	保留

地址偏移：004-007h　　属性：RO
默认值：003F0001h　　大小：32 位

位域	名称	访问	描述
31:24	Reserved	RO	保留
23:16	int_num	RO	支持的中断源个数。实际中断个数等于该字段的值加 1。
15:8	Reserved	RO	保留
7:0	apic_version	RO	中断控制器版本号

中断掩码寄存器

地址偏移：020-023h　　属性：R/W
默认值：FFFFFFFFh　　大小：32 位

位域	名称	访问	描述
31:0	int_mask	R/W	中断掩码寄存器的低 32 位 (bit[31:0])

地址偏移：024-027h　　属性：R/W
默认值：FFFFFFFFh　　大小：32 位

位域	名称	访问	描述
31:0	int_mask	R/W	中断掩码寄存器的高 32 位 (bit[63:32])

中断消息包使能寄存器

地址偏移：040-043h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	msi_en	R/W	中断消息包使能寄存器的低 32 位 (bit[31:0])

地址偏移：044-047h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	msi_en	R/W	中断消息包使能寄存器的高 32 位 (bit[63:32])

中断触发控制寄存器

地址偏移：060-063h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_edge	R/W	中断触发控制寄存器的低 32 位 (bit[31:0])

地址偏移：064-067h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_edge	R/W	中断触发控制寄存器的高 32 位 (bit[63:32])

中断清除寄存器

地址偏移：080-083h　　属性：WO
默认值：N/A　　大小：32 位

位域	名称	访问	描述
31:0	int_clear	WO	中断清除寄存器的低 32 位 (bit[31:0])

地址偏移：084-087h　　属性：WO
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_clear	WO	中断清除寄存器的高 32 位 (bit[63:32])

软中断寄存器

地址偏移：0A0-0A3h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	soft_int	R/W	软中断寄存器的低 32 位 (bit[31:0])

地址偏移：0A4-0A7h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	soft_int	R/W	软中断寄存器的高 32 位 (bit[63:32])

INT_AUTO_CTRL0 寄存器

地址偏移：0C0-0C3h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_auto_ctrl0	R/W	中断智能分发控制寄存器 0 的低 32 位 (bit[31:0])

地址偏移：0C4-0C7h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_auto_ctrl0	R/W	中断智能分发控制寄存器 0 的高 32 位 (bit[63:32])

INT_AUTO_CTRL1 寄存器

地址偏移：0C0-0C3h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_auto_ctrl1	R/W	中断智能分发控制寄存器 1 的低 32 位 (bit[31:0])

地址偏移：0C4-0C7h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_auto_ctrl1	R/W	中断智能分发控制寄存器 1 的高 32 位 (bit[63:32])

中断路由配置寄存器

地址偏移：100-13Fh　　属性：R/W
该寄存器用来配置各个中断的路由目的地，每个 IO 中断可以被路由到 NODE 传统 IO 中断的 INT0 或 INT1。该寄存器的每个字节低两位用来独立控制对应 IO 中断的路由，当 bit0 为 1 时代表路由到 INT0，当 bit1 为 1 时代表路由到 INT1。下表列出前三个 IO 中断的寄存器描述，其他中断以此类推。
地址偏移：100-103h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24			保留
17:16	SDIO1_DMA	R/W	SDIO1_DMA 中断路由配置寄存器
9:8	SDIO0_DMA	R/W	SDIO0_DMA 中断路由配置寄存器
1:0	eMMC_DMA	R/W	eMMC_DMA 中断路由配置寄存器

消息包中断向量配置寄存器

地址偏移：200-23Fh　　属性：R/W
该寄存器用来配置各个中断对应的消息包向量号，每个字节用来独立控制对应 IO 中断。下表列出前三个 IO 中断的寄存器描述，其他中断以此类推。
地址偏移：200-203h　　属性：R/W
默认值：03020100h　　大小：32 位

位域	名称	访问	描述
25:24		R/W	保留
17:16	SDIO1_DMA_int_vector	R/W	RIO1 MSI 中断向量配置寄存器
9:8	SDIO0_DMA_int_vector	R/W	SDIO_DMA MSI 中断向量配置寄存器
1:0	eMMC_DMA_int_vector	R/W	eMMC_DMA MSI 中断向量配置寄存器

路由到 INTn0 的中断在服务状态寄存器

地址偏移：300-303h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_isr_0	R/W	路由到 INTn0 的中断在服务状态寄存器的低 32 位 (bit[31:0])

地址偏移：304-307h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_isr_0	R/W	路由到 INTn0 的中断在服务状态寄存器的高 32 位 (bit[63:32])

路由到 INTn1 的中断在服务状态寄存器

地址偏移：320-323h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_isr_1	R/W	路由到 INTn1 的中断在服务状态寄存器的低 32 位 (bit[31:0])

地址偏移：324-327h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_isr_1	R/W	路由到 INTn1 的中断在服务状态寄存器的高 32 位 (bit[63:32])

中断请求寄存器

地址偏移：380-383h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_irr	R/W	中断请求寄存器的低 32 位 (bit[31:0])

地址偏移：384-387h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_irr	R/W	中断请求寄存器的高 32 位 (bit[63:32])

中断在服务状态寄存器

地址偏移：3A0-3A3h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_isr	R/W	中断在服务状态寄存器的低 32 位 (bit[31:0])

地址偏移：3A4-3A7h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_isr	R/W	中断在服务状态寄存器的高 32 位 (bit[63:32])

中断电平触发极性寄存器

地址偏移：3E0-3E3h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_polarity	R/W	中断电平触发极性寄存器的低 32 位 (bit[31:0])

地址偏移：3E4-3E7h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	int_polarity	R/W	中断电平触发极性寄存器的高 32 位 (bit[63:32])

8.1.2 设备中断类型

IO 设备中断都连接到了 IO 中断控制器上。对于 IO 设备来说，I2S DMA 中断为脉冲触发类型，GPIO 中断根据需要可以配置成电平触发或者脉冲触发，其余中断均为电平触发，且高电平有效。

对于 PCIe 设备，一种方式是通过中断线的方式将 PCIe 控制器的中断送给 IO 中断控制器；另一种方式是直接使用 PCIe 设备的 MSI 中断。

在 PCIe MSI 中断方式下，PCIe 控制器接收到 MSI 中断时，会将它直接转换成中断消息包发给 NODE 中断控制器。

8.1.3 中断分发模式

IO 模块支持双路中断输出，因此可以将不同的中断源或者同一个中断源的多次中断在这两路中断输出之间进行分发。南北桥支持中断硬件负载均衡功能，使得中断可以在软件预设的两路输出之间进行分发，分为四种模式：

1. 固定分发模式——按照中断路由配置寄存器配置的路由方式进行分发。在此种模式下，路由配置寄存器必须为 one-hot 编码，也就是说一个中断只能路由给一路中断输出。

2. 轮转分发模式——这种模式下每个新中断产生时，会按照 Route_Entry 寄存器中路由向量的配置，路由到下一个中断输出上。

3. 空闲分发模式——跳转到空闲的中断输出。这种模式下，每个新中断产生时，会按照 Entry 寄存器中路由向量的配置，首先检测下一个中断输出上是否已有未被处理的中断，如果没有，则路由到该中断输出；如果下一个中断输出上已经存在未被处理的中断，则继续检测下一个处理器核。

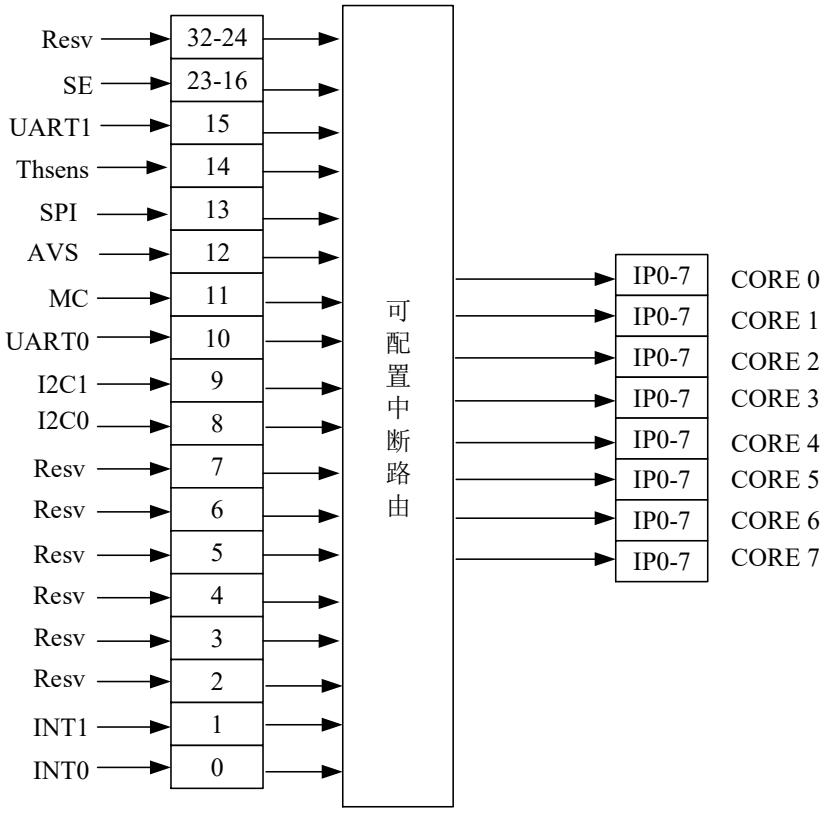
4. 忙碌分发模式——当前中断输出忙碌则跳转到其左边 (0->1->2->3) 的候选中断输出上。这种模式下每个新中断产生时，会首先检测上次中断的中断输出上是否已有未被处理的中断，如果没有，则继续中断该中断输出，如果存在未被处理的中断，则按照 Entry 寄存器的配置，路由到下一个中断输出上。

需要注意到的是，一旦配置完 AUTO_CTRL0/1 后不应该在运行中途修改。

建议软件使用固定分发模式。

8.2 NODE 传统 I/O 中断

龙芯 2K3000 芯片的传统中断支持 32 个中断源，以统一方式进行管理，如下图所示。



每一个内部结点对应一个中断路由模块，其寄存器的基址与其它的配置寄存器基址同样都是内部结点号位置的变化。

任意一个 IO 中断源可以被配置为是否使能，可以配置触发的方式以及被路由的目标处理器核中断脚。传统中断不支持中断的跨结点分发，只能中断同一个处理器结点内的处理器核。

中断相关配置寄存器都是以位的形式对相应的中断线进行控制，中断控制位连接及属性配置见下表。

中断使能（Enable）的配置有三个寄存器：Intenset、Intenclr 和 Inten。Intenset 设置中断使能，Intenset 寄存器写 1 的位对应的中断被使能。Intenclr 清除中断使能，Intenclr 寄存器写 1 的位对应的中断被清除。Inten 寄存器读取当前各中断使能的情况。

边沿触发的中断信号由 Intedge 配置寄存器来选择，写 1 表示边沿触发，写 0 表示电平触发。中断处理程序可以通过 Intenclr 的相应位来清除中断记录，清除中断的同时也会清除中断使能。

表 8-3 中断控制寄存器

位域	访问属性/缺省值				
	Intedge	Inten	Intenset	Intenclr	中断源
0	RW / 0	R / 0	RW / 0	RW / 0	INT0
1	RW / 0	R / 0	RW / 0	RW / 0	INT1
2	RW / 0	R / 0	RW / 0	RW / 0	-
3	RW / 0	R / 0	RW / 0	RW / 0	-
4	RW / 0	R / 0	RW / 0	RW / 0	-
5	RW / 0	R / 0	RW / 0	RW / 0	-
6	RW / 0	R / 0	RW / 0	RW / 0	-
7	RW / 0	R / 0	RW / 0	RW / 0	-
8	RW / 0	R / 0	RW / 0	RW / 0	I2C0
9	RW / 0	R / 0	RW / 0	RW / 0	I2C1
10	RW / 0	R / 0	RW / 0	RW / 0	UART0
11	RW / 0	R / 0	RW / 0	RW / 0	MC
12	RW / 0	R / 0	RW / 0	RW / 0	AVS
13	RW / 0	R / 0	RW / 0	RW / 0	SPI
14	RW / 0	R / 0	RW / 0	RW / 0	Thsens
15	RW / 0	R / 0	RW / 0	RW / 0	UART1
23 : 16	RW / 0	R / 0	RW / 0	RW / 0	SE[7:0]
31 : 24	RW / 0	R / 0	RW / 0	RW / 0	-

与核间中断类似，IO 中断的基地址同样可以使用 0x1FE00000 进行访问，也可以通过处理器核的 IOCSR 指令进行访问。不同结点使用的基地址规则与之前的配置寄存器基地址规则一致。

8.2.1 按地址访问

按地址访问方式下，基地址为 0x1FE00000。不同结点使用的基地址规则与之前的配置寄存器基地址规则一致。由于每个龙芯 2K3000 中包括 8 个核，被划分为 2 个内部结点。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]加上内部结点号。例如，核 4 至核 7 的基地址为 0x1FE10000，具体的寄存器对应于该基址上核 0 至核 3 的偏移。

表 8-4 IO 控制寄存器地址

名称	偏移地址	描述
Intisr	0x1420	32 位中断状态寄存器
Inten	0x1424	32 位中断使能状态寄存器
Intenset	0x1428	32 位设置使能寄存器
Intenclr	0x142c	32 位清除使能寄存器
Intedge	0x1434	32 位触发方式寄存器
COREO_INTISR	0x1440	路由给 COREO 的 32 位中断状态

CORE1_INTISR	0x1448	路由给 CORE1 的 32 位中断状态
CORE2_INTISR	0x1450	路由给 CORE2 的 32 位中断状态
CORE3_INTISR	0x1458	路由给 CORE3 的 32 位中断状态

龙芯 2K3000 每个内部结点对应 4 个核，上述的 32 位中断源可以通过软件配置选择期望中断的目标核。进一步，中断源可以选择路由到该结点内的逻辑核中断 INT0 到 INT7 中任意一个。32 个 I/O 中断源中每一个都对应一个 8 位的路由控制器，其格式和地址如下表所示。路由寄存器采用向量的方式进行路由选择，如 0x48 表示路由到 3 号逻辑核的 INT2 上。

采用位向量模式时，中断路由能够直接对应低 4 位中断引脚；而使用编码方式，则可以映射到 8 个中断中的任意一个，编码方式由 IOCSR[0x420][49]位控制使能。当该位使能时，下表的[7:4]由位图表示法变为数值编码法。可配置的数值 0-7 表示中断引脚 0-7。例如，在该模式下，0x28 表示路由到 3 号核的 INT2 上。

表 8-5 中断路由寄存器的说明

位域	说　明
3:0	路由的逻辑核向量号
7:4	路由的逻辑核中断引脚向量号

表 8-6 中断路由寄存器地址

名称	偏移地址	描述	名称	偏移地址	描述
Entry0	0x1400	INT0	Entry16	0x1410	SE-int0
Entry1	0x1401	INT1	Entry17	0x1411	SE-int1
Entry2	0x1402	-	Entry18	0x1412	SE-int2
Entry3	0x1403	-	Entry19	0x1413	SE-int3
Entry4	0x1404	-	Entry20	0x1414	SE-int4
Entry5	0x1405	-	Entry21	0x1415	SE-int5
Entry6	0x1406	-	Entry22	0x1416	SE-int6
Entry7	0x1407	-	Entry23	0x1417	SE-int7
Entry8	0x1408	I2C0	Entry24	0x1418	-
Entry9	0x1409	I2C1	Entry25	0x1419	-
Entry10	0x140a	UART0	Entry26	0x141a	-
Entry11	0x140b	MC	Entry27	0x141b	-
Entry12	0x140c	AVS	Entry28	0x141c	-
Entry13	0x140d	SPI	Entry29	0x141d	-
Entry14	0x140e	Thsens	Entry30	0x141e	-
Entry15	0x140f	UART1	Entry31	0x141f	-

8.2.2 配置寄存器指令访问

在龙芯 2K3000 中，可以使用配置寄存器指令，通过私有空间对配置寄存器进行访问。指令所使用的偏移地址与通过地址访问的方式相同。此外，为了方便用户的使用，对每个核不同的当前中断状态设置了专用的私有中断状态寄存器，如下表所示。

表 8-7 处理器核私有中断状态寄存器

名称	偏移地址	描述
perCore_INTISR	0x1010	路由给当前处理器核的 32 位中断状态

8.3 NODE 扩展 I/O 中断

除了兼容原有的传统 IO 中断方式，2K3000 支持扩展 I/O 中断，用于将 IO 设备的 256 位中断直接分发给各个处理器核，而不再通过传统的中断线进行转发，提升 IO 中断使用的灵活性。

内核在使用扩展 IO 中断前，需要使能“其他功能设置寄存器”中的对应位。该寄存器基地址为 0x1FE00000，偏移地址 0x0420。

表 8-8 其它功能设置寄存器

位域	字段名	访问	复位值	描述
48	EXT_INT_en	RW	0x0	扩展 IO 中断使能

8.3.1 按地址访问

以下是相关的扩展 IO 中断寄存器。与其它的配置寄存器一样，基地址可以使用 0x1FE00000，也可以通过处理器核的专用寄存器配置指令进行访问。不同结点使用的基地址规则与之前的配置寄存器基地址规则一致。由于每个龙芯 2K3000 中包括 8 个处理器核，被划分为 2 内部结点。当使用地址进行访问时，各个内部结点的基地址需要在地址的[19:16]上加上内部结点号。例如，核 4 至核 7 的基地址为 0x1FE10000，具体的寄存器对应于该基址上核 0 至核 3 的偏移。

表 8-9 扩展 IO 中断使能寄存器

名称	偏移地址	描述
EXT_IOIen[63:0]	0x1600	扩展 IO 中断[63:0]的中断使能配置
EXT_IOIen[127:64]	0x1608	扩展 IO 中断[127:64]的中断使能配置
EXT_IOIen[191:128]	0x1610	扩展 IO 中断[191:128]的中断使能配置
EXT_IOIen[255:192]	0x1618	扩展 IO 中断[255:192]的中断使能配置

表 8-10 扩展 IO 中断自动轮转使能寄存器

名称	偏移地址	描述
EXT_IOIbounce[63:0]	0x1680	扩展 IO 中断[63:0]的自动轮转使能配置
EXT_IOIbounce[127:64]	0x1688	扩展 IO 中断[127:64]的自动轮转使能配置

EXT_IOIbounce[191:128]	0x1690	扩展 IO 中断[191:128]的自动轮转使能配置
EXT_IOIbounce[255:192]	0x1698	扩展 IO 中断[255:192]的自动轮转使能配置

这里需要注意的是，在 bounce 使能的情况下，不能随意修改对应中断位的路由信息。

表 8-11 扩展 IO 中断状态寄存器

名称	偏移地址	描述
EXT_IOIsr[63:0]	0x1700	扩展 IO 中断[63:0]的中断状态
EXT_IOIsr[127:64]	0x1708	扩展 IO 中断[127:64]的中断状态
EXT_IOIsr[191:128]	0x1710	扩展 IO 中断[191:128]的中断状态
EXT_IOIsr[255:192]	0x1718	扩展 IO 中断[255:192]的中断状态

表 8-12 各处理器核的扩展 IO 中断状态寄存器

名称	偏移地址	描述
CORE0_EXT_IOIsr[63:0]	0x1800	路由至处理器核 0 的扩展 IO 中断[63:0]的中断状态
CORE0_EXT_IOIsr[127:64]	0x1808	路由至处理器核 0 的扩展 IO 中断[127:64]的中断状态
CORE0_EXT_IOIsr[191:128]	0x1810	路由至处理器核 0 的扩展 IO 中断[191:128]的中断状态
CORE0_EXT_IOIsr[255:192]	0x1818	路由至处理器核 0 的扩展 IO 中断[255:192]的中断状态
CORE1_EXT_IOIsr[63:0]	0x1900	路由至处理器核 1 的扩展 IO 中断[63:0]的中断状态
CORE1_EXT_IOIsr[127:64]	0x1908	路由至处理器核 1 的扩展 IO 中断[127:64]的中断状态
CORE1_EXT_IOIsr[191:128]	0x1910	路由至处理器核 1 的扩展 IO 中断[191:128]的中断状态
CORE1_EXT_IOIsr[255:192]	0x1918	路由至处理器核 1 的扩展 IO 中断[255:192]的中断状态
CORE2_EXT_IOIsr[63:0]	0x1A00	路由至处理器核 2 的扩展 IO 中断[63:0]的中断状态
CORE2_EXT_IOIsr[127:64]	0x1A08	路由至处理器核 2 的扩展 IO 中断[127:64]的中断状态
CORE2_EXT_IOIsr[191:128]	0x1A10	路由至处理器核 2 的扩展 IO 中断[191:128]的中断状态
CORE2_EXT_IOIsr[255:192]	0x1A18	路由至处理器核 2 的扩展 IO 中断[255:192]的中断状态
CORE3_EXT_IOIsr[63:0]	0x1B00	路由至处理器核 3 的扩展 IO 中断[63:0]的中断状态
CORE3_EXT_IOIsr[127:64]	0x1B08	路由至处理器核 3 的扩展 IO 中断[127:64]的中断状态
CORE3_EXT_IOIsr[191:128]	0x1B10	路由至处理器核 3 的扩展 IO 中断[191:128]的中断状态
CORE3_EXT_IOIsr[255:192]	0x1B18	路由至处理器核 3 的扩展 IO 中断[255:192]的中断状态

与传统 IO 中断类似，扩展 IO 中断的 256 位中断源也可以通过软件配置选择期望中断的目标处理器核。

但中断源并不可以单独选择路由到处理器核中断 INT0 到 INT7 中的任意一个，而是以组为单位进行 INT 中断的路由，以中断对应 ESTAT 的 IS2 到 IS9。下面是按组进行配置的中断引脚路由寄存器。

中断引脚路由位支持编码方式，由 IOCSR[0x420][49]位控制使能。当该位使能时，下表的[3:0]由位图表示法变为数值编码法。可配置的数值 0-7 表示中断引脚 0-7。例如，在该模式下，0x2 表示路由到 INT2 上。

表 8-13 中断引脚路由寄存器的说明

位域	说 明
3:0	路由的处理器核中断引脚向量号
7:4	保留

表 8-14 中断路由寄存器地址

名称	偏移地址	描述
EXT_IOImap0	0x14C0	EXT_IOI[31:0]的引脚路由方式
EXT_IOImap1	0x14C1	EXT_IOI[63:32]的引脚路由方式
EXT_IOImap2	0x14C2	EXT_IOI[95:64]的引脚路由方式
EXT_IOImap3	0x14C3	EXT_IOI[127:96]的引脚路由方式
EXT_IOImap4	0x14C4	EXT_IOI[159:128]的引脚路由方式
EXT_IOImap5	0x14C5	EXT_IOI[191:160]的引脚路由方式
EXT_IOImap6	0x14C6	EXT_IOI[223:192]的引脚路由方式
EXT_IOImap7	0x14C7	EXT_IOI[255:224]的引脚路由方式

每个中断源都另外对应一个 8 位的路由控制器，其格式和地址如表 8-15 所示。其中[7:4]用于在表 8-16 中选择真正的结点路由向量。路由寄存器采用向量的方式进行路由选择，如 0x48 表示路由到 EXT_IOI_node_type4 所指结点的 3 号处理器核。

表 8-15 中断目标处理器核路由寄存器的说明

位域	说 明
3:0	路由的处理器核向量号
7:4	路由的内部结点映射方式选择（选择下表中的一项）

需要注意的是，当使用轮转分发模式时（对应的 EXT_IOIbounce 为 1），在结点号与处理器核号的全映射模式上轮转。EXT_IOIbounce 的设置应该在相关的路由映射配置之后。需要注意的是，此处结点的计算为前面给出的 IOI_Node。

IOI_Node = (thread / 4) % 2

例如，当表 8-15

中的设置为 0x27，而表 8-16 中的 EXT_IOI_node_type2 的设置为 0x0003 时，使能轮转分发模式下，该中断将依次在核 0（内部结点 0 核 0，核号 0）、核 1（内部结点 0 核 1，核号 1）、核 2（内部结点 0 核 2，核号 2）、核 4（内部结点 1 核 0，核号 4）、核 5（内部结点 1 核 1，核号 5）、核 6（内部结点 1 核 2，核号 6）上轮转。

当使用固定分发模式时（对应的 EXT_IOIbounce 为 0），结点号的 bitmap 上只允许有一位为 1，或为全 0，对应本地触发。

表 8-16 中断目标处理器核路由寄存器地址

名称	偏移地址	描述
EXT_IOImap_Core0	0x1C00	EXT_IOI[0]的处理器核路由方式
EXT_IOImap_Core1	0x1C01	EXT_IOI[1]的处理器核路由方式

EXT_IOImap_Core2	0x1C02	EXT_IOI[2]的处理器核路由方式
……		
EXT_IOImap_Core254	0x1CFE	EXT_IOI[254]的处理器核路由方式
EXT_IOImap_Core255	0x1CFF	EXT_IOI[255]的处理器核路由方式

表 8-17 中断目标结点映射方式配置

名称	偏移地址	描述
EXT_IOI_node_type0	0x14A0	16 个结点的映射向量类型 0（软件配置）
EXT_IOI_node_type1	0x14A2	16 个结点的映射向量类型 1（软件配置）
EXT_IOI_node_type2	0x14A4	16 个结点的映射向量类型 2（软件配置）
……		
EXT_IOI_node_type15	0x14BE	16 个结点的映射向量类型 15（软件配置）

8.3.2 配置寄存器指令访问

使用处理器核的配置寄存器指令进行访问时，最大的不同在于对处理器核的中断状态寄存器的访问成为私有的访问，每个核都只需要向同一个地址发出查询请求就可以得到当前核的中断状态。

表 8-18 当前处理器核的扩展 IO 中断状态寄存器

名称	偏移地址	描述
perCore_EXT_IOIsr[63:0]	0x1800	路由至当前处理器核的扩展 IO 中断[63:0]的中断状态
perCore_EXT_IOIsr[127:64]	0x1808	路由至当前处理器核的扩展 IO 中断[127:64]的中断状态
perCore_EXT_IOIsr[191:128]	0x1810	路由至当前处理器核的扩展 IO 中断[191:128]的中断状态
perCore_EXT_IOIsr[255:192]	0x1818	路由至当前处理器核的扩展 IO 中断[255:192]的中断状态

8.3.3 扩展 IO 中断触发寄存器

为了支持扩展 IO 中断的动态分发，在配置寄存器中增加了一个扩展 IO 中断触发寄存器，用于将对应的 IO 中断置位。平时可以利用这个寄存器对中断进行调试或测试。

这个寄存器的说明如下：

表 8-19 扩展 IO 中断触发寄存器

名称	偏移地址	权限	描述
EXT_IOI_send	0x1140	WO	扩展 IO 中断设置寄存器 [7:0]为期望设置的中断向量

9 温度传感器

龙芯 2K3000 内部集成温度传感器，可以通过采样寄存器进行观测，同时内部实现了灵活的高低温中断报警机制。

下面列出温度传感器相关的寄存器以及说明，这些寄存器的基地址与 IO 中断控制器一致。

9.1 温度传感器配置寄存器

本寄存器用来配置温度传感器的一些控制参数。

地址偏移：0400h
默认值：0000_0000_0000_0001h

位域	名称	访问	描述
63:7	reserved	R/W	保留
6:4	cluster_sel	R/W	采样点选择：0 为本地监测点，对于电压检测可选值有 1-7，对于温度检测可选值有 1-7。
3:2	mode	R/W	工作模式控制： 00-温度采样 10-电压采样 其他-保留
1	rate	R/W	检测速率控制： 0-低速模式（10~20Hz） 1-高速模式（325~650Hz）
0	powerdown	R/W	低功耗控制，为 1 代表进入低功耗模式

9.2 温度传感器中断控制寄存器

本寄存器用来对温度传感器中断进行控制。

地址偏移：0408h
默认值：0000_0000_0000_0001h

位域	名称	访问	描述
63:41	reserved	R/W	保留
40	low_int_en	R/W	低温中断使能
39:32	temp_low	R/W	低温中断触发温度设置： [7]-温度正负指示，0 代表正值，1 代表负值； [6:0]-摄氏温度值
31:9	reserved	R/W	保留
8	high_int_en	R/W	高温中断使能
7:0	temp_high	R/W	高温中断触发温度设置： [7]-温度正负指示，0 代表正值，1 代表负值； [6:0]-摄氏温度值

9.3 温度传感器中断状态/清除寄存器

本寄存器用来对温度传感器中断进行控制。

地址偏移：0410h

默认值：0000_0000_0000_0000h

位域	名称	访问	描述
63:56	temperature	RO	摄氏温度值 最高位为符号位，0 代表正，1 代表负。当温度为负值时，按位取反后为真实温度。
55:	reserved	R/W	保留
54:52	thsens_outcluster	RO	传感器配置的监测点
51:49	reserved	R/W	保留
48	thsens_outmode	RO	传感器配置的监测模式 0：温度模式；1：电压模式
47	reserved	R/W	保留
46	thsens_overflow	R/W	传感器监测值溢出标志
45:32	thsens_data	R/W	传感器的原始数据
31:2	reserved	R/W	保留
1	int_high	R/W	中断状态指示，读取时获取高温中断状态，写 1 清零
0	int_low	R/W	中断状态指示，读取时获取低温中断状态，写 1 清零

9.4 高温自动降频设置

为了在高温环境中保证芯片的运行，可以设置令高温自动降频，使得芯片在超过预设范围时主动进行时钟分频，达到降低芯片翻转率的效果。

对于高温降频功能，有 4 组控制寄存器对其行为进行设置。每组寄存器包含以下四个控制位：

GATE：设置高温或低温的阈值。当输入温度高于高温阈值或低于低温阈值时，将触发分频操作；

EN：使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 2K3000 内部集成 1 个温度传感器，该寄存器用于配置选择哪个传感器的温度作为输入，因此只能选择 0。

FREQ：分频数。当触发分频操作时，使用预设的 FREQ 对时钟进行分频，分频的模式受 freqscale_mode_node 的控制。

其基地址为 0x1fe00000。

表 9-1 高温降频控制寄存器说明

寄存器	地址	控制	说明
高温降频控制寄存器 Thsens_freq_scale	0x1480	RW	四组设置优先级由高到低 [7:0]：Scale_gate0：高温阈值 0，超过这个温度将降频 [8:8]：Scale_en0：高温降频使能 0 [11:10]：Scale_Sel0：选择高温降频 0 的温度传感器输入源 [14:12]：Scale_freq0：降频时的分频值 [23:16]：Scale_gate1：高温阈值 1，超过这个温度将降频 [24:24]：Scale_en1：高温降频使能 1 [27:26]：Scale_Sel1：选择高温降频 1 的温度传感器输入源 [30:28]：Scale_freq1：降频时的分频值 [39:32]：Scale_gate2：高温阈值 2，超过这个温度将降频 [40:40]：Scale_en2：高温降频使能 2 [43:42]：Scale_Sel2：选择高温降频 2 的温度传感器输入源 [46:44]：Scale_freq2：降频时的分频值 [55:48]：Scale_gate3：高温阈值 3，超过这个温度将降频 [56:56]：Scale_en3：高温降频使能 3 [59:58]：Scale_Sel3：选择高温降频 3 的温度传感器输入源 [62:60]：Scale_freq3：降频时的分频值
Thsens_freq_scale_up	0x1490	RW	温度传感器控制寄存器高位 [7:0] Scale_Hi_gate0 高 8 位 [15:8] Scale_Hi_gate1 高 8 位 [23:16] Scale_Hi_gate2 高 8 位 [31:24] Scale_Hi_gate3 高 8 位 [39:32] Scale_Lo_gate0 高 8 位 [47:40] Scale_Lo_gate1 高 8 位 [55:48] Scale_Lo_gate2 高 8 位 [63:56] Scale_Lo_gate3 高 8 位

9.5 温度状态检测与控制

引脚 PROCHOTn 和 THERMTRIPn 用于温度状态检测与控制。其中 PROCHOTn 既可作为输入也可作为输出，THERMTRIPn 仅有输出功能。

PROCHOTn 作为输入时，芯片受外部温度检测电路的控制，外部温度检测电路需要降低芯片温度时可以置 PROCHOTn 为 0，芯片接收到该低电平后会采取降频措施，降频时的分频值由通过寄存器 prochotn_freq_scale 设置。PROCHOTn 作为输出时，芯片可输出高温中断，通过 prochotn_o_sel 寄存器从高温中断控制寄存器所设置的 4 个中断中选择一个作为对外发出的高温中断。

THERMTRIPn 作为输出，由芯片通过 thermtripn_o_sel 寄存器从高温中断控制寄存器所设置的 4 个中断中选择一个作为对外发出的高温中断。

虽然 THERMTRIPn 和 PROCHOTn 都是对外的高温中断，但是 THERMTRIPn 的紧急程度较 PROCHOTn 更高。PROCHOTn 置位时，外部温度控制电路还可以采取一定的措施，比如提高风扇转速。而 THERMTRIPn 置位时，外部电源控制电路应该直接采取紧急断电措施。

具体的控制寄存器如下：

表 9-2 温度状态检测与控制寄存器说明

寄存器	地址	控制	说明
温度状态检测与控制寄存器 Thsens_hi_ctrl	0x1498	RW	[0:0]：prochotn_oe PROCHOTn 引脚输出使能控制，0 为输出，1 为输入 [5:4]：prochotn_o_sel PROCHOTn 高温中断输出选择 [10:8]：prochotn_freq_scale：PROCHOTn 输入有效时的分频值 [17:16]：thermtripn_o_sel THERMTRIPn 高温中断输出选择

10 SPI 控制器

串行外围设备接口 SPI 总线技术是多种微处理器、微控制器以及外围设备之间的一种全双工、同步、串行数据接口标准。

10.1 访问地址

龙芯 2K3000 处理器集成了 2 个 SPI 控制器，均支持 4 线模式，SPI0 控制器的地址空间分配如表 10- 1 所示，SPI1 通过 PCI 设备的方式访问（D22:F0）。SPI0 控制器包括两个地址空间，分别如下：

表 10-1 SPI0 控制器地址空间分配

地址名称	地址范围	大小
SPI Memory	0X1C00_0000-0X1D00_0000	16MByte
SPI Register	0X1FE0_01F0-0X1FE0_01FF	16Byte

SPI Memory 地址空间是系统启动时处理器最先访问的地址空间， 0x1C000000 的地址被自动路由至 SPI0。

当需要对 SPI 进行其它操作时，比如发送命令，擦除 SPI Flash 等时，就要使用 SPI Register 空间对控制寄存器进行直接操作。

10.2 SPI 控制器结构

本系统集成的 SPI 控制器仅可作为主控端，所连接的是从设备。对于软件而言，SPI 控制器除了有若干 IO 寄存器外还有一段映射到 SPI Flash 的只读 memory 空间。如果将这段 memory 空间分配在 0x1c000000，复位后不需要软件干预就可以直接访问，从而支持处理器从 SPI Flash 启动。

10.3 配置寄存器

表 10-2 SPI 配置寄存器列表

偏移	名称	描述
0	SPCR	控制寄存器
1	SPSR	状态寄存器
2	TxFIFO/RxFIFO	数据寄存器
3	SPER	外部寄存器
4	SFC_PARAM	参数控制寄存器
5	SFC_SOFTCS	片选控制寄存器
6	SFC_TIMING	时序控制寄存器

10.3.1 控制寄存器（SPCR）

偏移地址：0x0

表 10-3 SPI 控制寄存器 (SPCR)

位域	名称	访问	初值	描述
7	spie	R/W	0	中断输出使能信号高有效
6	spe	R/W	0	系统工作使能信号高有效
5	-	-	0	保留
4	mstr	-	1	master 模式选择位，此位一直保持 1
3	cpol	R/W	0	时钟极性位
2	cpha	R/W	0	时钟相位位 1 则相位相反，为 0 则相同
1:0	spr	R/W	0	sclk_o 分频设定，需要与 sper 的 spre 一起使用

10.3.2 状态寄存器 (SPSR)

偏移地址：0x1

表 10-4 SPI 状态寄存器 (SPSR)

位域	名称	访问	初值	描述
7	spif	R/W	0	中断标志位 1 表示有中断申请，写 1 则清零
6	wcol	R/W	0	写寄存器溢出标志位为 1 表示已经溢出,写 1 则清零
5:4	-	-	0	保留
3	wffull	R	0	写寄存器满标志 1 表示已经满
2	wfempty	R	1	写寄存器空标志 1 表示空
1	rffull	R	0	读寄存器满标志 1 表示已经满
0	rfempty	R	1	读寄存器空标志 1 表示空

10.3.3 数据寄存器 (TxFIFO/RxFIFO)

偏移地址：0x2

表 10-5 SPI 数据寄存器 (TxFIFO/RXFIFO)

位域	名称	访问	初值	描述
7:0	TxFIFO RxFIFO	W R	-	数据发送端口 数据接收端口

10.3.4 外部寄存器 (SPER)

偏移地址：0x3

表 10-6 SPI 外部寄存器 (SPER)

位域	名称	访问	初值	描述
7:6	icnt	R/W	0	传输完多少个字节后发中断 00：1 01：2 10：3 11：4
5:3	-	-	-	保留
2	mode	R/W	0	spi 接口模式控制 0：采样与发送时机同时 1：采样与发送时机错开半周期

位域	名称	访问	初值	描述
1:0	spre	R/W	0	与 spr 一起设定分频的比率

表 10-7 SPI 分频系数

spre	00	00	00	00	01	01	01	01	10	10	10	10
spr	00	01	10	11	00	01	10	11	00	01	10	11
分频系数	2	4	16	32	8	64	128	256	512	1024	2048	4096

10.3.5 参数控制寄存器 (SFC_PARAM)

偏移地址：0x4

表 10-8 SPI 参数控制寄存器 (SFC_PARAM)

位域	名称	访问	初值	描述
7:4	clk_div	R/W	2	时钟分频数选择 分频系数与{spre, spr}组合相同
3	dual_io	R/W	0	双 I/O 模式，优先级高于快速读
2	fast_read	R/W	0	快速读模式
1	burst_en	R/W	0	SPI flash 支持连续地址读模式
0	memory_en	R/W	1	SPI flash 读使能，无效时 csn[0]可由软件控制。

10.3.6 片选控制寄存器 (SFC_SOFTCS)

偏移地址：0x5

表 10-9 SPI 片选控制寄存器 (SFC_SOFTCS)

位域	名称	访问	初值	描述
7:4	csn	R/W	0	csn 引脚输出值
3:0	csen	R/W	0	为 1 时对应位的 csn 线由 7:4 位控制

10.3.7 时序控制寄存器 (SFC_TIMING)

偏移地址：0x6

表 10-10 SPI 时序控制寄存器 (SFC_TIMING)

位域	名称	访问	初值	描述
7:4	samp_dly	RW	0	采样延迟，用于调整读的时序。 1，表示延迟一个单位； 2，表示延迟两个单位，以此类推
3	quad_io	RW	0	4 线模式使能，1 有效
2	tFAST	R/W	0	SPI flash 读采样模式 0：上沿采样，间隔半个 SPI 周期 1：上沿采样，间隔一个 SPI 周期
1:0	tCSH	R/W	3	SPI Flash 的片选信号最短无效时间，以分频后时钟周期 T 计算 00：1T 01：2T 10：4T 11：8T

10.3.8 自定义控制寄存器（CTRL）

偏移地址：0x8

表 10-11 SPI Flash 自定义控制寄存器

位域	名称	访问	初值	描述
7:4	nbyte	RW	0	一次传输的字节数
3:2	reserve	RW	0	保留
1	nbmode	RW	0	多字节传输模式
0	start	RW	0	开始多字节传输，完成后自动清零

10.3.9 自定义命令寄存器（CMD）

偏移地址：0x9

表 10-12 SPI Flash 自定义命令寄存器

位域	名称	访问	初值	描述
7:0	cmd	RW	0	设置发送给 spi flash 的命令

10.3.10 自定义数据寄存器 0（BUF0）

偏移地址：0xa

表 10-13 SPI Flash 自定义数据寄存器 0

位域	名称	访问	初值	描述
7:0	buf0	RW	0	向 SPI 发送写命令时，该寄存器配置发送的第一个字节的数据；向 SPI 发送读命令时，该寄存器存储第一个读回来的数据。

10.3.11 自定义数据寄存器 1（BUF1）

偏移地址：0xb

表 10-14 SPI Flash 自定义数据寄存器 1

位域	名称	访问	初值	描述
7:0	buf1	RW	0	向 SPI 发送写命令时，该寄存器配置发送的第二个字节的数据；向 SPI 发送读命令时，该寄存器存储第二个读回来的数据。

