龙芯中科
LOONGSON TECHNOLOGY

LOONGSON

龙芯 2K1500 处理器

用户手册

V1.0

2024 年 08 月

龙芯中科技术股份有限公司

北京市海淀区中关村环保科技示范园龙芯产业园 100095
Loongson Industrial Park, Zhongguancun Environmental Protection Park,
Haidian District, Beijing 10095, P.R.China
www.loongson.cn

版权声明

免责声明

本文档仅提供阶段性信息，所含内容可根据产品的实际情况随时更新，恕不另行通知。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

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

电话(Tel)：010-62546668

传真(Fax)：010-62600826

阅读指南

《龙芯 2K1500 处理器用户手册》主要介绍龙芯 2K1500 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、寄存器列表及位域等进行详细说明。

版本信息

版本信息	文档名	龙芯 2K1500 处理器用户手册
	版本号	V1.0
	创建人	芯片研发部

更新历史

序号	版本号	更新内容
1	V1.0	初版发布

手册信息反馈：service@loongson.cn

目录

图目录

表目录

1 概述

龙芯 2K1500 处理器(简称龙芯 2K1500)主要面向于工控领域应用。片内集成 2 个 LA264 处理器核，采用 LoongArch 指令系统（龙架构），最高主频 1.2GHz，64 位 DDR3 控制器，并集成各种系统 IO 接口。

龙芯 2K1500 的主要特征如下：

- 片内集成两个 64 位的双发射超标量 LA264 处理器核
- 片内集成共享的 2MB 二级 Cache
- 片内集成 64 位 800MHz 的 DDR3 控制器(支持 32 位模式下的 ECC)
- 1 个 x4 PCIe 3.0 接口，可拆分为 4 个独立 PCIe x1 接口，仅 RC 模式
- 1 个 x4 PCIe 3.0 接口，可拆分为 2 个独立 PCIe x1 接口，可作为 RC 或者 EP
- 1 个 4 通道 DMA
- 片内集成 1 个 SATA3.0 接口
- 片内集成 5 个 USB2.0 接口，其中 1 个为 OTG
- 片内集成 2 个 RGMII 千兆网接口
- 片内集成 RTC/HPET 模块
- 片内集成 3 个全功能 UART 接口
- 片内集成 1 个 NAND 控制器
- 片内集成 6 个 CAN 控制器，符合 CAN2.0A/B 规范
- 片内集成 6 个 PWM 控制器
- 片内集成 1 个 SDIO 控制器，兼容 SD Memory 4.0/MMC/SDIO 4.0 协议
- 片内集成 1 个 eMMC 控制器，兼容 eMMC 5.1 协议
- 片内集成 4 个 SPI 控制器，其中 2 个支持 QSPI
- 片内集成 4 个 I2C 控制器
- 片内集成 1 个 LIO 控制器
- 最多 96 个 GPIO 接口
- 片内集成 1 个温度传感器
- 采用 FCBGA608 封装，封装尺寸为 27mm x 27mm

1.1 体系结构框图

龙芯 2K1500 的结构如图 1- 1 所示。一级交叉开关连接两个处理器核、两个二级 Cache 以及 IO 子网络（Cache 访问路径）。二级交叉开关连接两个二级 Cache、内存控制器以及启动模块（SPI 或者 LIO）。IO 子网络采用南北桥结构，北桥包含 2 个 PCIe 和 DMA 模块，通

过北桥网络连接一级交叉开关，以减少处理器访问延迟。南桥包括 GMAC、SATA、USB、NAND、SDIO、eMMC、加解密以及 MISC 模块，通过南桥网络与北桥相连。

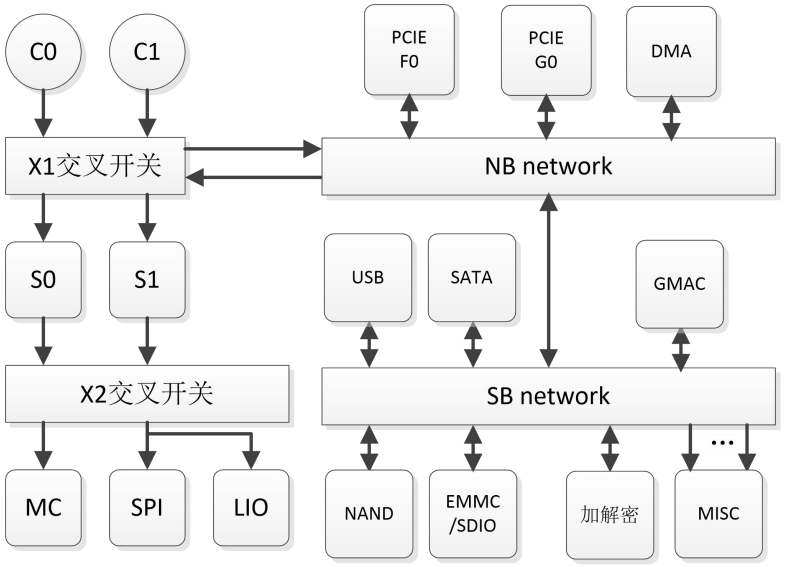


图 1-1 龙芯 2K1500 结构图

1.2 芯片主要功能

1.2.1 处理器核

- LA264
- 采用 LoongArch 指令系统（龙架构）
- 32KB 数据 Cache 和 32KB 的指令 Cache
- 2MB 共享二级 Cache
- 通过目录协议维护 I/O DMA 访问的 Cache 一致性

1.2.2 内存接口

- DDR3 控制器，最高工作频率 800MHz
- 支持 32 位模式下的 ECC
- 可配置为 64/32/16 位模式
- 支持命令调度

1.2.3 PCIe 接口

- 兼容 PCIe 3.0
- 双独立 X4 接口，均支持 X4/X2/X1 链路宽度自适应
- 其中一路 X4 接口可以配置为 4 个 X1 接口，4X1 模式下支持最高速率为 PCIe 2.0 速率，仅 RC 模式
- 其中一路 X4 接口可以配置为 2 个 X1 接口，2X1 模式下支持最高速率为 PCIe 2.0

速率，RC 或 EP 模式

1.2.4 SATA 控制器

- 1 个 SATA 端口
- 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

1.2.5 USB2.0 控制器

- 5 个独立的 USB2.0 的 HOST 端口
- 其中端口 0 固定为 OTG 工作模式
- 兼容 USB1.1 和 USB2.0
- 内部集成 XHCI 控制器

1.2.6 GMAC 控制器

- 两路 10/100/1000Mbps 自适应以太网 MAC
- 双网卡均兼容 IEEE 802.3
- 对外部 PHY 实现 RGMII 接口
- 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

1.2.7 NAND 控制器

- 最大支持单片 16GB NAND Flash
- 最大支持 4 个片选
- 支持 SLC 和 MLC
- 支持 512/2K/4K/8K 页

1.2.8 SPI 控制器

- 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
- 支持系统启动
- 支持标准读、连续地址读、快速读、双路 I/O 等 SPI Flash 读模式
- 其中 2 个支持 QSPI

1.2.9 UART

- 3 个全功能 UART 和流控 TXD,RXD,CTS, RTS, DSR,DTR,DCD, RI
- 最多 12 个 UART 接口
- 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

1.2.10 I2C 总线

- 兼容 SMBUS（100Kbps）
- 与 PHILIPS I2C 标准相兼容
- 履行双向同步串行协议
- 均支持主设备，其中 I2C2 也支持从设备
- 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

1.2.11 PWM

- 6 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

1.2.12 HPET

- 64 位计数器
- 支持 1 个周期性中断
- 支持 2 个非周期性中断

1.2.13 RTC

- 可产生 3 个计时中断

1.2.14 Watchdog

- 32 比特计数器及初始化寄存器

1.2.15 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

1.2.16 CAN

- 符合 CAN2.0A/B 规范
- 6 路 CAN 接口
- 支持中断

1.2.17 GPIO

- 12 个专用 GPIO 引脚
- 其余引脚与其他接口相复用，使用各个接口电压域
- 输入中断功能
- 中断极性、触发类型可设置

1.2.18 加解密模块

- AES、DES 算法支持
- RSA 算法支持

1.2.19 SDIO 控制器

- 1 路独立 SDIO 控制器
- 兼容 SD Memory 4.0/MMC/SDIO 4.0 协议

1.2.20 eMMC 控制器

- 1 路独立 eMMC 控制器
- 兼容 eMMC 5.1 协议

2 引脚定义

龙芯 2K1500 的引脚进行了大量的功能复用。对于有复用关系的引脚，在介绍完其本身的功能之外会同时在下方给出其他复用功能的描述。

2.1 约定

本章对龙芯 2K1500 引脚定义的说明使用以下约定：

- 信号名

信号名的选取以方便记忆和明确标识功能为原则。低有效信号以 n/N 结尾，高有效信号则不带 n/N。

- 类型

信号的输入输出类型由一个代码表示，见表 2- 1。

表 2- 1 信号类型代码

代码	描述
A	模拟
DIFF I/O	双向差分
DIFF IN	差分输入
DIFF OUT	差分输出
I	输入
I/O	双向
O	输出
OD	开漏输出
P	电源
G	地

2.2 DDR3 接口

表 2- 2 DDR3 SDRAM 控制器接口信号

信号名称	类型	描述
DDR_DQ[63:0]	I/O	DDR3 SDRAM 数据总线信号
DDR_DQSp[7:0] DDR_DQSn[7:0]	DIFF I/O	DDR3 SDRAM 数据选通
DDR_DQM[7:0]	O	DDR3 SDRAM 数据屏蔽
DDR_A[15:0]	O	DDR3 SDRAM 地址总线信号
DDR_BA[2:0]	O	DDR3 SDRAM 逻辑 BANK 地址信号
DDR_WEn	O	DDR3 SDRAM 写使能信号
DDR_CASn	O	DDR3 SDRAM 列地址选择信号
DDR_RASn	O	DDR3 SDRAM 行地址选择信号
DDR_CSn[1:0]	O	DDR3 SDRAM 片选信号
DDR_CKE[1:0]	O	DDR3 SDRAM 时钟使能信号

DDR_CKp[1:0] [7:6] DDR_CKn[1:0] [7:6]	DIFF OUT	DDR3 SDRAM 差分时钟输出信号
DDR_ODT[1:0]	O	DDR3 SDRAM ODT 信号
DDR_RESETn	O	DDR3 SDRAM 复位控制信号
DDR_REXT	I/O	外部参考电阻，通过 240ohm/1%电阻连至地(PCB 下拉)

2.3 PCIe 接口

表 2- 3 PCIe 总线信号

信号名称	类型	描述
PCIe0_REFCLKIN_P PCIe0_REFCLKIN_N	DIFF IN	系统参考时钟输入
PCIe1_REFCLKIN_P PCIe1_REFCLKIN_N	DIFF IN	PCIe1 PHY 参考时钟输入
PCIe_REFCLKOUT_P[5:0] PCIe_REFCLKOUT_N[5:0]	DIFF OUT	PCIe 参考时钟输出
PCIe1_REFRES	A	PCIe1 外部参考电阻,DIE 内部已做端接处理，PCB 应悬空
PCIe[1:0]_TXP[3:0] PCIe[1:0]_TXN[3:0]	DIFF OUT	PCIe 差分数据输出
PCIe[1:0]_RXP[3:0] PCIe[1:0]_RXN[3:0]	DIFF IN	PCIe 差分数据输入
PCIe[1:0]_RSTN	O	PCIe 复位

PCIe1_RSTN 接口与 GPIO43 有复用关系。

2.4 LIO 接口

表 2- 4 LIO 接口信号

信号名称	类型	信号描述
LIO_RDn	O	LIORDn 输出
LIO_WRn	O	LIOWRn 输出
LIO_DEN	O	LIO 数据使能
LIO_DIR	O	LIO 方向控制，0 代表读，1 代表写
LIO_ADLOCK	O	LIO 地址/数据选择信号
LIO_AD[15:0]	I/O	LIO 双向 AD 信号
LIO_A[6:0]	O	LIO 地址低位
LIO_CSn	O	LIO 片选信号
LIO_RDY	I	LIO 数据准备好输入

LIO 接口与 UART 以及 GPIO 有复用关系，如表 2- 5LIO 与 UART 复用关系和表 2- 6 LIO

与 GPIO 复用关系所示。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 5 LIO 与 UART 复用关系

信号名称	复用名称	复用类型	复用信号描述
LIO_DEN	UART1_TXD	O	串口数据输出
LIO_DIR	UART1_RXD	I	串口数据输入
LIO_ADLOCK	UART1_RTS	O	串口数据传输请求
LIO_AD00	UART1_DTR	O	串口初始化完成
LIO_AD01	UART1_RI	I	外部 MODEM 探测到振铃信号
LIO_AD02	UART1_CTS	I	设备接受数据就绪
LIO_AD03	UART1_DSR	I	设备初始化完成
LIO_AD04	UART1_DCD	I	外部 MODEM 探测到载波信号
LIO_AD05	UART2_TXD	O	串口数据输出
LIO_AD06	UART2_RXD	I	串口数据输入
LIO_AD07	UART2_RTS	O	串口数据传输请求
LIO_AD08	UART2_DTR	O	串口初始化完成
LIO_AD09	UART2_RI	I	外部 MODEM 探测到振铃信号
LIO_AD11	UART2_CTS	I	设备接受数据就绪
LIO_AD12	UART2_DSR	I	设备初始化完成
LIO_AD13	UART2_DCD	I	外部 MODEM 探测到载波信号

表 2- 6 LIO 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
LIO_A0	NODE_GPIO21	IO	通用输入输出 21
LIO_A1	NODE_GPIO22	IO	通用输入输出 22
LIO_A2	NODE_GPIO23	IO	通用输入输出 23
LIO_A3	NODE_GPIO24	IO	通用输入输出 24
LIO_A4	NODE_GPIO25	IO	通用输入输出 25
LIO_A5	NODE_GPIO26	IO	通用输入输出 26
LIO_A6	NODE_GPIO27	IO	通用输入输出 27
LIO_AD00	NODE_GPIO03	IO	通用输入输出 03
LIO_AD01	NODE_GPIO04	IO	通用输入输出 04
LIO_AD02	NODE_GPIO05	IO	通用输入输出 05
LIO_AD03	NODE_GPIO06	IO	通用输入输出 06
LIO_AD04	NODE_GPIO07	IO	通用输入输出 07
LIO_AD05	NODE_GPIO08	IO	通用输入输出 08
LIO_AD06	NODE_GPIO09	IO	通用输入输出 09
LIO_AD07	NODE_GPIO10	IO	通用输入输出 10
LIO_AD08	NODE_GPIO11	IO	通用输入输出 11
LIO_AD09	NODE_GPIO12	IO	通用输入输出 12
LIO_AD10	NODE_GPIO18	IO	通用输入输出 18
LIO_AD11	NODE_GPIO13	IO	通用输入输出 13
LIO_AD12	NODE_GPIO14	IO	通用输入输出 14
LIO_AD13	NODE_GPIO15	IO	通用输入输出 15

LIO_AD14	NODE_GPIO19	IO	通用输入输出 19
LIO_AD15	NODE_GPIO20	IO	通用输入输出 20
LIO_ADLOCK	NODE_GPIO02	IO	通用输入输出 02
LIO_CSN	NODE_GPIO28	IO	通用输入输出 28
LIO_DEN	NODE_GPIO00	IO	通用输入输出 00
LIO_DIR	NODE_GPIO01	IO	通用输入输出 01
LIO_RDY	NODE_GPIO29	IO	通用输入输出 29
LIO_WRN	NODE_GPIO16	IO	通用输入输出 16
LIO_RDN	NODE_GPIO17	IO	通用输入输出 17

2.5 GMAC 接口

表 2- 7GMAC 接口信号

信号名称	类型	描述
GMAC[1:0]_TXCK	O	RGMII 发送时钟
GMAC[1:0]_TCTL	O	RGMII 发送控制
GMAC[1:0]_TXD[3:0]	O	RGMII 发送数据
GMAC[1:0]_RXCK	I	RGMII 接收时钟
GMAC[1:0]_RCTL	I	RGMII 接收控制
GMAC[1:0]_RXD[3:0]	I	RGMII 接收数据
GMAC[1:0]_MDCK	O	SMA 接口时钟
GMAC[1:0]_MDIO	OD	SMA 接口数据

GMAC1 接口与 GPIO 有复用关系，如下表所示。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 8 GMAC1 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
GMAC1_TXCK	-	-	-
GMAC1_TCTL	GPIO13	I/O	通用输入输出 13
GMAC1_TXD[3:0]	GPIO[12:9]	I/O	通用输入输出 12-9
GMAC1_RXCK	-	-	-
GMAC1_RCTL	GPIO8	I/O	通用输入输出 8
GMAC1_RXD[3:0]	GPIO[7:4]	I/O	通用输入输出 7-4
GMAC1_MDCK	-	-	-
GMAC1_MDIO	-	-	-