10.3.12 自定义时序寄存器 0（TIMER0）

偏移地址：0xc

表 10-15 SPI Flash 自定义时序寄存器 0

位域	名称	访问	初值	描述
7:0	time0	RW	0	自定义命令所需时间值的低 8 位

10.3.13 自定义时序寄存器 1（TIMER1）

偏移地址：0xd

表 10-16 SPI Flash 自定义时序寄存器 1

位域	名称	访问	初值	描述

位域	名称	访问	初值	描述
7:0	Time1	RW	0	自定义命令所需时间值的中间 8 位

10.3.14 自定义时序寄存器 2（TIMER2）

偏移地址：0xe

表 10-17 SPI Flash 自定义时序寄存器 2

位域	名称	访问	初值	描述
7:0	Time2	RW	0	自定义命令所需时间值的高 8 位

10.4 接口时序

10.4.1 SPI 主控制器接口时序

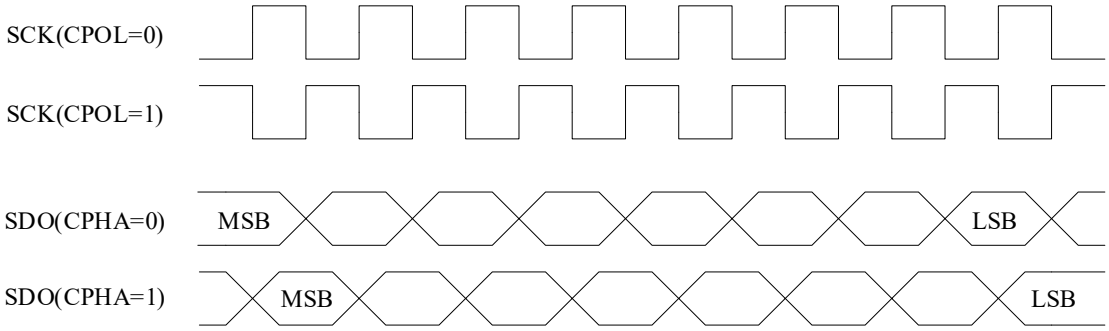


图 10-1 SPI 主控制器接口时序

10.4.2 SPI Flash 访问时序

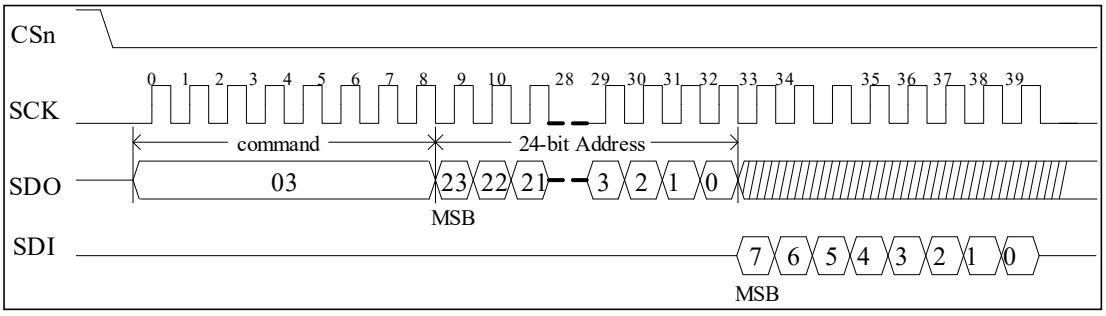


图 10-2 SPI Flash 标准读时序

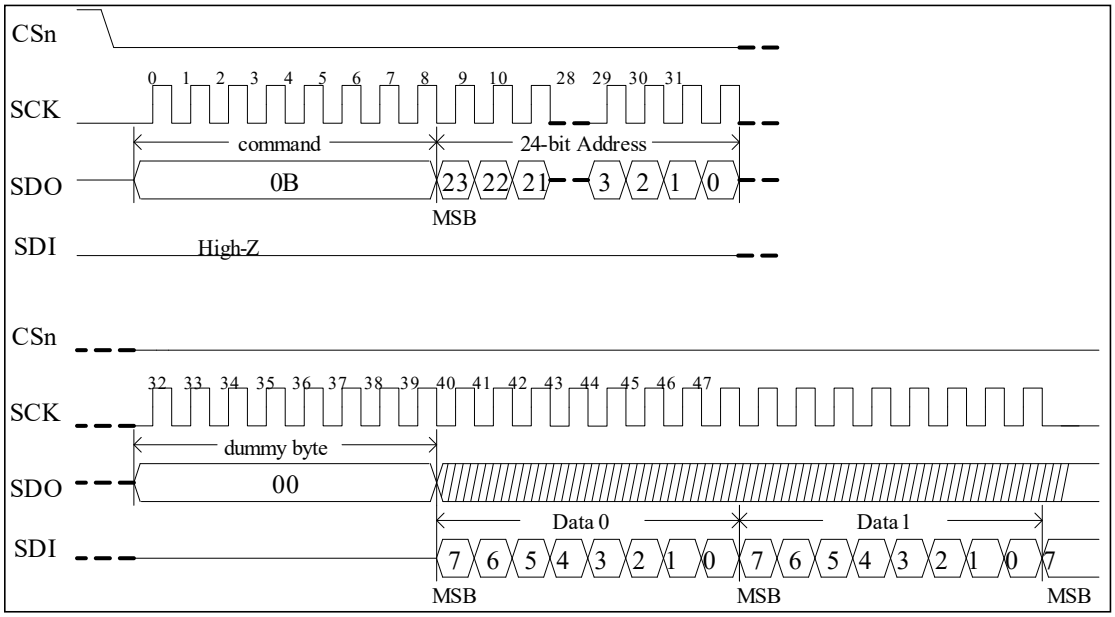


图 10-3 SPI Flash 快速读时序

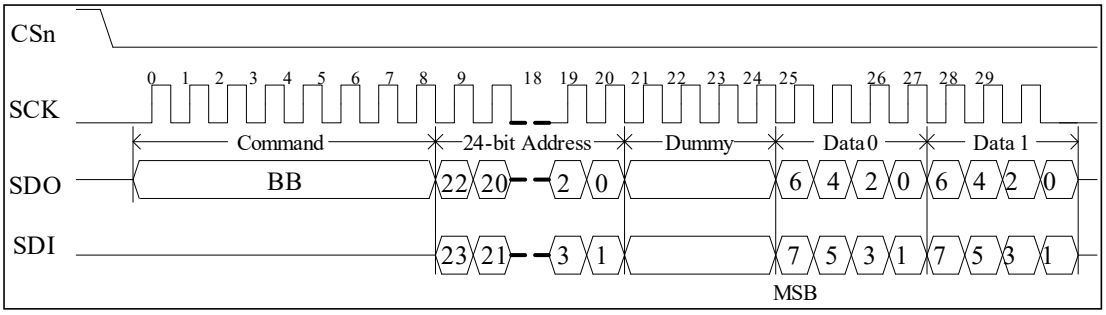


图 10-4 SPI Flash 双向 I/O 读时序

10.5 软件编程指南

10.5.1 SPI 主控制器的读写操作

- **模块初始化.**
  - 停止 SPI 控制器工作，对控制寄存器 spcr 的 spe 位写 0
  - 重置状态寄存器 spsr，对寄存器写入 8'b1100_0000
  - 设置外部寄存器 sper，包括中断申请条件 sper[7:6]和分频系数 sper[1:0]，具体参考寄存器说明
  - 配置 SPI 时序，包括 spcr 的 cpol、cpha 和 sper 的 mode 位。mode 为 1 时是标准 SPI 实现，为 0 时为兼容模式。
  - 配置中断使能，spcr 的 spie 位
  - 启动 SPI 控制器，对控制寄存器 spcr 的 spe 位写 1
- **模块的发送/传输操作**
  - 往数据传输寄存器写入数据

- 传输完成后从数据传输寄存器读出数据。由于发送和接收同时进行，即使 SPI 从设备没有发送有效数据也必须进行读出操作。

- **中断处理**
  - 接收到中断申请
  - 读状态寄存器 spsr 的值，若 spsr[2]为 1 则表示数据发送完成，若 spsr[0]为 1 则表示已经接收数据
  - 读或写数据传输寄存器
  - 往状态寄存器 spsr 的 spif 位写 1，清除控制器的中断申请

10.5.2 硬件 SPI Flash 读

- **初始化**
  - 将 SFC_PARAM 的 memory_en 位写 1。当 SPI 被选为启动设备时此位复位为 1。
  - 设置读参数（时钟分频、连续地址读、快速读、双 I/O、tCSH 等）。这些参数复位值均为最保守的值。
- **更改参数**

如果所使用的 SPI Flash 支持更高的频率或者提供增强功能，修改相应参数可以大大加快 Flash 的访问速度。参数的修改不需要关闭 SPI Flash 读使能（memory_en）。具体参考寄存器说明。

10.5.3 混合访问 SPI Flash 和 SPI 主控制器

- **对 SPI Flash 进行读以外的访问**

将 SPI Flash 读使能关闭后，软件就可直接控制 csn[0]，并通过 SPI 主控制器访问 SPI 总线。这意味着在进行此操作时，不能从 SPI Flash 中取指。

除了读以外，SPI Flash 还实现了很多命令（如擦除、写入），具体参见相关 Flash 的文档。

11 LocalIO 控制器

11.1 访问地址及引脚复用

表 11-1 LocalIO 地址空间分配

起始地址	结束名称	名称	说明
0x1D00_0000	0x1DFF_FFFF	存储空间	

对于 LocalIO 模块，使用时要注意将对应的引脚设置为相应的功能并且需要将 LIO 功能使能（IO 通用配置寄存器 1[35]设置为 1）

与 LocalIO 相关的引脚设置寄存器为 pad_sel[23]。

11.2 LocalIO 控制器功能概述

LocalIO 控制器提供了简单外设访问接口。它对外提供一个片选，具有可配置的数据位宽和访问延迟。其中 wait 参数指 liord 或 liowr 信号为低的周期数减一，读写时序可参考图 10-1，图 10-2。当数据位宽为 16 时，送出的地址由 CPU 物理地址右移一位得到。

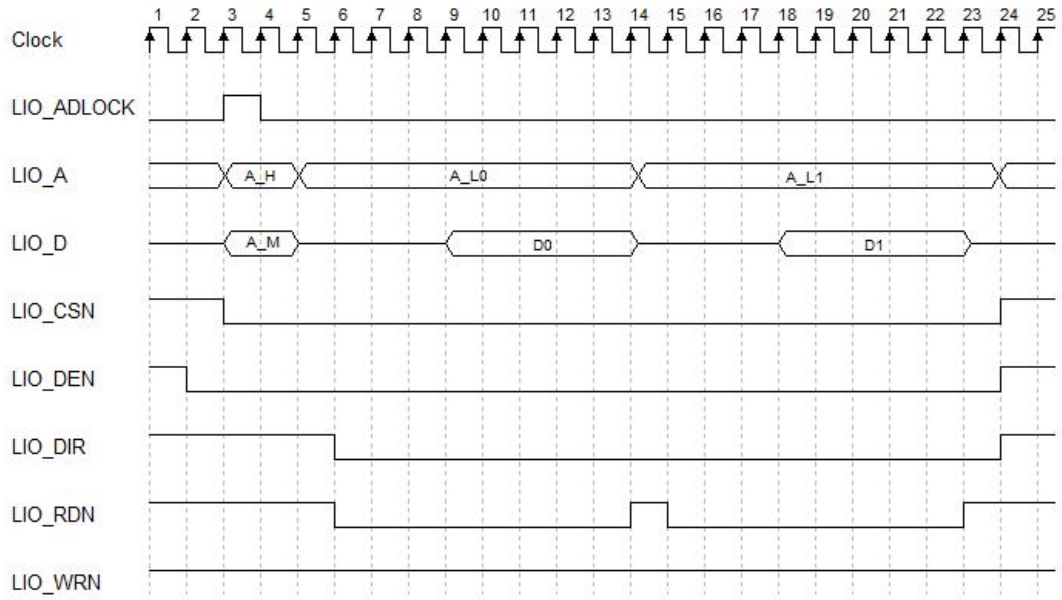


图 11-1 LocalIO 读时序

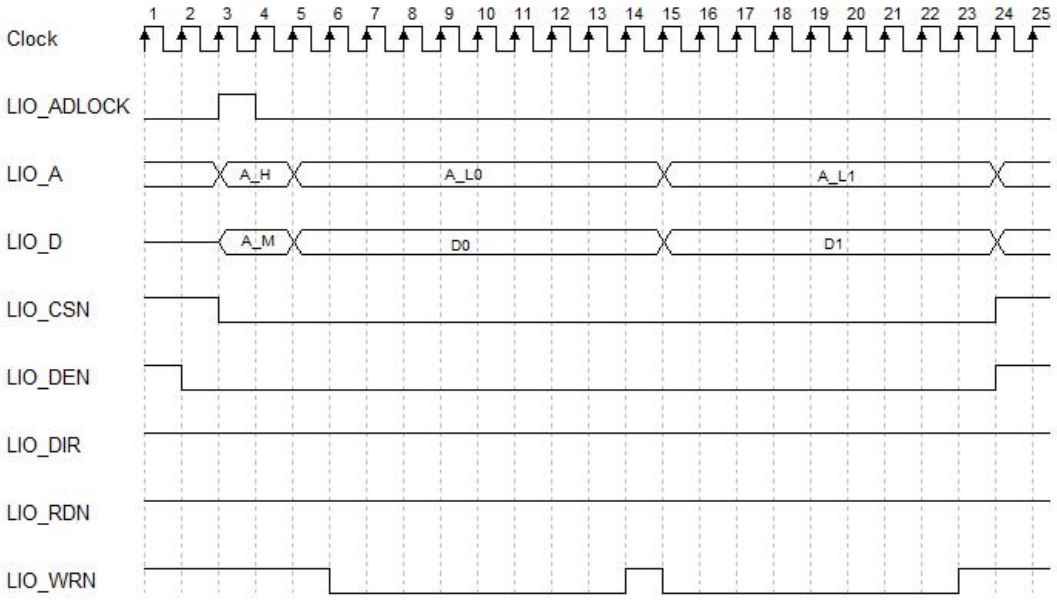


图 11-2 LocalIO 写时序

说明：

- 图中 clock 信号实际并不存在，只是为了方便时序描述
- A_H 代表地址 bit23-bit29(8 位模式) /bit24-bit30(16 位模式)
- A_M 代表地址 bit7-bit22(8 位模式)/bit8-bit23(16 位模式)
- A_L 代表低 7 位地址
- 在 big_mem 设置为 0 时，第 4 拍波形不存在，第 4 拍之后的波形向前推一拍
- LIO_WRN 和 LIO_RDN 低有效时间与 LIO clock_period_i 设置有关：

  LIO clock_period_i 设置为 1 时-低电平持续 8 拍；

  LIO clock_period_i 设置为 2 时-低电平持续 16 拍；

  LIO clock_period_i 设置为 3/0 时-低电平持续 32 拍；
- 一次 CS 有效期间可能出现多次读写操作，以上时序图仅作为一种示例。

12 AVS 控制器

12.1 访问地址及引脚复用

AVS 控制器寄存器物理地址基址为 0x1FE00160。

表 12-1 AVS 控制器地址空间分布

地址名称	地址范围	大小
AVS Register	0x1FE0_0160-0x1FE0_016F	16B

12.2 控制寄存器 (CSR)

中文名： 控制寄存器
寄存器位宽： [31：0]
偏移量： 0x0
复位值： 0x0

位域	位域名称	位宽	访问	描述
31	resyn	1	RW	1 表示在对设备进行通信前，先对设备进行重同步；0 则不进行重同步；默认为 0
30:29	mask_ack	19	RW	全写 1 表示 mask alert_ack
28	mask_i	1	RW	1 表示 mask alert_i
27	mask_s	1	RW	1 表示 mask alert_s
26	mask_c	1	RW	1 表示 mask alert_c
25	mask_a	1	RW	1 表示 mask alert_a
24	rx_ctrl	1	RW	控制器采样数据的方式，1 表示在 AVS 时钟的上沿开始采样数据，0 表示在 AVS 时钟的下沿开始采样数据；默认为 0
23:20	rx_delay	4	RW	延迟采样，1 表示延迟一个单位后采样数据；2 表示延迟两个单位后采样数据，以此类推；默认为 0
19:17	clk_div	3	RW	时钟分频系数：0x0，二分频；0x1，四分频；0x2，八分频；0x3，十六分频；0x4，三十二分频；0x5，六十四分频；默认二分频
16	Dmux	1	RW	1 表示接受设备返回的数据，0 反之。默认为 0
15:0	reserved	16	—	—

12.3 参数控制寄存器 (Mreg)

中文名： 参数控制寄存器
寄存器位宽： [31：0]
偏移量： 0x4
复位值： 0x0

位域	位域名称	位宽	访问	描述
31	TX_EN	1	RW	写 1 表示开始一次 AVS 通信
30:29	cmd	2	RW	0x0 表示写入设备并且生效；0x1 表示写入并保持，且不立即生效；0x2 保留；0x3 表示读。对于写，建议使 cmd 为 0
28	group	1	RW	0 为 AVS 协议指定的指令类别；1 表示制造厂商自定义的指令类别
27:24	cmd_type	4	RW	0x0 为电压调节命令（1LSB=1mv）；0x1 为压摆率命令，cmd_data 的高八位为上摆率，低八位为下摆率；0x2 为读电流命令（只读）；0x3 为读温度命令（只读）；0x4 为复位电压至默认值命令（只写）；0x5 为功率设置命令；0x6~0xd，保留；0xe 为设备状态清除命令；0xf 为读设备版本命令
23:20	rail_sel	4	RW	轨道选择，如电压轨。如果该值为 0xf，则表示选择全部的轨道
19:4	cmd_data	16	RW	写命令需要设置的值，如设置电压值大小；读命令，该值可设置为 0；状态清除命令需要将该值全部置为 1
3:0	reserved	4	—	—

12.4 数据寄存器（Sreg）

中文名：　　返回数据寄存器
寄存器位宽：　[31：0]
偏移量：　　0x8
复位值：　　0x00

位域	位域名称	位宽	访问	描述
31	busy	1	R	1 表示正在进行通信或控制器正在处理数据
30:29	slave_ack	2	RW	0x0 表示 CRC 校验 OK，并且读或写有效；0x1 表示 CRC 校验 OK，数据都 OK，但是设备资源不可用（busy 等）。如，超出电压范围的电压设置；0x2 表示 CRC 校验错误；0x3 表示 CRC 校验 OK，但是命令无效等
28	alert_i	1	RW	1 表示出现设备中断
27	alert_s	1	RW	1 表示出现设备状态响应
26	alert_c	1	RW	1 表示 CRC 校验错误
25	alert_a	1	RW	1 表示出现设备状态响应，包含设备自定义的状态响应
24:16	reserved	9	—	—
15:0	sdata	16	R	设备返回的数据。如，读电压命令，该值为设备返回的电压值

12.5 使用说明

以设置电压 1.8V 为例，CSR 寄存器设为 0x70000（下沿采样数据，16 分频）；然后将 Mreg 设为 0x80007080；等待 busy 为 0，最后读取 salve_ack 寄存器，若为 0，则表示设置成功。读电压需要设置 CSR 寄存器设为 0x70000；然后将 Mreg 设为 0xe0000000；等待 busy 为 0，最后读取 salve_ack 寄存器，若为 0，则表示读取成功，sdata 为设备返回的电压值。假如在一次通信中设备未发送 status frame，会出现 alert_s，alert_c， alert_a 一直被拉高的情况。

13 DDR4 控制器

13.1 地址空间

DDR4 内存控制器包括两个地址空间：内存控制器的寄存器配置空间和内存的存储空间。当配置参数 mc_disable_reg = 0 时，发给内存的访问都为寄存器配置访问；当配置参数 mc_disable_reg = 1 时，发给内存的访问都为内存的读写访问。

13.2 DDR4 SDRAM 控制器功能概述

龙芯 2K3000 处理器内部集成的内存控制器的设计遵守 DDR4/LPDDR4 SDRAM 行业标准（JESD79-4 和 JESD209-4）。在龙芯 2K3000 处理器中，所实现的所有内存读/写操作都遵守 JESD79-4 和 JESD209-4 的规定。

龙芯 2K3000 处理器内存控制器在具体选择使用不同内存芯片类型时，可以调整控制器参数设置进行支持。其中，支持的最大片选（CS_n）为 2，逻辑 RANK（CHIP ID）数为 8，行地址（ROW）数为 17，列地址（COL）数为 12，逻辑体选择（BANK）数为 2（DDR4）或 3（LPDDR4），逻辑体组（BANK Group）数为 2（仅 DDR4）。对于 LPDDR4 颗粒，不支持非 2 次幂容量的颗粒。对于 DDR4 支持单通道 64 位宽、32 位宽、16 位宽，对于 LPDDR4 支持双通道，每个通道 32 位宽。

13.3 DDR4 SDRAM 参数配置格式

内存控制器的参数列表

表 13-1 内存控制器软件可见参数列表

Offset	63:55	55:48	47:40	39:32	31:24	23:16	15:8	7:0
PHY								
0x0000							version (RD)	
0x0008		switch_byte48	x4_mode	ddr3_mode			capability (RD)	
0x0010							dram_init (RD)	init_start
0x0018								
0x0020							preamble2	rdfifo_valid
0x0028		rdfifo_empty (RD)				Overflow (RD)		
0x0030		dll_value (RD)	dll_init_done (RD)	dll_lock_mode	dll_bypass	dll_adj_cnt	dll_increment	dll_start_point
0x0038				dll_dbl_fix			dll_close_disable	dll_ck
0x0040								
0x0048							clken_ckca	
0x0050								
0x0058							clken_ds_0	
0x0060								
0x0068							clken_ds_1	
0x0070								
0x0078							clken_ds_2	
0x0080								
0x0088							clken_ds_3	

0x0090								
0x0098							clken_ds_4	ds4_en
0x00a0								
0x00a8							clken_ds_5	
0x00b0								
0x00b8							clken_ds_6	
0x00c0								
0x00c8							clken_ds_7	
0x00d0								
0x00d8							clken_ds_8	ecc_ds_en
0x00e0								vref_slice_enable
……								
0x0100					dll_1xdly_0	dll_1xgen_0	dll_wrdqs_0	dll_wrdq_0
0x0108		vref_sample_0	vrefclk_inv_0	dll_vref_0	vref_dly_0	dll_gate_0	dll_rddqs1_0	dll_rddqs0_0
0x0110	rdodt_ctrl_0	rdgate_len_0	rdgate_mode_0	rdgate_ctrl_0			dqs_oe_ctrl_0	dq_oe_ctrl_0
0x0118				rd_odt_len_add_0	dly_2x_0		redge_sel_0	rddqs_phase_0 (RD)
0x0120	w_dly_64_0[46:41]	w_dly_64_0[40:35]	w_dly_64_0[34:29]	w_dly_64_0[28:23]	w_dly_64_0[23:18]	w_dly_64_0[17:12]	w_dly_64_0[11:6]	w_dly_64_0[5:0]
0x0128		w_bdly0_0[15:12]	w_bdly0_0[11:8]	w_bdly0_0[7:4]	w_bdly0_0[3:0]	w_dq_en_0_8	w_dq_en_0_64	w_dly_64_0[52:47]
0x0130	w_bdly1_0[23:21]	w_bdly1_0[20:18]	w_bdly1_0[17:15]	w_bdly1_0[14:12]	w_bdly1_0[11:9]	w_bdly1_0[8:6]	w_bdly1_0[5:3]	w_bdly1_0[2:0]
0x0138						dq_dly_0_16		w_bdly1_0[26:24]
0x0140							rg_bdly_0[7:4]	rg_bdly_0[3:0]
0x0148								
0x0150	rdqsp_bdly_0[31:28]	rdqsp_bdly_0[27:24]	rdqsp_bdly_0[23:20]	rdqsp_bdly_0[19:16]	rdqsp_bdly_0[15:12]	rdqsp_bdly_0[11:8]	rdqsp_bdly_0[7:4]	rdqsp_bdly_0[3:0]
0x0158								rdqsp_bdly_0[35:32]
0x0160	rdqsn_bdly_0[31:28]	rdqsn_bdly_0[27:24]	rdqsn_bdly_0[23:20]	rdqsn_bdly_0[19:16]	rdqsn_bdly_0[15:12]	rdqsn_bdly_0[11:8]	rdqsn_bdly_0[7:4]	rdqsn_bdly_0[3:0]
0x0168								rdqsn_bdly_0[35:32]
0x0170	rdq_bdly_0[24:21]	rdq_bdly_0[20:18]	rdq_bdly_0[17:15]	rdq_bdly_0[14:12]	rdq_bdly_0[11:9]	rdq_bdly_0[8:6]	rdq_bdly_0[5:3]	rdq_bdly_0[2:0]
0x0178								rdq_bdly_0[27:26]
0x0180					dll_1xdly_1	dll_1xgen_1	dll_wrdqs_1	dll_wrdq_1
0x0188		vref_sample_1	vrefclk_inv_1	dll_vref_1	vref_dly_1	dll_gate_1	dll_rddqs1_1	dll_rddqs0_1
0x0190	rdodt_ctrl_1	rdgate_len_1	rdgate_mode_1	rdgate_ctrl_1			dqs_oe_ctrl_1	dq_oe_ctrl_1
0x0198				rd_odt_len_add_1	dly_2x_1		redge_sel_1	rddqs_phase_1 (RD)
0x01a0	w_dly_64_1[46:41]	w_dly_64_1[40:35]	w_dly_64_1[34:29]	w_dly_64_1[28:23]	w_dly_64_1[23:18]	w_dly_64_1[17:12]	w_dly_64_1[11:6]	w_dly_64_1[5:0]
0x01a8		w_bdly0_1[15:12]	w_bdly0_1[11:8]	w_bdly0_1[7:4]	w_bdly0_1[3:0]	w_dq_en_1_8	w_dq_en_1_64	w_dly_64_1[52:47]
0x01b0	w_bdly1_1[23:21]	w_bdly1_1[20:18]	w_bdly1_1[17:15]	w_bdly1_1[14:12]	w_bdly1_1[11:9]	w_bdly1_1[8:6]	w_bdly1_1[5:3]	w_bdly1_1[2:0]
0x01b8						dq_dly_1_16		w_bdly1_1[26:24]
0x01c0							rg_bdly_1[7:4]	rg_bdly_1[3:0]
0x01c8								
0x01d0	rdqsp_bdly_1[31:28]	rdqsp_bdly_1[27:24]	rdqsp_bdly_1[23:20]	rdqsp_bdly_1[19:16]	rdqsp_bdly_1[15:12]	rdqsp_bdly_1[11:8]	rdqsp_bdly_1[7:4]	rdqsp_bdly_1[3:0]
0x01d8								rdqsp_bdly_1[35:32]
0x01e0	rdqsn_bdly_1[31:28]	rdqsn_bdly_1[27:24]	rdqsn_bdly_1[23:20]	rdqsn_bdly_1[19:16]	rdqsn_bdly_1[15:12]	rdqsn_bdly_1[11:8]	rdqsn_bdly_1[7:4]	rdqsn_bdly_1[3:0]
0x01e8								rdqsn_bdly_1[35:32]
0x01f0	rdq_bdly_1[24:21]	rdq_bdly_1[20:18]	rdq_bdly_1[17:15]	rdq_bdly_1[14:12]	rdq_bdly_1[11:9]	rdq_bdly_1[8:6]	rdq_bdly_1[5:3]	rdq_bdly_1[2:0]
0x01f8								rdq_bdly_1[27:26]
0x0200					dll_1xdly_2	dll_1xgen_2	dll_wrdqs_2	dll_wrdq_2
0x0208		vref_sample_2	vrefclk_inv_2	dll_vref_2	vref_dly_2	dll_gate_2	dll_rddqs1_2	dll_rddqs0_2
0x0210	rdodt_ctrl_2	rdgate_len_2	rdgate_mode_2	rdgate_ctrl_2			dqs_oe_ctrl_2	dq_oe_ctrl_2

0x0218				rd_odt_len_add_2	dly_2x_2		redge_sel_2	rddqs_phase_2 (RD)
0x0220	w_dly_64_2[46:41]	w_dly_64_2[40:35]	w_dly_64_2[34:29]	w_dly_64_2[28:23]	w_dly_64_2[23:18]	w_dly_64_2[17:12]	w_dly_64_2[11:6]	w_dly_64_2[5:0]
0x0228		w_bdly0_2[15:12]	w_bdly0_2[11:8]	w_bdly0_2[7:4]	w_bdly0_2[3:0]	w_dq_en_2_8	w_dq_en_2_64	w_dly_64_2[52:47]
0x0230	w_bdly1_2[23:21]	w_bdly1_2[20:18]	w_bdly1_2[17:15]	w_bdly1_2[14:12]	w_bdly1_2[11:9]	w_bdly1_2[8:6]	w_bdly1_2[5:3]	w_bdly1_2[2:0]
0x0238						dq_dly_2_16		w_bdly1_2[26:24]
0x0240							rg_bdly_2[7:4]	rg_bdly_2[3:0]
0x0248								
0x0250	rdqsp_bdly_2[31:28]	rdqsp_bdly_2[27:24]	rdqsp_bdly_2[23:20]	rdqsp_bdly_2[19:16]	rdqsp_bdly_2[15:12]	rdqsp_bdly_2[11:8]	rdqsp_bdly_2[7:4]	rdqsp_bdly_2[3:0]
0x0258								rdqsp_bdly_2[35:32]
0x0260	rdqsn_bdly_2[31:28]	rdqsn_bdly_2[27:24]	rdqsn_bdly_2[23:20]	rdqsn_bdly_2[19:16]	rdqsn_bdly_2[15:12]	rdqsn_bdly_2[11:8]	rdqsn_bdly_2[7:4]	rdqsn_bdly_2[3:0]
0x0268								rdqsn_bdly_2[35:32]
0x0270	rdq_bdly_2[24:21]	rdq_bdly_2[20:18]	rdq_bdly_2[17:15]	rdq_bdly_2[14:12]	rdq_bdly_2[11:9]	rdq_bdly_2[8:6]	rdq_bdly_2[5:3]	rdq_bdly_2[2:0]
0x0278								rdq_bdly_2[27:26]
0x0280					dll_1xdly_3	dll_1xgen_3	dll_wrdqs_3	dll_wrdq_3
0x0288		vref_sample_3	vrefclk_inv_3	dll_vref_3	vref_dly_3	dll_gate_3	dll_rddqs1_3	dll_rddqs0_3
0x0290	rdodt_ctrl_3	rdgate_len_3	rdgate_mode_3	rdgate_ctrl_3			dqs_oe_ctrl_3	dq_oe_ctrl_3
0x0298				rd_odt_len_add_3	dly_2x_3		redge_sel_3	rddqs_phase_3 (RD)
0x02a0	w_dly_64_3[46:41]	w_dly_64_3[40:35]	w_dly_64_3[34:29]	w_dly_64_3[28:23]	w_dly_64_3[23:18]	w_dly_64_3[17:12]	w_dly_64_3[11:6]	w_dly_64_3[5:0]
0x02a8		w_bdly0_3[15:12]	w_bdly0_3[11:8]	w_bdly0_3[7:4]	w_bdly0_3[3:0]	w_dq_en_3_8	w_dq_en_3_64	w_dly_64_3[52:47]
0x02b0	w_bdly1_3[23:21]	w_bdly1_3[20:18]	w_bdly1_3[17:15]	w_bdly1_3[14:12]	w_bdly1_3[11:9]	w_bdly1_3[8:6]	w_bdly1_3[5:3]	w_bdly1_3[2:0]
0x02b8						dq_dly_3_16		w_bdly1_3[26:24]
0x02c0							rg_bdly_3[7:4]	rg_bdly_3[3:0]
0x02c8								
0x02d0	rdqsp_bdly_3[31:28]	rdqsp_bdly_3[27:24]	rdqsp_bdly_3[23:20]	rdqsp_bdly_3[19:16]	rdqsp_bdly_3[15:12]	rdqsp_bdly_3[11:8]	rdqsp_bdly_3[7:4]	rdqsp_bdly_3[3:0]
0x02d8								rdqsp_bdly_3[35:32]
0x02e0	rdqsn_bdly_3[31:28]	rdqsn_bdly_3[27:24]	rdqsn_bdly_3[23:20]	rdqsn_bdly_3[19:16]	rdqsn_bdly_3[15:12]	rdqsn_bdly_3[11:8]	rdqsn_bdly_3[7:4]	rdqsn_bdly_3[3:0]
0x02e8								rdqsn_bdly_3[35:32]
0x02f0	rdq_bdly_3[24:21]	rdq_bdly_3[20:18]	rdq_bdly_3[17:15]	rdq_bdly_3[14:12]	rdq_bdly_3[11:9]	rdq_bdly_3[8:6]	rdq_bdly_3[5:3]	rdq_bdly_3[2:0]
0x02f8								rdq_bdly_3[27:26]
0x0300					dll_1xdly_4	dll_1xgen_4	dll_wrdqs_4	dll_wrdq_4
0x0308		vref_sample_4	vrefclk_inv_4	dll_vref_4	vref_dly_4	dll_gate_4	dll_rddqs1_4	dll_rddqs0_4
0x0310	rdodt_ctrl_4	rdgate_len_4	rdgate_mode_4	rdgate_ctrl_4			dqs_oe_ctrl_4	dq_oe_ctrl_4
0x0318				rd_odt_len_add_4	dly_2x_4		redge_sel_4	rddqs_phase_4 (RD)
0x0320	w_dly_64_4[46:41]	w_dly_64_4[40:35]	w_dly_64_4[34:29]	w_dly_64_4[28:23]	w_dly_64_4[23:18]	w_dly_64_4[17:12]	w_dly_64_4[11:6]	w_dly_64_4[5:0]
0x0328		w_bdly0_4[15:12]	w_bdly0_4[11:8]	w_bdly0_4[7:4]	w_bdly0_4[3:0]	w_dq_en_4_8	w_dq_en_4_64	w_dly_64_4[52:47]
0x0330	w_bdly1_4[23:21]	w_bdly1_4[20:18]	w_bdly1_4[17:15]	w_bdly1_4[14:12]	w_bdly1_4[11:9]	w_bdly1_4[8:6]	w_bdly1_4[5:3]	w_bdly1_4[2:0]
0x0338						dq_dly_4_16		w_bdly1_4[26:24]
0x0340							rg_bdly_4[7:4]	rg_bdly_4[3:0]
0x0348								
0x0350	rdqsp_bdly_4[31:28]	rdqsp_bdly_4[27:24]	rdqsp_bdly_4[23:20]	rdqsp_bdly_4[19:16]	rdqsp_bdly_4[15:12]	rdqsp_bdly_4[11:8]	rdqsp_bdly_4[7:4]	rdqsp_bdly_4[3:0]
0x0358								rdqsp_bdly_4[35:32]
0x0360	rdqsn_bdly_4[31:28]	rdqsn_bdly_4[27:24]	rdqsn_bdly_4[23:20]	rdqsn_bdly_4[19:16]	rdqsn_bdly_4[15:12]	rdqsn_bdly_4[11:8]	rdqsn_bdly_4[7:4]	rdqsn_bdly_4[3:0]
0x0368								rdqsn_bdly_4[35:32]
0x0370	rdq_bdly_4[24:21]	rdq_bdly_4[20:18]	rdq_bdly_4[17:15]	rdq_bdly_4[14:12]	rdq_bdly_4[11:9]	rdq_bdly_4[8:6]	rdq_bdly_4[5:3]	rdq_bdly_4[2:0]
0x0378								rdq_bdly_4[27:26]
0x0380					dll_1xdly_5	dll_1xgen_5	dll_wrdqs_5	dll_wrdq_5
0x0388		vref_sample_5	vrefclk_inv_5	dll_vref_5	vref_dly_5	dll_gate_5	dll_rddqs1_5	dll_rddqs0_5

0x0390	rdodt_ctrl_5	rdgate_len_5	rdgate_mode_5	rdgate_ctrl_5			dqs_oe_ctrl_5	dq_oe_ctrl_5
0x0398				rd_odt_len_add_5	dly_2x_5		redge_sel_5	rddqs_phase_5 (RD)
0x03a0	w_dly_64_5[46:41]	w_dly_64_5[40:35]	w_dly_64_5[34:29]	w_dly_64_5[28:23]	w_dly_64_5[23:18]	w_dly_64_5[17:12]	w_dly_64_5[11:6]	w_dly_64_5[5:0]
0x03a8		w_bdly0_5[15:12]	w_bdly0_5[11:8]	w_bdly0_5[7:4]	w_bdly0_5[3:0]	w_dq_en_5_8	w_dq_en_5_64	w_dly_64_5[52:47]
0x03b0	w_bdly1_5[23:21]	w_bdly1_5[20:18]	w_bdly1_5[17:15]	w_bdly1_5[14:12]	w_bdly1_5[11:9]	w_bdly1_5[8:6]	w_bdly1_5[5:3]	w_bdly1_5[2:0]
0x03b8						dq_dly_5_16		w_bdly1_5[26:24]
0x03c0							rg_bdly_5[7:4]	rg_bdly_5[3:0]
0x03c8								
0x03d0	rdqsp_bdly_5[31:28]	rdqsp_bdly_5[27:24]	rdqsp_bdly_5[23:20]	rdqsp_bdly_5[19:16]	rdqsp_bdly_5[15:12]	rdqsp_bdly_5[11:8]	rdqsp_bdly_5[7:4]	rdqsp_bdly_5[3:0]
0x03d8								rdqsp_bdly_5[35:32]
0x03e0	rdqsn_bdly_5[31:28]	rdqsn_bdly_5[27:24]	rdqsn_bdly_5[23:20]	rdqsn_bdly_5[19:16]	rdqsn_bdly_5[15:12]	rdqsn_bdly_5[11:8]	rdqsn_bdly_5[7:4]	rdqsn_bdly_5[3:0]
0x03e8								rdqsn_bdly_5[35:32]
0x03f0	rdq_bdly_5[24:21]	rdq_bdly_5[20:18]	rdq_bdly_5[17:15]	rdq_bdly_5[14:12]	rdq_bdly_5[11:9]	rdq_bdly_5[8:6]	rdq_bdly_5[5:3]	rdq_bdly_5[2:0]
0x03f8								rdq_bdly_5[27:26]
0x0400					dll_1xdly_6	dll_1xgen_6	dll_wrdqs_6	dll_wrdq_6
0x0408		vref_sample_6	vrefclk_inv_6	dll_vref_6	vref_dly_6	dll_gate_6	dll_rddqs1_6	dll_rddqs0_6
0x0410	rdodt_ctrl_6	rdgate_len_6	rdgate_mode_6	rdgate_ctrl_6			dqs_oe_ctrl_6	dq_oe_ctrl_6
0x0418				rd_odt_len_add_6	dly_2x_6		redge_sel_6	rddqs_phase_6 (RD)
0x0420	w_dly_64_6[46:41]	w_dly_64_6[40:35]	w_dly_64_6[34:29]	w_dly_64_6[28:23]	w_dly_64_6[23:18]	w_dly_64_6[17:12]	w_dly_64_6[11:6]	w_dly_64_6[5:0]
0x0428		w_bdly0_6[15:12]	w_bdly0_6[11:8]	w_bdly0_6[7:4]	w_bdly0_6[3:0]	w_dq_en_6_8	w_dq_en_6_64	w_dly_64_6[52:47]
0x0430	w_bdly1_6[23:21]	w_bdly1_6[20:18]	w_bdly1_6[17:15]	w_bdly1_6[14:12]	w_bdly1_6[11:9]	w_bdly1_6[8:6]	w_bdly1_6[5:3]	w_bdly1_6[2:0]
0x0438						dq_dly_6_16		w_bdly1_6[26:24]
0x0440							rg_bdly_6[7:4]	rg_bdly_6[3:0]
0x0448								
0x0450	rdqsp_bdly_6[31:28]	rdqsp_bdly_6[27:24]	rdqsp_bdly_6[23:20]	rdqsp_bdly_6[19:16]	rdqsp_bdly_6[15:12]	rdqsp_bdly_6[11:8]	rdqsp_bdly_6[7:4]	rdqsp_bdly_6[3:0]
0x0458								rdqsp_bdly_6[35:32]
0x0460	rdqsn_bdly_6[31:28]	rdqsn_bdly_6[27:24]	rdqsn_bdly_6[23:20]	rdqsn_bdly_6[19:16]	rdqsn_bdly_6[15:12]	rdqsn_bdly_6[11:8]	rdqsn_bdly_6[7:4]	rdqsn_bdly_6[3:0]
0x0468								rdqsn_bdly_6[35:32]
0x0470	rdq_bdly_6[24:21]	rdq_bdly_6[20:18]	rdq_bdly_6[17:15]	rdq_bdly_6[14:12]	rdq_bdly_6[11:9]	rdq_bdly_6[8:6]	rdq_bdly_6[5:3]	rdq_bdly_6[2:0]
0x0478								rdq_bdly_6[27:26]
0x0480					dll_1xdly_7	dll_1xgen_7	dll_wrdqs_7	dll_wrdq_7
0x0488		vref_sample_7	vrefclk_inv_7	dll_vref_7	vref_dly_7	dll_gate_7	dll_rddqs1_7	dll_rddqs0_7
0x0490	rdodt_ctrl_7	rdgate_len_7	rdgate_mode_7	rdgate_ctrl_7			dqs_oe_ctrl_7	dq_oe_ctrl_7
0x0498				rd_odt_len_add_7	dly_2x_7		redge_sel_7	rddqs_phase_7 (RD)
0x04a0	w_dly_64_7[46:41]	w_dly_64_7[40:35]	w_dly_64_7[34:29]	w_dly_64_7[28:23]	w_dly_64_7[23:18]	w_dly_64_7[17:12]	w_dly_64_7[11:6]	w_dly_64_7[5:0]
0x04a8		w_bdly0_7[15:12]	w_bdly0_7[11:8]	w_bdly0_7[7:4]	w_bdly0_7[3:0]	w_dq_en_7_8	w_dq_en_7_64	w_dly_64_7[52:47]
0x04b0	w_bdly1_7[23:21]	w_bdly1_7[20:18]	w_bdly1_7[17:15]	w_bdly1_7[14:12]	w_bdly1_7[11:9]	w_bdly1_7[8:6]	w_bdly1_7[5:3]	w_bdly1_7[2:0]
0x04b8						dq_dly_7_16		w_bdly1_7[26:24]
0x04c0							rg_bdly_7[7:4]	rg_bdly_7[3:0]
0x04c8								
0x04d0	rdqsp_bdly_7[31:28]	rdqsp_bdly_7[27:24]	rdqsp_bdly_7[23:20]	rdqsp_bdly_7[19:16]	rdqsp_bdly_7[15:12]	rdqsp_bdly_7[11:8]	rdqsp_bdly_7[7:4]	rdqsp_bdly_7[3:0]
0x04d8								rdqsp_bdly_7[35:32]
0x04e0	rdqsn_bdly_7[31:28]	rdqsn_bdly_7[27:24]	rdqsn_bdly_7[23:20]	rdqsn_bdly_7[19:16]	rdqsn_bdly_7[15:12]	rdqsn_bdly_7[11:8]	rdqsn_bdly_7[7:4]	rdqsn_bdly_7[3:0]
0x04e8								rdqsn_bdly_7[35:32]
0x04f0	rdq_bdly_7[24:21]	rdq_bdly_7[20:18]	rdq_bdly_7[17:15]	rdq_bdly_7[14:12]	rdq_bdly_7[11:9]	rdq_bdly_7[8:6]	rdq_bdly_7[5:3]	rdq_bdly_7[2:0]
0x04f8								rdq_bdly_7[27:26]
0x0500					dll_1xdly_8	dll_1xgen_8	dll_wrdqs_8	dll_wrdq_8