2.6 SATA 接口

表 2- 9 SATA 接口信号

信号名称	类型	描述
SATA_REFCLKIN_P SATA_REFCLKIN_N	I	差分 25MHz 参考时钟输入(内部有备份时钟)
SATA_TXP SATA_TXN	DIFF OUT	SATA 差分数据输出

SATA_RXP SATA_RXN	DIFF IN	SATA 差分数据输入
SATA_LEDN	OD	SATA 工作状态，低表示有数据传输

SATA 接口的 SATA_LEDN 与 GPIO 有复用关系，如下表所示。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 10 SATA 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
SATA_LEDN	GPIO14	I/O	通用输入输出 14

2.7 USB 接口

表 2- 11 USB 接口信号

信号名称	类型	描述
USB2_REFRES	A	参考电阻
USB2_DP[4:0]	I/O	USB D+
USB2_DM[4:0]	I/O	USB D-
USB2_OC[3:1]	I	USB 过流检测输入，需注意该信号为高有效
USB2_OC0	O	OTG DRVVBUS 输出
USB2_ID	I	OTG ID 输入
USB2_VBUS	A	OTG VBUS 输入

2.8 SPI 接口

表 2- 12 SPI 接口信号

信号名称	类型	描述
SPI[3:0]_SCK	O	SPI 时钟输出
SPI[3:0]_CSn0	O	SPI 片选 0
SPI[3:0]_CSn1	O	SPI 片选 1
SPIO/3_CSn2/WPN	O	SPI 片选 2/写保护输出
SPIO/3_CSn3/HOLDN	O	SPI 片选 3/地址保持输入输出
SPI[3:0]_SDO	O	SPI 数据输出
SPI[3:0]_SDI	I	SPI 数据输入

2.9 I2C 接口

表 2- 13 I2C 接口信号

信号名称	类型	描述
I2C0_SCL	O	I2C0 时钟
I2C0_SDA	OD	I2C0 数据
I2C1_SCL	O	I2C1 时钟
I2C1_SDA	OD	I2C1 数据
I2C2_SCL	O	I2C2 时钟
I2C2_SDA	OD	I2C2 数据

I2C3_SCL	O	I2C3 时钟
I2C3_SDA	OD	I2C3 数据

I2C 与 GPIO 有复用，复用关系见下表。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 14 I2C 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
I2C0_SCL	GPIO16	I/O	通用输入输出 16
I2C0_SDA	GPIO17	I/O	通用输入输出 17
I2C1_SCL	GPIO18	I/O	通用输入输出 18
I2C1_SDA	GPIO19	I/O	通用输入输出 19
I2C2_SCL	GPIO27	I/O	通用输入输出 27
I2C2_SDA	GPIO28	I/O	通用输入输出 28
I2C3_SCL	GPIO29	I/O	通用输入输出 29
I2C3_SDA	GPIO30	I/O	通用输入输出 30

2.10 UART 接口

表 2- 15 UART 接口信

信号名称	类型	描述
UART[2:0]_TXD	O	串口数据输出
UART[2:0]_RXD	I	串口数据输入
UART[2:0]_RTS	O	串口数据传输请求
UART[2:0]_DTR	O	串口初始化完成
UART[2:0]_RI	I	外部 MODEM 探测到振铃信号
UART[2:0]_CTS	I	设备接受数据就绪
UART[2:0]_DSR	I	设备初始化完成
UART[2:0]_DCD	I	外部 MODEM 探测到载波信号

2K1500 有 3 个独立的全功能串口，UART0 可以选择 NODE 或者南桥， UART1 和 UART2 有独立引脚，也可以通过软件调整为与 LIO 复用引脚。串口通过设置可以工作在 2x4 和 4x2 模式，各种模式的管脚对应关系如下。其它引脚复用的 UART 接口的内部复用关系也如下表所示。

表 2- 16 UART 接口复用关系

1x8	2x4	4x2
TXD0(O)	TXD0(O)	TXD0(O)
RTS0(O)	RTS0(O)	TXD5(O)
DTR0(O)	TXD3(O)	TXD3(O)
RXD0(I)	RXD0(I)	RXD0(I)
CTS0(I)	CTS0(I)	RXD5(I)
DSR0(I)	RXD3(I)	RXD3(I)
DCD0(I)	CTS3(I)	RXD4(I)

RI0(I)	RTS3(O)	TXD4(O)

2.11 CAN 接口

表 2- 17 CAN 接口信号

信号名称	类型	描述
CAN0_RX	I	CAN 通道 0 数据接收
CAN0_TX	O	CAN 通道 0 数据发送
CAN1_RX	I	CAN 通道 1 数据接收
CAN1_TX	O	CAN 通道 1 数据发送
CAN2_RX	I	CAN 通道 2 数据接收
CAN2_TX	O	CAN 通道 2 数据发送
CAN3_RX	I	CAN 通道 3 数据接收
CAN3_TX	O	CAN 通道 3 数据发送
CAN4_RX	I	CAN 通道 4 数据接收
CAN4_TX	O	CAN 通道 4 数据发送
CAN5_RX	I	CAN 通道 5 数据接收
CAN5_TX	O	CAN 通道 5 数据发送

CAN 接口与 GPIO 有复用，如下表所示。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 18 CAN 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
CAN0_RX	GPIO32	I/O	通用输入输出 32
CAN0_TX	GPIO33	I/O	通用输入输出 33
CAN1_RX	GPIO34	I/O	通用输入输出 34
CAN1_TX	GPIO35	I/O	通用输入输出 35

2.12 NAND 接口

表 2- 19 NAND 接口信号

信号名称	类型	描述
NAND_CEn[3:0]	O	NAND 片选 3-0
NAND_CLE	O	NAND 命令锁存
NAND_ALE	I	NAND 地址锁存
NAND_WRn	O	NAND 写信号
NAND_RDn	I	NAND 读信号
NAND_RDYn[3:0]	I	NAND 准备好输入 3-0
NAND_D[7:0]	I/O	NAND 命令/地址/数据线

NAND 与 GPIO 和 eMMC 有复用，复用关系见下表。复用配置请参考表 4- 14 通用配置寄存器 0。

表 2- 20 NAND 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述

NAND_CEn[3:0]	GPIO[47:44]	I/O	通用输入输出 47-44
NAND_CLE	GPIO48	I/O	通用输入输出 48
NAND_ALE	GPIO49	I/O	通用输入输出 49
NAND_WRn	GPIO50	I/O	通用输入输出 50
NAND_RDn	GPIO51	I/O	通用输入输出 51
NAND_RDYn[3:0]	GPIO[55:52]	I/O	通用输入输出 55-52
NAND_D[7:0]	GPIO[63:56]	I/O	通用输入输出 63-56

表 2- 21 NAND 与 eMMC 复用关系

信号名称	复用名称	复用类型	复用信号描述
NAND_RDYn1	EMMC_CMD	I/O	双向命令信号
NAND_RDYn2	EMMC_CLK	O	时钟信号
NAND_RDYn3	EMMC_DS	I	数据选通信号
NAND_D[7:0]	EMMC_D[7:0]	I/O	双向数据信号

2.13 SDIO 接口

表 2- 22 SDIO 接口信号

信号名称	类型	描述
SDIO_CLK	O	SDIO 时钟输出
SDIO_CMD	I/O	SDIO 命令输入输出
SDIO_DATA[3:0]	I/O	SDIO 数据信号

SDIO 与 GPIO 有复用，复用关系见下表。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 23 SDIO 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
SDIO_CLK	GPIO41	I/O	通用输入输出 41
SDIO_CMD	GPIO40	I/O	通用输入输出 40
SDIO_DATA[3:0]	GPIO[39:36]	I/O	通用输入输出 39-36

2.14 GPIO

下表仅列出专用的 12 个 GPIO 引脚信号，其他 GPIO 为复用信号，可参考其他信号定义。默认情况下所有与 GPIO 复用的引脚为 GPIO 功能（GMAC1 除外），且都为输入状态。

表 2- 24 GPIO 信号

信号名称	类型	描述
GPIO00	I/O	通用输入输出
GPIO01	I/O	通用输入输出
GPIO02	I/O	通用输入输出
GPIO03	I/O	通用输入输出
GPIO15	I/O	通用输入输出
GPIO24	I/O	通用输入输出
GPIO25	I/O	通用输入输出