0x0508		vref_sample_8	vrefclk_inv_8	dll_vref_8	vref_dly_8	dll_gate_8	dll_rddqs1_8	dll_rddqs0_8
0x0510	rdodt_ctrl_8	rdgate_len_8	rdgate_mode_8	rdgate_ctrl_8			dqs_oe_ctrl_8	dq_oe_ctrl_8
0x0518				rd_odt_len_add_8	dly_2x_8		redge_sel_8	rddqs_phase_8 (RD)
0x0520	w_dly_64_8[46:41]	w_dly_64_8[40:35]	w_dly_64_8[34:29]	w_dly_64_8[28:23]	w_dly_64_8[23:18]	w_dly_64_8[17:12]	w_dly_64_8[11:6]	w_dly_64_8[5:0]
0x0528		w_bdly0_8[15:12]	w_bdly0_8[11:8]	w_bdly0_8[7:4]	w_bdly0_8[3:0]	w_dq_en_8_8	w_dq_en_8_64	w_dly_64_8[52:47]
0x0530	w_bdly1_8[23:21]	w_bdly1_8[20:18]	w_bdly1_8[17:15]	w_bdly1_8[14:12]	w_bdly1_8[11:9]	w_bdly1_8[8:6]	w_bdly1_8[5:3]	w_bdly1_8[2:0]
0x0538						dq_dly_8_16		w_bdly1_8[26:24]
0x0540							rg_bdly_8[7:4]	rg_bdly_8[3:0]
0x0548								
0x0550	rdqsp_bdly_8[31:28]	rdqsp_bdly_8[27:24]	rdqsp_bdly_8[23:20]	rdqsp_bdly_8[19:16]	rdqsp_bdly_8[15:12]	rdqsp_bdly_8[11:8]	rdqsp_bdly_8[7:4]	rdqsp_bdly_8[3:0]
0x0558								rdqsp_bdly_8[35:32]
0x0560	rdqsn_bdly_8[31:28]	rdqsn_bdly_8[27:24]	rdqsn_bdly_8[23:20]	rdqsn_bdly_8[19:16]	rdqsn_bdly_8[15:12]	rdqsn_bdly_8[11:8]	rdqsn_bdly_8[7:4]	rdqsn_bdly_8[3:0]
0x0568								rdqsn_bdly_8[35:32]
0x0570	rdq_bdly_8[24:21]	rdq_bdly_8[20:18]	rdq_bdly_8[17:15]	rdq_bdly_8[14:12]	rdq_bdly_8[11:9]	rdq_bdly_8[8:6]	rdq_bdly_8[5:3]	rdq_bdly_8[2:0]
0x0578								rdq_bdly_8[27:26]
……								
0x0700	ca_train_map	wrdqs_disable	ca_invert	ca_training_mode	leveling_cs	tLVL_DELAY	leveling_req (WR)	leveling_mode
0x0708						mpr_loc	leveling_done (RD)	leveling_ready (RD)
0x0710	leveling_resp_7	leveling_resp_6	leveling_resp_5	leveling_resp_4	leveling_resp_3	leveling_resp_2	leveling_resp_1	leveling_resp_0
0x0718								leveling_resp_8
0x0720								
……								
0x0800	dfe_ctrl_ds	pad_ctrl_ds				pad_ctrl_ck		
0x0808		pad_reset_po	pad_oplen_ca	pad_opdly_ca		pad_ctrl_ca		
0x0810	vref_ctrl_ds_3		vref_ctrl_ds_2		vref_ctrl_ds_1		vref_ctrl_ds_0	
0x0818	vref_ctrl_ds_7		vref_ctrl_ds_6		vref_ctrl_ds_5		vref_ctrl_ds_4	
0x0820							vref_ctrl_ds_8	
0x0828								
0x0830			pad_comp_o (RD)				pad_comp_i	
0x0838								
0x0840	pad_ctrl_ds_3		pad_ctrl_ds_2		pad_ctrl_ds_1		pad_ctrl_ds_0	
0x0848	pad_ctrl_ds_7		pad_ctrl_ds_6		pad_ctrl_ds_5		pad_ctrl_ds_4	
0x0850							pad_ctrl_ds_8	
……								
0x0900		rdedge_soft		rd_phase (RD)			clk_inv	
0x0908							rdedge_inv	
0x918	pm_lp_osc_dll_ds3		pm_lp_osc_dll_ds2		pm_lp_osc_dll_ds1		pm_lp_osc_dll_ds0	
0x920	pm_lp_osc_dll_ds7		pm_lp_osc_dll_ds6		pm_lp_osc_dll_ds5		pm_lp_osc_dll_ds4	
……								
0xa00	pm_lp_osc_dll_ds0_3		pm_lp_osc_dll_ds0_2		pm_lp_osc_dll_ds0_1		pm_lp_osc_dll_ds0_0	
0xa08	pm_lp_osc_dll_ds0_7		pm_lp_osc_dll_ds0_6		pm_lp_osc_dll_ds0_5		pm_lp_osc_dll_ds0_4	
0xa10	pm_lp_osc_dll_ds0_11		pm_lp_osc_dll_ds0_10		pm_lp_osc_dll_ds0_9		pm_lp_osc_dll_ds0_8	
0xa18	pm_lp_osc_dll_ds0_15		pm_lp_osc_dll_ds0_14		pm_lp_osc_dll_ds0_13		pm_lp_osc_dll_ds0_12	
0xa20	pm_lp_osc_count_ds0_3		pm_lp_osc_count_ds0_2		pm_lp_osc_count_ds0_1		pm_lp_osc_count_ds0_0	
0xa28	pm_lp_osc_count_ds0_7		pm_lp_osc_count_ds0_6		pm_lp_osc_count_ds0_5		pm_lp_osc_count_ds0_4	
0xa30	pm_lp_osc_count_ds0_11		pm_lp_osc_count_ds0_10		pm_lp_osc_count_ds0_9		pm_lp_osc_count_ds0_8	
0xa38	pm_lp_osc_count_ds0_15		pm_lp_osc_count_ds0_14		pm_lp_osc_count_ds0_13		pm_lp_osc_count_ds0_12	

0xa40	pm_lp_osc_dll_ds1_3	pm_lp_osc_dll_ds1_2	pm_lp_osc_dll_ds1_1	pm_lp_osc_dll_ds1_0
0xa48	pm_lp_osc_dll_ds1_7	pm_lp_osc_dll_ds1_6	pm_lp_osc_dll_ds1_5	pm_lp_osc_dll_ds1_4
0xa50	pm_lp_osc_dll_ds1_11	pm_lp_osc_dll_ds1_10	pm_lp_osc_dll_ds1_9	pm_lp_osc_dll_ds1_8
0xa58	pm_lp_osc_dll_ds1_15	pm_lp_osc_dll_ds1_14	pm_lp_osc_dll_ds1_13	pm_lp_osc_dll_ds1_12
0xa60	pm_lp_osc_count_ds1_3	pm_lp_osc_count_ds1_2	pm_lp_osc_count_ds1_1	pm_lp_osc_count_ds1_0
0xa68	pm_lp_osc_count_ds1_7	pm_lp_osc_count_ds1_6	pm_lp_osc_count_ds1_5	pm_lp_osc_count_ds1_4
0xa70	pm_lp_osc_count_ds1_11	pm_lp_osc_count_ds1_10	pm_lp_osc_count_ds1_9	pm_lp_osc_count_ds1_8
0xa78	pm_lp_osc_count_ds1_15	pm_lp_osc_count_ds1_14	pm_lp_osc_count_ds1_13	pm_lp_osc_count_ds1_12
0xa80	pm_lp_osc_dll_ds2_3	pm_lp_osc_dll_ds2_2	pm_lp_osc_dll_ds2_1	pm_lp_osc_dll_ds2_0
0xa88	pm_lp_osc_dll_ds2_7	pm_lp_osc_dll_ds2_6	pm_lp_osc_dll_ds2_5	pm_lp_osc_dll_ds2_4
0xa90	pm_lp_osc_dll_ds2_11	pm_lp_osc_dll_ds2_10	pm_lp_osc_dll_ds2_9	pm_lp_osc_dll_ds2_8
0xa98	pm_lp_osc_dll_ds2_15	pm_lp_osc_dll_ds2_14	pm_lp_osc_dll_ds2_13	pm_lp_osc_dll_ds2_12
0xaa0	pm_lp_osc_count_ds2_3	pm_lp_osc_count_ds2_2	pm_lp_osc_count_ds2_1	pm_lp_osc_count_ds2_0
0xaa8	pm_lp_osc_count_ds2_7	pm_lp_osc_count_ds2_6	pm_lp_osc_count_ds2_5	pm_lp_osc_count_ds2_4
0xab0	pm_lp_osc_count_ds2_11	pm_lp_osc_count_ds2_10	pm_lp_osc_count_ds2_9	pm_lp_osc_count_ds2_8
0xab8	pm_lp_osc_count_ds2_15	pm_lp_osc_count_ds2_14	pm_lp_osc_count_ds2_13	pm_lp_osc_count_ds2_12
0xac0	pm_lp_osc_dll_ds3_3	pm_lp_osc_dll_ds3_2	pm_lp_osc_dll_ds3_1	pm_lp_osc_dll_ds3_0
0xac8	pm_lp_osc_dll_ds3_7	pm_lp_osc_dll_ds3_6	pm_lp_osc_dll_ds3_5	pm_lp_osc_dll_ds3_4
0xad0	pm_lp_osc_dll_ds3_11	pm_lp_osc_dll_ds3_10	pm_lp_osc_dll_ds3_9	pm_lp_osc_dll_ds3_8
0xad8	pm_lp_osc_dll_ds3_15	pm_lp_osc_dll_ds3_14	pm_lp_osc_dll_ds3_13	pm_lp_osc_dll_ds3_12
0xae0	pm_lp_osc_count_ds3_3	pm_lp_osc_count_ds3_2	pm_lp_osc_count_ds3_1	pm_lp_osc_count_ds3_0
0xae8	pm_lp_osc_count_ds3_7	pm_lp_osc_count_ds3_6	pm_lp_osc_count_ds3_5	pm_lp_osc_count_ds3_4
0xaf0	pm_lp_osc_count_ds3_11	pm_lp_osc_count_ds3_10	pm_lp_osc_count_ds3_9	pm_lp_osc_count_ds3_8
0xaf8	pm_lp_osc_count_ds3_15	pm_lp_osc_count_ds3_14	pm_lp_osc_count_ds3_13	pm_lp_osc_count_ds3_12
0xb00	pm_lp_osc_dll_ds4_3	pm_lp_osc_dll_ds4_2	pm_lp_osc_dll_ds4_1	pm_lp_osc_dll_ds4_0
0xb08	pm_lp_osc_dll_ds4_7	pm_lp_osc_dll_ds4_6	pm_lp_osc_dll_ds4_5	pm_lp_osc_dll_ds4_4
0xb10	pm_lp_osc_dll_ds4_11	pm_lp_osc_dll_ds4_10	pm_lp_osc_dll_ds4_9	pm_lp_osc_dll_ds4_8
0xb18	pm_lp_osc_dll_ds4_15	pm_lp_osc_dll_ds4_14	pm_lp_osc_dll_ds4_13	pm_lp_osc_dll_ds4_12
0xb20	pm_lp_osc_count_ds4_3	pm_lp_osc_count_ds4_2	pm_lp_osc_count_ds4_1	pm_lp_osc_count_ds4_0
0xb28	pm_lp_osc_count_ds4_7	pm_lp_osc_count_ds4_6	pm_lp_osc_count_ds4_5	pm_lp_osc_count_ds4_4
0xb30	pm_lp_osc_count_ds4_11	pm_lp_osc_count_ds4_10	pm_lp_osc_count_ds4_9	pm_lp_osc_count_ds4_8
0xb38	pm_lp_osc_count_ds4_15	pm_lp_osc_count_ds4_14	pm_lp_osc_count_ds4_13	pm_lp_osc_count_ds4_12
0xb40	pm_lp_osc_dll_ds5_3	pm_lp_osc_dll_ds5_2	pm_lp_osc_dll_ds5_1	pm_lp_osc_dll_ds5_0
0xb48	pm_lp_osc_dll_ds5_7	pm_lp_osc_dll_ds5_6	pm_lp_osc_dll_ds5_5	pm_lp_osc_dll_ds5_4
0xb50	pm_lp_osc_dll_ds5_11	pm_lp_osc_dll_ds5_10	pm_lp_osc_dll_ds5_9	pm_lp_osc_dll_ds5_8
0xb58	pm_lp_osc_dll_ds5_15	pm_lp_osc_dll_ds5_14	pm_lp_osc_dll_ds5_13	pm_lp_osc_dll_ds5_12
0xb60	pm_lp_osc_count_ds5_3	pm_lp_osc_count_ds5_2	pm_lp_osc_count_ds5_1	pm_lp_osc_count_ds5_0
0xb68	pm_lp_osc_count_ds5_7	pm_lp_osc_count_ds5_6	pm_lp_osc_count_ds5_5	pm_lp_osc_count_ds5_4
0xb70	pm_lp_osc_count_ds5_11	pm_lp_osc_count_ds5_10	pm_lp_osc_count_ds5_9	pm_lp_osc_count_ds5_8
0xb78	pm_lp_osc_count_ds5_15	pm_lp_osc_count_ds5_14	pm_lp_osc_count_ds5_13	pm_lp_osc_count_ds5_12
0xb80	pm_lp_osc_dll_ds6_3	pm_lp_osc_dll_ds6_2	pm_lp_osc_dll_ds6_1	pm_lp_osc_dll_ds6_0
0xb88	pm_lp_osc_dll_ds6_7	pm_lp_osc_dll_ds6_6	pm_lp_osc_dll_ds6_5	pm_lp_osc_dll_ds6_4
0xb90	pm_lp_osc_dll_ds6_11	pm_lp_osc_dll_ds6_10	pm_lp_osc_dll_ds6_9	pm_lp_osc_dll_ds6_8
0xb98	pm_lp_osc_dll_ds6_15	pm_lp_osc_dll_ds6_14	pm_lp_osc_dll_ds6_13	pm_lp_osc_dll_ds6_12
0xba0	pm_lp_osc_count_ds6_3	pm_lp_osc_count_ds6_2	pm_lp_osc_count_ds6_1	pm_lp_osc_count_ds6_0
0xba8	pm_lp_osc_count_ds6_7	pm_lp_osc_count_ds6_6	pm_lp_osc_count_ds6_5	pm_lp_osc_count_ds6_4
0xbb0	pm_lp_osc_count_ds6_11	pm_lp_osc_count_ds6_10	pm_lp_osc_count_ds6_9	pm_lp_osc_count_ds6_8

0xbb8	pm_lp_osc_count_ds6_15		pm_lp_osc_count_ds6_14		pm_lp_osc_count_ds6_13		pm_lp_osc_count_ds6_12	
0xbc0	pm_lp_osc_dll_ds7_3		pm_lp_osc_dll_ds7_2		pm_lp_osc_dll_ds7_1		pm_lp_osc_dll_ds7_0	
0xbc8	pm_lp_osc_dll_ds7_7		pm_lp_osc_dll_ds7_6		pm_lp_osc_dll_ds7_5		pm_lp_osc_dll_ds7_4	
0xbd0	pm_lp_osc_dll_ds7_11		pm_lp_osc_dll_ds7_10		pm_lp_osc_dll_ds7_9		pm_lp_osc_dll_ds7_8	
0xbd8	pm_lp_osc_dll_ds7_15		pm_lp_osc_dll_ds7_14		pm_lp_osc_dll_ds7_13		pm_lp_osc_dll_ds7_12	
0xbe0	pm_lp_osc_count_ds7_3		pm_lp_osc_count_ds7_2		pm_lp_osc_count_ds7_1		pm_lp_osc_count_ds7_0	
0xbe8	pm_lp_osc_count_ds7_7		pm_lp_osc_count_ds7_6		pm_lp_osc_count_ds7_5		pm_lp_osc_count_ds7_4	
0xbf0	pm_lp_osc_count_ds7_11		pm_lp_osc_count_ds7_10		pm_lp_osc_count_ds7_9		pm_lp_osc_count_ds7_8	
0xbf8	pm_lp_osc_count_ds7_15		pm_lp_osc_count_ds7_14		pm_lp_osc_count_ds7_13		pm_lp_osc_count_ds7_12	
CTL								
0x1000	tMRD	tRP	tWLDQSEN	tMOD	tXPR		tCKE	tRESET
0x1008		tXSFAST	tCKSRX	tCKSRE	tCKOFF	tCKEV	tSTAB	tODTL
0x1010	tREFretention				tRFC		tREF	
0x1018	tCKESR	tXSRD	tXS		tRFC_dlr			tREF_IDLE
0x1020					tRDPDEN	tCPDED	tXPDLL	tXP
0x1028					tZQperiod	tZQCL	tZQCS	tZQ_CMD
……								
0x1040	tRCD	tRRD_S_slr	tRRD_L_slr	tRRD_dlr				tRAS_min
0x1048				tRTP	tWR_CRC_DM	tWR	tFAW_slr	tFAW
0x1050	tWTR_S_CRC_DM	tWTR_L_CRC_DM	tWTR_S	tWTR		tCCD_dlr	tCCD_S_slr	tCCD_L_slr
0x1058								
0x1060			tPHY_WRLAT	tWL		tRDDATA	tPHY_RDLAT	tRL
0x1068				tCAL				tPL
0x1070			tW2P_sameba	tW2W_sameba	tW2R_sameba	tR2P_sameba	tR2W_sameba	tR2R_sameba
0x1078			tW2P_samebg	tW2W_samebg	tW2R_samebg	tR2P_samebg	tR2W_samebg	tR2R_samebg
0x1080			tW2P_samec	tW2W_samec	tW2R_samec	tR2P_samec	tR2W_samec	tR2R_samec
0x1088								
0x1090			tW2P_samecs	tW2W_samecs	tW2R_samecs	tR2P_samecs	tR2W_samecs	tR2R_samecs
0x1098				tW2W_diffcs	tW2R_diffcs		tR2W_diffcs	tR2R_diffcs
……								
0x1100	mirror_dimm_cs_map		cs_ref	cs_resync	cs_zqcl	cs_zq	cs_mrs	cs_enable
0x1108	cke_map				cs_map			
0x1110	mirror_cs_enable	mirror_cs_sta (RO)	mirro_dimm_ctrl	cs2cid				cid_map
0x1118						rcd_cs	two_dimm	rdimm
0x1120	mrs_done (RD)	mrs_req (WR)	pre_all_done (RD)	pre_all_req (WR)	cmd_cmd	status_cmd (RD)	cmd_req (WR)	command_mode
0x1128	cmd_cke	cmd_a			cmd_ba	cmd_bg	cmd_c	cmd_cs
0x1130	cs_mrs_sequence				lpddr4_rddq_cal	mpr_rd_done (RO)	cmd_mpr_rd	cmd_pda
0x1138						cmd_dq0		
0x1140	mr_3_cs_0		mr_2_cs_0		mr_1_cs_0		mr_0_cs_0	
0x1148	mr_3_cs_1		mr_2_cs_1		mr_1_cs_1		mr_0_cs_1	
0x1150	mr_3_cs_2		mr_2_cs_2		mr_1_cs_2		mr_0_cs_2	
0x1158	mr_3_cs_3		mr_2_cs_3		mr_1_cs_3		mr_0_cs_3	
0x1160	mr_3_cs_4		mr_2_cs_4		mr_1_cs_4		mr_0_cs_4	
0x1168	mr_3_cs_5		mr_2_cs_5		mr_1_cs_5		mr_0_cs_5	
0x1170	mr_3_cs_6		mr_2_cs_6		mr_1_cs_6		mr_0_cs_6	
0x1178	mr_3_cs_7		mr_2_cs_7		mr_1_cs_7		mr_0_cs_7	
0x1180	mr_3_cs_0_ddr4		mr_2_cs_0_ddr4		mr_1_cs_0_ddr4		mr_0_cs_0_ddr4	

0x1188		mr_13_cs_0_lpddr4	mr_6_cs_0_ddr4		mr_5_cs_0_ddr4		mr_4_cs_0_ddr4	
0x1190	mr_3_cs_1_ddr4		mr_2_cs_1_ddr4		mr_1_cs_1_ddr4		mr_0_cs_1_ddr4	
0x1198		mr_13_cs_1_lpddr4	mr_6_cs_1_ddr4		mr_5_cs_1_ddr4		mr_4_cs_1_ddr4	
0x11a0	mr_3_cs_2_ddr4		mr_2_cs_2_ddr4		mr_1_cs_2_ddr4		mr_0_cs_2_ddr4	
0x11a8			mr_6_cs_2_ddr4		mr_5_cs_2_ddr4		mr_4_cs_2_ddr4	
0x11b0	mr_3_cs_3_ddr4		mr_2_cs_3_ddr4		mr_1_cs_3_ddr4		mr_0_cs_3_ddr4	
0x11b8			mr_6_cs_3_ddr4		mr_5_cs_3_ddr4		mr_4_cs_3_ddr4	
0x11c0	mr_3_cs_4_ddr4		mr_2_cs_4_ddr4		mr_1_cs_4_ddr4		mr_0_cs_4_ddr4	
0x11c8		ch1_mr_13_cs_0_lpddr4	mr_6_cs_4_ddr4		mr_5_cs_4_ddr4		mr_4_cs_4_ddr4	
0x11d0	mr_3_cs_5_ddr4		mr_2_cs_5_ddr4		mr_1_cs_5_ddr4		mr_0_cs_5_ddr4	
0x11d8		ch1_mr_13_cs_1_lpddr4	mr_6_cs_5_ddr4		mr_5_cs_5_ddr4		mr_4_cs_5_ddr4	
0x11e0	mr_3_cs_6_ddr4		mr_2_cs_6_ddr4		mr_1_cs_6_ddr4		mr_0_cs_6_ddr4	
0x11e8			mr_6_cs_6_ddr4		mr_5_cs_6_ddr4		mr_4_cs_6_ddr4	
0x11f0	mr_3_cs_7_ddr4		mr_2_cs_7_ddr4		mr_1_cs_7_ddr4		mr_0_cs_7_ddr4	
0x11f8			mr_6_cs_7_ddr4		mr_5_cs_7_ddr4		mr_4_cs_7_ddr4	
0x1200	pm_lvl_channel_sel	pm_two_channel	nc16_map	nc	channel_width	ba_xor_row_offset	addr_new	cs_place
0x1208		pm_nc8_map	ba_ch_xor	ch_offset	bg_xor_row	bg_xor_row_offset		addr_mirror
0x1210	addr_base_1				addr_base_0			
0x1218				stb_arprior_cfg				
0x1220	addr_mask_1				addr_mask_0			
0x1228	prior_age7		prior_age6		prior_age5		prior_age4	
0x1230			cs_diff	c_diff	bg_diff	ba_diff	row_diff	col_diff
0x1238				CF_confbus_timeout				
0x1240	WRQthreshold	tRDQidle	wr_pkc_num	rwq_arb	retry	no_dead_inorder	placement_en	stb_en/pbuf
0x1248		WRQ_halfth	WRQthreshold_L	WRQ_near_full	rankarb_age		rankarb_age_en	tRWGNTidle
0x1250	pref_low		tRDQwait	RDQthreshold_L	RDQthreshold	rd_pkg_num	rfifo_age	
0x1258	prior_age3		prior_age2		prior_age1		prior_age0	
0x1260	retry_cnt (RD)					rbuffer_max (RD)	rdfifo_depth	stat_en
0x1268	rdage_low		rdage_up		pref_near_full	hot_en	hot_minus	hot_count
0x1270	stb_page_close_timeout		stb_dbg_cnt_clear	stb_dbg_cnt_sel	stb_dbg_cnt			
0x1278	stb_hot_16	stb_hot_4/8	stb_active_16	stb_active_4/8	stb_distance	stb_context_grant	stb_cold_time_refresh	strdrec_soft_reqb_cold_time_init
0x1280	aw_512_align		rd_before_wr	ecc_enable		int_vector (RD)	int_trigger (RD)	int_enable
0x1288						reread_en	rdrec_interval	rdrec_soft_req
0x1290						int_cnt_fatal (RD)	int_cnt_err (RD)	int_cnt
0x1298	ecc_cnt_cs_7 (RD)	ecc_cnt_cs_6 (RD)	ecc_cnt_cs_5 (RD)	ecc_cnt_cs_4 (RD)	ecc_cnt_cs_3 (RD)	ecc_cnt_cs_2 (RD)	ecc_cnt_cs_1 (RD)	ecc_cnt_cs_0 (RD)
0x12a0	ecc_data_dir (RD)	ecc_code_dir (RD)		ecc_valid	reread_ecc (RD)	ecc_64[23:16] (RD)		ecc_code_64 (RD)
0x12a8	ecc_addr (RD)							
0x12b0	ecc_data[63:0] (RD)							
0x12b8	ecc_data[127:64] (RD)							
0x12c0	ecc_data[191:128] (RD)							
0x12c8	ecc_data[255:192] (RD)							
0x12d0	ecc_err_set[63:0]							
0c12d8								ecc_err_set[71:64]
……								
0x1300							ref_num	ref_sch_en
0x1308				pm_s3_mode	Status_sref_ch1 (RD)	srefresh_req_ch1	Status_sref_ch0 (RD)	srefresh_req_ch0
……								

0x1340	hardware_pd_7	hardware_pd_6	hardware_pd_5	hardware_pd_4	hardware_pd_3	hardware_pd_2	hardware_pd_1	hardware_pd_0
0x1348	power_sta_7 (RD)	power_sta_6 (RD)	power_sta_5 (RD)	power_sta_4 (RD)	power_sta_3 (RD)	power_sta_2 (RD)	power_sta_1 (RD)	power_sta_0 (RD)
0x1350	selfref_age		slowpd_age		fastpd_age		active_age	
0x1358				power_up				Age_step
0x1360	tCONF_IDLE				tLPMC_IDLE			
0x1368				half_frq_func_reg_sel	half_frq_auto_intev	half_frq_auto_rw	half_frq_reg_rw	sch_lp_en
0x1370							F0RC0A	F0RC3x
0x1378								F0RC09
0x1380								zq_overlap
0x1388								zq_stat_en
0x1390	zq_cnt_1 (RD)				zq_cnt_0 (RD)			
0x1398	zq_cnt_3 (RD)				zq_cnt_2 (RD)			
0x13a0	zq_cnt_5 (RD)				zq_cnt_4 (RD)			
0x13a8	zq_cnt_6 (RD)				zq_cnt_6 (RD)			
……								
0x13c0					odt_wr_cs_map			
0x13c8							odt_wr_length	odt_wr_delay
0x13d0					odt_rd_cs_map			
0x13d8							odt_rd_length	odt_rd_delay
……								
0x1400		resync_done	resync_req	tRESYNC_length	tRESYNC_delay	tRESYNC_shift	tRESYNC_max	tRESYNC_min
……								
0x1440					pre_predict		tm_cmdq_num	burst_length
0x1448								ca_timing
0x1450						wr/rd_dbi_en	ca_par_en	crc_en
0x1458							tCA_PAR	tWR_CRC
0x1460	bit_map_7	bit_map_6	bit_map_5	bit_map_6	bit_map_3	bit_map_2	bit_map_1	bit_map_0
0x1468	bit_map_15	bit_map_14	bit_map_13	bit_map_12	bit_map_11	bit_map_10	bit_map_9	bit_map_8
0x1470							bit_map_17	bit_map_16
0x1478								bitmap_mirror
0x1480				alertn_misc (RD)			alertn_cnt	alertn_clr
0x1488	alertn_addr (RD)							
……								
0x1498	mpr_train_data_ecc_ch0							
0x14a0	mpr_dm							
……								
0x14c0	mpr_train_data_0							
0x14c8	mpr_train_data_1							
0x14d0	mpr_train_data_2							
0x14d8	mpr_train_data_3							
0x14e0	mpr_train_data_4							
0x14e8	mpr_train_data_5							
0x14f0	mpr_train_data_6							
0x14f8	mpr_train_data_7							
0x1500	win0_base							
0x1508	win1_base							
0x1510	win2_base							

0x1518	win3_base							
0x1520	win4_base							
0x1528	win5_base							
0x1530	win6_base							
0x1538	win7_base							
……								
0x1580	win0_mask							
0x1588	win1_mask							
0x1590	win2_mask							
0x1598	win3_mask							
0x15a0	win4_mask							
0x15a8	win5_mask							
0x15b0	win6_mask							
0x15b8	win7_mask							
……								
0x1600	win0_mmap							
0x1608	win1_mmap							
0x1610	win2_mmap							
0x1618	win3_mmap							
0x1620	win4_mmap							
0x1628	win5_mmap							
0x1630	win6_mmap							
0x1638	win7_mmap							
……								
0x1680	write_train_data[63:0]							
0x1688	write_train_data[127:64]							
0x1690	write_train_data[191:128]							
0x1698	write_train_data[255:192]							
0x16a0	write_train_data[319:256]							
0x16a8	write_train_data[383:320]							
0x16b0	write_train_data[447:384]							
0x16b8	write_train_data[511:448]							
0x16c0	Write_train_ecc_data[64:0]							
0x16c8		train_write_n	obj_addr		train_ecc_cmd_len		rw_train_req	rw_train_mode/start
0x16d0	pm_train_base_addr							
……								
0x1700							acc_hp	acc_en
0x1708	acc_fake_b				acc_fake_a			
0x1710								
0x1718								
0x1720	addr_base_acc_1				addr_base_acc_0			
0x1728								
0x1730	addr_mask_acc_1				addr_mask_acc_0			
0x1738								
……								
0x1768	cke_map_ch1				cs_map_ch1			
0x1770				cs2cid_ch1				cid_map_ch1

0x1778								
0x1780				tPHY_WRLAT_ch1	tWL_ch1	tRDDATA_ch1	tPHY_RDLAT_ch1	tRL_ch1
0x1788								
0x1790					cs_mrs_sequence_ch1			
.....								
0x1800	pm_lp_clk_fsp_change_req	pm_lp_ca_train_clk_sel	pm_lp_ca_sample	pm_lp_cavref	pm_lp_cavref_set	pm_lp_clk_en	pm_lp_cke_disable	pm_lp_cacmd_done
0x1808	pm_osc_mrr_ready (RO)	pm_clk_fsp_change_done (RO)	pm_ca_sample_ready (RO)	pm_ca_vref_ready (RO)	pm_ca_training_cmd_ready (RO)	pm_lp_clk_stop_ready (RO)	pm_lpcke_ctrl_ready (RO)	pm_lp_ca_train_mrw_ready (RO)
0x1810		pm_CA_ODT_en	pm_derate_en	pm_odt_en	pm_osc_start_req	pm_lp_wdqs_model	pm_ref_control_en	pm_osc_control
0x1818				pm_lp_tMRW	pm_tODToff_max	pm_ODTLoff	pm_tODTon_min	pm_ODTLon
0X1820	pm_lp_tXCBT	PM_LP_Tckelodt	pm_lp_tCACD	pm_lp_tCAENT	pm_lp_tCKELCS	pm_lp_tDStrain	pm_lp_tCKPSTCS	pm_lp_tCKPRECS
0x1828	pm_lp_tCKFSP	pm_lp_tFC_long	pm_lp_tVREFCA_long	pm_lp_tVRCG_disable	pm_lp_tDQSCKEH	pm_lp_tDQSCKE	pm_lp_tCKEHDQS	pm_lp_tCKELCK
0x1830	pm_tDQSCK	pm_tWDQS_off	pm_tRFMab		pm_lp_tMR4_rdinterval			
0X1838	pm_tRAAIMT		pm_tRFMTH		pm_tRAAMMT		pm_tRAADEC	
0x1840				tOSCO	tOSC_wait		pm_OSC_interval	
0x1848				tRRD_derate	tRP_derate		tRCD_derate	tRAS_min_derate
0x1850							osc_close_done	ref_close_done
.....								
0x1870	pm_FIFO_data_read_all	pm_FIFO_data_read_one	pm_FIFO_read_req	pm_wr_fifo_wr_req	pm_FIFO_write_req	pm_mpc_cs	pm_lpfifo_reset	pm_mpc_fifo_en
0x1878					pm_FIFO_rd_done (RO)	pm_wr_fifo_wrready (RO)	pm_FIFO_wr_done (RO)	pm_FIFO_compare_ready (RO)
0x1880	pm_fifo_wr_data[63:0]							
0x1888	pm_fifo_wr_data[127:64]							
0x1890	pm_fifo_wr_data[191:128]							
0x1898	pm_fifo_wr_data[255:192]							
0x18a0	pm_fifo_wr_data[319:256]							
0x18a8	pm_fifo_wr_data[383:320]							
0x18b0	pm_fifo_wr_data[447:384]							
0x18b8	pm_fifo_wr_data[511:448]							
0x18c0	pm_fifo_rd_data[63:0]							
0x18c8	pm_fifo_rd_data[127:64]							
0x18d0	pm_fifo_rd_data[191:128]							
0x18d8	pm_fifo_rd_data[255:192]							
0x18e0	pm_fifo_rd_data[319:256]							
0x18e8	pm_fifo_rd_data[383:320]							
0x18f0	pm_fifo_rd_data[447:384]							
0x18f8	pm_fifo_rd_data[511:448]							
0x1900	pm_FIFO_compare_result_one[63:0]							
0x1908	pm_FIFO_compare_result_one[127:64]							
0x1910	pm_FIFO_compare_result_one[191:128]							
0x1918	pm_FIFO_compare_result_one[255:192]							
0x1920	pm_FIFO_compare_result_one[319:256]							
0x1928	pm_FIFO_compare_result_one[383:320]							
0x1930	pm_FIFO_compare_result_one[447:384]							
0x1938	pm_FIFO_compare_result_one[511:448]							
0x1940	pm_FIFO_compare_result_all_0[63:0]							
0x1948	pm_FIFO_compare_result_all_0[127:64]							
0x1950	pm_FIFO_compare_result_all_0[191:128]							
0x1958	pm_FIFO_compare_result_all_0[255:192]							

0x1960	pm_FIFO_compare_result_all_0[319:256]							
0x1968	pm_FIFO_compare_result_all_0[383:320]							
0x1970	pm_FIFO_compare_result_all_0[447:384]							
0x1978	pm_FIFO_compare_result_all_0[511:448]							
0x1980	pm_FIFO_compare_result_all_1[63:0]							
0x1988	pm_FIFO_compare_result_all_1[127:64]							
0x1990	pm_FIFO_compare_result_all_1[191:128]							
0x1998	pm_FIFO_compare_result_all_1[255:192]							
0x19a0	pm_FIFO_compare_result_all_1[319:256]							
0x19a8	pm_FIFO_compare_result_all_1[383:320]							
0x19b0	pm_FIFO_compare_result_all_1[447:384]							
0x19b8	pm_FIFO_compare_result_all_1[511:448]							
0x19c0	pm_FIFO_compare_result_all_2[63:0]							
0x19c8	pm_FIFO_compare_result_all_2[127:64]							
0x19d0	pm_FIFO_compare_result_all_2[191:128]							
0x19d8	pm_FIFO_compare_result_all_2[255:192]							
0x19e0	pm_FIFO_compare_result_all_2[319:256]							
0x19e8	pm_FIFO_compare_result_all_2[383:320]							
0x19f0	pm_FIFO_compare_result_all_2[447:384]							
0x19f8	pm_FIFO_compare_result_all_2[511:448]							
0x1a00	pm_FIFO_compare_result_all_3[63:0]							
0x1a08	pm_FIFO_compare_result_all_3[127:64]							
0x1a10	pm_FIFO_compare_result_all_3[191:128]							
0x1a18	pm_FIFO_compare_result_all_3[255:192]							
0x1a20	pm_FIFO_compare_result_all_3[319:256]							
0x1a28	pm_FIFO_compare_result_all_3[383:320]							
0x1a30	pm_FIFO_compare_result_all_3[447:384]							
0x1a38	pm_FIFO_compare_result_all_3[511:448]							
0x1a40	pm_FIFO_compare_result_all_4[63:0]							
0x1a48	pm_FIFO_compare_result_all_4[127:64]							
0x1a50	pm_FIFO_compare_result_all_4[191:128]							
0x1a58	pm_FIFO_compare_result_all_4[255:192]							
0x1a60	pm_FIFO_compare_result_all_4[319:256]							
0x1a68	pm_FIFO_compare_result_all_4[383:320]							
0x1a70	pm_FIFO_compare_result_all_4[447:384]							
0x1a78	pm_FIFO_compare_result_all_4[511:448]							
.....								
MON								
0x2000								cmd_monitor
0x2008								
0x2010	cmd_fbck[63:0] (RD)							
0x2018	cmd_fbck[127:64] (RD)							
0x2020					rw_switch_cnt (RD)			
.....								
0x2100								scheduler_mon
0x2108								
0x2110	sch_cmd_num (RD)							

0x2118	ba_conflict_all (RD)							
0x2120	ba_conflict_last1 (RD)							
0x2128	ba_conflict_last2 (RD)							
0x2130	ba_conflict_last3 (RD)							
0x2138	ba_conflict_last4 (RD)							
0x2140	ba_conflict_last5 (RD)							
0x2148	ba_conflict_last6 (RD)							
0x2150	ba_conflict_last7 (RD)							
0x2158	ba_conflict_last8 (RD)							
0x2160	rd_conflict (RD)							
0x2168	wr_conflict (RD)							
0x2170	rtw_conflict (RD)							
0x2178	wtr_conflict (RD)							
0x2180	rd_conflict_last1 (RD)							
0x2188	wr_conflict_last1 (RD)							
0x2190	rtw_conflict_last1 (RD)							
0x2198	wtr_conflict_last1 (RD)							
0x21a0	wr_rd_turnaround (RD)							
0x21a8	cs_turnaround (RD)							
0x21b0	bg_conflict (RD)							
……								
0x2300						sm_leveling		sm_init
0x2308								
0x2310		sm_rank_03		sm_rank_02		sm_rank_01		sm_rank_00
0x2318		sm_rank_07		sm_rank_06		sm_rank_05		sm_rank_04
0x2320		sm_rank_11		sm_rank_10		sm_rank_09		sm_rank_08
0x2328		sm_rank_15		sm_rank_14		sm_rank_13		sm_rank_12
0x2330		sm_rank_19		sm_rank_18		sm_rank_17		sm_rank_16
0x2338		sm_rank_23		sm_rank_22		sm_rank_21		sm_rank_20
0x2340		sm_rank_27		sm_rank_26		sm_rank_25		sm_rank_24
0x2348		sm_rank_31		sm_rank_30		sm_rank_29		sm_rank_28
……								
TST								
0x3000						lpbk_mode	lpbk_start	lpbk_en
0x3008	lpbk_correct (RD)				lpbk_counter (RD)			lpbk_error (RD)
0x3010	lpbk_data_en[63:0]							
0x3018								lpbk_data_en[71:64]
0x3020							lpbk_data_mask_en	
0x3028								
0x3030	Lpbk_dat_w0[63:0]							
0x3038	Lpbk_dat_w0[127:64]							
0x3040	Lpbk_dat_w1[63:0]							
0x3048	Lpbk_dat_w1[127:64]							
0x3050		lpbk_ecc_mask_w0	lpbk_dat_mask_w0				lpbk_ecc_w0	
0x3058		lpbk_ecc_mask_w1	lpbk_dat_mask_w1				lpbk_ecc_w1	
0x3060								prbs_23
0x3068						prbs_init		

……								
0x3100					fix_data_pattern_index	bus_width	page_size	test_engine_en
0x3108			cs_diff_tst	c_diff_tst	bg_diff_tst	ba_diff_tst	row_diff_tst	col_diff_tst
0x3120	addr_base_tst							
0x3128								
0x3130	user_data_pattern							
0x3138								
0x3140	valid_bits[63:0]							
0x3148								valid_bits[71:64]
0x3150	ctrl[63:0]							
0x3158	ctrl[127:64]							
0x3160	obs[63:0] (RD)							
0x3168	obs[127:64] (RD)							
0x3170	obs[191:128] (RD)							
0x3178	obs[255:192] (RD)							
0x3180	obs[319:256] (RD)							
0x3188	obs[383:320] (RD)							
0x3190	obs[447:384] (RD)							
0x3198	obs[511:448] (RD)							
0x31a0	obs[575:512] (RD)							
0x31a8	obs[639:576] (RD)							
0x31b0					obs[671:640] (RD)			
……								
0x3200								
0x3208								
0x3220	tud_i0							
0x3228	tud_i1							
0x3230	tud_o (RD)							
……								
0x3300	tst_300							
0x3308	tst_308							
0x3310	tst_310							
0x3318	tst_318							
0x3320	tst_320							
0x3328	tst_328							
0x3330	tst_330							
0x3338	tst_338							
0x3340	tst_340							
0x3348	tst_348							
0x3350	tst_350							
0x3358	tst_358							
0x3360	tst_360							
0x3368	tst_368							
0x3370	tst_370							
0x3378	tst_378							

13.4 软件编程指南

13.4.1 初始化操作

初始化操作由软件向寄存器 Init_start (0x010) 写入 0x2 时开始，在设置 Init_start 信号之前，必须将其它所有寄存器设置为正确的值。

软硬件协同的 DRAM 初始化过程如下：

(1) 设置 pm_clk_sel_ckca 和 pm_clk_sel_ds

(2) 设置 pm_phy_init_start 为 1，开始初始化 PHY

(3) 等待 DLL 主控模块锁定，即 pm_dll_init_done 为 1

(4) 等待所有时钟产生模块的 pm_dll_lock_*或者 pm_pll_lock_*变为 1

(5) 使能所有的 pm_clken_*

(6) 将 pm_init_start 设置为 1，内存控制器开始初始化

(7) 等待内存控制器初始化完成，即 pm_dram_init 的值与 pm_cs_enable 相同。

13.4.2 Leveling

Leveling 操作是在 DDR3/4 中，用于智能配置内存控制器读写操作中各种信号间相位关系的操作。通常它包括了 Write Leveling、Read Leveling 和 Gate Leveling。在本控制器中，只实现了 Write Leveling 与 Gate Leveling，Read Leveling 没有实现，软件需要通过判断读写的正确性来实现 Read Leveling 所完成的功能。除了在 Leveling 过程中操作的 DQS 相位、GATE 相位之外，还可以根据这些最后确认的相位来计算出写 DQ 相位、读 DQ 相位的配置方法。此外，本设计还支持 bit-deskew 功能，用于补偿一个 dataslice 内不同 bit 之间的延时差。

13.4.3 Write Leveling

Write Leveling 用于配置写 DQS 与时钟之间的相位关系，软件编程需要参照如下步骤。

(1) 完成控制器初始化，参见上一小节内容；

(2) 将 Dll_wrdqs_x (x = 0…8) 设置为 0x20；

(3) 将 Dll_wrdq_x (x = 0…8) 设置为 0x0；

(4) 设置 Lvl_mode 为 2’b01；

(5) 采样 Lvl_ready 寄存器，如果为 1，表示可以开始 Write Leveling 请求；

(6) 设置 Lvl_req 为 1；

(7) 采样 Lvl_done 寄存器，如果为 1，表示一次 Write Leveling 请求完成；

(8) 采样 Lvl_resp_x 寄存器，如果为 0，则将对应的 Dll_wrdq_x[6:0] 和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 1，然后转向 9；如果

为 1，则将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 0，然后继续将对应的 Dll_wrdq_x[6:0]和 dll_1xdly[6:0]增加 1，并重复执行 5-7 直至 Lvl_resp_x 为 1，然后转向 9。

(9) 将 Dll_wrdq_x 和 dll_1xdly 的值减 0x40，此时 Dll_wrdq_x 和 dll_1xdly 的值就应该是正确的设置值。

(10) 根据 DIMM 类型设置 pm_dly_2x，对于 0x0 边界右边的颗粒对应的 pm_dly_2x 值增加 0x010101。

(11) 将 Lvl_mode (0x700) 设置为 2'b00，退出 Write Leveling 模式。

13.4.4 Gate Leveling

Gate Leveling 用于配置控制器内使能采样读 DQS 窗口的时机，软件编程参照如下步骤。

(1) 完成控制器初始化，参见上一小节内容；

(2) 完成 Write Leveling，参见上一小节内容；

(3) 将 Dll_gate_x (x = 0…8) 设置为 0；

(4) 设置 Lvl_mode 为 2'b10；

(5) 采样 Lvl_ready 寄存器，如果为 1，表示可以开始 Gate Leveling 请求；

(6) 设置 Lvl_req 为 1；

(7) 采样 Lvl_done 寄存器，如果为 1，表示一次 Gate Leveling 请求完成；

(8) 采样 Lvl_resp_x[0]寄存器。如果第一次采样发现 Lvl_resp_x[0]为 1，则将对应的 Dll_gate_x[6:0]增加 1，并重复执行 6-8，直至采样结果为 0，否则进行下一步；

(9) 如果采样结果为 0，则将对应的 Dll_gate_x[6:0]增加 1，并重复执行 6-9；如果为 1，则表示 Gate Leveling 操作已经成功；

(10) 根据 pm_rddqs_phase 的值设置 pm_rdedge_sel

(11) 将 Dll_gate_x (x = 0…8) 减 0x20；

(12) 调整完毕后，再分别进行两次 Lvl_req 操作，观察 Lvl_resp_x[7:5]与 Lvl_resp_x[4:2]的值变化，如果各增加为 Burst_length/2，则继续进行第 13 步操作；如果不为 4，可能需要对 Rd_oe_begin_x 进行加一或减一操作，如果大于 Burst_length/2，很可能需要对 Dll_gate_x 的值进行一些微调；

(13) 将 Lvl_mode (0x700) 设置为 2'b00，退出 Gate Leveling 模式；

(14) 至此 Gate Leveling 操作结束。

13.4.5 功耗控制配置流程

首先需要设置 pm_pad_ctrl_ca[0]为 1，等待内存初始化完成之后，再设置 pm_pad_ctrl_ca[0]为 0。该功能只有在 DDR4 模式下使能了 CAL Mode 才可以使用。

13.4.6 单独发起 MRS 命令

对于 DDR4 模式时，内存控制器向内存发出的 MRS 命令次序分别为：

MR3_CS0、MR3_CS1、MR3_CS2、MR3_CS3、MR3_CS4、MR3_CS5、MR3_CS6、MR3_CS7、

MR6_CS0、MR6_CS1、MR6_CS2、MR6_CS3、MR6_CS4、MR6_CS5、MR6_CS6、MR6_CS7、

MR5_CS0、MR5_CS1、MR5_CS2、MR5_CS3、MR5_CS4、MR5_CS5、MR5_CS6、MR5_CS7、

MR4_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR4_CS4、MR4_CS5、MR4_CS6、MR4_CS7、

MR2_CS0、MR2_CS1、MR2_CS2、MR2_CS3、MR2_CS4、MR2_CS5、MR2_CS6、MR2_CS7、

MR1_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR1_CS4、MR1_CS5、MR1_CS6、MR1_CS7、

MR0_CS0、MR1_CS1、MR1_CS2、MR1_CS3、MR0_CS4、MR0_CS5、MR0_CS6、MR0_CS7。

其中，对应 CS 的 MRS 命令是否有效，是由 Cs_mrs 决定，只有 Cs_mrs 上对应每个片选的位有效，才会真正向 DRAM 发出这个 MRS 命令。对应的每个 MR 的值由寄存器 Mr*_cs*决定。这些值同时也用于初始化内存时的 MRS 命令。

具体操作如下：

(1) 将寄存器 Cs_mrs (0x1101) 、Mr*_cs* (0x1140 – 0x11f8) 设置为正确的值；

(2) 设置 Command_mode (0x0x1120) 为 1，使控制器进入命令发送模式；

(3) 采样 Status_cmd (0x1122) ，如果为 1，则表示控制器已进入命令发送模式，可以进行下一步操作，如果为 0，则需要继续等待；

(4) 写 Mrs_req (0x1126) 为 1，向 DRAM 发送 MRS 命令；

(5) 采样 Mrs_done (0x1127) ，如果为 1，则表示 MRS 命令已经发送完毕，可以退出，如果为 0，则需要继续等待；

(6) 设置 Command_mode (0x1120) 为 0，使控制器退出命令发送模式。

13.4.7 任意操作控制总线

内存控制器可以通过命令发送模式向 DRAM 发出任意的命令组合，软件可以设置 Cmd_cs、Cmd_cmd、Cmd_ba、Cmd_a (0x1128) ，在命令发送模式下向 DRAM 发出。

具体操作如下：

(1) 将寄存器 Cmd_cs、Cmd_cmd、Cmd_ba、Cmd_a (0x1128) 设置为正确的值；

(2) 设置 Command_mode (0x1120) 为 1，使控制器进入命令发送模式；

(3) 采样 Status_cmd (0x1122) ，如果为 1，则表示控制器已进入命令发送模式，可以进行下一步操作，如果为 0，则需要继续等待；

(4) 写 Cmd_req (0x1121) 为 1，向 DRAM 发送命令；

(5) 设置 Command_mode (0x1120) 为 0，使控制器退出命令发送模式。

13.4.8 自循环测试模式控制

自循环测试模式可以分别在测试模式下或者正常功能模式下使用，为此，本内存控制器分别实现了两套独立的控制接口，一套用于在测试模式下由测试端口直接控制，另一套用于在正常功能模式下由寄存器配置模块进行配置使能测试。

这两套接口的复用使用端口 test_phy 进行控制，当 test_phy 有效时，使用控制器的 test_*端口进行控制，此时的自测试完全由硬件控制；当 test_phy 无效时，使用软件编程的 pm_*的参数进行控制。使用测试端口的具体信号含义可以参考寄存器参数中的同名部分。

这两套接口从控制的参数来说基本一致，仅仅是接入点不同，在此介绍软件编程时的控制方法。具体操作如下：

(1) 将内存控制器所有的参数全部正确设置；

(2) 按照初始化流程等待时钟复位稳定；

(3) 将寄存器 Lpbk_en 设为 1；

(4) 将寄存器 Lpbk_start 设为 1；此时自循环测试正式开始。

(5) 到此为止自循环测试已经开始，软件需要经常检测是否有错误发生，具体操作如下：

采样寄存器 Lpbk_error，如果这个值为 1，表示有错误发生，此时可以通过 Lpbk_*等观测用寄存器来观测第一个出错时的错误数据和正确数据；如果这个值为 0，表示还没有出现过数据错误。

13.4.9 低功耗控制

寄存器基地址为 0x1FE00000，偏移地址为 0x1520，同样也可以使用配置寄存器指令进行访问，寄存器及其对应位如下。

寄存器	地址	控制	说明
内存控制器调度模块低功耗控制	0x1520	RW	1 为使能低功耗
内存控制器配置寄存器模块低功耗控制	0x1521	RW	1 为使能低功耗
降半频控制	0x1522	RW	[0]：降半频使能寄存器，1 为使能 [1]：根据访存空闲自动降半频使能寄存器，1 为使能 [2]：出错计数溢出时自动降半频使能寄存器，1 为使能
降半频频率选择	0x1523	RW	[0]：降半频软件控制，0 选择高频，1 选择低频 [7]：降半频切换频率完成标志，该位为只读位

自动降频功能需要对低频先初始化，其流程如下：

（1）将 pm_half_frq_reg_rw=1，切换到低频的寄存器读写；

（2）配置完后，再将 0x1fe01522[0]=1；

（3）然后将 0x1fe01523[0]=1；

（4）等待 0x1fe01523[7]；

（5）在低频进行所有读写训练，配置所有参数；

（6）然后将 0x1fe01523[0]=0；

（7）等待 0x1fe01523[7]；

（8）根据需求开启 0x1fe01522[2:1]。

14 IO 低速设备 (D2:F0)

14.1 IO 低速设备配置寄存器 (MISC-D2:F0)

IO 低速设备块包含多个低速设备。该总线控制器作为一个设备具有相应的配置头空间。配置头的默认值见表 13- 1.

表 14-1 IO 低速设备块的 PCI 配置寄存器 (MISC-D2:F0)

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A42h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	80h	RO
0Bh	BCC	Base Class Code	08h	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W,RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

PCICMD-PCI 命令寄存器

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 IO 低速设备块内部设备的访问。 0：禁止访问； 1:使能对 IO 低速设备块内部设备的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

14.2 内部设备地址路由

IO 低速设备块下挂载了多个低速设备，包括：UART、I2C、PWM、CAN、HPET、ACPI、RTC、DPM 和 GPIO。这些低速设备根据地址位的 bit[19:16]进行区分，对应关系为：

表 14-2 IO 低速设备地址路由

bit[19:16]	0	1	2	3	4	5	6	7	8
设备	UART	I2C	PWM	CAN	HPET	ACPI/RTC/DPM	GPIO	-	DMA
访问类型	B	B	W	W	W	W	B	保留	

对于 UART、I2C、PWM、CAN、ACPI/RTC 来说，由于包含多个控制器，它们需要进一步的路由。这些设备块的内部路由见表 14- 22。不同的设备块需要的路由地址位数是不同的。

后续章节将分别对这些低速设备进行介绍。

15 UART 控制器

15.1 概述

UART 控制器提供与 MODEM 或其他外部设备串行通信的功能，例如与另外一台计算机，以 RS232 为标准使用串行线路进行通信。该控制器在设计上能很好地兼容国际工业标准半导体设备 16550A。

2K3000 集成了 8 个 UART 接口和 8 个 UART 控制器。

15.2 访问地址及引脚复用

UART0 控制器寄存器物理地址基址为 0x1FE001E0，此外通过物理地址 0x1FE00100 可访问新增的两个寄存器 RFC 和 TFC。

其余 UART 控制器的访问基地址为 IO 低速设备块的基地址加偏移 0x0。

注意：UART 模块仅支持按字节访问。

各个 UART 控制器内部寄存器的物理地址构成如下：

表 15-1 UART 控制器物理地址构成

地址位	构成	备注
[15:13]	0	保留
[12:08]	UART 号	0x0 - 0x6，分别代表 UART1-7
[07:00]	REG	内部寄存器地址

对于各个 UART，使用时要注意将对应的引脚设置为相应的功能。

15.3 控制器结构

UART 控制器有发送和接收模块（Transmitter and Receiver）、MODEM 模块、中断仲裁模块（Interrupt Arbitrator）、和访问寄存器模块（Register Access Control），这些模块之间的关系见下图所示。主要模块功能及特征描述如下：

1） 发送和接收模块：负责处理数据帧的发送和接收。发送模块是将 FIFO 发送队列中的数据按照设定的格式把并行数据转换为串行数据帧，并通过发送端口送出去。接收模块则监视接收端信号，一旦出现有效开始位，就进行接收，并实现将接收到的异步串行数据帧转换为并行数据，存入 FIFO 接收队列中，同时检查数据帧格式是否有错。UART 的帧结构是通过行控制寄存器（LCR）设置的，发送和接收器的状态被保存在行状态寄存器（LSR）中

2） MODEM 模块：MODEM 控制寄存器（MCR）控制输出信号 DTR 和 RTS 的状态。MODEM 控制模块监视输入信号 DCD,CTS,DSR 和 RI 的线路状态，并将这些信号的状态记录在 MODEM 状态寄存器（MSR）的相对应位中。

3） 中断仲裁模块：当任何一种中断条件被满足，并且在中断使能寄存器（IER）中相应位置 1，那么 UART 的中断请求信号 UAT_INT 被置为有效状态。为了减少和外部软件的交互，UART 把中断分为四个级别，并且在中断标识寄存器（IIR）中标识这些中断。四个级别的中断按优先级级别由高到低的排列顺序为，接收线路状态中断；接收数据准备好中断；传送拥有寄存器为空中断；MODEM 状态中断。

4） 访问寄存器模块：当 UART 模块被选中时，CPU 可通过读或写操作访问被地址线选中的寄存器。

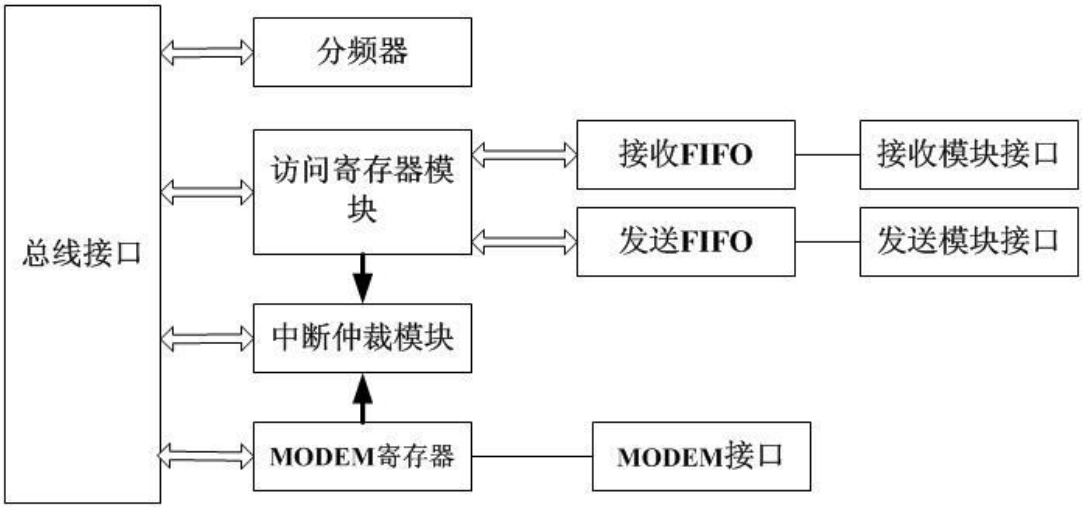


图 15-1 UART 控制器结构

15.4 寄存器描述

UART 接口模块配置寄存器，如下表所示。

表 15-2 UART 配置寄存器列表

地址偏移	寄存器名称	描述
0x00	DAT/DL_L	数据寄存器/分频系数低 8 位
0x01	IER/DL_H	中断使能寄存器/分频系数高 8 位
0x02	IIR	中断标识寄存器
0x02	FCR/DL_D	FIFO 控制寄存器/分频系统小数位
0x03	LCR	线路控制寄存器
0x04	MCR	Modem 控制寄存器
0x05	LSR	线路状态寄存器
0x06	MSR	Modem 状态寄存器

15.4.1 数据寄存器（DAT）

中文名： 数据传输寄存器

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0x00

表 15-3 数据传输寄存器

位域	位域名称	位宽	访问	描述
7:0	DATA	8	RW	写访问为发送数据、读访问为接收数据

15.4.2 中断使能寄存器（IER）

中文名： 中断使能寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x00

表 15-4 中断使能寄存器

位域	位域名称	位宽	访问	描述
7	ACTSE	1	RW	CTS 自动流控使能 ‘0’ - 关闭 ‘1’ - 打开 打开时，若 CTSn 的输入为 0，则暂停发送
6	ARTSE	1	RW	RTS 自动流控使能 ‘0’ - 关闭 ‘1’ - 打开 打开时，若 rxfifo 为满，则 RTSn 的输出为 0
5	TXDE	1	RW	发送状态 DMA 使能 ‘0’ - 关闭 ‘1’ - 打开 打开时，若 txfifo 未满，向 DMA 发送发送请求
4	RXDE	1	RW	接收状态 DMA 使能 ‘0’ - 关闭 ‘1’ - 打开 打开时，若 rxfifo 未满，向 DMA 发送接收请求
3	IME	1	RW	Modem 状态中断使能 ‘0’ - 关闭 ‘1’ - 打开
2	ILE	1	RW	接收器线路状态中断使能 ‘0’ - 关闭 ‘1’ - 打开
1	ITxE	1	RW	传输保存寄存器为空中断使能 ‘0’ - 关闭 ‘1’ - 打开
0	IRxE	1	RW	接收有效数据中断使能 ‘0’ - 关闭 ‘1’ - 打开

15.4.3 中断标识寄存器（IIR）

中文名： 中断源寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0xc1

表 15-5 中断源寄存器

位域	位域名称	位宽	访问	描述
7:4	Reserved	4	R	保留
3:1	II	3	R	中断源表示位，详见下表
0	INTp	1	R	中断表示位

表 15-6 中断控制功能表

Bit 3	Bit 2	Bit 1	优先级	中断类型	中断源	中断复位控制
0	1	1	1st	接收线路状态	奇偶、溢出或帧错误，或打断中断	读 LSR
0	1	0	2nd	接收到有效数据	FIFO 的字符个数达到 trigger 的水平	FIFO 的字符个数低于 trigger 的值
1	1	0	2nd	接收超时	在 FIFO 至少有一个字符，但在 4 个字符时间内没有任何操作，包括读和写操作	读接收 FIFO

0	0	1	3rd	传输保存寄存器为空	传输保存寄存器为空	写数据到 THR 或者多 IIR
0	0	0	4th	Modem 状态	CTS, DSR, RI or DCD.	读 MSR

15.4.4 FIFO 控制寄存器 (FCR)

中文名： FIFO 控制寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0xc0

表 15-7 FIFO 控制寄存器

位域	位域名称	位宽	访问	描述
7:6	TL	2	W	接收FIFO 提出中断申请的trigger 值 ‘00’ - 1 字节 ‘01’ - 2 字节 ‘10’ - 3 字节 ‘11’ - 4 字节
5:3	Reserved	3	W	保留
2	Txset	1	W	‘1’ 清除发送FIFO 的内容，复位其逻辑
1	Rxset	1	W	‘1’ 清除接收FIFO 的内容，复位其逻辑
0	Reserved	1	W	保留

15.4.5 线路控制寄存器 (LCR)

中文名： 线路控制寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x03

表 15-8 线路控制寄存器

位域	位域名称	位宽	访问	描述
7	dlab	1	RW	分频锁存器访问位 ‘1’ - 访问操作分频锁存器 ‘0’ - 访问操作正常寄存器
6	bcb	1	RW	打断控制位 ‘1’ - 此时串口的输出被置为 0（打断状态）. ‘0’ - 正常操作
5	spb	1	RW	指定奇偶校验位 ‘0’ - 不用指定奇偶校验位 ‘1’ - 如果 LCR[4]位是 1 则传输和检查奇偶校验位为 0。如果 LCR[4]位是 0 则传输和检查奇偶校验位为 1。
4	eps	1	RW	奇偶校验位选择 ‘0’ - 在每个字符中有奇数个 1）（包括数据和奇偶校验位（ ‘1’ - 在每个字符中有偶数个 1
3	pe	1	RW	奇偶校验位使能 ‘0’ - 没有奇偶校验位 ‘1’ - 在输出时生成奇偶校验位，输入则判断奇偶校验位
2	sb	1	RW	定义生成停止位的位数 ‘0’ - 1 个停止位 ‘1’ - 在 5 位字符长度时是 1.5 个停止位，其他长度是 2 个停止位

1:0	bec	2	RW	设定每个字符的位数 ‘00’ - 5 位 ‘01’ - 6 位 ‘10’ - 7 位 ‘11’ - 8 位

15.4.6 MODEM 控制寄存器 (MCR)

中文名： Modem 控制寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x00

表 15-9 Modem 控制寄存器

位域	位域名称	位宽	访问	描述
7:5	Reserved	3	W	保留
4	Loop	1	W	回环模式控制位 ‘0’ - 正常操作 ‘1’ - 回环模式。在在回环模式中，TXD 输出一直为 1，输出移位寄存器直接连到输入移位寄存器中。其他连接如下。 DTR DSR RTS CTS Out1 RI Out2 DCD
3	OUT2	1	W	在回环模式中连到 DCD 输入
2	OUT1	1	W	在回环模式中连到 RI 输入
1	RTSC	1	W	RTS 信号控制位
0	DTRC	1	W	DTR 信号控制位

15.4.7 线路状态寄存器 (LSR)

中文名： 线路状态寄存器

寄存器位宽： [7：0]

偏移量： 0x05

复位值： 0x00

表 15-10 线路状态寄存器

位域	位域名称	位宽	访问	描述
7	ERROR	1	R	错误表示位 ‘1’ - 至少有奇偶校验位错误，帧错误或打断中断的一个。 ‘0’ - 没有错误
6	TE	1	R	传输为空表示位 ‘1’ - 传输 FIFO 和传输移位寄存器都为空。给传输 FIFO 写数据时清零 ‘0’ - 有数据
5	TFE	1	R	传输 FIFO 位空表示位 ‘1’ - 当前传输 FIFO 为空，给传输 FIFO 写数据时清零 ‘0’ - 有数据

位域	位域名称	位宽	访问	描述
4	BI	1	R	打断中断表示位 ‘1’ －接收到 起始位＋数据＋奇偶位＋停止位都是 0，即有打断中断 ‘0’ － 没有打断
3	FE	1	R	帧错误表示位 ‘1’ － 接收的数据没有停止位 ‘0’ － 没有错误
2	PE	1	R	奇偶校验位错误表示位 ‘1’ － 当前接收数据有奇偶错误 ‘0’ － 没有奇偶错误
1	OE	1	R	数据溢出表示位 ‘1’ － 有数据溢出 ‘0’ － 无溢出
0	DR	1	R	接收数据有效表示位 ‘0’ － 在 FIFO 中无数据 ‘1’ － 在 FIFO 中有数据

对这个寄存器进行读操作时，LSR[4:1]和 LSR[7]被清零，LSR[6:5]在给传输 FIFO 写数据时清零，LSR[0]则对接收 FIFO 进行判断。

15.4.8 MODEM 状态寄存器 (MSR)

中文名： Modem 状态寄存器

寄存器位宽： [7：0]

偏移量： 0x06

复位值： 0x00

表 15-11 Modem 状态寄存器

位域	位域名称	位宽	访问	描述
7	CDCD	1	R	DCD 输入值的反，或者在回环模式中连到Out2
6	CRI	1	R	RI 输入值的反，或者在回环模式中连到OUT1
5	CDSR	1	R	DSR 输入值的反，或者在回环模式中连到DTR
4	CCTS	1	R	CTS 输入值的反，或者在回环模式中连到RTS
3	DDCD	1	R	DDCD 指示位
2	TERI	1	R	RI 边沿检测。RI 状态从低到高变化
1	DDSR	1	R	DDSR 指示位
0	DCTS	1	R	DCTS 指示位

15.4.9 分频锁存器

中文名： 分频锁存器低 8 位

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0x00

表 15-12 分频锁存器低 8 位寄存器

位域	位域名称	位宽	访问	描述
7:0	DL_L	8	RW	存放分频锁存器的低 8 位数值

中文名： 分频锁存器高 8 位

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x00

表 15-13 分频锁存器高 8 位寄存器

位域	位域名称	位宽	访问	描述
7:0	DL_H	8	RW	存放分频锁存器的高 8 位数值

中文名： 分频锁存器小数位

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x00

表 15-14 分频锁存器小数位寄存器

位域	位域名称	位宽	访问	描述
7:0	DL_D	8	RW	存放分频锁存器的小数部分二进制值 例如,分频小数值为 0.45,DL_D=0x73(即:0x100*0.45)

UART 分频由 DL_H、DL_L、DL_D 三个寄存器组成分频值寄存器 DL，则 UART 波特率为：CLKin/16*DL。例如，输入为 10MHz 时钟，目标波特率为 115200，则 DL=5.42535，故，DL_H=0x0，DL_L=0x5，DL_D=0x6c（即：0x100*0.42535）。波特率配置值与期望值的误差在 5%以内，否则无法识别所有数据位，将导致串口显示乱码。该芯片中 UART0 的输入时钟频率为 100MHz，其余 UART 的输入时钟频率为 48MHz。

16 CAN

龙芯 2K3000 集成了 4 路 CAN FD 接口控制器。CAN 总线是由发送数据线 TX 和接收数据线 RX 构成的串行总线，可发送和接收数据。

16.1 访问地址及引脚复用

4 个 CAN 控制器内部寄存器的物理地址构成如下：

表 16-1 CAN 内部寄存器物理地址构成

地址位	构成	备注
[15:13]	0	保留
[12:08]	CAN 编号	分别为 0x10-0x13
[07:00]	REG	内部寄存器地址

对于 CAN 模块，使用时要注意将对应的引脚设置为相应的功能。

16.2 CAN FD 控制器特性

此 CAN FD 控制器主要特性有：

支持 CAN FD 协议并兼容 CAN2.0 协议；

RX 缓存区大小为 32（宽）x20（深）；

8 个可容纳满载荷 CAN FD 报文（64 字节）的 TX 缓存区；

支持 ISO 与 non-ISO CAN FD 协议；

支持接收添加时间戳与定时发送功能；

支持 Loopback mode、Bus monitoring mode、ACK forbidden mode、Self-test mode 与 Restricted operation mode；

17 I2C 控制器

17.1 概述

本章给出 I2C 的详细描述和配置使用。I2C 总线是由数据线 SDA 和时钟 SCL 构成的串行总线，可发送和接收数据。器件与器件之间进行双向传送，最高传送速率 400kbps。

本系统芯片集成了 4 个 I2C 接口（I2C0~I2C3）及控制器，4 个 I2C 控制器均可以做主设备（master），可通过引脚与其他 I2C 设备进行数据的交换。

17.2 访问地址及引脚复用

I2C0 和 I2C1 内部寄存器的物理地址构成如下：

地址位	构成	备注
[31:04]	I2C 基地址	0x1fe00120 代表 I2C0;0x1fe00130 代表 I2C1
[03]	保留	
[02:00]	REG	内部寄存器地址

I2C2 和 I2C3 通过 APB 设备进行访问，两个 I2C 控制器内部寄存器的物理地址构成如下：

地址位	构成	备注
[18:16]	APB 模块内路由	3’b001 代表 I2C 模块
[15:09]	保留	
[08]	I2C 编号	0x0 代表 I2C2；0x1 代表 I2C3
[07:03]	保留	
[02:00]	REG	内部寄存器地址

对于 I2C 模块，使用时要注意将对应的引脚设置为相应的功能。

I2C0 作为 slave 时，reg0[13:0]为温度传感器的原始数据。

I2C2-3 作为 slave 时，reg0[127:0]={toy,49’h0,rdy,overflow,摄氏温度}。

17.3 I2C 主控制器结构

I2C 主控制器的结构，主要模块有，时钟发生器（Clock Generator）、字节命令控制器（Byte Command Controller）、位命令控制器（Bit Command controller）、数据移位寄存器（Data Shift Register）。其余为 APB 总线接口和一些寄存器。

1） 时钟发生器模块：产生分频时钟，同步位命令的工作。

2） 字节命令控制器模块：将一个命令解释为按字节操作的时序，即把字节操作分解为位操作。

3） 位命令控制器模块：进行实际数据的传输，以及位命令信号产生。

4） 数据移位寄存器模块：串行数据移位。

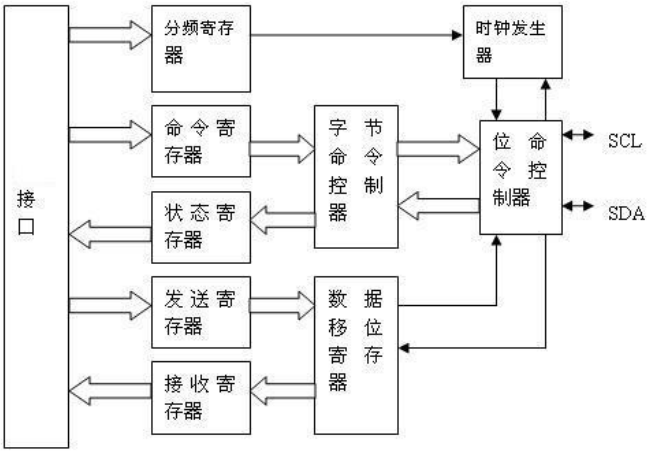


图 17-1 I2C 主控制器结构

17.4 I2C 主控制器寄存器说明

17.4.1 分频锁存器低字节寄存器 (PRERlo)

中文名：分频锁存器低字节寄存器

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0xff

位域	位域名称	位宽	访问	描述
7:0	PRERlo	8	RW	存放分频锁存器的低 8 位

17.4.2 分频锁存器高字节寄存器 (PRERhi)

中文名：分频锁存器高字节寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0xff

位域	位域名称	位宽	访问	描述
7:0	PRERhi	8	RW	存放分频锁存器的高 8 位

假设分频锁存器的值为 prescale，输入时钟的频率为 clock_a（I2C0/1 频率为 100MHz，I2C2/3 频率为 48MHz），SCL 总线的输出频率为 clock_s，则输入输出时钟频率大致满足如下关系（由于 SCL 物理信号存在上升下降沿，导致频率计算有一定误差）：

Prcescale = clock_a/ (5.5*clock_s) -1

17.4.3 控制寄存器 (CTR)

中文名：控制寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x00

位域	位域名称	位宽	访问	描述
7	EN	1	RW	模块工作使能位 为 1 正常工作模式, 0 对分频寄存器进行操作
6	IEN	1	RW	中断使能位为 1 则打开中断
5:0	Reserved	6	RW	保留

17.4.4 发送数据寄存器 (TXR)

中文名：发送寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:1	DATA	7	W	存放下个将要发送的字节
0	DRW	1	W	当数据传送时，该位保存的是数据的最低位； 当地址传送时，该位指示读写状态

17.4.5 接受数据寄存器 (RXR)

中文名：接收寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

位域	位域名称	位宽	访问	描述
7:0	RXR	8	R	存放最后一个接收到的字节

17.4.6 命令控制寄存器 (CR)

中文名：命令寄存器

寄存器位宽： [7：0]

偏移量： 0x04

复位值： 0x00

位域	位域名称	位宽	访问	描述
7	STA	1	W	产生 START 信号
6	STO	1	W	产生 STOP 信号
5	RD	1	W	产生读信号
4	WR	1	W	产生写信号
3	ACK	1	W	控制是否产生应答信号（0 代表产生 ACK，1 代表不产生 ACK）

2:1	Reserved	2	W	保留
0	IACK	1	W	产生中断应答信号

都是在 I2C 发送数据后硬件自动清零。

17.4.7 状态寄存器（SR）

中文名：状态寄存器

寄存器位宽：［7：0］

偏移量：　　0x04

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7	RxACK	1	R	收到应答位 1 没收到应答位 0 收到应答位
6	Busy	1	R	I2c 总线忙标志位 1 总线在忙 0 总线空闲
5	AL	1	R	当 I2C 核失去 I2C 总线控制权时，该位置 1
4:2	Reserved	3	R	保留
1	TIP	1	R	指示传输的过程 1 表示正在传输数据 0 表示数据传输完毕
0	IF	1	R	中断标志位，一个数据传输完，或另外一个器件发起数据传输，该位置 1

17.4.8 从模式控制寄存器（SLV_CTRL）

中文名：从模式控制寄存器

寄存器位宽：［7：0］

偏移量：　　0x07

复位值：　　0x00

位域	位域名称	位宽	访问	描述
7	slv_en	1	RW	I2C2 从模式使能，高有效
6:0	slv_addr	7	RW	I2C2 从模式地址

*该寄存器可用于 I2C0/2/3

18 PWM 控制器

18.1 概述

2K3000 芯片里实现了 4 路脉冲宽度调节/计数控制器，以下简称 PWM。每一路 PWM 工作和控制方式完全相同。每路 PWM 有一路脉冲宽度输出信号和一路待测脉冲输入信号。时钟频率为 48MHz，计数寄存器和参考寄存器均 32 位数据宽度。

18.2 访问地址及引脚复用

PWM 控制器内部寄存器的物理地址构成如下：

地址位	构成	备注
[15:11]	0	保留
[10:8]	PWM 编号	取值 0x0-0x3 分别代表 PWM0-PWM3
[7:4]	0	保留
[03:00]	REG	内部寄存器地址

对于 PWM 模块，使用时要注意将对应的引脚设置为相应的功能。

18.3 寄存器描述

每路控制器共有五个寄存器，具体描述如下：

表 18-1 PWM 寄存器列表

名称	地址	宽度	访问	说明
Low_buffer	Base + 0x4	32	R/W	低脉冲缓冲寄存器
Full_buffer	Base + 0x8	32	R/W	脉冲周期缓冲寄存器
CTRL	Base + 0xC	11	R/W	控制寄存器

表 18-2 PWM 控制寄存器设置

位域	名称	访问	复位值	说明
0	EN	R/W	0	计数器使能位 置 1 时：CNTR 用来计数 置 0 时：CNTR 停止计数（输出保持）
2：1		Reserved	2’b0	预留
3	OE	R/W	0	脉冲输出使能控制位,低有效 置 0 时：脉冲输出使能 置 1 时：脉冲输出屏蔽
4	SINGLE	R/W	0	单脉冲控制位 置 1 时：脉冲仅产生一次 置 0 时：脉冲持续产生
5	INTE	R/W	0	中断使能位 置 1 时：当 full_pulse 到 1 时送中断 置 0 时：不产生中断
6	INT	R/W	0	中断位 读操作：1 表示有中断产生,0 表示没有中断 写入 1：清中断

7	RST	R/W	0	使得 Low_level 和 full_pluse 计数器重置 置 1 时：计数器重置（从 buffer 读，输出低电平） 置 0 时：计数器正常工作
8	CAPTE	R/W	0	测量脉冲使能 置 1 时：测量脉冲模式 置 0 时：非测量脉冲模式（一般而言则是脉冲输出模式）
9	INVERT	R/W	0	输出翻转使能 置 1 时：使脉冲在输出去发生信号翻转（周期以高电平开始） 置 0 时：使脉冲保持原始输出（周期以低电平开始）
10	DZONE	R/W	0	防死区功能使能 置 1 时：该计数模块需要启用防死区功能 置 0 时：该模块无需防死区功能

18.4 功能说明

18.4.1 脉宽调制功能

Low_buffer 和 Full_buffer 寄存器可以由系统编程写入获得初始值。系统编程写入完毕后，模块内部的 low_level 和 full_pulse 寄存器分别从 Low_buffer 和 Full_buffer 缓冲寄存器中读取初值，之后在系统时钟驱动下不断自减（初始输出低电平）。当 low_level 寄存器到达 1 之后，输出变为高电平，此时 full_pulse 仍在自减。当 full_pulse 寄存器到达 1 之后，输出变为低电平，low_level 和 full_pulse 又分别从 Low_buffer 和 Full_buffer 缓冲寄存器中读取初值，然后重新开始不断自减，控制器就产生连续不断的脉冲宽度输出。当 full_pulse 寄存器的值等于 1 的时候，可以配置产生一个中断，从而作为定时器使用。

例：如果要产生宽度为系统时钟周期 50 倍的高脉宽和 90 倍的低脉宽，在 low_buffer 中应该配置初始值 90，在 full_buffer 寄存器中配置初始值（50+90）=140.

值得说明的是，由于两个缓冲寄存器的写入有先后之分，在某些特殊的情况下（比如写入时刻刚好是旧脉冲结束时）会使得输出脉冲有异于预期。推荐的做法是在向缓冲寄存器写入新数前，将控制寄存器 EN 位写 0，在写入新数之后再将 EN 位写 1。值得说明的是，即使没有重写 EN 位，紊乱的脉冲输出最多只会维持一个周期。

如果对两个缓冲寄存器都写 0，则输出为低电平；如果对 low_buffer 写 0，对 full_buffer 写 1，则输出高电平；如果写入 Low_buffer 的值不小于 full_buffer，则输出低电平。但这三类数值都是不推荐的。

此外，缓冲寄存器的数值写入应当先于 CTRL 控制寄存器。

18.4.2 脉冲测量功能

待测脉冲信号连在 PWM 输入信号接口上，在设置完 CTRL 控制寄存器后，在系统时钟的驱动下，Low_level 和 full_pulse 寄存器开始不断自增。当检测到输入脉冲信号上跳变时，将 Low_level 寄存器的值传送到 low_buffer 寄存器中；当检测到输入脉冲信号下跳变时，

将 full_pulse 寄存器的值传送到 full_buffer 寄存器中，并将 Low_level 和 full_pulse 寄存器置 1，重新开始计数。

例：如果要输入脉冲为系统时钟 50 倍的高脉宽和 90 倍的低脉宽，在 low_buffer 中最终读出的值为 90，在 full_buffer 寄存器中读出的值为（50+90）=140.

待测脉冲应当是周期信号，且脉冲周期不应超出 32 位计数器能计量的范围。

每次测量均是从下跳变开始，到下一个下跳变结束。由于测量及缓冲的需要，在连续测量两个脉冲周期后，low_buffer 和 full_buffer 寄存器中存储的才是正确的脉冲参数。

若出现持续的周期超过 0xFFFF_FFF9 的脉冲，控制寄存器 INT 位会被置 1，表示待测脉冲超出了计量范围。

18.4.3 防死区功能

四路 PWM 都配备了防死区功能，可以防止四路脉冲输出同时发生跳变。

将四路模块分别标记为 PWM_0、PWM_1、PWM_2、PWM_3，它们的优先级为 0>1>2>3，即若要同时产生跳变，在 PWM_0 跳变之后 PWM_1 才能跳变（低优先级的信号被“抹去”一个或多个系统时钟），依此类推。该优先级是固化的，不可配置。

一个典型的防死区示例如下（PWM_*为未开防死区的输出，PWM_*’为打开防死区后的输出）：

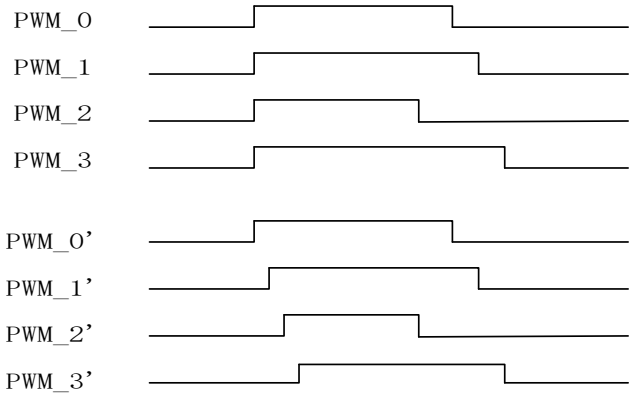


图 18-1 防死区功能

19 HPET 控制器

19.1 概述

HPET（High Precision Event Timer，高精度事件定时器）定义了一组新的定时器，这组定时器被操作系统使用，用来给线程调度，内核以及多媒体定时器服务器等产生中断。操作系统可以将不同的定时器分配给不同的应用程序使用。通过配置，每个定时器都能独立产生中断。

这组定时器由一个向上累加的主计时器（up-counter）以及一组比较器构成。这个计时器以固定的频率（48MHz）向上累加，因此当软件两次读取计时器的值时，除非遇到计时器溢出，否则第二次读取的值总是比第一次读取的值大。而每个定时器都包含一个 match 寄存器以及一个比较器。当 match 寄存器的值与主计时器相等时，那么定时器产生中断。部分定时器可产生周期性中断。

内部包括一个 64 位的主计数器（main count）以及三个 32 位的比较器（comparator）。在这三个比较器中，比较器 0 支持周期性中断（periodic-capable）和非周期性中断，其他两个比较器支持非周期性中断。

19.2 访问地址

HPET 模块的访问基地址为 IO 低速设备块的基地址加偏移 0x40000。HPET 控制器内部寄存器的物理地址构成如下：

地址位	构成	备注
[15:08]	0	保留
[07:00]	REG	内部寄存器地址

19.3 寄存器描述

下表列出了 HPET 的寄存器：

寄存器偏移地址	寄存器	类型
000-007h	General Capabilities and ID Register	只读
008-00Fh	Reserved	
010-017h	General Configuration Register	读/写
018-01Fh	Reserved	R/WC
020-027h	General Interrupt Status Register	R/W
028-0EFh	Reserved	
0F0-0F7h	Main Counter Value Register	R/W

100-107h	Timer 0 Configuration and Capability Register	R/W
108-10Fh	Timer 0 Comparator Value Register	R/W
110-11Fh	Reserved	
120-127h	Timer 1 Configuration and Capability Register	R/W
128-12Fh	Timer 1 Comparator Value Register	R/W
130-13Fh	Reserved	
140-147h	Timer 2 Configuration and Capability Register	R/W
148-14Fh	Timer 2 Comparator Value Register	R/W
150-15Fh	Reserved	

若系统在状态转换过程中需要保存这些寄存器的的值以便随后恢复，那么操作系统负责保存这些寄存器的值，硬件无需保存这些寄存器的值。因此当系统处于 S3，S4，S5 状态时，这些寄存器无需维持。

19.3.1 General Capabilities and ID Register

位	名称	描述	读写特性
63：32	COUNTER_CLK_PERIOD	Main Counter Tick Period：这个域标示了主计时器的计时频率，以 fs（10^-15s）为单位。	RO
31：16	VENDOR_ID		RO
15：14	Reserved		
13	COUNT_SIZE_CAP	Counter Size:主计时器的宽度； 0：32 bits 1：64 bits	RO
12:8	NUM_TIM_CAP	Num of Timer：定时器的个数；这个域的值指示最后一个定时器的编号，2K3000 中有三个定时器，因此这个域的值是 2。	RO
7:0	REV_ID	版本号；不可为 0	RO

19.3.2 General Configuaration Register

位	名称	描述	读写特性
63：1	Reserved		
0	ENABLE_CNF	Overal Enable：用来使能所有定时器产生中断。如果为 0，主计时器停止计时且所有的定时器都不再产生中断。 0：主计时器停止计时且所有的定时器都不再产生中断； 1：主计时器计时且允许定时器产生中断；	R/W

19.3.3 General Interrupt Status Register

位	名称	描述	读写特性
63：3	Reserved		
2	T2_INT_STS	Timer 2 Interrupt Active:功能同 T0_INT_STS	R/WC

1	T1_INT_STS	Timer 1 Interrupt Active:功能同 T0_INT_STS	R/WC
0	T0_INT_STS	Timer 0 Interrupt Active:功能依赖于这个定时器的中断触发模式是电平触发还是边沿触发： 如果是电平触发模式： 这位默认是 0。当对应的定时器发生中断，那么有硬件将其置 1.一旦被置位，软件往这位写 1 将会清空这位。往这位写 0，则无意义。 如果边沿触发模式： 软件将忽略这位。软件通常往这位写 0.	R/WC