GPIO26	I/O	通用输入输出
GPIO31	I/O	通用输入输出
GPIO42	I/O	通用输入输出
NODE_GPIO30	I/O	通用输入输出
NODE_GPIO31	I/O	通用输入输出

2.15 PWM

表 2- 25 PWM 信号

信号名称	类型	描述
PWM[5:0]	O	PWM 输出

PWM 与 GPIO 有复用，复用关系如下。复用配置请参考表 4- 16 引脚复用配置寄存器。

表 2- 26 PWM 与 GPIO 复用关系

信号名称	复用名称	复用类型	复用信号描述
PWM[3:0]	GPIO[23:20]	I/O	通用输入输出 23-20

2.16 PLL 电源接口

表 2- 27 PLL 电源接口

信号名称	类型	描述
PLL_NODE_VDD	P	NODE PLL 电源
PLL_SATA_VDD	P	SATA PLL 电源
PLL_DDR_VDD	P	DDR PLL 电源
PLL_NODE_VSS	P	NODE PLL 地
PLL_SATA_VSS	P	SATA PLL 地
PLL_DDR_VSS	P	DDR PLL 地

2.17 测试接口

表 2- 28 测试接口

信号名称	类型	描述
ACPI_DOTESTn	I	测试模式控制(ACPI 电压域) 0：测试模式 1：功能模式

2.18 JTAG 接口

表 2- 29 JTAG 接口

信号名称	类型	描述
JTAG_SEL	I	JTAG 选择(1：JTAG；0：CPU_JTAG)
JTAG_TCK	I	JTAG 时钟
JTAG_TDI	I	JTAG 数据输入
JTAG_TMS	I	JTAG 模式
JTAG_TRST	I	JTAG 复位

JTAG_TDO	O	JTAG 数据输出

2.19 时钟信号

表 2- 30 时钟信号

信号名称	类型	描述
SYS_SYSCLK	I	100MHz 参考时钟
SYS_TESTCLK	I	测试时钟输入，功能模式下板级必须将该引脚下拉至 GND

2.20 RTC 相关信号

表 2- 31 时钟信号

信号名称	类型	描述
RTC_XI	I/O	32.768KHz 晶体输入接口
RTC_XO	I/O	32.768KHz 晶体输出接口

2.21 系统相关信号

表 2- 32 系统相关信号

信号名称	类型	描述
SYS_CLKMODE	I	PLL 时钟输入选择 0：SYS_SYSCLK 1：PCIE0_REFCLKIN_N/P
CHIP_CONFIG[1:0]	I	PLL 时钟配置输入 00=低频模式 01=高频模式 10=软件模式(DFT) 11=bypass 模式
CHIP_CONFIG[3:2]	I	启动选择输入 00=LIO 01=SPI(DFT) 10=SDIO 11=NAND
CHIP_CONFIG[5:4]	I	NAND 类型选择 00=512Mb(page 512B) 01=1Gb(page 2KB) 10=16Gb(page 4KB) 11=128Gb(page 8KB)
CHIP_CONFIG6	I	PHY 内部参考时钟选择输入 0：SYS_SYSCLK 1：PCIE0_REFCLKIN_N/P
CHIP_CONFIG7	I	必须为 0
CHIP_CONFIG8	I	PCIe1 模式选择输入

		0：EP mode 1：RC mode
CHIP_CONFIG9	I	NAND ECC 功能使能输入， 1=enable 0=disable(DFT)

2.22 外设功能复用表

模块层次的功能复用关系如下表所示：

表 2- 33 外设功能复用表

功能 0	功能 1	功能 2	功能 3	功能 4	功能 5
DDR3					
PCIex4		4*PCIex1			
PCIex4	GPIO(1)	2*PCIex1			
SATA	GPIO(1)				
USB					
GMAC0					
GMAC1	GPIO(10)				
Local Bus	GPIO(30)		UART1-2		
NAND	GPIO(20)	EMMC			
SPI0-3					
RTC					
I2C0-3	GPIO(8)				
CAN0	GPIO(2)				
CAN1	GPIO(2)				
CAN2-5					
			UART0(8)	UART0(4)	UART0(2)
					UART5(2)
				UART3(4)	UART3(2)
					UART4(2)
			UART1(8)	UART1(4)	UART1(2)
					UART8(2)
				UART6(4)	UART6(2)
					UART7(2)
			UART2(8)	UART2(4)	UART2(2)
					UART11(2)
				UART9(4)	UART9(2)
					UART10(2)
JTAG（LA264）		JTAG	JTAG（LA132）		
	GPIO(12)				
PWM0-1	GPIO(2)				

PWM2-3	GPIO(2)				
PWM4-5					
SDIO	GPIO(6)				

3 时钟结构

3.1 芯片时钟结构

龙芯 2K1500 由一个 100MHz 单端时钟或者一个 100MHz 差分时钟作为参考时钟，内部共有 3 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 3 个 PLL 的用途分别为：

- ✓ 一个 PLL 用于产生 node 和 eMMC 时钟，node 时钟经过各自分频供 CPU 核、二级 Cache、一二级交叉开关、IO 子网络、加解密模块以及 LA132 使用；
- ✓ 一个 PLL 产生 GMAC 控制器、SATA 以及 USB 的时钟；
- ✓ 一个 PLL 产生 DDR 时钟；

除了内部的 PLL 之外，对于 SATA、PCIe、USB 这几个采用 PHY 自己产生时钟的模块，也使用外部输入的参考时钟进行了参考时钟通路设计，该参考时钟可以通过 CHIP_CONFIG6 在系统差分和单端参考时钟间进行选择。

还有一个 MISC 时钟，直接使用 100MHz 参考时钟通过分频得到。

芯片时钟结构图如图 3- 1 所示，

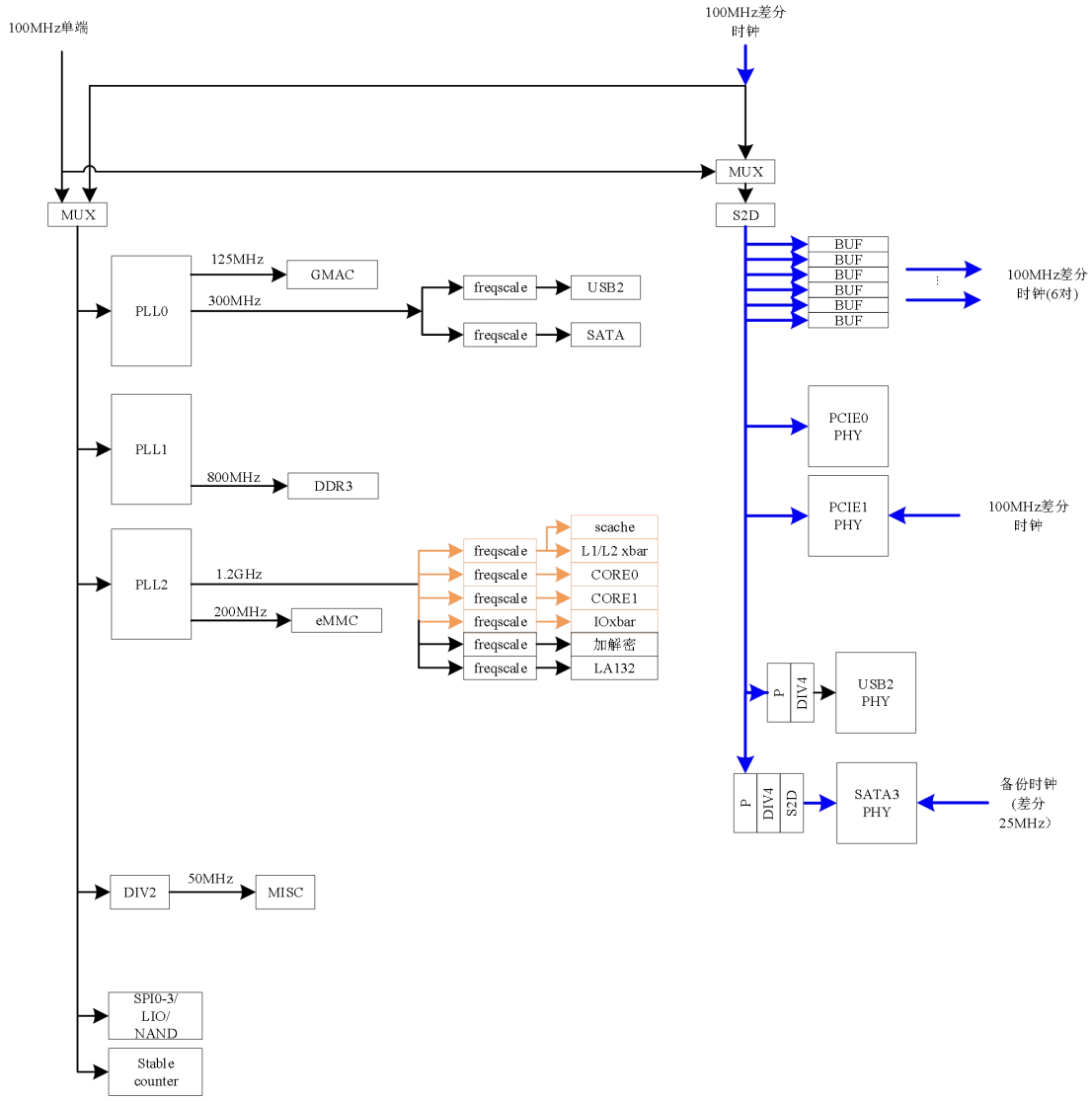


图 3- 1 芯片时钟结构图

3.2 系统参考时钟

芯片的系统参考时钟有两种选择方式，一种是选择单端输入时钟 SYS_SYSCLK，另一种是选择差分输入时钟 PCIe0_REFCLKIN_P/N，选择信号为 SYS_CLKMODE。无论哪种方式，必须保证系统参考时钟的频率为 100MHz。

3.3 内部网络时钟

芯片内部包含了 3 个主要 PLL，这 3 个 PLL 以系统参考时钟作为输入，用于产生芯片内部网络需要的各个时钟。每个 PLL 最多可以提供 3 个时钟输出。

这 3 个 PLL 的用途分别为：

一个 PLL 用于产生 node 和 eMMC 时钟，node 时钟经过各自分频供 CPU 核、二级 Cache、

一二级交叉开关、IO 子网络、加解密模块以及 LA132 使用；

一个 PLL 产生 GMAC 控制器、SATA 以及 USB 的时钟；

一个 PLL 产生 DDR 时钟；

PLL 的结构如图 3- 2 所示，

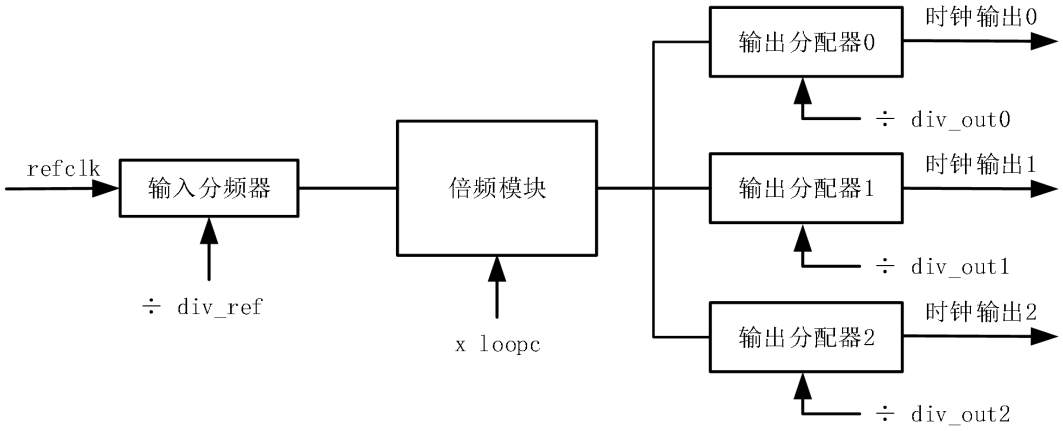


图 3- 2PLL 结构图

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 20 ~ 40MHz 的范围内，倍频模块倍频后的频率（refclk / div_ref * loopc）在 1.2GHz ~ 3.2GHz 的范围内。

PLL相关的配置及说明见表 3- 1。这些配置通过 PLL0-2 配置寄存器(参见 4.21-4.23 章节)进行控制。

表 3- 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[1:0]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。

根据小数环路分频使能的设置,环路分频系数计算公式有所不同。

当小数环路分频未使能时,环路分频系数为：

DIV_LOOPC = 4*loopc

当小数环路分频使能时,环路分频系数为：

DIV_LOOPC = 4*(loopc+frac_input/(2^16)+!(dither_disable[1])

第 n 路输出时钟频率计算公式如下:

f_outn = fref/ref_div*DIV_LOOPC/(2*div_outn)

在参数配置时要求 VCO 的振荡频率(fref/ref_div*DIV_LOOPC)在 1.6-3.2GHz 之间.

3.4 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。

3.5 PCIe PHY 参考时钟

2K1500 的 PCIe 有 2 个 PHY，它们共用内部参考时钟。PCIe1 PHY 的参考时钟可以从下面两个时钟源进行选择：

1. 外部 100MHz 差分输入(PCIe1_REFCLKIN_P/N)
2. 内部系统参考时钟

3.6 USB PHY 参考时钟

USB PHY 的参考时钟只能选择内部系统参考时钟经过 4 分频后得到的 25M 单端时钟。

3.7 SATA PHY 参考时钟

SATA PHY 的参考时钟可以从下面两个时钟源进行选择：

1. 外部 25MHz 差分输入(SATA_REFCLKIN_P/N)
2. 内部系统参考时钟经过 4 分频后得到的 25M 差分时钟

4 芯片配置寄存器

龙芯 2K1500 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 4.13 节配置寄存器的基地址为 0x1fe00000 或 0x3ff00000，4.14 至 4.39 节配置寄存器的基地址为 0x5ff00000。

下面详细介绍这些配置寄存器。

4.1 版本寄存器

版本寄存器。

地址偏移：0000h

默认值：012h

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：37fh

表 4- 2 芯片特性寄存器

位域	字段名	访问	描述
0	Centigrade	R	为 1 时，表示 CSR[0x428]有效
1	Node counter	R	为 1 时，表示 CSR[0x408]有效
2	MSI	R	为 1 时，表示 MSI 可用
3	EXT_IOI	R	为 1 时，表示 EXT_IOI 可用
4	IPI_percore	R	为 1 时，表示通过 CSR 私有地址进行 IPI 发送
5	Freq_percore	R	为 1 时，表示通过 CSR 私有地址调整频率
6	Freq_scale	R	为 1 时，表示动态分频功能可用
7	DVFS_v1	R	为 1 时，表示动态调频 v1 可用
8	Tsensor	R	为 1 时，表示温度传感器可用
9	中断译码	R	中断引脚译码模式可用

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_3531_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“2K1500”

4.5 功能设置寄存器

地址偏移：0180h

默认值：3D00_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_defult_confspace	RW	将所有内存访问路由至配置空间
6		RW	保留
7	MC0_resetn	RW	MC0 软件复位（低有效）
8	MC0_clken	RW	是否使能 MC0
43:9	-	RW	保留
63:56	Cpu_version	R	CPU 版本

4.6 温度采样寄存器

地址偏移：0198h

默认值：0000_0000_0000_0000h

表 4- 6 温度采样寄存器

位域	字段名	访问	描述
15:0		R	保留
19:16		R	保留
20	dotest	R	Dotest 引脚状态
21		R	
31:22		R	保留
45:32	thsens_data	R	传感器的原始数据

46	thsens_overflow	R	传感器监测值溢出标志
47	reserved	R	保留
48	thsens_outmode	R	传感器配置的监测模式 0：温度模式；1：电压模式
51:49	reserved	R	保留
54:52	thsens_outcluster	R	传感器配置的监测点
55	reserved	R	保留
63:56	temperature	R	摄氏温度值

4.7 处理器核分频设置寄存器

以下寄存器用于处理器核动态分频使用，使用该寄存器对处理器核进行调频设置，可以在 100ns 内完成变频操作，没有其它额外开销。

地址偏移：01d0h

默认值：FFFF_FFFF_FFFF_FFFFh

表 4- 7 处理器核软件分频设置寄存器

位域	字段名	访问	描述
2:0	core0_freqctrl	RW	核 0 分频控制值
3	core0_en	RW	核 0 时钟使能
6:4	core1_freqctrl	RW	核 1 分频控制值
7	core1_en	RW	核 1 时钟使能
			软件分频后的时钟频率值等于原来的（分频控制值+1）/8