各个定时器的中断触发模式由各自 Configuartion and Capability 寄存器的 Tn_TYPE_CNF 位确定。

19.3.4 Main Counter Value Register

位	名称	描述	读写特性
63：0	Main_Counter_Val	主计时器的值；只有当主计时器停止计时时，才允许修改这个寄存器的值。	R/W

19.3.5 Timer N Configuration and Capabilities Register

位	名称	描述	读写特性
63：9	Reserved		
8	Tn_32MODE_CNF	Timer n 32-bit 模式（N 为 0-2）。当定时器为 32 位时，这位为 0，且只读	RO
7	Reserved		RO
6	Tn_VAL_SET_CNF	Timer N Value Set（N 为 0-2）:只有能产生周期性中断的定时器才会使用这个域。通过对这位写 1，软件能直接修改周期性定时期的累加器。软件无需对这位清 0 只有 Timer 0 能产生周期性中断，因此对 Timer0 来讲，这位是可读可写。而对于 Timer1， Timer2，这位默认为 0，且为只读。	R/W
5	Tn_SIZE_CAP	Timer N Size；Timer N 的宽度（N 为 0-2）。 0：32 位宽。	RO
4	Tn_PER_INT_CAP	Timer N Periodic Interrupt Capable（N 为 0-2）： 1：定时器能产生周期性中断； 0：定时器不能产生周期性中断；	RO
3	Tn_TYPE_CNF	Timer N type（N 为 0-2）： 如果对应的 Tn_PER_INT_CAP 位为 0，那么这位为只读，且默认为 0. 若对应的 Tn_PER_INT_CAP 位为 1,那么这位可读可写。用作使能相应的定时器产生周期性中断。	R/W

		1：使能定时器产生周期性中断 0：使能定时器产生非周期性中断	
2	Tn_INT_ENB_CNF	Timer N interrupt Enable (N 为 0-2) :使能定时器产生中断	R/W
1	Tn_INT_TYPE_CNF	Timer N Interrupt Type (N 为 0-2) : 0：定时器的中断触发模式为边沿触发；这意位着对应的定时器将产生边沿触发中断。若另外的的中断产生，那么将产生另外的边沿。 1：定时器的中断触发模式为电平触发；这意味着对应的定时器将产生电平触发中断。这个中断将一直有效直到被软件清掉 (General Interrupt Status Register) 。	
0	Reserved		

19.3.6 Timer N Comparator Value Register

位	名称	描述	读写特性
63：32	Reserved		
31：0	Tn_Com_VAL	Tn_Comparator value (N 为 0-2) ：定时器比较器的值； 当对应的定时器配置为非周期性模式时： ◆ 这个寄存器的值将与主计时器寄存器的值做比较； ◆ 若主计时器的值与比较器的值相等时，则产生定时中断（如果；对应的中断使能打开）。 ◆ 比较器的值不会因为中断的产生而发生变化 若对应的定时器配置为周期性模式时： 当主计时器的值域比较器的值相等时，产生中断(如果对应的中断使能被打开）； 如果产生中断，那么比较器的值将累加最后一次软件写入比较器的值。比如当比较器的值被写入 0x0123h， 那么当主计时器的值为 0x123h 时，产生中断； 比较器的值被硬件修改为 0x246h； 当主计时器的值达到 0x246h 时，产生另外一个中断； 比较器的值被硬件修改为 0x369h。 ◆ 只要产生中断，那么比较器的值都会累加；直到比较器的值达到最大 (0xffffffff) ，那么累加器的值将会继续累加。比如当比较器的值是 FFFF0000h，而最后一次由软件写入比较器的值是 20000。当中断发生后，比较器的值变为 00010000h。	R/W

20 GPIO

龙芯 2K3000 共有 58 个 GPIO 引脚（不包含 ACPI 和 SE 专用 GPIO），4 个为专用 GPIO，其余 54 个与其他功能复用。需注意，GPIO 引脚的编号分为 0-39 和 46-63 两部分。

GPIO 引脚由一组寄存器控制，包括：GPIO 方向控制（GPIO_OEN）、GPIO 输出值（GPIO_O）、GPIO 输入值（GPIO_I）、GPIO 输入中断使能控制（GPIO_INT_EN）、GPIO 输入中断极性控制（GPIO_INT_POL）、GPIO 输入中断边沿性控制（GPIO_INT_EDGE）、GPIO 输入中断清除（GPIO_INT_CLR）、GPIO 输入中断状态（GPIO_INT_STS）。

表 20-1 GPIO 控制寄存器

<table>
<tr><th>寄存器</th><th>大小（位）</th><th>描述</th></tr>
<tr><td>GPIO_OEN</td><td>1</td><td>GPIO 输出使能，低有效。</td></tr>
<tr><td>GPIO_O</td><td>1</td><td>GPIO 输出值。</td></tr>
<tr><td>GPIO_I</td><td>1</td><td>GPIO 输入值。</td></tr>
<tr><td>GPIO_INT_EN</td><td>1</td><td>GPIO 中断使能。</td></tr>
<tr><td>GPIO_INT_POL</td><td>1</td><td>GPIO 中断极性。</td></tr>
<tr><td>GPIO_INT_EDGE</td><td>1</td><td>GPIO 中断边沿性。与 GPIO 中断极性配合控制 GPIO 中断状态的产生。
<table>
<tr><td>POL</td><td>EDGE</td><td>描述</td></tr>
<tr><td>0</td><td>0</td><td>低电平触发中断</td></tr>
<tr><td>1</td><td>0</td><td>高电平触发中断</td></tr>
<tr><td>0</td><td>1</td><td>下降沿触发中断</td></tr>
<tr><td>1</td><td>1</td><td>上升沿触发中断</td></tr>
</table></td></tr>
<tr><td>GPIO_INT_CLR</td><td>1</td><td>GPIO 中断状态清除</td></tr>
<tr><td>GPIO_INT_STS</td><td>1</td><td>GPIO 中断状态</td></tr>
</table>

20.1 访问地址

GPIO 的访问基地址等于 IO 低速设备块的基地址加偏移 0x60000。

2K3000 提供了两种方式来控制 GPIO 引脚。一种是按位控制每个 GPIO 引脚，一种是按字节控制每个 GPIO 引脚。南桥是通过提供两个地址空间来映射 GPIO 控制寄存器实现该功能的。一种是按位映射，一种是按字节来索引控制寄存器的每个比特位。对应的，GPIO 内部的地址空间也分为两部分。

推荐使用后一种方式来控制器 GPIO 引脚。

GPIO 模块的内部寄存器物理地址构成如下：

地址空间	说明
0x800-0xF00	按字节控制寄存器地址
0x0 - 0x70	按位控制寄存器地址

表 20-2 按位控制 GPIO 配置寄存器地址

地址偏移	寄存器	大小（位）	描述
0x00	GPIO_OEN	64	GPIO 输出使能，低有效。每位控制一个 GP IO 引脚。

0x10	GPIO_O	64	GPIO 输出值。每位控制一个 GPIO 引脚。
0x20	GPIO_I	64	GPIO 输入值。每位控制一个 GPIO 引脚。
0x30	GPIO_INT_EN	64	GPIO 中断使能。每位控制一个 GPIO 引脚。
0x40	GPIO_INT_POL	64	GPIO 中断极性。每位控制一个 GPIO 引脚。
0x50	GPIO_INT_EDGE	64	GPIO 中断边沿性。每位控制一个 GPIO 引脚。
0x60	GPIO_INT_CLR	64	GPIO 中断清除。每位控制一个 GPIO 引脚。
0x70	GPIO_INT_STS	64	GPIO 中断状态。每位控制一个 GPIO 引脚。

表 20-3 按字节控制 GPIO 配置寄存器地址

地址偏移	寄存器	大小（字节）	描述
0x800	GPIO_OEN	64	GPIO 输出使能，低有效。每个字节控制一个 GPIO 引脚。
0x900	GPIO_O	64	GPIO 输出值。每个字节控制一个 GPIO 引脚。
0xA00	GPIO_I	64	GPIO 输入值。每个字节控制一个 GPIO 引脚。
0xB00	GPIO_INT_EN	64	GPIO 中断使能。每个字节控制一个 GPIO 引脚。
0xC00	GPIO_INT_POL	64	GPIO 中断极性。每个字节控制一个 GPIO 引脚。
0xD00	GPIO_INT_EDGE	64	GPIO 中断边沿性。每个字节控制一个 GPIO 引脚。
0xE00	GPIO_INT_CLR	64	GPIO 中断清除。每个字节控制一个 GPIO 引脚。
0xF00	GPIO_INT_STS	64	GPIO 中断状态。每个字节控制一个 GPIO 引脚。

20.2 控制寄存器

GPIO 方向控制

地址偏移：00-03h　　属性：R/W
默认值：FFFFFFF0h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_OEN	R/W	对应于 GPIO[31:0]的方向控制。 0：输出 1：输入

地址偏移：04-07h　　属性：R/W
默认值：FFFFFFFFh　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_OEN	R/W	对应于 GPIO[63:32]的方向控制。 0：输出 1：输入

GPIO 输出值

地址偏移：10-13h　　属性：R/W
默认值：0000000Fh　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_O	R/W	对应于 GPIO[31:0]的输出值。

地址偏移：14-17h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_O	R/W	对应于 GPIO[63:32]的输出值。

GPIO 输入值

地址偏移：20-23h　　属性：RO
默认值：N/A　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_I	RO	对应于 GPIO[31:0]的输入值。

地址偏移：24-27h　　属性：RO
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_I	RO	对应于 GPIO[63:32]的输入值。

GPIO 中断使能

地址偏移：30-33h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_EN	R/W	对应于 GPIO[31:0]的输入中断使能。 0：关闭中断 1：使能中断

地址偏移：34-37h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_EN	R/W	对应于 GPIO[63:32]的输入中断使能。 0：关闭中断 1：使能中断

GPIO 中断极性

地址偏移：40-43h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_POL	R/W	对应于 GPIO[31:0]的中断极性。 与中断边沿性配合，组成四种中断触发方式，见下文 GPIO 中断边沿。

地址偏移：44-47h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_POL	R/W	对应于 GPIO[63:32]的中断极性。 与中断边沿性配合，组成四种中断触发方式，见下文 GPIO 中断边沿。

GPIO 中断边沿

地址偏移：50-53h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_EDGE	R/W	对应于 GPIO[31:0]的中断边沿。 与中断极性配合，组成四种中断触发方式。

POL	EDGE	描述
0	0	低电平触发中断
1	0	高电平触发中断
0	1	下降沿触发中断
1	1	上升沿触发中断

地址偏移：54-57h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_EDGE	R/W	对应于 GPIO[63:32]的中断边沿。 与中断极性配合，组成四种中断触发方式。 POL / EDGE / 描述 0 / 0 / 低电平触发中断 1 / 0 / 高电平触发中断 0 / 1 / 下降沿触发中断 1 / 1 / 上升沿触发中断

GPIO 中断清除

地址偏移：60-63h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_CLR	R/W	对应于 GPIO[31:0]的中断清除。 写 1 清除相应 GPIO 位上的中断，随后硬件会自动置 0 中断清除寄存器相应位，不需软件再作处理。

地址偏移：64-67h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_CLR	R/W	对应于 GPIO[63:32]的中断极性。 写 1 清除相应 GPIO 位上的中断，随后硬件会自动置 0 中断清除寄存器相应位，不需软件再作处理。

GPIO 中断状态

地址偏移：70-73h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_STS	R/W	对应于 GPIO[31:0]的中断状态。 1：有中断 0：无中断

地址偏移：74-77h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:0	GPIO_INT_STS	R/W	对应于 GPIO[63:32]的中断状态。 1：有中断 0：无中断

21 DMA 控制器

21.1 概述

2K3000 的 DMA 可以实现从设备到内存，内存到设备的数据传输。支持 DMA 的设备包括 CAN0-3 和 UART1-7。DMA 内部有 4 个通道。

每个通道在配置寄存器都可以配置其优先级，每当遇到两个优先级不同的通道发出 DMA 传输请求时，高优先级会赢得仲裁，低优先级请求会在传输高优先级请求的传输间隙开启。相同优先级的通道，编号越小优先级越高。

DMA 具备循环模式，可用于处理循环缓冲区和连续数据传输，当传输个数变为 0 时，该通道所有寄存器会被复位为传输开启时的值。

21.2 寄存器定义

DMA 控制器的访问基地址等于 IO 低速设备块的基地址加偏移 0x80000。

部分表格内以 x 代替通道编号 1-4。

表 21- 1 DMA 寄存器列表

偏移	名称	描述
0x00	DMA_ISR	DMA 中断状态寄存器
0x04	DMA_IFCR	DMA 中断标志清除寄存器
0x08	DMA_CCR1	DMA 通道 1 配置寄存器
0x0c	DMA_CNDTR1	DMA 通道 1 传输数量寄存器
0x10	DMA_CPAR1	DMA 通道 1 外设地址寄存器
0x14	DMA_CMAR1	DMA 通道 1 储存地址寄存器
0x18		
0x1c	DMA_CCR2	DMA 通道 2 配置寄存器
0x20	DMA_CNDTR2	DMA 通道 2 传输数量寄存器
0x24	DMA_CPAR2	DMA 通道 2 外设地址寄存器
0x28	DMA_CMAR2	DMA 通道 2 储存地址寄存器
0x2c		
0x30	DMA_CCR3	DMA 通道 3 配置寄存器
0x34	DMA_CNDTR3	DMA 通道 3 传输数量寄存器
0x38	DMA_CPAR3	DMA 通道 3 外设地址寄存器
0x3c	DMA_CMAR3	DMA 通道 3 储存地址寄存器
0x40		
0x44	DMA_CCR4	DMA 通道 4 配置寄存器
0x48	DMA_CNDTR4	DMA 通道 4 传输数量寄存器
0x4c	DMA_CPAR4	DMA 通道 4 外设地址寄存器
0x50	DMA_CMAR4	DMA 通道 4 储存地址寄存器

偏移	名称	描述
0x54		

21.2.1 DMA 中断状态寄存器 (DMA_ISR)

偏移量： 0x00

复位值： 0x00000000

表 21- 2 DMA 中断状态寄存器

位域	名称	访问	描述
4*x - 1	TEIFx	R	通道 x 传输错误标志 0：通道 x 无传输错误事件； 1：通道 x 有传输错误事件。
4*x - 2	HTIFx	R	通道 x 传输过半标志 0：通道 x 无传输过半事件； 1：通道 x 有传输过半事件。
4*x - 3	TCIFx	R	通道 x 传输完成标志 0：通道 x 无传输完成事件； 1：通道 x 有传输完成事件。
4*x - 4	GIFx	R	通道 x 全局中断标志 0：通道 x 无传输错误/过半/完成事件； 1：通道 x 有传输错误/过半/完成事件。

21.2.2 DMA 中断标志清除寄存器 (DMA_IFCR)

偏移量： 0x04

复位值： 0x00000000

表 21- 3 DMA 中断标志清除寄存器

位域	名称	访问	描述
4*x - 1	CTEIFx	RW	清除通道 x 传输错误标志 0：无效； 1：清除 DMA_ISR 寄存器中对应的传输错误事件标志。
4*x - 2	CHTIFx	RW	清除通道 x 传输过半标志 0：无效； 1：清除 DMA_ISR 寄存器中对应的传输过半事件标志。
4*x - 3	CTCIFx	RW	清除通道 x 传输完成标志 0：无效； 1：清除 DMA_ISR 寄存器中对应的传输完成事件标志。
4*x - 4	CGIFx	RW	清除通道 x 全局中断标志 0：无效； 1：清除 DMA_ISR 寄存器中对应的传输错误/过半/完成事件标志。

21.2.3 DMA 通道 x 配置寄存器 (DMA_CCRx)

偏移量： 0x08 + 0x14 * (x - 1)

复位值： 0x00000000

表 21- 4 DMA 通道 x 配置寄存器

位域	名称	访问	描述
31:15	Reserved	-	-
14	MEM2MEM	RW	存储器到存储器模式 该位由软件配置和清除。 0：非存储器到存储器模式； 1：启动存储器到存储器模式。
13:12	PL	RW	通道优先级 该位域由软件配置和清除。 00：低；01：中；10：高；11：最高。
11:10	MSIZE	RW	存储器数据宽度 该位域由软件配置和清除。 00：8 位；01：16 位；10：32 位；11：保留。
9:8	PSIZE	RW	外设数据宽度 该位域由软件配置和清除。 00：8 位；01：16 位；10：32 位；11：保留。
7	MINC	RW	存储器地址增量模式 该位域由软件配置和清除。 0：无效；1：有效。
6	PINC	RW	外设地址增量模式 该位域由软件配置和清除。 0：无效；1：有效。
5	CIRC	RW	循环模式 该位域由软件配置和清除。 0：无效；1：有效。
4	DIR	RW	数据传输方向 该位域由软件配置和清除。 0：从外设读；1：从储存器读。
3	TEIE	RW	传输错误中断使能 该位域由软件配置和清除。 0：无效；1：有效。
2	HTIE	RW	传输过半中断使能 该位域由软件配置和清除。 0：无效；1：有效。
1	TCIE	RW	传输完成中断使能 该位域由软件配置和清除。 0：无效；1：有效。
0	EN	RW	通道开启 该位域由软件配置和清除。 0：无效；1：有效。

21.2.4 DMA 通道 x 传输数量寄存器 (DMA_CNDTRx)

偏移量： 0x0c + 0x14 * (x - 1)

复位值： 0x00000000

表 21- 5 DMA 通道 x 传输数量寄存器

位域	名称	访问	描述
31:16	Reserved	-	-
15:0	NDT	RW	数据传输个数 数据传输个数为 0 到 65535，这个寄存器只能通道停用时写入。 通道开启后变为只读，此时读数为待传输个数。

21.2.5 DMA 通道 x 外设地址寄存器 (DMA_CPARx)

偏移量： 0x10 + 0x14 * (x - 1)

复位值： 0x00000000

表 21- 6 DMA 通道 x 外设地址寄存器

位域	名称	访问	描述
31:16	Reserved	-	-
15:0	PADDR	RW	外设地址 外设数据起始地址。 通道开启后变为只读。

21.2.6 DMA 通道 x 储存器地址寄存器 (DMA_CMARx)

偏移量： 0x14 + 0x14 * (x - 1)

复位值： 0x00000000

表 21- 7 DMA 通道 x 储存器地址寄存器

位域	名称	访问	描述
31:16	Reserved	-	-
15:0	MADDR	RW	储存器地址 储存器数据起始地址。 通道开启后变为只读。

21.3 功能描述

21.3.1 配置流程

假设我们要启动一次 DMA 传输，我们需要执行下列初始化操作：

1) 在 DMA_CPARx 设置外设寄存器地址；
2) 在 DMA_CMARx 设置存储器寄存器地址；
3) 在 DMA_CNDTRx 设置要传输的数据个数；
4) 在 DMA_CCRx 设置要通道优先级；
5) 在 DMA_CCRx 设置要数据传输方向，循环模式，外设和存储器增量模式，外设和存储器数据宽度，传输中断使能；
6) 在 DMA_CCRx 设置 ENABLE,启动该通道；

21.3.2 宽度和对齐方式

以 PINC=MINC=1 为例，不同的 PSIZE 和 MSIZE 会按下表所示方式对齐。

表 21- 8 DMA 宽度和对齐方式

源宽度	目的宽度	传输数目	源地址/数据	传输操作	目的地址/数据
8	8	4	0x0：0xB0 0x1：0xB1 0x2：0xB2 0x3：0xB3	1：0x0 读 0xB0，0x0 写 0xB0； 2：0x1 读 0xB1，0x1 写 0xB1； 3：0x2 读 0xB2，0x2 写 0xB2； 4：0x3 读 0xB3，0x3 写 0xB3。	0x0：0xB0 0x1：0xB1 0x2：0xB2 0x3：0xB3
8	16	4	0x0：0xB0 0x1：0xB1 0x2：0xB2 0x3：0xB3	1：0x0 读 0xB0，0x0 写 0x00B0； 2：0x1 读 0xB1，0x2 写 0x00B1； 3：0x2 读 0xB2，0x4 写 0x00B2； 4：0x3 读 0xB3，0x6 写 0x00B3。	0x0：0x00B0 0x2：0x00B1 0x4：0x00B2 0x6：0x00B3
8	32	4	0x0：0xB0 0x1：0xB1 0x2：0xB2 0x3：0xB3	1：0x0 读 0xB0，0x0 写 0x000000B0； 2：0x1 读 0xB1，0x4 写 0x000000B1； 3：0x2 读 0xB2，0x8 写 0x000000B2； 4：0x3 读 0xB3，0xc 写 0x000000B3。	0x0：0x000000B0 0x4：0x000000B1 0x8：0x000000B2 0xc：0x000000B3
16	8	4	0x0：0xB1B0 0x2：0xB3B2 0x4：0xB5B4 0x6：0xB7B6	1：0x0 读 0xB1B0，0x0 写 0xB0； 2：0x1 读 0xB3B2，0x1 写 0xB2； 3：0x2 读 0xB5B4，0x2 写 0xB4； 4：0x3 读 0xB7B6，0x3 写 0xB6。	0x0：0xB0 0x1：0xB2 0x2：0xB4 0x3：0xB6
16	16	4	0x0：0xB1B0 0x2：0xB3B2 0x4：0xB5B4 0x6：0xB7B6	1：0x0 读 0xB1B0，0x0 写 0xB1B0； 2：0x1 读 0xB3B2，0x2 写 0xB3B2； 3：0x2 读 0xB5B4，0x2 写 0xB5B4； 4：0x3 读 0xB7B6，0x3 写 0xB7B6。	0x0：0xB1B0 0x2：0xB3B2 0x4：0xB5B4 0x6：0xB7B6
16	32	4	0x0：0xB1B0 0x2：0xB3B2 0x4：0xB5B4 0x6：0xB7B6	1：0x0 读 0xB1B0，0x0 写 0x0000B1B0； 2：0x1 读 0xB3B2，0x1 写 0x0000B3B2； 3：0x2 读 0xB5B4，0x2 写 0x0000B5B4； 4：0x3 读 0xB7B6，0x3 写 0x0000B7B6。	0x0：0x0000B1B0 0x4：0x0000B3B2 0x8：0x0000B5B4 0xc：0x0000B7B6
32	8	4	0x0：0xB3B2B1B0 0x4：0xB7B6B5B4 0x8：0xBbBaB9B8 0xc：0xBfBeBdBc	1：0x0 读 0xB3B2B1B0，0x0 写 0xB0； 2：0x1 读 0xB7B6B5B4，0x1 写 0xB4； 3：0x2 读 0xBbBaB9B8，0x2 写 0xB8； 4：0x3 读 0xBfBeBdBc，0x3 写 0xBc。	0x0：0xB0 0x1：0xB4 0x2：0xB8 0x3：0xBc
32	16	4	0x0：0xB3B2B1B0 0x4：0xB7B6B5B4 0x8：0xBbBaB9B8 0xc：0xBfBeBdBc	1：0x0 读 0xB3B2B1B0，0x0 写 0xB1B0； 2：0x1 读 0xB7B6B5B4，0x2 写 0xB5B4； 3：0x2 读 0xBbBaB9B8，0x4 写 0xB9B8； 4：0x3 读 0xBfBeBdBc，0x6 写 0xBdBc。	0x0：0xB1B0 0x2：0xB5B4 0x4：0xB9B8 0x6：0xBdBc
32	32	4	0x0：0xB3B2B1B0 0x4：0xB7B6B5B4 0x8：0xBbBaB9B8 0xc：0xBfBeBdBc	1：0x0 读 0xB3B2B1B0，0x0 写 0xB3B2B1B0； 2：0x1 读 0xB7B6B5B4，0x4 写 0xB7B6B5B4； 3：0x2 读 0xBbBaB9B8，0x8 写 0xBbBaB9B8； 4：0x3 读 0xBfBeBdBc，0xc 写 0xBfBeBdBc。	0x0：0xB3B2B1B0 0x4：0xB7B6B5B4 0x8：0xBbBaB9B8 0xc：0xBfBeBdBc

21.3.3 通道映射

外设通道映射见下表。

表 21- 9 外设请求的通道映射

CH1	CH2	CH3	CH4
UART1 RX	UART2 RX	UART1 TX	UART2 TX
UART3 RX	UART4 RX	UART3 TX	UART4 TX
UART5 RX	UART6 RX	UART5 TX	UART6 TX
UART7 RX	-	UART7 TX	-
CAN0 RX	CAN1 RX	CAN2 R	CAN3 RX

22 电源管理模块

桥片电源管理模块提供系统功耗管理功能。支持 Advanced Configuration and Power Interface, Version 4.0a (ACPI)，提供相应的功耗管理功能。

- 系统休眠与唤醒，支持ACPI S3（待机到内存），ACPI S4（待机到硬盘），ACPI S5（软关机），并且支持电源失效检测和自动系统恢复。支持多种唤醒方式（USB，GMAC，电源开关等）。
- 系统时钟控制，模块时钟门控，多种方式调节频率。
- 集成一个看门狗。最大定时时间约82s。

22.1 访问地址

电源管理模块的访问基地址为 IO 低速设备块的基地址加偏移 0x50000。

注意：电源管理模块支持按 4/1 字节访问。当 ACPI_EN 为 0 时，所有寄存器不可访问。

电源管理模块的内部寄存器物理地址构成如下：

地址位	构成	备注
[15:13]	0	保留
[12]	内部路由	0：ACPI 1：DPM
[11:8]	0	保留
[7:0]	REG	内部寄存器地址

22.2 电源级别

表 22-1 ACPI 状态说明

状态	描述
G0/S0	全部工作，该模式下系统全部工作。
G1/S1	暂不支持
G1/S3	Suspend to RAM (STR)，上下文保存到内存
G1/S4	Suspend to Disk (STD)，保存到硬盘，除唤醒电路全部掉电
G2/S5	Soft off，只有唤醒电路上电
G3	Mechanical off，所有供电失效

22.3 ACPI 寄存器描述

本节介绍电源管理相关寄存器。寄存器电压域表示寄存器的该位所属电压域。

PMCON_SOC : SOC General PM Configuration Register

地址偏移	电压域	属性
0x00	SOC	R/W，RO

位域	描述
25	**PWRBTN_LVL - RO** 该位指示当前 PWRBTNn 信号状态。
24:0	保留

PMCON_RESUME : RESUME General PM Configuration Register

地址偏移	电压域	属性
0x04	ACPI	R/W，RO，R/WC

位域	描述
31:14	保留
13	**VSB_GATEn_EN - R/W** 是否使能 VSB_GATEn 功能。0：关闭；1：使能。 如果 RSMRSTn 有效过，该位为 1。重新上电后由系统配置该位。如果主板使用 VSB_GATEn 引脚作为电源管理控制信号，系统软件必须将该位写 1。
12:11	**VSB_GATEn_SLEEP_DLY - R/W** 用来控制 S0 到 S3 时，VSB_GATEn 信号下降沿相对 S3n 下降沿的提前时间 2’b00：休眠时提前 31.25ms； 2’b01：休眠时提前 62.5ms； 2’b10：休眠时提前 125ms； 2’b11：休眠时提前 250ms； 如果 RSMRSTn 有效过，该字段为 2’b0。重新上电后由系统配置该字段。
10:9	**VSB_GATEn_WAKE_DLY - R/W** 用来控制 S3 到 S0 时，VSB_GATEn 信号上升沿相对 S3n 上升沿的延后时间 2’b00：唤醒时延后 125ms； 2’b01：唤醒时延后 250ms； 2’b10：唤醒时延后 500ms； 2’b11：唤醒时延后 1s； 如果 RSMRSTn 有效过，该字段为 2’b0。重新上电后由系统配置该字段。
8	保留
7	**USB_GMAC_OK - R/W** 如果 RSMRSTn 有效过，该位为 0，表示 USB 和 GMAC 没有配置，不能唤醒系统。重新上电后由系统配置该位。
6	**CTT_STS - R/WC** 系统在 S0 状态时发生 temperature trip，系统进入 G2/S5 状态，该位为重新上电后系统检测记录事件状态
5	**CTT_EN - R/W** 使能 temperature trip 保护机制
4:3	保留
2	**SRS (System Reset Status) - R/WC** 0：SYS_RESETn 没有被按下 1：SYS_RESETn 被按下过，系统重新复位后需检查此位并作出相应清除操作。
1	**PWROK_FLR (PWROK Failure) - R/WC** 当系统在 S0 状态时，PWROK 信号变无效该位置 1，软件通过写 1 将该位清除。
0	**DRAM_INIT - R/W** 该位不影响硬件功能，PMON 在进行 DRAM 初始化前将该位置 1，结束 DRAM 初始化后将该位写 0，软件可通过此位检查 DRAM 初始化是否被打断过。

PMCON_RTC : RTC General PM Configuration Register

地址偏移	电压域	属性
0x08	RTC	R/W，R/WC

位域	描述
31:13	保留
12	RTC_DLY_BYPASS - R/W 是否将 RTC 域输入的关键信号进行延迟控制，默认值为 1
11	RESUME_2POWER - R/W RESUME 域与 ACPI 域是否为单独电压域，默认值为 1
10	RTC_REG_LP - R/W 控制 RTC 模块中寄存器是否进入低功耗模式 0：非低功耗 1：低功耗

9	CMOS_CLKDISABLE - R/W 控制 CMOS 模块的时钟是否关闭 0：打开 1：关闭
8:7	保留
6:5	**S3_ASSERTION_WIDTH - R/W** 这 2 bit 值代表 S3n 信号从有效到重新无效的最小时间间隔。 11: 1s 10: 125ms 01: 1ms 00: 60us
4:3	**S4_ASSERTION_WIDTH - R/W** 这 2 bit 值代表 S4n 信号从有效到重新无效的最小时间间隔。 11: 4 s 10: 2 s 01: 1 s 00: 125 us
2	**S4_ASSERTION_EN - R/W** 0: S4n 信号从有效到重新无效的间隔为几个 RTC 周期。 1: S4n 信号从有效到重新无效的间隔为 S4_ASSERTION_WIDTH 决定。
1	**PWR_FLR (Power Failure) - R/WC.** 该位在 RTC 域，只能被 RTC_RSTn 复位。 如果置 1 表示系统发生过电源失效（进入 G3 状态），即除 RTC 以外所有供电失效过（RSMRSTn 有效过），软件通过写 1 将该位清除。
0	**AFTERG3_EN - R/W** 该位决定系统进入 G3 状态后重新供电后的动作。 0：系统在供电恢复后将自动回复到 S0 状态。 1：系统将恢复到 S5 状态，如果发生电源失效时系统在 S4 状态，重新供电后系统恢复到 S4 状态。 该位会被 power button override 和 thermal trip 事件置 1。

PM1_STS : Power Management 1 Status Register

地址偏移	电压域	属性
0x0C	ACPI/RTC/SOC	R/WC

位域	描述	电压域
31:16	Reserved	
15	**WAK_STS (Wake Status) - R/WC** 0：软件写 1 将该位清除。 1：如果系统从任何一个休眠状态被唤醒事件唤醒，硬件将该位置 1。	ACPI
14	**PCIEXP_WAKE_STS - R/WC.** 1：PCIe 唤醒事件发生 0：软件写 1 将该位清除	ACPI
13:12	Reserved	
11	**PRBTNOR_STS (Power Button Override Status) - R/WC** 0：软件写 1 将该位清除 1：当 power button override 发生时，该位置 1，系统无条件进入 G2/S5 状态，同时将 AFTERG3_EN 位置 1。	RTC
10	**RTC_STS (RTC Status) - R/WC** 0：软件写 1 将该位清除 1: 当 RTC 产生 alarm 时该位置 1。此外当 RTC_EN 有效时，该位产生唤醒事件。	ACPI
9	Reserved	
8	**PWRBTN_STS (Power Button Status) - R/WC** 0：软件写 1 将该位清除。Thermal trip 会清除该位。 1：当按下 PWRBTNn 保持 16ms 以上（4s 以下）时，该位会置 1。 在 S0 状态时，当 PWRBTN_EN 和 PWRBTN_STS 同时有效时，将产生中断。 在 S3-S5 任何休眠状态时，如果 PWRBTN_STS 置位，系统将恢复。	ACPI
7:1	保留	

0	**TMROF_STS (PM Timer Overflow Status) - R/WC** 0：软件写 1 将该位清除 1：当 24bit 计数器（一个时钟周期 20ns）的最高位翻转时，该位置 1。记时功能推荐使用 HPET 完成。	SOC

PM1_EN : Power Management 1 Enable Register

地址偏移	电压域	属性
0x10	ACPI/RTC/SOC	R/W

位域	描述	电压域
31:15	保留	
14	**PCIEXP_WAKE_DIS - R/W** 当置位时，不产生PCIe唤醒事件，但是该位的值不影响PCIEXP_WAKE_STS的值。	ACPI
13:11	保留	
10	**RTC_EN (RTC Event Enable) - R/W** RTC 唤醒和中断使能	rtc
9	保留	
8	**PWRBTN_EN (Power Button Enable) - R/W.** PWRBTN 中断事件产生使能，该位不影响 PWRBTN 唤醒事件产生。	ACPI
7:1	保留	
0	**TMROF_EN (PM Timer Overflow Enable) - R/W.** 如果该位置位，TMROF_STS 将产生中断。	SOC

PM1_CNT : Power Management 1 Control Register

地址偏移	电压域	属性
0x14	ACPI/RTC/SOC	R/W

位域	描述	电压域
31:14	保留	
13	**SLP_EN (Sleep Enable) - R/W.** 该位写 1 将会使系统进入 SLP_TYP 声明的休眠状态，进入相关休眠状态后该位自动恢复为 0。	ACPI
12:10	**SLP_TYP (Sleep Type) - R/W.** 该 3bit 表示系统的休眠状态。 000: 表示 S0 状态。 001: Reserved. 010: Reserved. 011: Reserved. 100: Reserved. 101: Suspend-to-RAM. S3n 信号有效，进入 S3 状态。 110: Suspend-to-Disk. S3n, S4n 信号有效，进入 S4 状态。 111: Soft Off. S3n, S4n, S5n 信号有效，进入 S5 状态。	rtc
9:1	Reserved	
0	**INT_EN - R/W** 中断使能开关，使能电源管理控制器中断信号的产生。	SOC

PM1_TMR : Power Management 1 Timer

地址偏移	电压域	属性
0x18	SOC	RO

位域	描述
31:24	保留
23:0	**TMR_VAL (Timer Value) - RO.** 计数器计数，周期 8ns。当 23 位翻转时，置位 TNROF_STS 位。 推荐使用 HPET。

GPE0_STS : General Purpose Event0 Status Register

地址偏移	电压域	属性
0x28	ACPI	R/WC

位域	描述
31:16	GPI_STS - R/WC 0：软件写 1 将该位清除。 1：当 ACPI_GPIO 发生中断事件时这些位被置位，当 GPI_EN 位有效时，产生唤醒事件或中断。
15:10	**USB[5:0]_STS - R/WC.** 只有第 10 位有意义，15:11 位暂无意义。 0：软件写 1 将该位清除。 1：当 USB 发生 wake 事件时这些位被置位，当 USBx_EN 位有效时，产生唤醒事件或中断。
9:6	保留
5	**GMAC2_STS - R/WC.** 0：软件写 1 将该位清除。 1：当 GMAC2 发生 wake 事件时这些位被置位，当 GMAC2_EN 位有效时，产生唤醒事件或中断。
4	保留
3	**CTW_STS - R/WC.** thermal warning 发生
2	**CTA_STS - R/WC.** thermal alert 发生
1:0	保留

GPE0_EN : General Purpose Event0 Status Register

地址偏移	电压域	属性
0x2C	ACPI/RTC	R/W

位域	描述	电压域
31:16	GPI_EN - R/W 0：无效 1：使能 GPIO 中断唤醒事件，GPIO 中断类型可配置	
15:10	**USB[5 :0]_EN - R/W.** 0：无效 1：使能 USBx_STS 产生唤醒事件，当回到 S0 将产生中断信号。	RTC
9:6	保留	
5	**GMAC2_EN - R/W.** 0：无效 1：使能 GMAC2_STS 产生唤醒事件，当回到 S0 将产生中断信号。	RTC
4	保留	
3	**CTW_EN - R/W** 使能 THERMAL WARNING 产生中断。	
2	**CTA_EN - R/W** 使能 THERMAL ALERT 产生中断。	
1:0	保留	

RST_CNT : Reset Control Register

地址偏移	电压域	属性
0x30	SOC	R/W

位域	描述
31:2	保留
1	**WD_EN - R/W** Watch dog 功能使能
0	**OS_RST - R/W** 软件写该位使系统复位。

WD_SET : Watch Dog Set Register

地址偏移	电压域	属性
0x34	SOC	WO
位域	**描述**	
31:1	保留	
0	当 WD_EN 为 1 时，写该位将重填内部 watch dog 计数器，充填的值为 WD_Timer. 注意，watch dog 计数器的工作频率为 50MHz。	

WD_Timer : Watch Dog Timer Register

地址偏移	电压域	属性
0x38	SOC	R/W
位域	**描述**	
31:0	该寄存器的值为 watch dog 重填的值，复位值为全 1 。	

THSENS_CNT : CPU Thermal Sensor Control Register

地址偏移	电压域	属性
0x4C	SOC	R/W

位域	描述
31:24	WARNING_TMP - R/W CPU 温度超过该值，如果被使能，将产生中断。如果温度位降低到该值以下，不会重复产生中断。 推荐操作：系统采取降低功耗操作。
23:16	ALERT_TMP - R/W CPU 温度超过该值，如果被使能，将产生中断。如果温度位降低到该值以下，不会重复产生中断。 推荐操作：采取正常关闭系统操作
15:8	TRIP_TMP - R/W CPU 温度超过该值，如果被使能，系统无条件进入 G2/S5 状态。
7:0	保留

GEN_RTC_1 : General RTC Register 1

地址偏移	电压域	属性
0x50	RTC	R/W
位域	**描述**	
31:0	RTC 通用寄存器	

GEN_RTC_2 : General RTC Register 2

地址偏移	电压域	属性
0x54	RTC	R/W
位域	**描述**	
31:0	RTC 通用寄存器	

ACPI_GPIO_O : ACPI 域 GPIO 输出

地址偏移	电压域	属性
0x80	ACPI	R/W
位域	**描述**	
31:16	保留	
15:0	ACPI 域 GPIO 输出值	

ACPI_GPIO_OEN ： ACPI 域 GPIO 输出使能

地址偏移	电压域	属性
0x84	ACPI	R/W
位域	**描述**	
31:16	保留	
15:0	ACPI 域 GPIO 输出使能控制，低有效	

ACPI_GPIO_I ： ACPI 域 GPIO 输入

地址偏移	电压域	属性
0x88	ACPI	R/W
位域	**描述**	
31:16	保留	
15:0	ACPI 域 GPIO 输入值	

ACPI_GPIO_POL ： ACPI 域 GPIO 中断极性

地址偏移	电压域	属性
0x90	ACPI	R/W
位域	**描述**	
31:16	保留	
15:0	ACPI 域 GPIO 中断极性控制，1 代表高电平/上升沿，0 代表低电平/下降沿	

ACPI_GPIO_EDGE ： ACPI 域 GPIO 中断边沿设置

地址偏移	电压域	属性
0x94	ACPI	R/W
位域	**描述**	
31:16	保留	
15:0	ACPI 域 GPIO 中断边沿设置，1 代表边沿类型中断，0 代表电平类型中断	

ACPI_GPIO_DUALEDGE ： ACPI 域 GPIO 中断双沿设置

地址偏移	电压域	属性
0x98	ACPI	R/W
位域	**描述**	
31:16	保留	
15:0	ACPI 域 GPIO 中断是否为双沿触发，1 代表双沿触发，0 代表单沿触发	

23 RTC

23.1 概述

实时时钟（RTC）单元可以在主板上电后进行配置，当主板断电后，该单元仍然运作，可以仅靠板上的电池供电就正常运行。RTC 单元运行时电流仅几个微安。

RTC 包含振荡器，结合外部 32.768KHZ 晶体产生工作时钟。该时钟用于时间信息的维护以及产生各种定时和计数中断。

RTC 模块中包含两个计数器，分别为 TOY（Time of Year）计数器和 RTC 计数器。其中 TOY 计数器按年月日时分秒计数，精度为以 0.1 秒；RTC 计数器以 32.768KHz 时钟计数，宽度为 32 位。

23.2 访问地址

RTC 模块的访问基地址为 IO 低速设备块的基地址加偏移 0x50100。RTC 模块的内部寄存器物理地址构成如下：

地址位	构成	备注
[15:9]	0	保留
[8]	1	保留
[7:0]	REG	内部寄存器地址

注：当 ACPI_EN 为 0 时，所有寄存器不可访问。

23.3 寄存器描述

23.3.1 寄存器地址列表

名称	偏移地址	位宽	RW	描述
sys_toytrim	0x20	32	RW	软件必须初始化为 0
sys_toywrite0	0x24	32	W	TOY 低 32 位数值写入
sys_toywrite1	0x28	32	W	TOY 高 32 位数值写入
sys_toyread0	0x2C	32	R	TOY 低 32 位数值读出
sys_toyread1	0x30	32	R	TOY 高 32 位数值读出
sys_toymatch0	0x34	32	RW	TOY 定时中断 0
sys_toymatch1	0x38	32	RW	TOY 定时中断 1
sys_toymatch2	0x3C	32	RW	TOY 定时中断 2
sys_rtcctrl	0x40	32	RW	TOY 和 RTC 控制寄存器 软件必须初始化
sys_rtctrim	0x60	32	RW	软件必须初始化为 0
sys_rtcwrite0	0x64	32	W	RTC 定时计数写入
sys_rtcread0	0x68	32	R	RTC 定时计数读出
sys_rtcmatch0	0x6C	32	RW	RTC 时钟定时中断 0
sys_rtcmatch1	0x70	32	RW	RTC 时钟定时中断 1
sys_rtcmatch2	0x74	32	RW	RTC 时钟定时中断 2

23.3.2 SYS_TOYWRITE0

中文名： TOY 计数器低 32 位数值

寄存器位宽： [31：0]

偏移量： 0x24

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:26	TOY_MONTH	W		月，范围 1~12
25:21	TOY_DAY	W		日，范围 1~31
20:16	TOY_HOUR	W		小时，范围 0~23
15:10	TOY_MIN	W		分，范围 0~59
9:4	TOY_SEC	W		秒，范围 0~59
3:0	TOY_MILLISEC	W		0.1 秒，范围 0~9

23.3.3 SYS_TOYWRITE1

中文名： TOY 计数器高 32 位数值

寄存器位宽： [31：0]

偏移量： 0x28

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:0	TOY_YEAR	W		年，范围 0～16383

23.3.4 SYS_TOYREAD0

中文名： TOY 计数器低 32 位数值

寄存器位宽： [31：0]

偏移量： 0x2C

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:26	TOY_MONTH	R		月，范围 1~12
25:21	TOY_DAY	R		日，范围 1~31
20:16	TOY_HOUR	R		小时，范围 0~23
15:10	TOY_MIN	R		分，范围 0~59
9:4	TOY_SEC	R		秒，范围 0~59
3:0	TOY_MILLISEC	R		0.1 秒，范围 0~9

23.3.5 SYS_TOYREAD1

中文名： TOY 计数器高 32 位数值

寄存器位宽： [31：0]