4.8 处理器核复位控制寄存器

以下寄存器用于处理器核软件控制复位使用。需要复位时，先将对应核的 resetn 置 0，再将 resetn_pre 置 0，等待 500 微秒后，将 resetn_pre 置 1，再将 resetn 置 1 即可完成整个复位过程。

地址偏移：01d8h

默认值：FFFF_FFFF_FFFF_FFFFh

表 4- 8 处理器核软件分频设置寄存器

位域	字段名	访问	描述
0	Core0_resetn_pre	RW	核 0 复位辅助控制
1	Core0_resetn	RW	核 0 复位
2	Core1_resetn_pre	RW	核 1 复位辅助控制
3	Core1_resetn	RW	核 1 复位

4.9 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：0044_BF7C_C000_00F0h

表 4- 9 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
9	disable_0x3ff0	RW	禁止通过基地址 0x3ff0_0000 对配置寄存器空间的路由
33:32	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
38:34	LIO rom_count_init_i	RW	ROM 数据读取延迟（boot 时钟周期数）
39	LIO rom_width16	RW	Rom 数据位宽： 0：8 位 1：16 位
44:40	LIO io_count_init_i	RW	IO 数据读取延迟（boot 时钟周期数）
45	LIO io_width_16	RW	IO 数据位宽： 0：8 位 1：16 位
46	LIO iopf_en	RW	保留
47	LIO big_mem	RW	LIO big_mem 模式控制
51:48	LIO adlock_cfg	RW	ADLOCK 时间软件配置
60:52	Reserved	R/W	保留
61	enable_gather	RW	boot 读响应聚合使能

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_0000_0000_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_cpujtag	RW	完全禁用 CPU_JTAG 调试接口
2	disable_LA132	RW	完全禁用 LA132
3	disable_jtagl32	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			保留
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			
21	stable_resetn	RW	稳定时钟复位控制
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			
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	freqscale_mode_node	RW	结点的调频模式选择 0：(n+1)/8 1：1/(n+1)
37	freqscale_mode_LA132	RW	LA132 的调频模式选择 0：(n+1)/8 1：1/(n+1)
39:38			
40	freqscale_mode_stable	RW	Stable clock 的调频模式选择 0：(n+1)/8 1：1/(n+1)
43:41			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	BRIDGE_INT_en	RW	南北桥 IO 中断使能
49	INT_encode	RW	使能中断引脚编码模式
53:52			
54			
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 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- 11 FUSE0 观测寄存器

位域	字段名	访问	描述
127:0	Fuse_0	RW	

4.12 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- 12 FUSE1 观测寄存器

位域	字段名	访问	描述
127:0	Fuse_1	RW	

4.13 通用配置寄存器 0

通用配置寄存器 0，对北桥设备进行配置。

地址偏移：0420h

默认值：0400_0000_CCCC_3CE0h

表 4- 13 通用配置寄存器 0

位域	名称	访问	描述
63:60	Reserved	R/W	保留
57:50	Reserved	R/W	保留
49	pcie_g0_p1_clk_ok	RO	pcie_g0 端口 1 时钟 ready 0：没有时钟 1：时钟正常
48	pcie_g0_p0_clk_ok	RO	pcie_g0 端口 0 时钟 ready 0：没有时钟 1：时钟正常
47:44	Reserved	R/W	保留
43	pcie_f0_p3_clk_ok	RO	pcie_f0 端口 3 时钟 ready 0：没有时钟 1：时钟正常
42	pcie_f0_p2_clk_ok	RO	pcie_f0 端口 2 时钟 ready 0：没有时钟 1：时钟正常
41	pcie_f0_p1_clk_ok	RO	pcie_f0 端口 1 时钟 ready 0：没有时钟 1：时钟正常
40	pcie_f0_p0_clk_ok	RO	pcie_f0 端口 0 时钟 ready 0：没有时钟 1：时钟正常
39:37	Reserved	R/W	保留

36	pcie_g0_uca_en	R/W	pcie_g0 uncache 访问加速使能 0：关闭访问加速 1：打开访问加速
35	Reserved	R/W	保留
33	pcie_f0_uca_en	R/W	pcie_f0 uncache 访问加速使能 0：关闭访问加速 1：打开访问加速
32:28	Reserved	R/W	保留
27	pcie_g0_p1_clken	R/W	使能 pcie_g0 的 port1 时钟 0：关闭时钟 1：打开时钟
26	pcie_g0_p0_clken	R/W	使能 pcie_g0 的 port0 时钟 0：关闭时钟 1：打开时钟
25	pcie_g0_enable	R/W	使能 pcie_g0 控制器 0：禁止访问 1：允许访问
24	pcie_g0_soft_reset	R/W	pcie_g0 的软件复位 0：解除复位 1：保持复位
23:20	Reserved	R/W	保留
15	pcie_g0_refclk_sel	R/W	PCIe G0 参考时钟源选择： 0：选择内部参考时钟； 1：选择从 PAD 输入参考时钟； 如果 fix_fc_mode 为 1,那么该信号恒为 1。
13	pcie_f0_p3_clken	R/W	使能 pcie_f0 的 port3 时钟 0：关闭时钟 1：打开时钟
12	pcie_f0_p2_clken	R/W	使能 pcie_f0 的 port2 时钟 0：关闭时钟 1：打开时钟
11	pcie_f0_p1_clken	R/W	使能 pcie_f0 的 port1 时钟 0：关闭时钟 1：打开时钟
10	pcie_f0_p0_clken	R/W	使能 pcie_f0 的 port0 时钟 0：关闭时钟 1：打开时钟
9	pcie_f0_enable	R/W	使能 pcie_f0 控制器 0：禁止访问 1：允许访问
8	pcie_f0_soft_reset	R/W	pcie_f0 的软件复位 0：解除复位 1：保持复位
7	Reserved	R/W	保留
6	Reserved	R/W	保留
5	Reserved	R/W	保留
4	Reserved	R/W	保留
3	Reserved	R/W	保留
2	pcie_f0_refclk_sel	R/W	PCIe F0 的参考时钟选择，必须配置为 0
1	Reserved	R/W	保留
0	default_route_cfg0	R/W	使用固定地址访问 PCIe 设备。 0：使用 PCI 配置头对设备地址进行配置 1：使用固定地址访问设备 如果 fix_default_route 为 1,那么该信号恒为 1。

			该位必须设置为 0。

4.14 通用配置寄存器 1

本寄存器包含 USB、SATA、GMAC、SPI 相关的配置信息。

地址偏移：0430h

默认值：1000_0900_0009_99F2h

表 4- 14 通用配置寄存器 1

位域	名称	访问	描述
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
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
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	spi1_soft_reset	R/W	SPI1 控制器软件复位 0：解除复位 1：保持复位
54	spi3_soft_reset	R/W	SPI3 控制器软件复位 0：解除复位 1：保持复位
53	Reserved	R/W	保留
52	otp_access	R/W	芯片内 antifuse 读写控制 0：禁止访问 antifuse 1：允许访问 antifuse
51:48	encrypt_soft_reset	R/W	ENCRYPT 控制器软件复位 0：解除复位 1：保持复位
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
43	sata0_clken	R/W	SATA0 时钟使能 0：没有时钟 1：时钟正常
42	sata0_en	R/W	SATA0 访问使能 0：禁止访问 1：允许访问
41	Reserved	R/W	保留
40	sata0_cntl_soft_reset	R/W	SATA0 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
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
35:34	Reserved	R/W	保留
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	spi3_enable	R/W	使能 SPI3 接口 0：禁止访问 1：允许访问
31	spi3_uca_en	R/W	SPI3 uncache 加速使能 0：关闭访问加速 1：打开访问加速
30	spi1_uca_en	R/W	SPI1 uncache 加速使能 0：关闭访问加速 1：打开访问加速
29	conf_uca_en	R/W	配置寄存器 uncache 加速使能 0：关闭访问加速 1：打开访问加速
28	misc_uca_en	R/W	低速 misc 设备 uncache 加速使能 0：关闭访问加速 1：打开访问加速
26	gmac_uca_en	R/W	gmac uncache 加速使能