偏移量： 0x30

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:0	TOY_YEAR	R		年，范围 0～16383

23.3.6 SYS_TOYMATCH0/1/2

中文名： TOY 计数器中断寄存器 0/1/2

寄存器位宽： [31：0]

偏移量： 0x34/38/3C

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:26	YEAR	RW		年，范围 0～63
25:22	MONTH	RW		月，范围 1~12
21:17	DAY	RW		日，范围 1~31
16:12	HOUR	RW		小时，范围 0~23
11:6	MIN	RW		分，范围 0~59
5:0	SEC	RW		秒，范围 0~59

23.3.7 SYS_RTCCTRL

中文名： RTC 定时器中断寄存器 0/1/2

寄存器位宽： [31：0]

偏移量： 0x40

复位值：无

位域	位域名称	访问	缺省	描述
31:24	保留	R	0	保留，置 0
23	ERS	R	0	REN (bit13) 写状态
22:21	保留	R	0	保留，置 0
20	RTS	R	0	Sys_rtctrim 写状态
19	RM2	R	0	Sys_rtcmatch2 写状态
18	RM2	R	0	Sys_rtcmatch2 写状态
17	RM0	R	0	Sys_rtcmatch0 写状态
16	RS	R	0	Sys_rtcwrite 写状态
15	保留	R	0	保留，置 0
14	保留	R	0	保留，置 0
13	REN	R/W	0	RTC 使能，高有效。需要初始化为 1
12	保留	R	0	保留，置 0
11	TEN	R/W	0	TOY 使能，高有效。需要初始化为 1
10	保留	R	0	保留，置 0
9	保留	R	0	保留，置 0
8	EO	R/W	0	0：32.768k 晶振禁止； 1：32.768k 晶振使能
7	保留	R	0	保留，置 0
6	保留	R	0	保留，置 0
5	32S	R	0	0：32.768k 晶振不工作； 1：32.768k 晶振正常工作。

4	保留	R	0	保留，置 0
3	TM2	R	0	Sys_toymatch2 写状态
2	TM1	R	0	Sys_toymatch1 写状态
1	TM0	R	0	Sys_toymatch0 写状态
0	TS	R	0	Sys_toywrite 写状态

23.3.8 SYS_RTCWRITE

中文名： RTC 计数器写入端口

寄存器位宽： [31：0]

偏移量： 0x64

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:0	RTC	W		

23.3.9 SYS_RTCREAD

中文名：RTC 计数器读出端口

寄存器位宽： [31：0]

偏移量： 0x68

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:0	RTC	R		

23.3.10 SYS_RTCMATCH0/1/2

中文名： RTC 定时器中断寄存器 0/1/2

寄存器位宽： [31：0]

偏移量： 0x6C/70/74

复位值： 0x00000000

位域	位域名称	访问	缺省	描述
31:26	RTC	RW		

24 GMAC 控制器 (D3:F0/1)

桥片集成了 2 个 GMAC 控制器，即 GMAC0/1，分别为 Device 3 的功能 0/1。

24.1 GMAC 配置寄存器

表 24-1 GMAC 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A23h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	00h	RO
0Bh	BCC	Base Class Code	02h	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W,RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

PCICMD-PCI 命令寄存器 (GMAC-D3:F0, D3:F1)

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 GMAC 控制寄存器的访问。 0：禁止访问； 1：使能对 GMAC 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

25 USB 控制器 (D4:F0, D25:F0)

25.1 总体概述

2K3000 的 USB 端口特性如下：

- USB Rev 3.0 协议
- 兼容 USB Rev 1.1 、USB Rev 2.0 协议
- 兼容 XHCI Rev1.1 协议
- 支持 4 个 USB3.0 端口，每个端口都可挂 SS、LS、FS 或 HS 设备
- 支持 4 个 USB2.0 端口，每个端口都可挂 LS、FS 或 HS 设备

25.2 XHCI 控制器

25.2.1 XHCI 配置寄存器

表 25-1 USB-XHCI 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A34h (USB3) 7A44h (USB2)	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	30h	RO
0Ah	SCC	Sub Class Code	03h	RO
0Bh	BCC	Base Class Code	0Ch	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W,RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

25.2.2 PCICMD-PCI 命令寄存器 (USB XHCI-D25:F0)

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 USB XHCI 控制寄存器的访问。 0：禁止访问； 1：使能对 USB XHCI 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

25.2.3 XHCI 基本操作寄存器

本节不具体列出协议里的寄存器的描述，可以在 XHCI Rev1.1 协议中查找，下表列出了相关域和在协议中的章节。

地址偏移	简称	默认值	描述
0 to CAPLENGTH	Capability Registers	Up to 256 Bytes	Refer to xhci specification Section5.3
CAPLENGTH to CAPLENGTH+BFFh	Operational Registers	Up to 3K Bytes	Refer to xhci specification Section5.4
Pointed to by the Capability Registers	RUN-time Registers	Up to 32800 Bytes	Refer to xhci specification Section5.5
Pointed to by the Capability Registers	Doorbell Array	Up to 1K Bytes	Refer to xhci specification Section5.6

26 图形处理器 (D6:F0)

26.1 概述

2K3000 采用龙芯第二代 GPU 架构，支持图形渲染和通用计算，主要特性如下：

- 支持 OpenGL 3.3 3D 图形流水线
  - 支持 VS/GS/FS 着色器
  - 支持兼容模式
- 集成独立的 2D 贴图流水线
  - 支持单色/模板填充
  - 支持源/目标区域重叠
  - 支持 90/180/270 旋转
  - 支持放大贴图
- 支持 OpenCL 3.0 计算流水线
  - 支持 4 个计算核并发
  - 支持 1-3 维任务空间
- 单个流处理器内集成
  - 64 个 32 位 ALU，支持单精度浮点运算
  - 4 个 64 位 ALU，支持双精度浮点运算（不含除法）
  - 16 个特殊函数单元（SFU），支持三角函数、指数、对数、开方、倒数
  - 4 个张量单元（TU），共 2048 个 INT8 乘加部件
  - 4 个纹理单元
  - 128KB 共享存储器
  - 16KB 纹理/数据缓存
- 集成 MMU，支持 40 位虚地址空间，支持 8 个图形/计算进程
- 2K3000 硬件配置
  - 两个流处理器

26.2 GPU 配置寄存器

表 26-1 GPU 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A35h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO

0Ah	SCC	Sub Class Code	02h	RO
0Bh	BCC	Base Class Code	03h	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W,RO
18h-1Fh	GMEM_BAR	Graphic Memory Base Address Register	0000000000000004h	R/W,RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

26.2.1 PCICMD-PCI 命令寄存器 (GPU-D6:F0)

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 GPU 控制寄存器的访问。 0：禁止访问； 1：使能对 GPU 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

27 显示控制器 (D6:F1)

27.1 概述

显示控制器从内存中取帧缓冲和光标信息输出到外部显示接口上。

显示控制器支持的特性包括：

- 一路 HDMI 接口显示
- 一路 EDP 接口显示
- 一路 DP 接口显示
- 每路显示最大支持至 4K@30Hz
- RGB444,RGB555,RGB565,RGB888 四种色深
- 输出抖动和伽马校正
- 可切换的双路线性帧缓冲
- 中断和软复位

27.2 DC 配置寄存器 (D6:F1)

表 27-1 DC 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A46h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	03h	RO
0Bh	BCC	Base Class Code	0Ch	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W, RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

27.2.1 PCICMD-PCI命令寄存器 (DC-D6:F1)

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留

1	Memory Space Enable	R/W	该位用来控制是否使能对 DC 控制寄存器的访问。 0：禁止访问； 1：使能对 DC 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

27.3 寄存器地址空间分布

寄存器类别		地址空间	基地址
DC	DC0	0x0000~0x03ff	0x0000
	DC1	0x0400~0x07ff	
	reserved	0x0800~0x0fff	
HDMI	HDMI0	0x1000~0x13ff	0x1000
	reserved	0x1400~0x1fff	
EDP/DP	EDP	0x2000~0x23ff	0x2000
	DP	0x2400~0x27ff	
	reserved	0x2800~0x2fff	
HDMI PHY	PHY0	0x3000~0x33ff	0x3000
	reserved	0x3400~0x3fff	
EDP/DP PHY	PHY0	0x4000~0x43ff	0x4000
	PHY1	0x4400~0x47ff	
	reserved	0x4800~0x4fff	

注：DC0 对应 EDP，DC1 对应 HDMI 和 DP;以下所有与光标相关的寄存器、DC GPIO 配置相关寄存器和中断寄存器均位于 DC0 地址空间内

27.4 DC 控制寄存器

27.4.1 帧缓冲配置寄存器

寄存器名	偏移地址	R/W	描述	复位值
frameBufferConfig	0x0	R/W	帧缓冲配置寄存器	00000000h
frameBufferConfig	**bit**	**描述**		**初始值**
Reset	20	从值 1 变为 0 时软复位		0
DC_DMA	17:16	DMA 取数字节数 0：256 1：128 2：64 3：32		0
Gamma	12	写 1 使能伽马校正		0
FB number	11	指示当前正在使用的缓冲区号，只读		0
Switch Panel	9	写 1 表示使用另一路显示输出，两路配合可实现两路显示交换、拷贝		0
Output Enable	8	写 1 使能显示输出		0
FB switch	7	写 1 使能缓冲区切换，下一帧完成切换		0
TileModeEn	4	0:Linear 模式 1:tile 4x4 模式		0
Format	2:0	色深格式： 0：none 1：RGB444 2：RGB555 3：RGB565 4：RGB888		0

27.4.2 帧缓冲地址寄存器 0

寄存器名	偏移地址	R/W	描述	复位值
frameBufferAddr0	0x4	R/W	帧缓冲地址寄存器 0。	00000000h
frameBufferAddr0	**bit**	**描述**		**初始值**
Address	31:0	缓冲区 0 在内存中的物理地址		0

27.4.3 帧缓冲高地址寄存器 0

寄存器名	偏移地址	R/W	描述	复位值
frameBufferAddr0High	0xc	R/W	帧缓冲高地址寄存器 0。	00000000h
frameBufferAddr0High	bit	描述		初始值
Address	31:0	缓冲区 0 在内存中的物理地址		0

27.4.4 帧缓冲地址寄存器 1

寄存器名	偏移地址	R/W	描述	复位值
frameBufferAddr1	0x8	R/W	帧缓冲地址寄存器 1。	00000000h
frameBufferAddr1	**bit**	**描述**		**初始值**
Address	31:0	缓冲区 1 在内存中的物理地址		0

27.4.5 帧缓冲高地址寄存器 1

寄存器名	偏移地址	R/W	描述	复位值
frameBufferAddr1High	0x10	R/W	帧缓冲地址寄存器 1。	00000000h
frameBufferAddr1High	**bit**	**描述**		**初始值**
Address	31:0	缓冲区 1 在内存中的物理地址		0

27.4.6 Meta0 低地址寄存器

寄存器名	偏移地址	R/W	描述	复位值
Meta0BufferAddrLow	0x14	R/W	Meta0 低地址寄存器	00000000h
Meta0BufferAddrLow	**bit**	**描述**		**初始值**
Address	31:0	Meta0 缓冲区在内存中的物理地址		0

27.4.7 Meta0 高地址寄存器

寄存器名	偏移地址	R/W	描述	复位值
Meta0BufferAddrHigh	0x1c	R/W	Meta0 高地址寄存器	00000000h
Meta0BufferAddrHigh	**bit**	**描述**		**初始值**
Address	31:0	Meta0 缓冲区在内存中的物理地址高位		0

27.4.8 Meta1 低地址寄存器

寄存器名	偏移地址	R/W	描述	复位值
Meta1BufferAddrLow	0x18	R/W	Meta1 低地址寄存器	00000000h
Meta1BufferAddrLow	**bit**	**描述**		**初始值**
Address	31:0	Meta1 缓冲区在内存中的物理地址		0

27.4.9 Meta1 高地址寄存器

寄存器名	偏移地址	R/W	描述	复位值
Meta1BufferAddrHigh	0x20	R/W	Meta1 高地址寄存器	00000000h
Meta1BufferAddrHigh	**bit**	**描述**		**初始值**
Address	31:0	Meta1 缓冲区在内存中的物理地址高位		0

27.4.10 帧缓冲跨度寄存器

寄存器名	偏移地址	R/W	描述	复位值
frameBufferStride	0x24	R/W	帧缓冲跨度寄存器：	00000000h
frameBufferStride	**bit**	**描述**		**初始值**
Stride	31:0	显示屏一行的字节数。 需要注意的是根据像素计算出的字节数应按 256 字节向上取证。		0

27.4.11 帧缓冲初始字节寄存器

寄存器名	偏移地址	R/W	描述	复位值
frameBufferOrigin	0x28	R/W	帧缓冲初始字节寄存器	00000000h
frameBufferOrigin	**bit**	**描述**		**初始值**
Origin	31:0	显示屏左侧原有字节数，一般配 0 即可。		0

27.4.12 颜色抖动配置寄存器

寄存器名	偏移地址	R/W	描述	复位值
ditherConfig	0x40	R/W	颜色抖动配置寄存器	00000000h
ditherConfig	**bit**	**描述**		**初始值**
Enable	31	写 1 使能颜色抖动功能。		0
RedSize	19:16	红色域宽度		0
GreenSize	11:8	绿色域宽度		0
BlueSize	3:0	蓝色域宽度		0

27.4.13 颜色抖动查找表低位寄存器

寄存器名	偏移地址	R/W	描述	复位值
ditherTableLow	0x44	R/W	颜色抖动查找表低位寄存器	00000000h
ditherTableLow	**bit**	**描述**		**初始值**
Y1_X3	31:28	坐标 (3,1) 处的比较值。		0
Y1_X2	27:24	坐标 (2,1) 处的比较值。		0
Y1_X1	23:20	坐标 (1,1) 处的比较值。		0
Y1_X0	19:16	坐标 (0,1) 处的比较值。		0
Y0_X3	15:12	坐标 (3,0) 处的比较值。		0
Y0_X2	11:8	坐标 (2,0) 处的比较值。		0
Y0_X1	7:4	坐标 (1,0) 处的比较值。		0
Y0_X0	3:0	坐标 (0,0) 处的比较值。		0

27.4.14 颜色抖动查找表高位寄存器

寄存器名	偏移地址	R/W	描述	复位值
ditherTableHigh	0x48	R/W	颜色抖动查找表高位寄存器	00000000h
ditherTableLow	**bit**	**描述**		**初始值**
Y3_X3	31:28	坐标 (3,3) 处的比较值。		0
Y3_X2	27:24	坐标 (2,3) 处的比较值。		0
Y3_X1	23:20	坐标 (1,3) 处的比较值。		0
Y3_X0	19:16	坐标 (0,3) 处的比较值。		0
Y2_X3	15:12	坐标 (3,2) 处的比较值。		0
Y2_X2	11:8	坐标 (2,2) 处的比较值。		0
Y2_X1	7:4	坐标 (1,2) 处的比较值。		0
Y2_X0	3:0	坐标 (0,2) 处的比较值。		0

27.4.15 颜色抖动说明

颜色抖动功能用于根据一定规则来增强像素值。

在下面的这个例子中，将解释颜色抖动功能的实现步骤。

首先，确定是对哪一数据位进行增强（即该数据位加 1），为此需要配置寄存器 Display Dither Configuration。比如配置 RedSize 为 6（1 到 8 之间，包含 1 和 8），则表明对从 MSB 位数起第 6 位进行增强，即 RedColor[7:0]的 RedColor[2]位增强。GreenSize 和 BlueSize 同理。

其次，需要建立查找表，即配置寄存器 Display Dither Table。

查找表包含 16 个条目，即 16 个阈值，每个条目 4 位宽。

查找表通过屏幕像素计数器的横坐标 x 的最低两位 x[1:0]和纵坐标 y 的最低两位 y[1:0]进行索引，得到一个阈值 U[3:0]。

对应屏幕（x，y）位置的像素值的最低四位被拿来跟查到的这个阈值比较。

如果 RedColor[3:0] > U[3:0]，并且 RedColor[7:2]不是 111111b，则 RedColor[2]位加 1，实现颜色增强。

27.4.16 液晶面板配置寄存器

寄存器名	偏移地址	R/W	描述	复位值
panelConfig	0x4c	R/W	液晶面板配置寄存器	101h
panelConfig	**bit**	**描述**		**初始值**
ClockPol	9	时钟极性，写 1 取反		0
ClockEn	8	时钟使能，写 1 使能		1
DEPol	1	数据使能极性，写 1 取反		0
DE	0	数据使能，写 1 使能		1

27.4.17 水平显示宽度寄存器

寄存器名	偏移地址	R/W	描述	复位值
HDisplay	0x54	R/W	行水平显示宽度寄存器	00000000h
HDisplay	**Bit**	**描述**		**初始值**
Total	28:16	显示屏一行的总像素数（包括非显示区）		0
Display	12:0	显示屏一行中显示区的像素数		0

27.4.18 行同步配置寄存器

寄存器名	偏移地址	R/W	描述	复位值
Hsync	0x58	R/W	行同步配置寄存器	40000000h
Hsync	**Bit**	**描述**		**初始值**
Pol	31	行同步极性，写 1 取反		0
Pulse	30	行同步使能，写 1 使能		1
End	28:16	行同步结束时的像素数		0
Start	12:0	行同步开始时的像素数		0

27.4.19 垂直显示高度寄存器

寄存器名	偏移地址	R/W	描述	复位值
VDisplay	0x68	R/W	垂直显示高度寄存器	00000000h
VDisplay	**Bit**	**描述**		**初始值**
Total	27:16	显示屏一列的总像素数（包括非显示区）		0
Display	11:0	显示屏一列中显示区的像素数		0

27.4.20 场同步配置寄存器

寄存器名	偏移地址	R/W	描述	复位值
VSync	0x6c	R/W	场同步配置寄存器	40000000h
VSync	**Bit**	**描述**		**初始值**
Pol	31	场同步极性，写 1 取反		0
Pulse	30	场同步使能，写 1 使能		1
End	27:16	场同步结束时的像素数		0
Start	11:0	场同步开始时的像素数		0

27.4.21 行场同步偏移配置寄存器

寄存器名	偏移地址	R/W	描述	复位值
SyncDeviation	0x5c	R/W	场同步配置寄存器	00000000h
SyncDeviation	**Bit**	**描述**		**初始值**
SyncDeviation En	31	行场同步偏移使能，不使能则场同步信号按像素行起落		0
SyncDeviationNum	12:0	行场同步偏移值，单位为像素点，数值为行同步起始位置减场同步起始位置		0

27.4.22 当前显示位置寄存器

寄存器名	偏移地址	R/W	描述	复位值
DisplayCurrentLocation	0x70	R	当前显示位置寄存器	00000000h
DisplayCurrentLocation	**Bit**	**描述**		**初始值**
Current Location X	31:16	当前 X 的位置		0
Current Location Y	15:0	当前 Y 的位置		0

27.4.23 伽马校正目录寄存器

寄存器名	偏移地址	R/W	描述	复位值
GammaIndex	0x74	R/W	伽马校正目录寄存器	00000000h
GammaIndex	**bit**	**描述**		**初始值**
Index	7:0	表示从 0-255 颜色值之间的哪一项开始进行 Gamma 调整，一般设 0。只需配一次，此后该值硬件会自增。		0

27.4.24 伽马校正值寄存器

寄存器名	偏移地址	R/W	描述	复位值
GammaData	0x78	R/W	伽马校正值寄存器	00000000h
GammaIndex	**bit**	**描述**		**初始值**
Red	23:16	Gamma 调整的红色域，将 Gamma Index 指示的值调整为当前域的值		0
Green	15:8	Gamma 调整的绿色域，将 Gamma Index 指示的值调整为当前域的值		0
Blue	7:0	Gamma 调整的蓝色域，将 Gamma Index 指示的值调整为当前域的值		0

Gamma 调整模块包含三个查找表，一个负责红色，一个负责绿色，一个负责蓝色。

查找表可以通过寄存器改写。查找表只可写。

考虑一个 Gamma 调整如下：（原色，调整色）。当设置 Gamma 颜色查找表时，我们应该按照颜色值大小顺序配置寄存器.。

如果我们想要一个调整为 （0，0） （1，2） （2，5） （3，6）……，先对寄存器 Gamma Index 配置 0，表示 gamma 调整从 0 开始，然后对 Gamma Data 寄存器配置 256 次完成整个配置过程：0，2，5，6……

27.4.25 中断寄存器 0

寄存器名	偏移地址	R/W	描述	复位值
Interrupt	0x7c	R/W	中断寄存器 0	00000000h
Interrupt	**bit**	**描述**		**初始值**
Display1_Vsync	0	1 号显示单元产生了 Vsync；写 1 清 0		0
Dispaly1_Hsync	1	1 号显示单元产生了 Hsync；写 1 清 0		0
Display0_Vsync	2	0 号显示单元产生了 Vsync；写 1 清 0		0
Display0_Hysnc	3	0 号显示单元产生了 Hsync；写 1 清 0		0
CursorReadEnd	4	光标数据读取结束；写 1 清 0		0
FB1ReadEnd	5	1 号显示单元帧缓冲读取结束；写 1 清 0		0
FB0ReadEnd	6	0 号显示单元帧缓冲读取结束；写 1 清 0		0
DB1Underflow	7	1 号显示单元数据缓冲区下溢；写 1 清 0		0
DB0Underflow	8	0 号显示单元数据缓冲区下溢；写 1 清 0		0
DB1FUnderflow	9	1 号显示单元数据缓冲区致命下溢；写 1 清 0		0
DB0FUnderflow	10	0 号显示单元数据缓冲区致命下溢；写 1 清 0		0
i2c_int	12:11	I2C 中断；写 1 清 0		0
HDMI0_hotplug	13	HDMI0 热插拔；写 1 清 0		0
HDMI1_hotplug	14	HDMI1 热插拔；写 1 清 0		0
VGA HotPlug	15	VGA 热插拔；写 1 清 0		0
En0	16	第 0 位中断使能；写 1 使能，写 0 屏蔽		0
En1	17	第 1 位中断使能；写 1 使能，写 0 屏蔽		0
En2	18	第 2 位中断使能；写 1 使能，写 0 屏蔽		0
En3	19	第 3 位中断使能；写 1 使能，写 0 屏蔽		0
En4	20	第 4 位中断使能；写 1 使能，写 0 屏蔽		0
En5	21	第 5 位中断使能；写 1 使能，写 0 屏蔽		0
En6	22	第 6 位中断使能；写 1 使能，写 0 屏蔽		0
En7	23	第 7 位中断使能；写 1 使能，写 0 屏蔽		0
En8	24	第 8 位中断使能；写 1 使能，写 0 屏蔽		0
En9	25	第 9 位中断使能；写 1 使能，写 0 屏蔽		0
En10	26	第 10 位中断使能；写 1 使能，写 0 屏蔽		0
En11	27	第 11 位中断使能；写 1 使能，写 0 屏蔽		0
En12	28	第 12 位中断使能；写 1 使能，写 0 屏蔽		0
En13	29	第 13 位中断使能；写 1 使能，写 0 屏蔽		0
En14	30	第 14 位中断使能；写 1 使能，写 0 屏蔽		0
En15	31	第 15 位中断使能；写 1 使能，写 0 屏蔽		0

27.4.26 中断寄存器 1

寄存器名	偏移地址	R/W	描述	复位值
Interrupt	0xdc	R/W	中断寄存器 1	00000000h
Interrupt	**bit**	**描述**		**初始值**
edp_hotplug	0	edp 热插拔，写 1 清 0		0
dp_hotplug	1	dp 热插拔，写 1 清 0		0

edp_hpd_irq	2	edp 中断请求，写 1 清 0	0
dp_hpd_irq	3	dp 中断请求，写 1 清 0	0

27.4.27 中断使能寄存器

寄存器名	偏移地址	R/W	描述	复位值
Interrupt	0xe0	R/W	中断寄存器 1 使能寄存器	00000000h
Interrupt	**bit**	**描述**		**初始值**
edp_hotplug_en	0	写 1 使能 edp_hotplug		0
dp_hotplug_en	1	写 1 使能 dp_hotplug		0
edp_hpd_irq_en	2	写 1 使能 edp_hpd_irq		0
dp_hpd_irq_en	3	写 1 使能 dp_hpd_irq		0

27.4.28 通用控制寄存器

寄存器名	偏移地址	RW	描述	复位值
generalCtrl	0x90	RW	通用寄存器	00000000h
generalCtrl	**bit**	**描述**		**初始值**
clk_disable1	1	写 1 关闭 DC1 时钟		0
clk_disable0	0	写 1 关闭 DC0 时钟		0

27.4.29 GPIO 输入输出寄存器

寄存器名	偏移地址	RW	描述	复位值
GPIO	0x94	RW	GPIO 输入输出寄存器	00000000h
GPIO	**bit**	**描述**		**初始值**
GPIO3	3	GPIO_EN 为 1 时，该值作为输入值;GPIO_EN 为 0 时，该值作为输出值;DC1 使用		0
GPIO2	2	GPIO_EN 为 1 时，该值作为输入值;GPIO_EN 为 0 时，该值作为输出值;DC1 使用		0
GPIO1	1	GPIO_EN 为 1 时，该值作为输入值;GPIO_EN 为 0 时，该值作为输出值;DC0 使用		0
GPIO0	0	GPIO_EN 为 1 时，该值作为输入值;GPIO_EN 为 0 时，该值作为输出值;DC0 使用		0

27.4.30 GPIO 输入输出使能寄存器

寄存器名	偏移地址	RW	描述	复位值
GPIO_EN	0x98	RW	GPIO 输入输出使能寄存器;	00000000h
GPIO_EN	**bit**	**描述**		**初始值**
GPIO_EN3	3	1 表示输入使能值;0 表示输出使能;DC1 使用		0
GPIO_EN2	2	1 表示输入使能值;0 表示输出使能;DC1 使用		0
GPIO_EN1	1	1 表示输入使能值;0 表示输出使能;DC0 使用		0
GPIO_EN0	0	1 表示输入使能值;0 表示输出使能;DC0 使用		0

27.4.31 HDMI 热插拔状态寄存器

寄存器名	偏移地址	RO	描述	复位值
HDMI HPSR	0xd0	RO	HDMI 热插拔状态寄存器	00000000h
HDMI HotPlugStatusReg	**bit**	**描述**		**初始值**
HDMI1 HotPlug Status	1	HDMI1 热插拔状态 1：有插入 0：无插入		0
HDMI0 HotPlug Status	0	HDMI0 热插拔状态 1：有插入 0：无插入		0

27.4.32 场同步计数寄存器

寄存器名	偏移地址	R/W	描述	复位值
Vsync Counter	0xd8	R	Vsync 计数寄存器	00000000h
Vsync Counter	**Bit**	**描述**		**初始值**
Vsync Counter	31：0	Vsync 计数		0

27.4.33 pixelclk 选择寄存器

寄存器名	偏移地址	R/W	描述	复位值
dc_pixelclk_sel	0xe4	R/W	pixelclk 选择寄存器	00000000h
dc_pixelclk_sel	**Bit**	**描述**		**初始值**
dc0_pixelclk_sel	0	0 表示 dc0 的 pixelclk 由 edp_phy (dp_phy0) 提供;1 表示 dc0 的 pixelclk 由 HDMI_phy 提供		0
dc1_pixelclk_sel	1	0 表示 dc1 的 pixelclk 由 dp_phy (dp_phy1) 提供;1 表示 dc1 的 pixelclk 由 HDMI_phy 提供		0

28 HDA 控制器 (D6:F2,D7:F0)

龙芯 2K3000 内集成两个 HDA 控制器，其中一个专门给 HDMI 和 DP 使用 (D6：F2) ，另一个通过 HDA 引脚与外部 CODEC 连接 (D7：F0) 。

HDA 控制器兼容 High Definition Audio Specification Revision 1.0a。

28.1 HDA 配置寄存器 (D7:F0)

表 28-1 HDA 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A07h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	03h	RO
0Bh	BCC	Base Class Code	04h	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W,RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：

1. HDA 配置头只在相关引脚配置为 HDA 模式下可见。
2. 表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

28.1.1 PCICMD-PCI 命令寄存器 (HDA-D7:F0)

地址偏移：04-05h　　属性：R/W, RO

默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 HDA 控制寄存器的访问。 0：禁止访问； 1：使能对 HDA 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

28.2 HDA 控制寄存器描述

HDA 控制寄存器的设计完全按照 HD audio Rev 1.0 规范进行设计的，下表列举了主要的寄存器的参数信息，具体的参考 HD audio Rev 1.0 手册，下表仅列举了与虚拟声卡相关的寄存器参数信息。

28.2.1 ELD 0 MEM 寄存器（虚拟声卡 0）

地址偏移：0x2200-0x2253　　属性：R/W, RO
默认值：0　　大小：84 字节

位域	名称	访问	描述
XXX	XXX	XXX	详见 HDA1.0a 协议

28.2.2 ELD0 MEM 有效寄存器（虚拟声卡 0）

地址偏移：0x2254-0x2254　　属性：R/W, RO
默认值：0　　大小：8 位

位域	名称	访问	描述
7:1	Reserved	RO	保留
0	ELD0 Mem Valid	R/W	该位用来控制 ELD0 MEM 寄存器中的内容是否有效 0：ELD0 Mem 内容无效 1：ELD0 Mem 内容有效

28.2.3 ELD1 MEM 寄存器（虚拟声卡 1）

地址偏移：0x2300-0x2353　　属性：R/W, RO
默认值：0　　大小：84 字节

位域	名称	访问	描述
XXX	XXX	XXX	详见 HDA1.0a 协议

28.2.4 ELD1 MEM 有效寄存器（虚拟声卡 1）

地址偏移：0x2354-0x2354　　属性：R/W, RO
默认值：0　　大小：8 位

位域	名称	访问	描述
7:1	Reserved	RO	保留
0	ELD1 Mem Valid	R/W	该位用来控制 ELD1 MEM 寄存器中的内容是否有效 0：ELD1 Mem 内容无效 1：ELD1 Mem 内容有效

28.2.5 Codec Unsupported Response 内容寄存器

地址偏移：0x2400-0x2403　　属性：R/W, RO
默认值：0x00000000　　大小：32 位

位域	名称	访问	描述
31:0	Codec Unsupported Response	R/W	虚拟声卡不支持的命令所返回的响应内容

28.2.6 Codec Unsupported Response 有效寄存器

地址偏移：0x2404-0x2404　　属性：R/W, RO

默认值：0　　大小：8 位

位域	名称	访问	描述
7:1	Reserved	RO	保留
0	Codec Unsupported Response Valid	R/W	该位用来控制虚拟声卡对于不支持的命令所返回的响应内容是否有效 0：Response 内容无效 1：Response 内容有效

28.2.7 Codec IID 初始值寄存器

地址偏移：0x2408-0x23b　　属性：R/W
默认值：0　　大小：32 位

位域	名称	访问	描述
31:0	Codec IID Init Value	R/W	虚拟声卡 IID 寄存器的初始值，由 BIOS 进行设置

29 I2S 控制器 (D7:F0)

29.1 概述

龙芯 2K3000 中 I2S 控制器支持 DMA 传输，支持多家公司的 codec 芯片。I2S 控制器支持主或从模式。主模式时由 I2S 控制器产生位时钟信号、左右声道选择时钟信号和数据信号，从模式时 I2S 控制器接收位时钟信号、左右声道选择时钟信号和数据信号。主模式时，codec 系统时钟由控制器提供，从模式时，系统时钟可由控制器或晶振提供。I2S 的功能特性包括：

1. 支持 8、16、18、20、24、32 位的音频数据采样位宽。
2. 支持 8、16、32 位的左右声道处理字宽。
3. 包含两个缓存 FIFO，FIFO 的缓存容量为 8bytes。
4. I2S 的中断处理模式可配，在 I2S 的发送和接收中断功能都使能后，当两个通道的缓存 fifo 为满仍要写以及为空仍要读时，则向 CPU 发出中断信号。
5. I2S 可以为 codec 芯片提供系统时钟，时钟频率可配。

29.2 I2S 配置寄存器

表 29-1 I2S 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A27h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	01h	RO
0Bh	BCC	Base Class Code	04h	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W, RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：

1. I2S 配置头只在 hda0_i2s_en 配置为 0 时可见。
2. 表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

29.3 PCICMD-PCI 命令寄存器

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 I2S 控制寄存器的访问。 0：禁止访问； 1：使能对 I2S 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

29.4 I2S 控制器寄存器

I2S 包含 5 个寄存器，定义如下表所示。

表 29-2 寄存器定义

寄存器名称	偏移地址	读/写 (R/W)	功能描述	复位值
IISVersion	0x0000	R/W	I2S 标识寄存器	32'h0
IISConfig	0x0004	R/W	I2S 配置寄存器	32'h0
IISControl	0x0008	R/W	I2S 控制寄存器	32'h0
IISRxData	0x000c	R/W	I2S 接收数据寄存器（用于 DMA 接收数据）	32'h0
IISTxData	0x0010	R/W	I2S 发送数据寄存器（用于 DMA 发送数据）	32'h0
IISConfig1	0xd014	R/W	I2S 配置寄存器 1	32' h0

I2S 标识寄存器允许主控机读取接收器的相关工作信息。它标识了 IIS 的地址位宽，数据位宽以及版本号等信息。

表 29-3 标识寄存器

寄存器名称	偏移地址	读/写 (R/W)	功能描述
IISVersion	位	缺省值	描述
ADRW	9:8	2'h0	地址总线宽度： 00：地址宽度 8 位 01：地址宽度 16 位 10：地址宽度 32 位 11：地址宽度 64 位
DATW	5:4	2'h0	数据宽度： 00：地址宽度 8 位 01：地址宽度 16 位 10：地址宽度 32 位 11：地址宽度 64 位
VER	3:0	4'h0	I2S 版本号

配置寄存器 0 是配置 I2S 的声道字长，发送和接收音频数据的采样深度以及位时钟的分频系数。

表 29-4 配置寄存器 0

寄存器名称	偏移地址	读/写 (R/W)	功能描述
IISConfig0	位	缺省值	描述
LR_LEN	31:24	'h0	左右声道处理的字长。
TX_RES_DEPTH	23:16	'h0	TX 采样深度设置： IIS 采样数据长度，有效范围为 8-32,如果发送的数据宽度小于采样数据长度，则低位补 0；如果发送的数据宽度大于采样数据长度，则低位忽略。
BCLK_RATIO	15:8	'h0	位时钟 (BCLK) 分频系数： 位时钟分频系数，分频数为 MCLK 时钟频率除以 2x (RATIO+1)
RX_RES_DEPTH	7:0	'h0	RX 采样深度设置： IIS 采样数据长度，有效范围为 8-32,如果接收到的数据宽度小于采样数据长度，则低位补 0；如果接收到的数据宽度大于采样数据长度，则低位忽略。

控制寄存器用于配置 IIS 的工作使能信号，缓存 FIFO 的存储状态以及中断相关信息状态。

表 29-5 控制寄存器

寄存器名称	偏移地址	读/写 (R/W)	功能描述
IISControl	位	缺省值	描述
MCLK_READY	16	R	系统时钟 (MCLK) 输出稳定标志，为 1 时时钟稳定输出，为 0 时输出时钟不可用
MASTER	15	'h0	1：IIS 工作于主模式 0：IIS 工作于从模式
MSB/LSB	14	'h0	1：高位在左端 0：高位在右端
RX_EN	13	'h0	控制器接收使能，为 1 时有效，开始接收数据
TX_EN	12	'h0	控制器发送使能，为 1 时有效，开始发送数据
RX_DMA_EN	11	'h0	DMA 接收使能，为 1 时有效
Reserved	10: 9	'h0	
CLK_READY	8	R	位时钟和声道选择时钟输出稳定标志，为 1 时时钟稳定输出，为 0 时输出时钟不可用
TX_DMA_EN	7	'h0	DMA 发送使能，为 1 时有效
Reserved	6:5	'h0	
RESETn	4	'h0	IIS 控制器软复位
MCLK_EN	3	'h0	使能时钟输出
RX_INT_EN	1	'h0	RX 中断使能，为 1 时使能中断，为 0 时禁止
TX_INT_EN	0	'h0	TX 中断使能，为 1 时使能中断，为 0 时禁止

配置寄存器 1 是配置系统时钟 (MCLK) 的分频系数。

表 29-6 配置寄存器 1

寄存器名称	偏移地址	读/写 (R/W)	功能描述
IISConfig1	位	缺省值	描述
MCLK_RATIO	15:0	'h0	系统时钟 (MCLK) 分频系数整数部分： 分频数为总线时钟频率除以系统时钟向下取整的值，系统时钟作为 Codec 的 sysclk
MCLK_RATIO_FRAC	32:16	'h0	系统时钟 (MCLK) 分频系数小数部分： 分频数为总线时钟频率除以系统时钟得小数部分乘以 2^16

29.5 DMA 命令寄存器

该寄存器用来控制 I2S 内部的 DMA 控制器（共 2 个），其中 DMA0 用于放音，DMA1 用于录音。DMA 控制器的详细描述见下节。

偏移量：0x100/0x110 (DMA0/DMA1)
复位值：00000000h

位域	名称	访问	描述
63:5	ask_addr	R/W	DMA 描述符地址的 bit[63:5]，低 5 位为 0
5	Reserved	R/W	保留
4	dma_stop	R/W	停止 DMA 操作。DMA 控制器完成当前数据读写后停止。
3	dma_start	R/W	开始 DMA 操作。DMA 控制器读取描述符地址 (ask_addr) 后将些位清零。
2	ask_valid	R/W	DMA 工作寄存器写回到 (ask_addr) 所指向的内存，完成后清零。
1	Reserved	R/W	保留
0	dma_64bit	R/W	DMA 控制器 64 位地址支持

29.6 配置操作

I2S 正常工作，需要先配置好 CODEC 芯片，然后配置 I2S 控制器的的配置寄存器和控制寄存器。

2K3000 芯片通过 I2S 接口和 CODEC 芯片通信，CODEC 芯片作为 I2S 总线上的从设备，详细地址、寄存器和配置方法请参考具体 CODEC 芯片的数据手册。配置完 CODEC 之后，需要对 I2S 控制器进行配置。可以将一些配置信息写入标志寄存器（I2SVersion），以供查询。先配置好 I2SConfig0 和 I2SConfig1 寄存器，再配置 I2SControl 寄存器。

I2SConfig 寄存器建议 LR_LEN 和 RES_DEPTH 配成一样，不至于在传输过程中丢数据或者空数据。MCLK 时钟是从内部时钟 I2S_CLOCK 分频得到，根据配置 CODEC 的采样频率、采样深度和倍频系数来计算，计算公式如下：

MCLK=256xfs（或者 512xfs）或者（768xfs）；（详细见 CODEC 手册推荐配置）

MCLK_RATIO = [Freq_I2S / (256xfs)];

MCLK_RATIO_FRAC = (Freq_I2S / (256xfs) - [Freq_I2S / (256xfs)]) x 2^16 ;

BCLK 时钟根据配置 CODEC 的采样深度来计算，计算公式如下：

BCLK = MASTER_RES_DEPTH x 2 xfs;

BCLK_RATIO= Freq_MCLK / (RES_DEPTH x 2 x fs) /2 - 1;

其中 fs 为配置的采样频率（即音频文件的码率）。。

1. I2S 控制器配置流程如下：先配置 I2SConfig0、I2SConfig1 寄存器、MSB/LSB 寄存器；
2. 如果是主模式或从模式采用控制器 MCLK 时，使能 MCLK_EN 寄存器，等待 MCLK_READY 为 1 进行下一步，否则跳过该步骤；
3. 如果是主模式，配置 MASTER 为 1，等待 CLK_READY 为 1 进行下一步，否则跳过该步骤；
4. 先配置 DMA， 然后配置 CODEC 芯片，使能 I2SControl 相关寄存器。

29.7 DMA 控制器

29.7.1 DMA 控制器结构描述

芯片中包含 2 个 DMA 控制器，用来实现内存与 I2S 之间数据搬移，可以节省资源提高系统数据传输的效率。

DMA 的传送数据的过程由三个阶段组成：

1. 传送前的预处理：由 CPU 配置 DMA 描述符相关的寄存器。
2. 数据传送：在 DMA 控制器的控制下自动完成。
3. 传送结束处理：发送中断请求。

本 DMA 控制器限定为以字（4Byte）为单位的数据搬运。

DMA 控制器支持 64 位地址空间，这主要通过 dma_64bit 来控制，当该位设置为 1 时表示 DMA 控制器工作在 64 位地址空间，反之为 32 位地址空间。在 64 位地址模式下，需要扩展 DMA_ORDER_ADDR 和 DMA_SADDR 为 64 位寄存器。

29.7.2 DMA 描述符

DMA_ORDER_ADDR_LOW

偏移地址：　0x0
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:1	dma_order_addr	31	R/W	存储器内部下一描述符地址寄存器（低 32 位）
0	dma_order_en	1	R/W	描述符是否有效信号

说明：存储下一个 DMA 描述符的地址，dma_order_en 是下个 DMA 描述符的使能位，如果该位为 1 表示下个描述符有效，该位为 0 表示下个描述符无效，不执行操作，地址 16 字节对齐。在配置 DMA 描述符时，该寄存器存放的是下个描述符的地址，执行完该次 DMA 操作后，通过判断 dma_order_en 信号确定是否开始下次 DMA 操作。在 64 位地址模式下，该寄存器存储低 32 位地址。

DMA_SADDR

偏移地址：0x4
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:0	dma_saddr	32	R/W	DMA 操作的系统内存地址（低 32 位）

说明：DMA 操作分为：内存读：从内存中读数据，保存在 DMA 控制器的缓存中，然后写入 I2S 设备，该寄存器指定了读内存的地址；内存写：从 I2S 设备读数据保存在 DMA 缓存中，当 DMA 缓存中的数据超过一定数目，就往内存中写，该寄存器指定了写内存的地址。在 64 位地址模式下，该寄存器存储低 32 位地址。

DMA_DADDR

偏移地址：0x8
复位值：0x00000000

位域	位域名称	位宽	访问	描述
27:0	dma_daddr	28	R/W	DMA 操作的 I2S 设备地址

DMA_LENGTH

偏移地址：0xc
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:0	dma_length	32	R/W	传输数据长度寄存器

说明：代表一块被搬运内容的长度，单位是字。当搬运完 length 长度的字之后，开始下个 step 即下一个循环。开始新的循环，则再次搬运 length 长度的数据。当 step 变为 1，单个 DMA 描述符操作结束，开始读下个描述符。

DMA_STEP_LENGTH

偏移地址：0x10
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:0	dma_step_length	32	R/W	数据传输间隔长度寄存器

说明：间隔长度说明两块被搬运内存数据块之间的长度，前一个 step 的结束地址与后一个 step 的开始地址之间的间隔。

DMA_STEP_TIMES

偏移地址：0x14
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:0	dma_step_times	32	R/W	数据传输循环次数寄存器

说明：循环次数说明在一次 DMA 操作中需要搬运的块的数目。如果只想搬运一个连续的数据块，循环次数寄存器的值可以赋值为 1。

DMA_CMD

偏移地址：0x18
复位值：　0x00000000

位域	位域名称	位宽	访问	描述
14:13	dma_cmd	2	R/W	源、目的地址生成方式
12	dma_r_w	1	R/W	DMA 操作类型，“1”为读 ddr2 写设备，“0”为读设备写 ddr2
11:8	dma_write_state	4	R/W	DMA 写数据状态
7:4	dma_read_state	4	R/W	DMA 读数据状态
3	dma_trans_over	1	R/W	DMA 执行完被配置的所有描述符操作
2	dma_single_trans_over	1	R/W	DMA 执行完一次描述符操作
1	dma_int	1	R/W	DMA 中断信号
0	dma_int_mask	1	R/W	DMA 中断是否被屏蔽掉

说明：dma_single_trans_over=1 指一次 DMA 操作执行结束，此时 length=0 且 step_times=1，开始取下个 DMA 操作的描述符。下个 DMA 操作的描述符地址保存在 DMA_ORDER_ADDR 寄存器中，如果 DMA_ORDER_ADDR 寄存器中 dma_order_en=0，则 dma_trans_over=1，整个 DMA 操作结束，没有新的描述符要读；如果 dma_order_en=1，则 dma_trans_over 置为 0，开始读下个 DMA 描述符。dma_int 为 DMA 的中断，如果没有中断屏蔽，在一次配置的 DMA 操作结束后发生中断。CPU 处理完中断后可以直接将其置低，也可以

等到 DMA 进行下次传输时自动置低。dma_int_mask 为对应 dma_int 的中断屏蔽。dma_read_state 说明了 DMA 当前的读状态。dma_write_state 说明了 DMA 当前的写状态。

DMA 写状态（WRITE_STATE[3:0]）描述，DMA 包括以下几个写状态：

Write_state	[3:0]	描述
Write_idle	4’h0	写状态正处于空闲状态
W_ddr_wait	4’h1	DMA 判断需要执行读设备写内存操作，并发起写内存请求，但是内存还没准备好响应请求，因此 DMA 一直在等待内存的响应
Write_ddr	4’h2	内存接收了 DMA 写请求，但是还没有执行完写操作
Write_ddr_end	4’h3	内存接收了 DMA 写请求，并完成写操作，此时 DMA 处于写内存操作完成状态
Write_dma_wait	4’h4	DMA 发出将 DMA 状态寄存器写回内存的请求，等待内存接收请求
Write_dma	4’h5	内存接收写 DMA 状态请求，但是操作还未完成
Write_dma_end	4’h6	内存完成写 DMA 状态操作
Write_step_end	4’h7	DMA 完成一次 length 长度的操作（也就是说完成一个 step）

DMA 读状态（READ_STATE[3:0]）描述，DMA 包括以下几个读状态：

Read_state	[3:0]	描述
Read_idle	4’h0	读状态正处于空闲状态
Read_ready	4’h1	接收到开始 DMA 操作的 start 信号后，进入准备好状态，开始读描述符
Get_order	4’h2	向内存发出读描述符请求，等待内存应答
Read_order	4’h3	内存接收读描述符请求，正在执行读操作
Finish_order_end	4’h4	内存读完 DMA 描述符
R_ddr_wait	4’h5	DMA 向内存发出读数据请求，等待内存应答
Read_ddr	4’h6	内存接收 DMA 读数据请求，正在执行读数据操作
Read_ddr_end	4’h7	内存完成 DMA 的一次读数据请求
Read_dev	4’h8	DMA 进入读设备状态
Read_dev_end	4’h9	设备返回读数据，结束此次读设备请求
Read_step_end	4’ha	结束一次 step 操作，step times 减 1

DMA_ORDER_ADDR_HIGH

偏移地址：　0x20
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:0	dma_order_addr	32	R/W	存储器内部下一个描述符地址寄存器（高 32 位）

DMA_SADDR_HIGH

偏移地址：　0x24
复位值：0x00000000

位域	位域名称	位宽	访问	描述
31:0	dma_saddr	32	R/W	DMA 操作的内存地址（高 32 位）

30 SATA 控制器 (D8:F0)

SATA 的特性包括：

- 支持SATA 1代1.5Gbps、SATA2代3Gbps和SATA3代6Gbps的传输
- 兼容串行ATA 3.3规范和AHCI 1.3.1规范

30.1 SATA 配置寄存器

表 31-1 SATA 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A18h	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	01h	RO
0Ah	SCC	Sub Class Code	06h	RO
0Bh	BCC	Base Class Code	01h	RO
0Ch	CLS	Cache Line Size	10h	RO
0Eh	HEADTYP	Header Type	00h	RO
24h-27h	ABAR	AHCI Base Address	00000000h	R/W,RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	00h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

30.1.1 PCICMD-PCI 命令寄存器 (SATA-D8:F0)

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 SATA 控制寄存器的访问。 0：禁止访问； 1：使能对 SATA 控制寄存器的访问。在将该位配置为 1 之前，必须先配置 PCNL_BAR 寄存器。
0	Reserved	RO	保留

30.2 SATA 控制器内部寄存器描述

SATA 的基地址是由 SATA 的 BAR0 给定，寄存器的定义和协议标准定义完全一致。

偏移地址	位宽	名称	描述
0x000	32	CAP	HBA 特性寄存器
0x004	32	GHC	全局 HBA 控制寄存器
0x008	32	IS	中断状态寄存器
0x00c	32	PI	端口寄存器
0x010	32	VS	AHCI 版本寄存器
0x014	32	CCC_CTL	命令完成合并控制寄存器
0x018	32	CCC_PORTS	命令完成合并端口寄存器
0x024	32	CAP2	HBA 特性扩展寄存器
0x0A0	32	BISTAFR	BIST 激活 FIS
0x0A4	32	BISTCR	BIST 控制寄存器
0x0A8	32	BISTCTR	BIST FIS 计数寄存器
0x0AC	32	BISTSR	BIST 状态寄存器
0x0B0	32	BISTDECR	BIST 双字错计数寄存器
0x0BC	32	OOBR	OOB 寄存器
0x0E0	32	TIMER1MS	1ms 计数寄存器
0x0E8	32	GPARAM1R	全局参数寄存器 1
0x0EC	32	GPARAM2R	全局参数寄存器 2
0x0F0	32	PPARAMR	端口参数寄存器
0x0F4	32	TESTR	测试寄存器
0x0F8	32	VERIONR	版本寄存器
0x0FC	32	IDR	ID 寄存器
0x100	32	P0_CLB	命令列表基地址低 32 位
0x104	32	P0_CLBU	命令列表基地址高 32 位
0x108	32	P0_FB	FIS 基地址低 32 位
0x10C	32	P0_FBU	FIS 基地址高 32 位
0x110	32	P0_IS	中断状态寄存器
0x114	32	P0_IE	中断使能寄存器
0x118	32	P0_CMD	命令寄存器
0x120	32	P0_TFD	任务文件数据寄存器
0x124	32	P0_SIG	签名寄存器
0x128	32	P0_SSTS	SATA 状态寄存器
0x12C	32	P0_SCTL	SATA 控制寄存器
0x130	32	P0_SERR	SATA 错误寄存器
0x134	32	P0_SACT	SATA 激活寄存器
0x138	32	P0_CI	命令发送寄存器
0x13C	32	P0_SNTF	SATA 命令通知寄存器
0x170	32	P0_DMACR	DMA 控制寄存器
0x178	32	P0_PHYCR	PHY 控制寄存器
0x17C	32	P0_PHYSR	PHY 状态寄存器

31 PCIe 控制器 (D9-14:F0)

2K3000 的 PCIe 分为 2 个模块：PCIe0，PCIe1，共 8 个 lane。

PCIe0 包括 4 个 lane，可当作一个 x4 的 PCIe 或者 4 个 x1 的 PCIe 使用，这个模块也可以被复用为一个 X4 的 RapidIO 接口（见 RapidIO 的章节）。其中，端口 0 在非 x4 模式下控制 lane0,在 x 4 模式下控制 lane0~3;在非 x4 模式下,端口 1 控制 lane1,端口 2 控制 lane2,端口 3 控制 lane3。

PCIe1 包括 4 个 lane，可当作一个 x4 的 PCIe 或者 2 个 x1 的 PCIe 使用，这个模块也可以被复用为一个 X4 的 RapidIO 接口（见 RapidIO 的章节）。其中，端口 0 在非 x4 模式下控制 lane0，在 x4 模式下控制 lane0~3；在非 x4 模式下，端口 1 控制 lane1， lane2 和 lane3 不可用。

PCIe 共包含 6 个端口 (Port) ，每个端口对应一个 PCIe 控制器。2K3000 的 PCIe 控制仪器作为 RC 使用，内部都有一个 TYPE1 类型的配置头。每个 PCIe 端口均有自己独立的 PCI 地址空间。

31.1 PCI 配置寄存器

下表列出 PCIe 端口的配置头缺省值，不同端口的 Device ID 可能不同，其他字段都相同。

表 32-1 PCIe 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	见寄存器描述	RO
04h-05h	PCICMD	PCI Command	0000h	R/W, RO
06h-07h	PCISTS	PCI Status	0010h	RO
08h	RID	Revision ID	01h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	04h	RO
0Bh	BCC	Base Class Code	06h	RO
0Ch	CLS	Cache Line Size	00h	RO
0Dh	PLT	Primary Latency Timer	00h	RO
0Eh	HEADTYP	Header Type	01h	RO
10h-17h	CNL_BAR	Control Block Base Address Register	0000000000000004h	R/W, RO
18h	PBNUM	Primary Bus Number	00h	R/W
19h	SBNUM	Secondary Bus Number	00h	R/W
1Ah	SuBNUM	Subordinate Bus Number	00h	R/W
1Bh	SLT	Secondary Latency Timer	00h	RO
1Ch	IOBASE	I/O Base	01h	R/W
1Dh	IOLMT	I/O Limit	01h	R/W
1Eh-1Fh	SSTS	Secondary Status	0000h	RO
20h-21h	MBASE	Memory Base	0000h	R/W
22h-23h	MLMT	Memory Limit	0000h	R/W
25h-24h	PMBASE	Prefetchable Memory Base	0000h	R/W
27h-26h	PMLMT	Prefetchable Memory Limit	0000h	R/W

28h-2Bh	PMBU32	Prefetchable Memory Base Upper 32 Bits	00000000h	R/W
2Ch-2Fh	PMLU32	Prefetchable Memory Limit Upper 32 Bits	00000000h	R/W
30h-31h	IOBU	I/O Base Upper 16 Bits	0000h	R/W
32h-33h	IOLMTU	I/O Limit Upper 16 Bits	0000h	R/W
34h	CAPP	Capabilities Pointer	40h	RO
3Ch	INT_LN	Interrupt Line	FFh	R/W
3Dh	INT_PN	Interrupt Pin	01h	RO
3Eh-3Fh	BCTRL	Bridge Control Register	0000h	R/W

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

DID-设备标识寄存器（PCIe）

地址偏移：02-03h　　属性：RO
默认值：见描述　　大小：16 位

位域	名称	访问	描述
15:0	DID	RO	PCIe 设备标识寄存器。各个 PCIe 端口对应的 DID 见表 22-2.

表 32-2 PCIe 端口 DID 表

PCI 设备号	描述	设备标识寄存器
D9:F0	PCIe0 端口 0	7A99h
D10:F0	PCIe0 端口 1	7A89h
D11:F0	PCIe0 端口 2	7A89h
D12:F0	PCIe0 端口 3	7A89h
D13:F0	PCIe1 端口 0	7A99h
D14:F0	PCIe1 端口 1	7A89h

31.2 地址空间划分

桥片中的 PCIe 控制器内有标准的 PCIe 配置头，因此 PCIe 控制器的内部寄存器以及其下游设备的地址空间都通过其配置头的信息来管理。配置头中地址相关的寄存器在 PCI 设备扫描时确定。

每个 PCIe 端口作为桥片中的独立设备，每个端口都包含一个 PCIe 配置头。当 F0、F1 中的 PCIe 工作在 X4 模式时，其他 X1 的端口软件不可见，只有当 PCIe 工作在 X1 模式时才可以访问其他 X1 端口。

对于每一个 PCIe 端口，其地址空间可以分为以下几部分：

配置头地址空间：该部分空间对应 PCIe 的配置头，通过配置请求来访问，最大 4KB，用于访问标准配置头。

配置访问地址空间：该部分地址空间用于通过配置请求访问 PCIe 控制器的下游设备配置头信息。根据下游设备的 Bus 号，由 PCIe 控制器决定发送 TYPE0 类型还是 TYPE1 类型的配置访问。

以上两个地址空间的地址由配置地址空间基地址、BUS 号、设备号、功能号以及寄存器偏移地址计算得出，访问以字为单位。

PCIe 控制器内部寄存器空间：该部分地址空间用于访问 PCIe 控制器的内部寄存器。这些寄存器用于控制 PCIe 控制器的行为和特性，与 PCIe 配置头空间属于两个地址空间。该地址空间为 MEM 类型，64 位地址空间，大小为 4KB，基地址等于 64 位 BAR0 的值，该值在初始化时由 PCI 扫描软件分配。

MEM 地址空间：该部分地址空间包含了 PCIe 控制器下游设备的所有 MEM 地址空间。对于 32 位地址空间，由 PCIe 配置头的 memory base 和 memory limit 决定；对于 64 位地址空间，由 PCIe 配置头的 prefetchable memory base（组合 upper 32bits）和 prefetchable memory limit（组合 upper 32bits）决定。该段地址空间由 PCIe 配置头的 command 寄存器 bit1 位来使能控制。

IO 地址空间：该部分地址空间包含了 PCIe 控制器下游设备的所有 IO 地址空间。由 PCIe 配置头的 io base（组合 upper 16bits）和 io limit（组合 upper 16bits）决定。该段地址空间由 PCIe 配置头的 command 寄存器 bit0 位来使能控制。

对于 MEM 地址空间和 IO 地址空间来说，如果在 X1 工作模式下，某个 X1 端口下游没有连接设备，通过设置 command 寄存器的 bit0 和 bit1 为 0 即可禁用其 MEM 和 IO 地址空间。

31.3 内部寄存器定义

桥片的每个 PCIe 端口都有自己的端口控制寄存器。如上文所述，每个端口的控制寄存器的基址由配置头的 BAR0 决定。

PCIe 端口控制寄存器 0

偏移地址：0x0
默认值：0x40ff0044

Bits	复位值	属性	名称	描述
0	0	R/W	Rx_lane_flip_en	PCIe 接收线反转
1	0	R/W	Tx_lane_flip_en	PCIe 发送线反转
2	1	R/W	Sys_aux_pwr_det	指示拥有辅助电源 (Vaux)
3	0 (RC) 1 (EP)	R/W	App_ltssm_enable	PCIe 端口链路建立使能
11:4	0x4	R/W	Reserved	需设置为 0
12	0	R/W	App_req_enter_L1	要求 PCIe 链路进入 L1
13	0	R/W	App_ready_enter_L23	已经准备好让 PCIe 链路进入 L23
14	0	R/W	App_req_exit_L1	要求 PCIe 端口退出 L1
15	0	R/W	Soft_reset_en	软复位使能，高有效
23:16	0xff	R/W	Reserved	保留
24	0	R/W	at_en	PCIe 端口在 RC 工作模式下入站地址转换使能
25	0	R/W	bus_error_en	PCIe 读返回总线错误的处理方式 0：读数据返回全 1，不产生总线错例外 1：产生总线错例外 当 PCIe 控制器完成总线扫描和初始化后，此位可以修改为 1
29：26	0	R/W	Reserved	保留
30	0x1	R/W	error response select	0：出现错误时，读数据返回全 0 1：出现错误时，读数据返回全 1
31	0x0	RO	Reserved	保留

PCIe 端口控制寄存器 1

通过对相应位写入 1 产生一个时钟周期的脉冲，控制 PCIe 接口进行相应的操作。写此寄存器时，一次仅能有 1 位被置 1。

偏移地址：0x4
默认值：0x0

Bits	复位值	属性	名称	描述
0	0	R/W1P	App_unlock_msg	在 PCIe 端口上发送解锁消息，高有效。该位在软件写 1 后会自动清 0
1	0	R/W1P	Apps_pm_xmt_turnoff	在 PCIe 端口上发送 PM_Turn_Off 消息，高有效。该位在软件写 1 后会自动清 0
2	0	R/W1P	App_init_rst	在 PCIe 端口上发送热复位消息，高有效。该位在软件写 1 后会自动清 0
3	0	R/W1P	Soft_reset	对 PCIe 端口进行软复位
4	0	R/W1P	Apps_pm_xmt_pme	将 PCIe 端口从 D1 或者 D2 或者 D3 状态中唤醒，并发送 PM_PME 消息，高有效。
31:5	0x0	RO	Reserved	保留

PCIe 端口状态寄存器 0

偏移地址：0x8
默认值：0x20f

Bits	名称	描述
1:0	Cfg_pwr_ind	系统电源状态指示，与 Slot Control register bits [9:8]相同 00b: Reserved 01b: On 10b: Blink 11b: Off
3:2	Cfg_atten_ind	Attention 按钮状态指示，与 Slot Control register bits [7:6]相同 00b: Reserved 01b: On 10b: Blink 11b: Off
4	Cfg_pwr_ctrl	系统电源控制器状态 0: Power On 1: Power Off
5	Pm_xtlh_block_tlp	禁止发出请求指示
6	Cfg_bus_master_en	PCI 主设备使能 1: enabled 0: disabled
7	Cfg_mem_space_en	内存映射访问使能 1: enabled 0: disabled
10:8	Cfg_max_rd_req_size	最大读请求大小，对应 Device Control register 寄存器的 Max_Read_Request_Size 位域.
13:11	Cfg_max_payload_size	最大数据负载，对应 Device Control register 寄存器的 Max_Payload_Size 位域
14	Cfg_rcb	对应 Link Control register 的 RCB 位域
15	Rdlh_link_up	数据链路层状态 1: Link is up 0: Link is down
18:16	Pm_curnt_state	当前电源状态
23:19	Cfg_aer_int_msg_num	根错误状态寄存器的 31:27 位
28:24	Cfg_pcie_cap_int_msg_num	PCIe 能力寄存器 13:9 位

29	rfc_update0	此位为 1 表示 rfc_data0 有更新的流控令牌包的内容。 rfc_update0、rfc_data0 被用作观察流控令牌的发放
30	rfc_update1	此位为 1 表示 rfc_data1 有更新的流控令牌包的内容。 rfc_update1、rfc_data1 被用作观察流控令牌的发放
31	Reserved	保留

PCIe 端口状态寄存器 1

偏移地址：0xc
默认值：0x01000000

Bits	名称	描述
5:0	Xmlh_ltssm_state	LTSSM 状态机的当前状态
6	Xmlh_link_up	物理链路状态指示 1: Up 0: Down
7	rdlh_link_up	数据链路状态指示 1：Up 0：Down
8	Cfg_eml_control	对 应 the Slot Control register 的 Electromechanical interlock control 位状态.
16:9	Cfg_pbus_num	设备所属总线号
21:17	Cfg_pbus_dev_num	设备号
24:22	Pm_dstate	电源管理状态指示 000b: D0 001b: D1 010b: D2 011b: D3 100b: Uninitialized Other values: Not applicable
25	Pm_status	电源管理状态位
26	Pm_pme_en	PME 使能指示
27	Aux_pm_en	辅助电源使能指示
28	Cfg_link_auto_bw_int	链路自治带宽状态寄存器更新指示
29	Cfg_bw_mgt_int	链路带宽管理状态寄存器更新指示
30	Reserved	保留
31	cfg_link_eq_req_int	链路均衡训练请求指示 此位为 1 表示在暂时无法做链路均衡参数训练的状态下收到了链路均衡训练的请求

用户定义消息 ID 寄存器

偏移地址：0x10

Bits	名称	属性	描述
15:0	radm_msg_req_id	RO	访问 radm_msg_index 指定的接收用户定义消息组所存储的用户定义消息的 ID
17:16	radm_msg_index	RW	访问的接收用户定义消息组的编号 X1、X4 控制器只允许使用 0h X8 控制器允许使用 0h、1h X16 控制器允许使用 0h ~ 3h
31:18	Reserved	RO	保留

流控观察寄存器 0

偏移地址：0x14

Bits	名称	属性	描述
31:0	rfc_data1	RO	记录最近一次同一个符号时间里收到 2 个流控包时的第二个流控包的内容

PCIe 端口中断状态寄存器

偏移地址：0x18

Bits	名称	描述
0	aer_rc_err_int	当 RC 产生或收到错误消息并且 Root Error Command 寄存器中对应的错误报告使能被打开时置位。此位仅在 RC 工作模式下使用。
1	aer_rc_err_msi	当 RC 产生或收到错误消息并且 MSI 被使能且 Root Error Command 寄存器中对应的错误报告使能被打开时置位。此位仅在 RC 工作模式下使用。
2	sys_err_rc	此位报告检测到系统错误。当 Root Control 寄存器的对应位使能被打开并且 PCIe 总线上如何设备产生：可修复错误、致命错误、非致命错误时，或者 RC 产生内部错误时置位。
3	pme_int	收到 PME 中断。当下列条件满足时，此位置位： 1. Command 寄存器中 INTx 未被禁止； 2. Root Control 寄存器中 PME 中断使能位被打开； 3. 收到 PME 消息（Root Status 寄存器中 PME 状态位被置位）。
4	pme_msi	收到 PME 中断。当下列条件满足时，此位置位： 1. Command 寄存器中 INTx 被禁止,MSI 被使能； 2. Root Control 寄存器中 PME 中断使能位被打开； 3. 收到 PME 消息（Root Status 寄存器中 PME 状态位被置位）。
5	vendor_msg0	用户定义消息组 0 收到用户定义消息
6	rc_core_wake	从电源管理中唤醒
7	INTA	INTA 中断
8	INTB	INTB 中断
9	INTC	INTC 中断
10	INTD	INTD 中断
11	radm_correctable_err	此 PCIe 端口收到可修复错误消息
12	radm_nonfatal_err	此 PCIe 端口收到非致命错误消息
13	radm_fatal_err	此 PCIe 端口收到致命错误消息
14	pm_pme	收到 PM_PME 消息
15	pm_to_ack	收到 PM_TO_Ack 消息
16	hp_pme	当下列条件满足时，此位置位： 1. PCI 电源管理控制和状态寄存器中 PME 使能被打开； 2. Slot Status 寄存器中的任意 1 位由 0 变 1 并且对应位的通知使能被打开。
17	hp_int	当下列条件满足时，此位置位： 1. Command 寄存器中 INTx 未被禁止； 2. Slot Control 寄存器中热插拔中断使能被打开； 3. Slot Status 寄存器中的任意 1 位为 1，并且对应位的通知使能被打开。
18	hp_msi	当下列条件满足时，此位置位： 1. MSI 使能被打开； 2. Slot Control 寄存器中热插拔中断使能被打开； 3. Slot Status 寄存器中的任意 1 位从 0 变 1，并且对应位的通知使能被打开。
19	link_auto_bw_int	当链路的自治带宽状态寄存器被更新并且链路的自治带宽中断被使能时置位仅在 RC 模式下使用。
20	bw_mgt_int	当链路的带宽状态寄存器被更新并且链路的带宽中断被使能时置位。仅在 RC 模式下使用。
21	gm_composer_lookup_err	此位被置位表示此 PCIe 端口在向外发送数据响应时溢出。这通常表示此 PCIe 端口接收端收到的 Non-Posted 请求的数量超出了端口流控限制。

22	radmx_composer_lookup_err	此位被置位表示此 PCIe 端口接收送数据响应时溢出。这通常表示此 PCIe 端口发送端发送的 Non-Posted 请求的数量超出了端口流控限制。
23	phy_int	PHY 中断
24	cds_wait_timeou	在数据链路层 link up 状态下，如果有 TLP 待发送，且待发送的 TLP 等待令牌的时间超过设置的时间阈值
25	ltssm_l2_to_detect	LTSSM 状态从 L2 状态退出到设备检测状态
26	pm_turn_off	收到 PM_Turn_Off 消息
27	Link_req_rst_not_fall	PCIe 端口物理链路断裂
28	Link_eq_req_int	此位被置位表面链路重新进行了 Gen3 的 equalization 参数训练
31:29	Reserved	保留

PCIe 端口中断状态清除寄存器

偏移地址：0x1c

R/W1C

在某一位写入 1 则清除 PCIe 端口中断状态寄存器的对应位。

PCIe 端口中断掩码寄存器

偏移地址：0x20

R/W

PCIe 端口中断状态寄存器对应的中断掩码。哪 1 位置 1 并且中断状态寄存器的相应位也为 1 时，此 PCIe 端口产生中断。

PCIe 端口控制寄存器 2

偏移地址：0x24　　默认值：　0x48111a

Bits	名称	属性	描述
31:26	int_offset	RW	中断号偏移设置
25	int_mode1_en	RW	中断模式 1
24	int_mode0_en	RW	中断模式 0
23：20	rx data queue error mask	RW	从高位到低位分别对 par_er、ecrc_err、tlp_ablrt、dllp_abort 错误进行屏蔽
19	rxstandby_en	RW	此位为 1 时，PCIe 控制器在切换速率时会将 PIPE 接口上的 rxstandby 信号置为有效
18：5	Reserved	RW	保留
4	tlp_extract_protect	RW	此位为 1 时，在进入 L1 状态的过程中不会看到 TLP 的情况下停止对 TLP 的提取
3	link_down_prot_en	RW	此位为 1 时，当链路断开时，由总线保护逻辑接收并处理对外部设备的访问，以防止出现内部总线卡死的情况
2	Header_ro_wen	RW	此位为 1，将是 header 中的只读位变为可写
1	cfg0_busnum_is_zero	RW	此位为 1 将发送的 TYPE0 的配置访问中的 BUS NUM 强制设置为 0
0	addr_trans_bypass	RW	此位为 1 表示旁路控制器内部的地址转换功能

PCIe 端口控制和状态寄存器

偏移地址：0x28

Bits	复位值	属性	名称	描述
0	1	R/W	func_bypass	置 1 时，取消内部接口请求间的先后顺序限制（读可能会越过写，写也可能越过读）
25:1				Reserved
26		RO	max_lane_mode	PCIe 端口在最大链路宽度模式工作
27		RO	Is_rc	PCIe 端口在 RC 模式工作
28		RO	L0s	PCIe 端口处于 L0s 功耗状态
29		RO	L1	PCIe 端口处于 L1 功耗状态
30		RO	L2	PCIe 端口处于 L2 功耗状态
31		RO	L2_exit	PCIe 端口退出 L2 功耗状态

用户定制消息发送寄存器 0

偏移地址：0x38

此寄存器用于在发送用户定制 PCIe 消息前存储 PCIe 传输层协议消息包头的头 4 个字节的信息。

Bits	名称	描述
1:0	ven_msg_fmt	PCIe 协议中传输层协议包头中的 Fmt 域
6:2	ven_msg_type	PCIe 协议中传输层协议包头中的 Type 域
9:7	ven_msg_tc	PCIe 协议中传输层协议包头中的 TC 域
10	ven_msg_td	PCIe 协议中传输层协议包头中的 TD 域
11	ven_msg_ep	PCIe 协议中传输层协议包头中的 EP 域
13:12	ven_msg_attr	PCIe 协议中传输层协议包头中的 Attr 域
23:14	ven_msg_len	PCIe 协议中传输层协议包头中的 Length 域
31:24	Reserved	保留

用户定制消息发送寄存器 1

此寄存器用于在发送用户定制 PCIe 消息前存储 PCIe 传输层协议消息包头的第 4 到第 7 字节的消息。

偏移地址：0x3c

Bits	名称	描述
2:0	ven_msg_fun_num	PCIe 协议中传输层协议包头中的 Function Number 域
10:3	ven_msg_tag	PCIe 协议中传输层协议包头中的 Tag 域
18:11	ven_msg_code	PCIe 协议中传输层协议包头中的 Message Code 域
22:19		Reserved
23	ven_msg_req_valid	此位置 1 表示当前用户定制消息的所有参数均已经配置完毕，要求 PCIe 端口发送此用户定制消息。消息发送完毕后，此位自动清 0。
31:24	Reserved	保留

用户定制消息发送寄存器 2

偏移地址：0x40

用于存储待发送用户定制消息所携带数据的低 32 位。

用户定制消息发送寄存器 3

偏移地址：0x44

用于存储待发送用户定制消息所携带数据的高 32 位。

流控观察寄存器 1

偏移地址：0x48

Bits	名称	属性	描述
31:0	rfc_data0	RO	记录最近一次同一个符号时间里收到的第一个流控包的内容

MSI 消息发送寄存器

偏移地址：0x5c

用于存储要发送的 MSI 消息的包头参数和触发 MSI 消息的发送。仅在 EP 模式下使用（2K3000 中不可用）。在发送 MSI 消息前，需要系统软件按照 PCIe 协议为控制器配置 MSI 消息使用的地址并将 EP 的中断方式设置为 MSI 方式。

Bits	名称	描述
4:0	ven_msi_vector	PCI 协议 MSI 包中的 Vector 域
7:5	ven_msg_tc	PCIe 协议中传输层协议包头中的 TC 域
10:8	ven_msg_func_num	PCIe 协议中传输层协议包头中的 Function Number 域
11	ven_msi_valid	此位置 1 表示当前 MSI 消息的所有参数均已经配置完毕，要求 PCIe 端口发送此 MSI 消息。消息发送完毕后，此位自动清 0。
31:12		Reserved

地址译码掩码寄存器 0

偏移地址：0x68

用于存储此 PCIe 端口作为 RC 时入站地址转换的地址掩码的低 32 位。

地址译码掩码寄存器 1

偏移地址：0x6c

R/W

用于存储此 PCIe 端口作为 RC 时入站地址转换的地址掩码的高 32 位。

addr_mask[51:0] = {addr_mask_h[19:0],addr_mask_l[31:0]}

当 at_en 为 1 时，addr_mask 决定来自设备的外部 PCIe 地址的 63～12 位中哪些位需要进行替换。

地址译码转换地址寄存器 0

偏移地址：0x70

R/W

用于存储此 PCIe 端口作为 RC 时入站地址转换的转换地址的低 32 位。

地址译码转换地址寄存器 1

偏移地址：0x74

R/W

用于存储此 PCIe 端口作为 RC 时入站地址转换的转换地址的高 32 位。

addr_tran[51:0] = {addr_tran_h[19:0],addr_tran_l[31:0]}

addr_tran 在 at_en 为 1 时，用来设定来自设备的 PCIe 地址的 63～12 位中需要被替换的值。

接收用户定义消息数据负载读取端口 0

偏移地址：0x78

R/W

用于读取 radm_msg_index 所指定的接收组所存储的用户定义消息数据的低 32 位。

接收用户定义消息数据负载读取端口 1

偏移地址：0x7c

R/W

用于读取 radm_msg_index 所指定的接收组存储的用户定义消息数据的高 32 位。

extioi_addr_trans_l

偏移地址：0x80

R/W

extioi_addr_trans 的低 32 位。

extioi_addr_trans_h

偏移地址：0x84

R/W

extioi_addr_trans 的高 32 位。

ir_addr32

偏移地址：0x88

R/W

ir_addr32。

PHY 参数观测寄存器 0

偏移地址：0xa0

RO

用于观测由 lane_index 选中 lane 的参数设置。

Bits	名称	描述
7:0	ltx_c-1	选中 lane 本地的 TX C-1 系数
15:8	ltx_c0	选中 lane 本地的 TX C0 系数
23:16	ltx_c+1	选中 lane 本地的 TX C+1 系数
27：24	ltx_pset	选中的 lane 本地的 TX preset 预设值
31:28	lrx_rhint	选中 lane 本地的 rx preset hint 值

PHY 参数观测寄存器 1

偏移地址：0xa4

RO

用于观测由 lane_index 选中 lane 的参数设置。

Bits	名称	描述
7:0	ltx_lf	选中 lane 本地的 TX LF 参数
15:8	ltx_fs	选中 lane 本地的 TX FS 参数
23:16	rtx_lf	选中 lane 对端的 TX LF 参数
31:24	rtx_fs	选中 lane 对端的 TX FS 参数

PHY 参数观测寄存器 2

偏移地址：0xa8

用于观测由 lane_index 选中 lane 的参数设置。

Bits	名称	描述
7:0	rtx_c-1	选中 lane 对端的 TX C-1 系数
15:8	rtx_c0	选中 lane 对端的 TX C0 系数
23:16	rtx_c+1	选中 lane 对端的 TX C+1 系数
31：24	rtx_pset	选中的 lane 对端的 TX preset 预设值

PHY 参数观测寄存器 3

偏移地址：0xac

用于观测由 lane_index 选中 lane 的参数设置。

Bits	名称	属性	描述
7：0	lane_index	RW	用于配置要观测的 lane 的编号 编号从 0 开始
15：8	pset0_fom	RO	选中 lane 对端在使用 preset_0 时本地评估的眼宽数值
23：16	pset1_fom	RO	选中 lane 对端在使用 preset_1 时本地评估的眼宽数值
31：24	pset2_fom	RO	选中 lane 对端在使用 preset_2 时本地评估的眼宽数值

PHY 参数观测寄存器 4

偏移地址：0xb0

用于观测由 lane_index 选中 lane 的参数设置。

Bits	名称	属性	描述
7：0	pset3_fom	RO	选中 lane 对端在使用 preset_3 时本地评估的眼宽数值
15：8	pset4_fom	RO	选中 lane 对端在使用 preset_4 时本地评估的眼宽数值
23：16	pset5_fom	RO	选中 lane 对端在使用 preset_5 时本地评估的眼宽数值
31：24	pset6_fom	RO	选中 lane 对端在使用 preset_6 时本地评估的眼宽数值

PHY 参数观测寄存器 5

偏移地址：0xb4

用于观测由 lane_index 选中 lane 的参数设置。

Bits	名称	属性	描述
7：0	pset7_fom	RO	选中 lane 对端在使用 preset_7 时本地评估的眼宽数值
15：8	pset8_fom	RO	选中 lane 对端在使用 preset_8 时本地评估的眼宽数值
23：16	pset9_fom	RO	选中 lane 对端在使用 preset_9 时本地评估的眼宽数值
31：24	pset10_fom	RO	选中 lane 对端在使用 preset_10 时本地评估的眼宽数值

物理层链路观测寄存器 0

偏移地址：0xb8

用于观测物理层链路的状态变化

Bits	名称	属性	描述
5：0	link state d3	RO	
7：6	link speed d3	RO	0：Gen1，1：Gen2，2：Gen3，3：Gen4
13：8	link state d2	RO	
15：14	link speed d2	RO	
21：16	link state d1	RO	
23：22	link speed d1	RO	
29：24	link state d0	RO	
31：30	link speed d0	RO	

物理层链路观测寄存器 1

偏移地址：0xbc

用于观测物理层链路的状态变化

Bits	名称	属性	描述
5：0	link state d7	RO	
7：6	link speed d7	RO	0：Gen1，1：Gen2，2：Gen3，3：Gen4
13：8	link state d6	RO	
15：14	link speed d6	RO	
21：16	link state d5	RO	
23：22	link speed d5	RO	
29：24	link state d4	RO	
31：30	link speed d4	RO	

物理层链路观测寄存器 2

偏移地址：0xc0

用于观测物理层链路退出 L0 状态原因。

Bits	名称	属性	描述
8：0	L0 lose case	RO	[0]：rcvd_2_unexpect_ts [1]：xdlh_xmlh_start_link_retrain [2]：directed_recovery [3]：rmlh_deskew_alignment_err [4]：go_recovery_speed_change [5]：rmlh_goto_recovery [6]：directed_equalization [7]：rtlh_req_link_retrain [8]：eidle_inferred_recovery
15：9	Reserved	RO	保留
24：16	last L0 lose case	RO	
31：25	Reserved	RO	保留

数据链路层观测寄存器 0

偏移地址：0xc4

Bits	名称	属性	描述
15：0	replay_cnt	RO	数据链路层产生超时重传的计数值
31：16	nack_cnt	RO	数据链路层给出 NACK 应答的计数值

RO 属性的写计数值

偏移地址：0xd0

Bits	名称	属性	描述
31：0	p_counter_ro	RO	协议层从 PCIe 链路接收到的带有 RO 属性的写请求的计数

IDO 属性的写计数值

偏移地址：0xd4

Bits	名称	属性	描述
31：0	p_counter_ido	RO	协议层从 PCIe 链路接收到的带有 IDO 属性的写请求的计数

接收通道写计数值

偏移地址：0xd8

Bits	名称	属性	描述
31：0	p_counter	RO	协议层从 PCIe 链路接收到的写请求的计数

接收通道写计数值

偏移地址：0xdc
默认值：0x00080000

Bits	名称	属性	描述
23：0	cdts_timeout_val	RW	发送通道在链路完好情况下获取令牌的超时时间 默认值为 8ms
30：24	Reserved	RW	保留
31	cdts_wait_timeout_en	RW	此位为 1 时，当发送通道的请求在链路完好的情况下超过 cdts_timeout_val 设定的时候还没有获得令牌时将触发总线保护机制工作

32 LPC 控制器 (D23:F0)

LPC 控制器具有以下特性：

- 符合LPC1.1规范
- 支持LPC访问超时计数器
- 支持Memory Read/write访问类型
- 支持Firmware Memory Read/Write访问类型（单字节）
- 支持I/O read/write访问类型
- 支持TPM I/O read/write访问类型
- 支持Memory访问类型地址转换
- 支持Serial IRQ规范，支持17个中断源

32.1 LPC 配置寄存器 (D23:F0)

表 33-1 LPC 控制器的配置寄存器

地址偏移	简称	描述	默认值	访问类型
00h-01h	VID	Vendor ID	0014h	RO
02h-03h	DID	Device ID	7A0Ch	RO
04h-05h	PCICMD	PCI Command	0001h	R/W, RO
08h	RID	Revision ID	00h	RO
09h	PI	Programming Interface	00h	RO
0Ah	SCC	Sub Class Code	01h	RO
0Bh	BCC	Base Class Code	06h	RO
0Ch	CLS	Cache Line Size	00h	RO
0Eh	HEADTYP	Header Type	00h	RO
10h-17h	FIXCREG	Fixed ControlRegister	0000000010002004h	RO
18h-1Fh	FIXMREG	Fixed MemoryRegister	0000000012000004h	RO
20h-27h	FIXIOREG	Fixed I/ORegister	000000FDFC000001h	RO
2Ch-2Dh	SVID	Subsystem Vendor ID	0000h	RO
2Eh-2Fh	SID	Subsystem Identification	0000h	RO
3Ch	INT_LN	Interrupt Line	00h	R/W
3Dh	INT_PN	Interrupt Pin	01h	RO

注：表中未列出的地址空间表示保留。

下面列出与 PCI 配置头规范稍有不同的寄存器及其描述。

32.1.1 PCICMD-PCI 命令寄存器 (LPC-D23:F0)

地址偏移：04-05h　　属性：R/W, RO
默认值：0000h　　大小：16 位

位域	名称	访问	描述
15:2	Reserved	RO	保留
1	Memory Space Enable	R/W	该位用来控制是否使能对 LPC 控制寄存器和 MEM 空间的访问。 0：禁止访问； 1：使能对 LPC 控制寄存器和 MEM 空间的访问。

0	I/O Space Enable	R/W	该位用来控制是否使能对 LPC I/O 空间的访问。LPC I/O 空间的地址固定从 I/O 空间的地址 0 开始。 0：禁止访问； 1：使能对 LPCI/O 空间的访问。

32.1.2 FIXCREG-Fixed 控制寄存器

该寄存器不作为 LPC 配置头的 BAR 使用。

地址偏移：10-17h　　属性：RO
默认值：0000000010002004h　　大小：64 位

位域	名称	访问	描述
63:0	Reserved	RO	保留。

32.1.3 FIXMREG-Fixed MEM 寄存器

该寄存器不作为 LPC 配置头的 BAR 使用。

地址偏移：18-1Fh　　属性：RO
默认值：0000000012000004h　　大小：64 位

位域	名称	访问	描述
63:0	Reserved	RO	保留。

32.1.4 FIXIOREG-Fixed I/O寄存器

该寄存器不作为 LPC 配置头的 BAR 使用。

地址偏移：20-27h　　属性：RO
默认值：000000FDFC000001h　　大小：64 位

位域	名称	访问	描述
63:0	Reserved	RO	保留。

FIXCREG、FIXMREG、FIXIOREG 的地址和 PCI 配置头的 BAR 寄存器相同，但这几个寄存器不作为 LPC 配置头的 BAR 寄存器使用。软件可以通过修改 PCI 配置读函数的方法来绕过该硬件 bug，使得上层软件不受影响。

32.2 LPC 地址空间

LPC 控制器包括三个地址空间：控制寄存器空间、MEM 空间、I/O 空间。

LPC 控制寄存器空间用来配置 LPC 控制器。LPC 控制寄存器空间位于桥片的固定设备地址空间内，起始地址为 0x1000,2000，大小为 4KB。

LPC MEM 空间用来访问 LPC 总线上挂载的 Memory/Firmware Memory 设备。LPC MEM 空间位于桥片的固定设备地址空间内，起始地址为 0x1200,0000，大小为 32MB。处理器发往 LPC MEM 空间的访问会被转换成 LPC 协议的 Memory 访问发往 LPC 总线。LPC 控制器发出哪种类型的 Memory 访问，由 LPC 控制器的控制寄存器决定。处理器发往这个地址空间的地址可以进行地址转换。转换后的地址由 LPC 控制器的配置寄存器 LPC_MEM_TRANS 设置。

LPC I/O 空间用来访问 LPC 总线上挂载的 I/O 设备，LPC I/O 空间的地址从 PCI I/O 空间的 0 地址开始，大小为 128KB。处理器发往该空间的访问会被转换成 LPC 协议的 I/O 访问发到 LPC 总线。其中 LPCI/O 空间的低 64KB 空间用来访问 LPCI/O 设备，高 64KB 空间用来访问 TPM 设备。

32.3 LPC 中断

LPC 控制器内部包括两类中断：SIRQ 中断和访问超时中断。LPC 控制器共支持 17 个 SIRQ 中断，对应中断相关寄存器的比特位[16:0]。访问超时中断对应中断相关寄存器的比特位[17]。

SIRQ 中断为电平触发中断，触发电平的值可由寄存器配置。软件应先配置好 SIRQ 中断的触发电平，然后再使能 LPC 控制器的 SIRQ 中断。SIRQ 中断不需要软件清除。