			0：关闭访问加速 1：打开访问加速
25	sata_uca_en	R/W	sata uncache 加速使能 0：关闭访问加速 1：打开访问加速
24	usb_uca_en	R/W	usb uncache 加速使能 0：关闭访问加速 1：打开访问加速
23	gmac2_clken	R/W	gmac2 时钟使能 0：没有时钟 1：时钟正常
22	gmac2_sdb_flowctrl	R/W	gmac2 流控使能 0：流控关闭 1：流控打开
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	otg_clken	R/W	otg 时钟使能 0：没有时钟 1：时钟正常
14	otg_en	R/W	otg 访问使能 0：禁止访问 1：允许访问
13	Reserved	R/W	保留
12	otg_cntl_soft_reset	R/W	otg 控制器软件复位 0：解除复位 1：保持复位
11	usb2_clken	R/W	usb2 时钟使能 0：没有时钟 1：时钟正常
10	usb2_en	R/W	usb2 访问使能 0：禁止访问 1：允许访问
9	usb2_phy_soft_re	R/W	usb2 PHY 软件复位

	set		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
3:1	Reserved	R/W	保留
0	default_route_cfgl	R/W	使用固定地址访问 USB、SATA、GMAC 等设备。 0：使用 PCI 配置头对设备地址进行配置 1：使用固定地址访问设备 如果 fix_default_route 为 1,那么该信号恒为 1。 该位必须设置为 0。

4.15 引脚复用配置寄存器

地址偏移：0440h

默认值：0000_0000_FFFF_FFFFh

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- 15 引脚复用配置寄存器

位域	名称	访问	描述
58	uart8_enable	R/W	UART8 控制器使能，引脚 UART1_RTS/CTS 对应的两线串口
57	uart7_enable	R/W	UART7 控制器使能，引脚引脚 UART1_RI/DCD 对应的两线串口
56	uart6_enable	R/W	UART6 控制器使能，引脚 UART1_DTR/DSR（RI/DCD）对应的两线（四线）串口
55	Reserved	R/W	保留
54	uart5_enable	R/W	UART5 控制器使能，引脚 UART0_RTS/CTS 对应的两线串口
53	uart4_enable	R/W	UART4 控制器使能，引脚引脚 UART0_RI/DCD 对应的两线串口
52	uart3_enable	R/W	UART3 控制器使能，引脚 UART0_DTR/DSR（RI/DCD）对应的两线（四线）串口
51	Reserved	R/W	保留

50	uart0_sel	R/W	UART0 控制器选择 0：选择 node 的 uart0 1：选择 io 的 uart0
33	can1_sel	R/W	CAN1 引脚的工作模式选择 0：工作在 GPIO 模式 1：工作在 CAN 模式
32	can0_sel	R/W	CAN0 引脚的工作模式选择 0：工作在 GPIO 模式 1：工作在 CAN 模式
31	uart11_enable	R/W	UART11 控制器使能，引脚 UART2_RTS/CTS 对应的两线串口
30	uart10_enable	R/W	UART10 控制器使能，引脚引脚 UART2_RI/DCD 对应的两线串口
29	uart9_enable	R/W	UART9 控制器使能，引脚 UART2_DTR/DSR（RI/DCD）对应的两线（四线）串口
28	Reserved	R/W	保留
27:26	nand_sel	R/W	NAND 引脚的工作模式选择 00：工作在 GPIO 模式 10：工作在 NAND 模式 11：工作在 eMMC 模式
25	sdio_sel	R/W	SDIO 引脚的工作模式选择 0：工作在 GPIO 模式 1：工作在 SDIO 模式
21:20	lio_sel	R/W	LIO 引脚的工作模式选择 00/11：工作在 GPIO 模式 01：工作在 LIO 模式 10：工作在 UART 模式
14	pcie1_rstn_sel	R/W	引脚 PCIe1_RSTN 的工作模式选择 0：工作 GPIO 模式 1：工作 PCIe1_RSTN 模式
13	sata_ledn_sel	R/W	引脚 SATA_LEDn 的工作模式选择 0：工作在 GPIO 模式 1：工作在 SATA 模式
10	gmac1_sel	R/W	GMAC1 引脚的工作模式选择 0：工作在 GPIO 模式 1：工作在 GMAC 模式
9	i2c3_sel	R/W	引脚 I2C3_SCL/SDA 的工作模式选择 0：工作在 GPIO 模式 1：工作在 I2C 模式
8	i2c2_sel	R/W	引脚 I2C2_SCL/SDA 的工作模式选择 0：工作在 GPIO 模式 1：工作在 I2C 模式
7	i2c1_sel	R/W	引脚 I2C1_SCL/SDA 的工作模式选择 0：工作在 GPIO 模式 1：工作在 I2C 模式
6	i2c0_sel	R/W	引脚 I2C0_SCL/SDA 的工作模式选择 0：工作在 GPIO 模式 1：工作在 I2C 模式
3	pwm3_sel	R/W	引脚 PWM3 的工作模式选择 0：工作在 GPIO 模式 1：工作在 PWM 模式
2	pwm2_sel	R/W	引脚 PWM2 的工作模式选择 0：工作在 GPIO 模式 1：工作在 PWM 模式
1	pwm1_sel	R/W	引脚 PWM1 的工作模式选择 0：工作在 GPIO 模式 1：工作在 PWM 模式
0	pwm0_sel	R/W	引脚 PWM0 的工作模式选择

			0：工作在 GPIO 模式 1：工作在 PWM 模式

4.16 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 3 种配置， 01 代表选择 USB_OC1 信号，10 代表选择 USB_OC2 信号，11 代表选择 USB_OC3 信号，00 禁止使用（USB_OC0 作为 OTG DRVVBUS 输出）。每组对应一个 USB 端口。

地址偏移：0448h

默认值：0000_0000h

表 4- 16 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

4.17 OTG 预取配置

本寄存器包含 OTG 内部预取模块的相关配置信息。

地址偏移：0450h

默认值：C708_8834h

表 4- 17 OTG 预取配置

位域	名称	访问	描述
31	stop_cpu_rd_for_dma_wr	R/W	当存在未响应的 DMA 写请求时，使能对处理器读访问的阻塞
30	invld_pref_on_cpu_wr	R/W	当处理器写控制器内部寄存器时，无效掉预取的读数据
29	stop_cpu_wr_for_dma_wr	R/W	当存在未响应的 DMA 写请求时，使能对处理器写访问的阻塞（当 bit31 有效时才生效）
26:24	pref_cond	R/W	使能对 4/16/32 字节访问的预取 bit26：4 字节访问 bit25：16 字节访问 bit24：32 字节访问
21:16		R/W	当处理器需要读取控制器内部寄存器时，会阻止控制器进行 DMA 写操作。此参数配置的是允许处理器访问被阻塞的周期数（该值乘以 4），当阻塞时间到达后，停止后续的 DMA 写操作。
15:12	pref_limit	R/W	预取请求地址边界配置。地址边界等于 4K*2^N。
11:8	pref_max_num	R/W	预取的最大个数
7:4	pref_num_on_static	R/W	静态预取策略下，一次预取的个数

3	pref_static_en	R/W	使能静态预取策略
2	invld_pref_on_dma_wr	R/W	写请求无效所有的读预取数据
1	pref_disable	R/W	关闭预取
0	rd_wait_wr	R/W	读请求必须等待所有的写请求完成（写响应到达），即使读写地址不冲突。当读写地址冲突时，读请求会被强制等待写请求完成。

4.18 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 MISC 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- 18 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

4.19 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- 19 DMA 一致性配置寄存器

位域	名称	访问	描述
31:30	encrypt_cc	R/W	加解密模块一致性控制
29:28	emmc_cc	R/W	EMMC 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
5：4	pcie_g0_cc	R/W	PCIe F1 两个端口一致性控制
3：0	pcie_f0_cc	R/W	PCIe F0 四个端口一致性控制

4.20 PLL0 配置寄存器

该寄存器用来设置 PLL0，其中输出时钟 0 用来控制 USB2/SATA3 的控制器时钟，输出时钟 1 用来产生 GMAC 需要的 125MHz 时钟。

地址偏移：0x0480

默认值：0000_0000h

表 4- 20 PLL0 配置寄存器

位域	名称	访问	描述
63:46	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

4.21 PLL1 配置寄存器

该寄存器用来设置 PLL1，其中输出时钟 1 用来产生 DDR 需要的时钟。

地址偏移：0x0490

默认值：0000_0000h

表 4- 21 PLL1 配置寄存器

位域	名称	访问	描述
63:52	Reserved	R/W	保留
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

4.22 PLL2 配置寄存器

该寄存器用来设置 PLL2，其中输出时钟 1 用来作为内部网络的时钟，输出时钟 2 用来作为 eMMC 的时钟。

地址偏移：0x04a0

默认值：0000_0000h

表 4- 22 PLL2 配置寄存器

位域	名称	访问	描述
63:46	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

4.23 FREQSCALE 配置寄存器

该寄存器用于配置时钟的分频系数以及 CPU 核的时钟使能。

地址偏移：0x04d0

默认值：FFFF_FF99_FFFF_FFB7h

表 4- 23 FRESCALE 配置寄存器

位域	名称	访问	描述
63:44	Reserved	R/W	保留
43	encrypt_clk_en	R/W	加解密模块时钟使能
42:40	encrypt_freqscale	R/W	加解密模块时钟分频数
35	io_clk_en	R/W	IO 桥网络时钟使能
34:32	io_network_freqscale	R/W	IO桥网络时钟分频数
25:23	boot_freqscale	R/W	boot时钟分频数
22:20	apb_freqscale	R/W	apb时钟分频数
18:15	Reserved	R/W	保留
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_frescale	R/W	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.