访问超时中断为边沿触发中断，因此，如果发生 LPC 访问超时中断，则软件需要写中断清除寄存器的 bit[17]来清除该中断。

32.4 LPC 控制寄存器

32.4.1 控制寄存器 0

地址偏移：00-03h　　属性：R/W
默认值：0000FFFFh　　大小：32 位

位域	名称	访问	描述
31	SIRQ_EN	R/W	SIRQ 中断使能控制
23	LPC_MEM_TRANS_EN	R/W	LPC Memory 空间地址转换使能
22:16	LPC_MEM_TRANS	R/W	LPC Memory 空间地址转换后的高 7 位地址(bit[31:25]) .
15:0	LPC_SYNC_TIMEOUT	R/W	LPC 访问超时的阈值（最小为 64）

32.4.2 控制寄存器 1

地址偏移：04-07h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31	FIRMWARE_TYPE	R/W	LPC Memory 空间 Firmware Memory 访问类型设置
17:0	LPC_INT_EN	R/W	LPC 中断使能，每个比特位对应一个中断源。对于每个中断源， 0：关闭中断； 1：使能中断。

32.4.3 LPC 中断状态寄存器

地址偏移：08-0Bh　　属性：RO
默认值：00000000h　　大小：32

位域	名称	访问	描述
17:0	LPC_INT_SRC	RO	LPC 中断源指示，每个比特位对应一个中断源。对于每个中断源， 0：没有中断； 1：有中断。

32.4.4 LPC 中断清除寄存器

地址偏移：0C-0Fh　　属性：WO
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
17	LPC_TIMEOUT_INT_CLEAR	WO	LPC 访问超时中断清除（写 1 清除）。比特 17 对应 LPC 访问超时中断，写 1 清除，写 0 无效。

32.4.5 LPC SIRQ 中断极性寄存器

地址偏移：10-13h　　属性：R/W
默认值：0000FFFBh　　大小：32 位

位域	名称	访问	描述
16:0	SIRQ_INT_POLARITY	R/W	LPC SIRQ 中断极性寄存器，每个比特位对应一个中断源。对于每个中断源， 0：低电平触发； 1：高电平触发。

33 eMMC 控制器 (D28:F0)

33.1 功能特性

- 8位预分频逻辑（频率=系统时钟/ (p+1) ）
- DMA数据传输模式
- 专用独立DMA通道
- 1位/4位/8位的总线模式

33.2 寄存器描述

eMMC 控制器的寄存器详细说明如下:

33.2.1 eMMC_CON 控制寄存器

地址偏移：00-03h　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_CON	位	缺省值	描述
•	31:9	0x0	
soft_rst	8	0x0	软件复位，整个模块复位。复位完成后硬件自动清零
Reserved	7:1	0x0	
enclk	0	0x0	SD 时钟输出使能

33.2.2 eMMC 预分频寄存器

地址偏移：04-07h　　属性：R/W
默认值：0x1　　大小：32 位

EMMC_PRE	位	缺省值	描述
emmc_clk_rev_en	31	0x1	SDR 模式时，该位为 1，表示控制器输出的 eMMC 数据与 eMMC 时钟下降沿对齐；该位为 0，表示控制器输出的 eMMC 数据与 eMMC 时钟上升沿对齐。DDR 模式，该位必须置为 1。
Reserved	30:10	0x0	
emmc_pre	9:0	0x1	eMMC 时钟预分频值，输出频率=PCLK/预分频值

33.2.3 eMMC 命令参数寄存器

地址偏移：0x08-0Bh　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_CMD_ARG	位	缺省值	描述
sdi_cmd_arg	31:0	0x0	命令参数

33.2.4 eMMC 命令控制寄存器

地址偏移：0C-0Fh　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_CMD_CON	位	缺省值	描述
Reserved	31:18	0x0	
func_num_abort	17:15	0x0	eMMC 卡时中断的功能号，用于多块读写时，硬件自动发送停止命令。如果 auto_stop_en 为 0，则此位无效
emmc_en	14	0x0	eMMC 使能信号。用于多块读写时，硬件自动发送停止命令，软件需配置改位为 0。如果 auto_stop_en 为 0，则此位无效。
check_on	13	0x0	是否检查 CRC，为 1 时有效
Auto_stop_en	12	0x0	硬件自动发送停止命令，多块读写时，是否硬件自动发送停止命令，为 1 时有效
Reserved	11	0x0	
long_rsp	10	0x0	是否为 136 位长响应，为 1 时表示长消息回复
Wait_rsp	9	0x0	决定是否主机等待相应，为 1 是表示等待消息回复
CMST	8	0x0	命令开始，置 1 时开始，命令结束后硬件自动清零
cmd_index	7:0	0x0	带开始 2 位的命令索引（共 8 位）

33.2.5 eMMC 命令状态寄存器

地址偏移：10-03h　　属性：RO
默认值：0x0　　大小：32 位

EMMC_CMD_STA	位	缺省值	描述
Reserved	31:13	0x0	
cmd_sent_fin	14	0x0	命令发送完成（包含响应）标志位，为 1 表示命令发送完成及响应完成
auto_stop	13	0x0	硬件自动发送停止命令标志位，为 1 表示硬件自动发送停止命令，为 0 则没有
rsp_crc_err	12	0x0	响应 CRC 错误，接收到的响应 CRC 错误。为 1 时表示响应 CRC 错误，为 0 时未发现
cmd_end	11	0x0	命令发送完成（不关心响应）。为 1 时表示命令发送完成，为 0 时未完成。
cmd_tout	10	0x0	命令超时。命令响应超时（64 个时钟周期），或者 R1b 类型的命令，忙等待超时，为 1 时表示响应超时，为 0 时未超时。
rsp_fin	9	0x0	响应结束，接收完成从设备的返回信息。为 1 时表示响应结束，为 0 时未完成。

cmd_on	8	0x0	命令传输标志位。为 1 时表示传输进行中，为 0 表示结束。
rsp_index	7:0	0x0	从设备返回的带开始 2 位的响应索引（共 8 位）

33.2.6 eMMC 命令响应寄存器 0

地址偏移：14-17h　　属性：RO
默认值：0x0　　大小：32 位

EMMC_RESP0	位	缺省值	描述
sdi_resp0	31:0	0x0	卡状态[31:0]（短），卡状态[127:96]（长）长响应的配置间 sdi_cmd_con[10]

33.2.7 eMMC 命令响应寄存器 1

地址偏移：18-1Bh　　属性：RO
默认值：0x0　　大小：32 位

EMMC_RESP1	位	缺省值	描述
sdi_resp1	31:0	0x0	未使用（短），卡状态[95:64]（长）长响应的配置间 sdi_cmd_con[10]

33.2.8 eMMC 命令响应寄存器 2

地址偏移：1C-1Fh　　属性：RO
默认值：0x0　　大小：32 位

EMMC_RESP2	位	缺省值	描述
sdi_resp2	31:0	0x0	未使用（短），卡状态[63:32]（长）长响应的配置间 sdi_cmd_con[10]

33.2.9 eMMC 命令响应寄存器 3

地址偏移：20-23h　　属性：RO
默认值：0x0　　大小：32 位

EMMC_RESP3	位	缺省值	描述
sdi_resp3	31:0	0x0	未使用（短），卡状态[31:0]（长）长响应的配置间 sdi_cmd_con[10]

33.2.10 eMMC 命令数据超时寄存器

地址偏移：24-27h　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_DTIMER	位	缺省值	描述
Reserved	31:28	0x0	
ext_dtimer	27:24	0x0	数据超时计数值，sdi_dtimer 设置为最大值时方可使用
sdi_dtimer	23:0	0x0	数据超时计数值，用分频后的时钟计数

33.2.11 eMMC 块大小寄存器

地址偏移：28-2Bh　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_BSIZE	位	缺省值	描述
Reserved	31:12	0x0	
sdi_bsize	11:0	0x0	块大小值 (0~4095)

33.2.12 eMMC 数据控制寄存器

地址偏移：2C-2Fh　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_DAT_CON	位	缺省值	描述
Reserved	31:21	0x0	
wide_mode_8b	26	0x0	wide_mode_8b 为 1，wide_mode 为 0，表示八线模式（eMMC 模式有效）
resume_rw	20	0x0	eMMC 挂起回复读写标志位。为 1 时，eMMC 挂起后恢复之前的写操作；为 0 时，恢复之前的读操作
IO_resume	19	0x0	eMMC 恢复请求。在 eMMC 设备进入挂起状态后，将此位写 1，并且 IO_suspend 位写 0 后，eMMC 设备恢复之前的操作。
IO_suspend	18	0x0	eMMC 挂起请求。写 1 后控制器会在合适的时机发送 CMD52 命令，通知 eMMC 设备进入挂起状态。恢复操作时需要将此位写 0.
RwaitReq	17	0x0	读等待请求。写 1 后控制器会在合适的时机将 DAT2 拉低，通知 eMMC 设备进入读等待状态。写 0 后恢复之前的读操作。
wide_mode	16	0x0	位宽选择位。为 1 表示 4 线模式，为 0 表示单线模式。
DMA_en	15	0x0	DMA 使能。为 1 时表示使能 DMA，为 0 表示禁止 DMA
DTST	14	0x0	数据传输开始，写 1 时数据传输开始，数据传输结束后硬件清零。
Reserved	13:12	0x0	
Blk_num	11:0	0x0	读写操作的块数。

33.2.13 eMMC 数据计数寄存器

地址偏移：30-33h　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_DAT_CNT	位	缺省值	描述
Reserved	31:24	0x0	
blk_num_cnt	23:12	0x0	当前传输块的字节数
blk_cnt	11:0	0x0	当前传输的块数

33.2.14 eMMC 数据状态寄存器

地址偏移：34-37h　　属性：RO
默认值：0x0　　大小：32 位

EMMC_DAT_STA	位	缺省值	描述
Reserved	31:17	0x0	
suspend_on	16	0x0	为 1 时表示正在挂起状态
rst_suspend	15	0x0	为 1 表示正在挂起复位。用于 eMMC 设备挂起后，控制器复位 FIFO 和 DMA 请求
R1b_tout	14	0x0	为 1 表示 R1b 类型命令超时
data_start	13	0x0	为 1 表示数据传输开始
R1b_fin	12	0x0	检测到带 busy 状态的命令完成。当发送带 busy 状态的命令时，此位为 0；当 busy 状态结束时变成 1
auto_stop	11	0x0	为 1 时表示硬件正在自动发送停止命令
Reserved	10	0x0	
r_wait_req	9	0x0	读等待发生。发送读等待请求信号到 eMMC 卡
EMMC_int	8	0x0	eMMC 中断标志位。为 1 表示检测到中断
crc_sta	7	0x0	数据发送后，从设备返回 CRC 错误
dat_crc	6	0x0	数据接收 CRC 错误
dat_tout	5	0x0	数据传输超时。为 1 时表示数据超时。
dat_fin	4	0x0	数据传输结束标志位（比如编程时）。为 1 时标志忙结束
busy_fin	3	0x0	编程错误标志位（比如编程时）。为 1 时标志忙结束
prog_err	2	0x0	编程错误标志位，为 1 时表示编程错误
tx_dat_on	1	0x0	Tx 数据发送中，为 1 时表示正在发送，为 0 时发送完成
rx_dat_on	0	0x0	Rx 数据接收中，为 1 时表示正在接收，为 0 时发送完成。

33.2.15 eMMC FIFO 状态寄存器

地址偏移：38-3Bh　　属性：RO
默认值：0x0　　大小：32 位

EMMC_FIFO_STA	位	缺省值	描述
Reserved	31:12	0x0	
tx_full	11	0x0	Tx FIFO 满标志位
tx_empty	10	0x0	Tx FIFO 空标志位
Reserved	9	0x0	
rx_full	8	0x0	Rx FIFO 满标志
rx_empty	7	0x0	Rx FIFO 空标志位
Reserved	6:0	0x0	

33.2.16 eMMC 中断寄存器

地址偏移：3C-3Fh　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_INT_MASK	位	缺省值	描述
Reserved	31:10	0x0	
R1b_fin_int	9	0x0	检测到 busy 结束中断，写 1 清零
rsp_crc_int	8	0x0	命令响应 CRC 错误中断，写 1 清零
cmd_tout_int	7	0x0	命令超时中断，写 1 清零
cmd_fin_int	6	0x0	发送完成中断，硬件清零
EMMC_int	5	0x0	检测到 eMMC 中断，写 1 清零
prog_err_int	4	0x0	编程错误中断，写 1 清零
crc_sta_int	3	0x0	数据发送后从设备返回 CRC 错误中断，写 1 清零
dat_crc_int	2	0x0	数据接收 CRC 错误中断，写 1 清零
dat_tout_int	1	0x0	数据超时中断，写 1 清零
dat_fin_int	0	0x0	数据完成中断，硬件清零

33.2.17 eMMC 命令数据寄存器

地址偏移：40-43h　　属性：RO
默认值：0x0　　大小：32 位

EMMC_DAT	位	缺省值	描述
emmc_dat	31:0	0x0	eMMC 控制器发送或者接收的数据（用于 DMA 操作）

33.2.18 eMMC 中断寄使能存器

地址偏移：64-67h　　属性：R/W
默认值：0x0　　大小：32 位

EMMC_INT_EN	位	缺省值	描述
Reserved	31:10	0x0	
R1b_fin_int_en	9	0x0	Busy 结束中断使能，为 1 时有效
rsp_crc_int_en	8	0x0	命令响应 CRC 错误中断使能，为 1 时有效
cmd_tout_int_en	7	0x0	命令超时中断使能，为 1 时有效
cmd_fin_int_en	6	0x0	命令发送完成中断使能，为 1 时有效
EMMC_int_en	5	0x0	eMMC 中断使能，为 1 时有效
prog_err_int_en	4	0x0	SD 卡编程错误中断使能，为 1 时有效
crc_sta_int_en	3	0x0	数据发送后从设备返回 CRC 错误中断使能，为 1 时有效
dat_cec_int_en	2	0x0	数据接收 CRC 错误中断使能，为 1 时有效
dat_tout_int_en	1	0x0	数据超时中断使能，为 1 时有效
dat_fin_int_en	0	0x0	数据完成中断使能，为 1 时有效

33.2.19 DLL master 锁定值寄存器

地址偏移：F0-F3h 属性：R
默认值：0x0 大小：32 位

dll_master_val	位	缺省值	描述
reserved	31:4	0x0	
dll_init_done	8	0x0	DLL master 锁定完成标志位
pm_dll_value	7：0	0x0	DLL master 锁定值

33.2.20 DLL 控制寄存器

地址偏移：F4-F7h 属性：R/W
默认值：0x0 大小：32 位

dll_con	位	缺省值	描述
reserved	31:30	0x0	
resync_dll_rd	29	0x0	内部采样时钟 DLL 重同步使能位
dll_bypass_rd	28	0x0	采样时钟 DLL bypass。该位置 1，DLL 控制器将不对 DLL 参数值进行微调。
resync_dll_pad	27	0x0	pad 时钟 DLL 重同步使能位
dll_bypass_pad	26	0x0	pad 时钟 DLL bypass。该位置 1，DLL 控制器将不对 DLL 参数值进行微调。
pm_init_start	25	0x0	DLL master 初始化开始位
pm_dll_lock_mode	24	0x0	DLL master 锁定模式。0，锁定一个周期；1，锁定半个周期
pm_dll_start_point	23:16	0x0	DLL master 初始化的起点值。该值应该低于一个周期的延迟值
pm_dll_increment	15:8	0x0	DLL master 初始化的步进值
pm_dll_adj_cnt	7:0	0x0	刷新锁定值的时间间隔

33.2.21 DLL 延迟参数寄存器

地址偏移：F8-FBh 属性：R/W
默认值：0x0 大小：32 位

param_delay	位	缺省值	描述
reserved	31:16	0x0	
clk_rd_delay	15:8	0x0	内部采样时钟延迟参数，用于调整控制器采样数据的采样点。DDR 模式下，该值一般比 clk_pad_delay 大。
clk_pad_delay	7:0	0x0	时钟延迟参数。DDR 模式下，此时的时钟与内部参考时钟的相位关系应该为 90° 。例如,内部参考时钟为 50MHz (20ns) ，此时的 clk_pad_delay 值 应该为 50 (50*100ps) 。

33.2.22 总线模式选择寄存器

地址偏移：FC-FFh 属性：R/W
默认值：0x0 大小：32 位

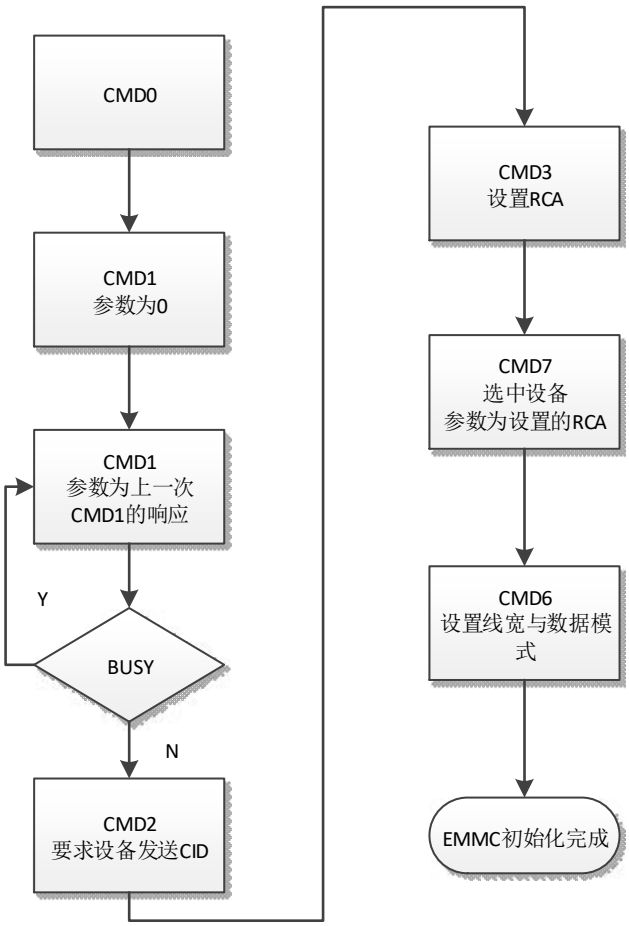
sdio_emmc_sel	位	缺省值	描述
reserved	31:4	0x0	
bus_sel	1	0x0	SDIO 与 eMMC 总线模式选择。0 表示 eMMC 总线模式；1 表示 SDIO 总线模式
data_mode	0	0x0	数据模式选择。0，表示 SDR 数据模式； 1，表示 DDR 数据模式

33.3 软件配置流程

33.3.1 eMMC 正常读写

1. 配置 emmc_con，使能时钟
2. 配置 emmc_pre，输出预设频率的时钟
3. 配置 emmc_bsize，设置一块数据的大小，单位为字节
4. 配置 emmc_dtimer，设置超时计数，最大为 100ms
5. 配置 emmc_int_en，设置中断使能，设置读写完成及错误中断使能
6. 配置 emmc_ dat_con，设置线宽数据模式
7. 配置 bus_sel，设置 DDR 或 SDR 模式
8. eMMC 设备初始化：初始化流程需要对命令通道进行操作，先配置 emmc_cmd_arg，填写命令参数，再 emmc_cmd_con，开始发送命令，通过检测 emmc_int_msk 检查命令发送完成中断；命令完成后读 emmc_rsp0~3，读 eMMC 发回来的回复
9. 初始化完成后，可发送数据相关命令进行数据操作。配置 DMA（注：SDIO 控制器基址加上 0x800 为 DMA 读，基址加上 0x400 为 DMA 写；其余配置及使用方式同 34 节 DMA 控制器）与 eMMC 控制器。命令发送同样时通过配置 emmc_cmd_arg 和 emmc_cmd_con 来完成。数据收发是否完成，通过检测 emmc_int_msk 来判断。

33.3.2 eMMC 初始化流程



33.3.3 DDR 模式设置

在开启 DDR 模式前，需要先初始化 eMMC 设备，同时设置 eMMC 设备为 DDR 模式；

然后初始化 DLL，设置延迟值，最后将控制器设置为 DDR 模式。流程如下：

1. DLL 设置:pm_dll_lock_mode 置 1 锁定半个周期时钟;pm_dll_start_point 置为 0x10（该值应该大于 0 小于半周期）；pm_dll_adj_cnt 置为 0xff；pm_dll_increment，pm_dll_bypass，pm_dll_resync 都置为 1；最后 pm_init_start 置 1 开始 DLL 初始化;
2. DLL 锁定:DLL 设置完成后，检测 dll_init_done 寄存器，该值为 1 表示 DLL 完成锁定
3. 配置延迟值：将 DLL 锁定值 pm_dll_value 除以 2 后的值写入 clk_pad_delay；clk_rd_delay 应该比 pm_dll_value/2 大，具体值视不同芯片和环境条件（PCB 走线，工作温度等）而定，软件需要根据实际情况对 clk_rd_delay 进行调整
4. data_mode 置 1,开启 DDR 模式

34 SDIO 控制器 (D28:F1/2)

34.1 功能概述

龙芯 2K3000 集成了两个 SDIO 控制器，用于 SD Memory 和 SDIO 卡的读写。SDIO 控制器特性如下：

- 8字（32字节）数据发送/接收FIFO
- 扩展的256位SD卡状态寄存器
- 8位预分频逻辑（频率=系统时钟/（p+1））
- DMA数据传输模式
- 专用独立DMA通道
- 1位/4位（宽总线）的SD模式

34.2 SDIO 协议概述

SDIO 是一个串行通信方式，主设备和从设备通过消息传递来实现数据和状态的传输。如下图是一个写多块数据的示意框图，过程如下：

1. 主设备通过命令线发送写命令消息给从设备
2. 从设备接收完消息之后通过命令线发送应答消息给主设备
3. 主设备接收到正确的应答消息后，通过数据线发送一块数据（512K Byte 或者更多）给从设备，并且检测数据线忙状态
4. 从设备接收到正确的数据后会进入编程状态，此时将数据线置为忙状态，不再响应主设备的数据请求
5. 主设备检测到从设备编程完成，继续发送下一块数据。
6. 主设备发送完最后一块数据时，通过命令线发送停止命令给从设备，收到正确应答之后完成这次多块写操作。

34.3 寄存器描述

SDIO 控制器的寄存器详细说明如下:

34.3.1 SDIO 控制寄存器

地址偏移：00-03h　　属性：R/W
默认值：0x0　　大小：32 位

SDI_CON	位	缺省值	描述
•	31:9	0x0	
soft_rst	8	0x0	软件复位，整个模块复位。复位完成后硬件自动清零

Reserved	7:1	0x0	
enclk	0	0x0	SD 时钟输出使能

34.3.2 SDIO 预分频寄存器

地址偏移：04-07h　　属性：R/W
默认值：0x01　　大小：32 位

SDI_PRE	位	缺省值	描述
sdio_clk_rev_en	31	0x1	SDR 模式时，该位为 1，表示控制器输出的 sdio 数据与 sdio 时钟下降沿对齐；该位为 0，表示控制器输出的 sdio 数据与 sdio 时钟上升沿对齐。DDR 模式，该位必须置为 1。
Reserved	30:10	0x0	
Sdi_pre	9:0	0x1	SDIO 时钟预分频值，输出频率=PCLK/预分频值

34.3.3 SDIO 命令参数寄存器

地址偏移：08-0Bh　　属性：R/W
默认值：0x0　　大小：32 位

SDI_CMD_ARG	位	缺省值	描述
sdi_cmd_arg	31:0	0x0	命令参数

34.3.4 SDIO 命令控制寄存器

地址偏移：0C-0Fh　　属性：R/W
默认值：0x0　　大小：32 位

SDI_CMD_CON	位	缺省值	描述
Reserved	31:19	0x0	
cmd6_data_en	18	0x0	写 1 使能 cmd6 数据传输
func_num_abort	17:15	0x0	SDIO 卡时中断的功能号，用于多块读写时，硬件自动发送停止命令。如果 auto_stop_en 为 0，则此位无效
sdio_en	14	0x0	SDIO 使能信号。用于多块读写时，硬件自动发送停止命令，为 1 时发送 CMD52，为 0 是发送 CMD12。如果 auto_stop_en 为 0，则此位无效
check_on	13	0x0	是否检查 CRC，为 1 时有效
Auto_stop_en	12	0x0	硬件自动发送停止命令，多块读写时，是否硬件自动发送停止命令，为 1 时有效
Reserved	11	0x0	
long_rsp	10	0x0	是否为 136 位长响应，为 1 时表示长消息回复
Wait_rsp	9	0x0	决定是否主机等待相应，为 1 是表示等待消息回复
CMST	8	0x0	命令开始，置 1 时开始，命令结束后硬件自动清零
cmd_index	7:0	0x0	带开始 2 位的命令索引（共 8 位）

34.3.5 SDIO 命令状态寄存器

地址偏移：10-13h　　属性：RO
默认值：0x0　　大小：32 位

SDI_CMD_STA	位	缺省值	描述
Reserved	31:13	0x0	
cmd_sent_fin	14	0x0	命令发送完成（包含响应）标志位，为 1 表示命令发送完成及响应完成
auto_stop	13	0x0	硬件自动发送停止命令标志位，为 1 表示硬件自动发送停止命令，为 0 则没有
rsp_crc_err	12	0x0	响应 CRC 错误，接收到的响应 CRC 错误。为 1 时表示响应 CRC 错误，为 0 时未发现
cmd_end	11	0x0	命令发送完成（不关心响应）。为 1 时表示命令发送完成，为 0 时未完成。
cmd_tout	10	0x0	命令超时。命令响应超时（64 个时钟周期），或者 R1b 类型的命令，忙等待超时，为 1 时表示响应超时，为 0 时未超时。
rsp_fin	9	0x0	响应结束，接收完成从设备的返回信息。为 1 时表示响应结束，为 0 时未完成。
cmd_on	8	0x0	命令传输标志位。为 1 时表示传输进行中，为 0 表示结束。
rsp_index	7:0	0x0	从设备返回的带开始 2 位的响应索引（共 8 位）

34.3.6 SDIO 命令响应寄存器 0

地址偏移：14-17h　　属性：RO
默认值：0x0　　大小：32 位

SDI_RESP0	位	缺省值	描述
sdi_resp0	31:0	0x0	卡状态[31:0]（短），卡状态[127:96]（长）长响应的配置间 sdi_cmd_con[10]

34.3.7 SDIO 命令响应寄存器 1

地址偏移：18-1Bh　　属性：RO
默认值：0x0　　大小：32 位

SDI_RESP1	位	缺省值	描述
sdi_resp1	31:0	0x0	未使用（短），卡状态[95:64]（长）长响应的配置间 sdi_cmd_con[10]

34.3.8 SDIO 命令响应寄存器 2

地址偏移：1C-1Fh　　属性：RO

默认值：0x0　　大小：32 位

SDI_RESP2	位	缺省值	描述
sdi_resp2	31:0	0x0	未使用（短），卡状态[63:32]（长）长响应的配置间 sdi_cmd_con[10]

34.3.9 SDIO 命令响应寄存器 3

地址偏移：20-23h　　属性：RO
默认值：0x0　　大小：32 位

SDI_RESP3	位	缺省值	描述
sdi_resp3	31:0	0x0	未使用（短），卡状态[31:0]（长）长响应的配置间 sdi_cmd_con[10]

34.3.10　SDIO 命令数据超时寄存器

地址偏移：24-27h　　属性：R/W
默认值：0x0　　大小：32 位

SDI_DTIMER	位	缺省值	描述
Reserved	31:28	0x0	
ext_dtimer	27:24	0x0	数据超时计数值，sdi_dtimer 设置为最大值时方可使用
sdi_dtimer	23:0	0x0	数据超时计数值，用分频后的时钟计数

34.3.11　SDIO 块大小寄存器

地址偏移：28-2Ch　　属性：R/W
默认值：0x0　　大小：32 位

SDI_BSIZE	位	缺省值	描述
Reserved	31:12	0x0	
sdi_bsize	11:0	0x0	块大小值 (0~4095)

34.3.12　SDIO 数据控制寄存器

地址偏移：2C-2Fh　　属性：R/W
默认值：0x0　　大小：32 位

SDI_DAT_CON	位	缺省值	描述
Reserved	31:21	0x0	
sdio_esp_rdy	25	0x0	写 1 表示对 SDIO 设备发送 CMD53 进行写操作时，并且写数据发送完成后，该设备不会拉低数据线 0 进入 busy 状态
sdio_esp_card	24	0x0	写 1 表示 SDIO 特殊设备，与 sdio_esp_rdy 配合使用

resume_rw	20	0x0	SDIO 挂起回复读写标志位。为 1 时，SDIO 挂起后恢复之前的写操作；为 0 时，恢复之前的读操作
IO_resume	19	0x0	SDIO 恢复请求。在 SDIO 设备进入挂起状态后，将此位写 1，并且 IO_suspend 位写 0 后，SDIO 设备恢复之前的操作。
IO_suspend	18	0x0	SDIO 挂起请求。写 1 后控制器会在合适的时机发送 CMD52 命令，通知 SDIO 设备进入挂起状态。恢复操作时需要将此位写 0.
RwaitReq	17	0x0	读等待请求。写 1 后控制器会在合适的时机将 DAT2 拉低，通知 SDIO 设备进入读等待状态。写 0 后恢复之前的读操作。
wide_mode	16	0x0	位宽选择位。为 1 表示 4 线模式，为 0 表示单线模式。
DMA_en	15	0x0	DMA 使能。为 1 时表示使能 DMA，为 0 表示禁止 DMA
DTST	14	0x0	数据传输开始，写 1 时数据传输开始，数据传输结束后硬件清零。
Reserved	13:12	0x0	
Blk_num	11:0	0x0	读写操作的块数。

34.3.13 SDIO 数据计数寄存器

地址偏移：30-33h 属性：R/W
默认值：0x0 大小：32 位

SDI_DAT_CNT	位	缺省值	描述
Reserved	31:24	0x0	
blk_num_cnt	23:12	0x0	当前传输块的字节数
blk_cnt	11:0	0x0	当前传输的块数

34.3.14 SDIO 数据状态寄存器

地址偏移：34-37h 属性：RO
默认值：0x0 大小：32 位

SDI_DAT_STA	位	缺省值	描述
Reserved	31:17	0x0	
suspend_on	16	0x0	为 1 时表示正在挂起状态
rst_suspend	15	0x0	为 1 表示正在挂起复位。用于 SDIO 设备挂起后，控制器复位 FIFO 和 DMA 请求
R1b_tout	14	0x0	为 1 表示 R1b 类型命令超时
data_start	13	0x0	为 1 表示数据传输开始
R1b_fin	12	0x0	检测到带 busy 状态的命令完成。当发送带 busy 状态的命令时，此位为 0；当 busy 状态结束时变成 1

auto_stop	11	0x0	为 1 时表示硬件正在自动发送停止命令
Reserved	10	0x0	
r_wait_req	9	0x0	读等待发生。发送读等待请求信号到 SDIO 卡
SDIO_int	8	0x0	SDIO 中断标志位。为 1 表示检测到中断
crc_sta	7	0x0	数据发送后，从设备返回 CRC 错误
dat_crc	6	0x0	数据接收 CRC 错误
dat_tout	5	0x0	数据传输超时。为 1 时表示数据超时。
dat_fin	4	0x0	数据传输结束标志位（比如编程时）。为 1 时标志忙结束
busy_fin	3	0x0	编程错误标志位（比如编程时）。为 1 时标志忙结束
prog_err	2	0x0	编程错误标志位，为 1 时表示编程错误
tx_dat_on	1	0x0	Tx 数据发送中，为 1 时表示正在发送，为 0 时发送完成
rx_dat_on	0	0x0	Rx 数据接收中，为 1 时表示正在接收，为 0 时发送完成。

34.3.15 SDIO FIFO 状态寄存器

地址偏移：38-3Bh 属性：RO
默认值：0x0 大小：32 位

SDI_FIFO_STA	位	缺省值	描述
Reserved	31:12	0x0	
tx_full	11	0x0	Tx FIFO 满标志位
tx_empty	10	0x0	Tx FIFO 空标志位
Reserved	9	0x0	
rx_full	8	0x0	Rx FIFO 满标志
rx_empty	7	0x0	Rx FIFO 空标志位
Reserved	6:0	0x0	

34.3.16 SDIO 中断寄存器

地址偏移：3C-3Fh 属性：R/W
默认值：0x0 大小：32 位

寄存器名称	偏移地址	读/写 (R/W)	功能描述	复位值
SDI_INT_MASK	0x3c	R/W	SDIO 中断寄存器	0x0

SDI_INT_MASK	位	缺省值	描述
Reserved	31:10	0x0	
R1b_fin_int	9	0x0	检测到 busy 结束中断，写 1 清零
rsp_crc_int	8	0x0	命令响应 CRC 错误中断，写 1 清零
cmd_tout_int	7	0x0	命令超时中断，写 1 清零
cmd_fin_int	6	0x0	发送完成中断，硬件清零

SDIO_int	5	0x0	检测到 SDIO 中断，写 1 清零
prog_err_int	4	0x0	SD 卡编程错误中断，写 1 清零
crc_sta_int	3	0x0	数据发送后从设备返回 CRC 错误中断，写 1 清零
dat_crc_int	2	0x0	数据接收 CRC 错误中断，写 1 清零
dat_tout_int	1	0x0	数据超时中断，写 1 清零
dat_fin_int	0	0x0	数据完成中断，硬件清零

34.3.17 SDIO 命令数据寄存器

地址偏移：40-43h 属性：RO
默认值：0x0 大小：32 位

SDI_DAT	位	缺省值	描述
sdi_dat	31:0	0x0	SDIO 控制器发送或者接收的数据（用于 DMA 操作）

34.3.18 SDIO 中断寄使能存器

地址偏移：64-67h 属性：R/W
默认值：0x0 大小：32 位

SDI_INT_EN	位	缺省值	描述
Reserved	31:10	0x0	
R1b_fin_int_en	9	0x0	Busy 结束中断使能，为 1 时有效
rsp_crc_int_en	8	0x0	命令响应 CRC 错误中断使能，为 1 时有效
cmd_tout_int_en	7	0x0	命令超时中断使能，为 1 时有效
cmd_fin_int_en	6	0x0	命令发送完成中断使能，为 1 时有效
SDIO_int_en	5	0x0	SDIO 中断使能，为 1 时有效
prog_err_int_en	4	0x0	SD 卡编程错误中断使能，为 1 时有效
crc_sta_int_en	3	0x0	数据发送后从设备返回 CRC 错误中断使能，为 1 时有效
dat_cec_int_en	2	0x0	数据接收 CRC 错误中断使能，为 1 时有效
dat_tout_int_en	1	0x0	数据超时中断使能，为 1 时有效
dat_fin_int_en	0	0x0	数据完成中断使能，为 1 时有效

34.3.19 DLL master 锁定值寄存器

地址偏移：F0-F3h 属性：R
默认值：0x0 大小：32 位

dll_master_val	位	缺省值	描述
reserved	31:4	0x0	
dll_init_done	8	0x0	DLL master 锁定完成标志位
pm_dll_value	7：0	0x0	DLL master 锁定值

34.3.20 DLL 控制寄存器

地址偏移：F4-F7h 属性：R/W
默认值：0x0 大小：32 位

dll_con	位	缺省值	描述
reserved	31:30	0x0	
resync_dll_rd	29	0x0	内部采样时钟 DLL 重同步使能位
dll_bypass_rd	28	0x0	采样时钟 DLL bypass。该位置 1，DLL 控制器将不对 DLL 参数值进行微调。
resync_dll_pad	27	0x0	pad 时钟 DLL 重同步使能位
dll_bypass_pad	26	0x0	pad 时钟 DLL bypass。该位置 1，DLL 控制器将不对 DLL 参数值进行微调。
pm_init_start	25	0x0	DLL master 初始化开始位
pm_dll_lock_mode	24	0x0	DLL master 锁定模式。0，锁定一个周期；1，锁定半个周期
pm_dll_start_point	23:16	0x0	DLL master 初始化的起点值。该值应该低于一个周期的延迟值
pm_dll_increment	15:8	0x0	DLL master 初始化的步进值
pm_dll_adj_cnt	7:0	0x0	刷新锁定值的时间间隔

34.3.21 DLL 延迟参数寄存器

地址偏移：F8-FBh 属性：R/W
默认值：0x0 大小：32 位

param_delay	位	缺省值	描述
reserved	31:16	0x0	
clk_rd_delay	15:8	0x0	内部采样时钟延迟参数，用于调整控制器采样数据的采样点。DDR 模式下，该值一般比 clk_pad_delay 大。
clk_pad_delay	7:0	0x0	时钟延迟参数。DDR 模式下，此时的时钟与内部参考时钟的相位关系应该为 90° 。例如,内部参考时钟为 50MHz (20ns) ，此时的 clk_pad_delay 值 应该为 50 (50*100ps) 。

34.3.22 总线模式选择寄存器

地址偏移：FC-FFh 属性：R/W
默认值：0x0 大小：32 位

sdio_emmc_sel	位	缺省值	描述
reserved	31:4	0x0	
bus_sel	1	0x0	SDIO 与 eMMC 总线模式选择。0 表示 SDIO 总线模式；1 表示 eMMC 总线模式。切换至 eMMC 模式时，eMMC 最多只能支持四线模式。

data_mode	0	0x0	数据模式选择。0，表示 SDR 数据模式； 1，表示 DDR 数据模式

34.4 软件编程指南

34.4.1 SD Memory 卡软件编程说明

SD Memory 卡要想正常工作，必须要先初始化。初始化的过程需要发送不同的命令序列来配置从设备。初始化完成之后就可以正常工作了。

配置寄存器的流程如下：

1. 配置 sdi_con，使能输出时钟
2. 配置 sdi_pre，设置一个分频系数，如果时序不满足，可以设置输出反向时钟来调整时序。
3. 配置 sdi_int_en，使能命令、数据完成及其他中断。
4. 按照协议要求初始化控制器

发送命令的配置寄存器过程如下：

- 根据发送的命令，配置 cmd_arg 寄存器
- 配置 cmd_con 寄存器，发送命令
- 读 sdi_int_msk 寄存器，检查是否传输完成，是否有错误
- 如果需要，读 sdi_rsp 寄存器

初始化的流程如下：

CMD0 → CMD8 → ACMD41 (即 CMD55 → CMD41) → CMD2 → CMD3

→ CMD7 → ACMD6（用于配置是否用 4bit 数据线传输）

5. 进行数据操作之前需要配置 Bsize 寄存器，Dtimer 寄存器
6. 数据的操作必须要配置 DMA，配置 dat_con 寄存器并配置 DMA（注：SDIO 控制器基址加上 0x800 为 DMA 读，基址加上 0x400 为 DMA 写；其余配置及使用方式同 29.7 节 DMA 控制器）
7. 读 sdi_int_msk 寄存器，检测是否传输完成，是否有错误。
8. 没有错误则完成一次数据传输，不需要软件发送停止命令

34.4.2 SDIO 卡软件编程说明

SDIO 卡的初始化流程和 SD memory 卡不同，其初始化流程如下：

1. 配置 sdi_con，使能输出时钟。
2. 配置 sdi_pre，设置一个分频系数，如果时序不满足，可以设置输出反向
3. 时钟来调整时序。

4. 配置 sdi_int_en，使能命令、数据完成及其他中断。

5. 初始化流程如下：

发送命令的配置寄存器过程如下：

- 根据发送的命令，配置 cmd_arg 寄存器
- 配置 cmd_con 寄存器，发送命令
- 读 sdi_int_msk 寄存器，检查是否传输完成，是否有错误
- 如果需要，读 sdi_rsp 寄存器

初始化的流程如下（对 CCCR 的操作）：

6. CMD52（复位） CMD5（等待上电完成） CMD3（获取 RCA） CMD7（选择相应 RCA 的卡） CMD52（配置是否用 4bit 数据线传输） CMD52（配置读写数据的块大小） CMD52（打开 IO 中断使能）进行数据操作之前需要配置 Bsize 寄存器，Dtimer 寄存器

7. 数据的操作必须要配置 DMA，配置 dat_con 寄存器并配置 DMA（注：读操作时先配置 DMA，写操作时先配置 dat_con）

8. 发送读写数据的命令时，如果需要硬件自动发送停止命令，需要配置 auto_stop 和 sdio_en。读写数据时需要先读写支持的 Function，配置相应的 FBR 的指针寄存器，再发送多块读写（CMD53）或者单块读写（CMD52）命令进行读写。

9. 读 sdi_int_msk 寄存器，检测是否传输完成，是否有错误。

10. 没有错误则完成一次数据传输，不需要软件发送停止命令

11. 如果检测到 IO 中断，控制器会置起响应的中断，但是不会停止当前的操作。

12. 对于读等待，控制器会在合适的时机将 DAT2 拉低，通知 SDIO 卡停止发送数据。所以如果当前正在传输数据过程中，控制器可能会在当前一块数据传输结束后，发出读等待信号，这时才不会接收下一块数据。

13. 对于挂起和恢复操作。有可能出现挂起嵌套情况，比如说操作 1 被操作 2 中断而挂起，然后操作 2 被操作 3 中断而挂起。挂起时中断的现场需要软件保存（如当前的读写标志位，当前传输的块数，地址等），进行入栈操作。恢复时需要软件再按相应的顺序出栈。

34.4.3 DDR 模式设置

在开启 DDR 模式前，需要先初始化 sdio 设备，同时设置 sdio 设备为 DDR 模式；

然后初始化 DLL，设置延迟值，最后将控制器设置为 DDR 模式。流程如下：

1. DLL 设置:pm_dll_lock_mode 置 1 锁定半个周期时钟;pm_dll_start_point 置为 0x10（该值应该大于 0 小于半周期）；pm_dll_adj_cnt 置为 0xff；pm_dll_increment，pm_dll_bypass，pm_dll_resync 都置为 1；最后 pm_init_start 置 1 开始 DLL 初始化;

2. DLL 锁定:DLL 设置完成后，检测 dll_init_done 寄存器，该值为 1 表示 DLL 完成锁定

3. 配置延迟值：将 DLL 锁定值 pm_dll_value 除以 2 后的值写入 clk_pad_delay；clk_rd_delay 应该比 pm_dll_value/2 大，具体值视不同芯片和环境条件（PCB 走线，工作温度等）而定，软件需要根据实际情况对 clk_rd_delay 进行调整

4. data_mode 置 1,开启 DDR 模式

修订记录

版本号	更新内容
V1.0	发布版本

技术支持

可通过邮箱向我司提交芯片手册和产品使用的问题，并获取技术支持。

服务邮箱：service@loongson.cn

龙芯中科技术股份有限公司

Loongson Technology Corporation Limited

地址：北京市海淀区中关村环保科技示范园龙芯产业园 2 号楼

Building No.2, Loongson Industrial Park,

Zhongguancun Environmental Protection Park, Haidian District, Beijing

电话（Tel）：010-62546668

传真（Fax）：010-62600826