4.24 PCIe_F0 配置寄存器 0

本组寄存器包含对 PCIe_F0 的控制信号和状态采集信息。

地址偏移：0x0580

默认值：0000_0191h

表 4- 24 PCIe_F0 配置寄存器 0

位域	名称	访问	描述
31:18	Reserved	R/W	保留
17	bar0_acc_sel	R/W	0：在 EP 模式下 PCIe 主机从控制器内部对 Bar0 寄存器进行访问； 1：在 EP 模式下 PCIe 主机通过 2K1500 的内部网络访问控制器的 Bar0 寄存器；
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
15:13	Reserved	R/W	保留
12	ref_master	R/W	0：与 PRG 共用参考电阻；1：使用 PHY 的 PAD 上的参考电阻
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

4.25 PCIe_F0 配置寄存器 1

本组寄存器包含对 PCIe_F0 的控制信号和状态采集信息。

地址偏移：0x0588

默认值：0006_0000h

表 4- 25 PCIe_F0 配置寄存器 1

位域	名称	访问	描述
63:36	Reserved	R/W	保留
35	phy_ready_p3	RO	此位为 1 表示 PHY 的 lane3 准备好
34	phy_ready_p2	RO	此位为 1 表示 PHY 的 lane2 准备好
33	phy_ready_p1	RO	此位为 1 表示 PHY 的 lane1 准备好
32	phy_ready_p0	RO	此位为 1 表示 PHY 的 lane0 准备好
31	pipe_rst_p3	R/W	向此位写入 1 将置 port3 的 PIPE 接口复位信号有效，再写入 0 则结束 port3 的 PIPE 接口的复位
30	pipe_rst_p2	R/W	向此位写入 1 将置 port2 的 PIPE 接口复位信号有效，再写入 0 则结束 port2 的 PIPE 接口的复位
29	pipe_rst_p1	R/W	向此位写入 1 将置 port1 的 PIPE 接口复位信号有效，再写入 0 则结束 port1 的 PIPE 接口的复位
28	pipe_rst_p0	R/W	向此位写入 1 将置 port0 的 PIPE 接口复位信号有效，再写入 0 则结束 port0 的 PIPE 接口的复位
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	Reserved	R/W	保留
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	ep_phy_cfg_soft	RW	0：EP 模式时不需要对 PHY 进行配置 1：EP 模式时需要对 PHY 进行配置
22	warm_wait_bypass	R/W	0：热复位时需要等待 PHY 的 PLL 工作正常 1：热复位时不等待 PHY 的 PLL 的状态
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
19:15	Reserved	R/W	保留
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

4.26 PCIe_F0 PHY 配置控制寄存器

本组寄存器用来控制产生 PCIe_F0 PHY 内部控制寄存器的配置访问操作。

地址偏移：0x0590

默认值：0000_0000h

表 4- 26 PCIe_F0 PHY 配置寄存器

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态
55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

PCIe/SATA/USB3 PHY 以及 PRG 的内部寄存器均通过以上软件接口（接口定义相同，但是每个 PHY 和 PRG 对应的访问寄存器的地址不同）进行读写配置。具体的配置流程如下：

对于写操作

1. 将 phy_cfg_write 置为 1,要写的数据写入 phy_cfg_data 寄存器,要写的地址写入 phy_cfg_adr 寄存器
2. 将 phy_cfg_trigger 先置 1 再置 0
3. 等待 phy_cfg_done 变为 1

对于读操作

1. 将 phy_cfg_write 置为 0,要读的地址写入 phy_cfg_adr 寄存器
2. 将 phy_cfg_trigger 先置 1 再置 0
3. 等待 phy_cfg_done 变为 1
4. phy_cfg_data 寄存器中即为从 PHY 中读出的数据

需注意，在配置之前先将 apb_clken 置为 1，且 apb_resetn 置为 0,同时需保证在 APB 配置过程中这两位不变。配置完成后可将 apb_clken 置为 0。

4.27 PCIe_G0 配置寄存器 0

本组寄存器包含对 PCIe_G0 的控制信号和状态采集信息。

地址偏移：0x05e0

默认值：0000_0191h

表 4- 27 PCIe_G0 配置寄存器 0

位域	名称	访问	描述
31:18	Reserved	R/W	保留
17	bar0_acc_sel	RW	0：在 EP 模式下 PCIe 主机从控制器内部对 Bar0 寄存器进行访问； 1：在 EP 模式下 PCIe 主机通过 2K1500 的内部网络访问控制器的 Bar0 寄存器；
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
15:13	Reserved	R/W	保留
12	ref_master	R/W	0：与 PRG 共用参考电阻；1：使用 PHY 的 PAD 上的参考电阻
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

4.28 PCIe_G0 配置寄存器 1

本组寄存器包含对 PCIe_G0 的控制信号和状态采集信息。

地址偏移：0x05e8

默认值：0006_0000h

表 4- 28 PCIe_G0 配置寄存器 1

位域	名称	访问	描述
63:36	Reserved	R/W	保留
35	phy_ready_p3	RO	此位为 1 表示 PHY 的 lane3 准备好
34	phy_ready_p2	RO	此位为 1 表示 PHY 的 lane2 准备好
33	phy_ready_p1	RO	此位为 1 表示 PHY 的 lane1 准备好
32	phy_ready_p0	RO	此位为 1 表示 PHY 的 lane0 准备好
31	pipe_rst_p3	R/W	向此位写入 1 将置 port3 的 PIPE 接口复位信号有效，再写入 0 则结束 port3 的 PIPE 接口的复位
30	pipe_rst_p2	R/W	向此位写入 1 将置 port2 的 PIPE 接口复位信号有效，再写入 0 则结束 port2 的 PIPE 接口的复位
29	pipe_rst_p1	R/W	向此位写入 1 将置 port1 的 PIPE 接口复位信号有效，再写入 0 则结束 port1 的 PIPE 接口的复位
28	pipe_rst_p0	R/W	向此位写入 1 将置 port0 的 PIPE 接口复位信号有效，再写入 0 则结束 port0 的 PIPE 接口的复位
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	Reserved	R/W	保留
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	ep_phy_cfg_soft	R/W	0：EP 模式时不需要对 PHY 进行配置 1：EP 模式时需要对 PHY 进行配置
22	warm_wait_bypass	R/W	0：热复位时需要等待 PHY 的 PLL 工作正常 1：热复位时不等待 PHY 的 PLL 的状态
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
19:13	Reserved	R/W	保留
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	保留
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

4.29 PCIe_G0 PHY 配置控制寄存器

本组寄存器用来控制产生 PCIe_G0 PHY 内部控制寄存器的配置访问操作。

地址偏移：0x05f0

默认值：0000_0000h

表 4- 29 PCIe_G0 PHY 配置寄存器

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态
55:52	Reserved	R/W	保留
51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

4.30 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- 30 PCIe_F0 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 PCIe_F0 的 port3-0 13:12 对应 PCIe_G0 的 port1-0 其他位，保留

4.31 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_0000_3c8d_d001h

表 4- 31 PAD 驱动配置寄存器

位域	名称	访问	描述
63:48	pad3v3_ctrl	R/W	3.3V pad 驱动能力控制 0:COMPEN 1:COMPTQ 2:FASTFRZ 3:FREEZE 7:4 RASRCN 11:8 RASRCP 12:SLEEP
47:32	pad1v8_ctrl	R/W	1.8V pad 驱动能力控制 0:COMPEN 1:COMPTQ 2:FASTFRZ 3:FREEZE 7:4 RASRCN 11:8 RASRCP 12:SLEEP
31	Reserved	R/W	保留
30	pad_ctrl_emmc	R/W	eMMC PAD 驱动能力控制
29:28	pad_ctrl_gmac	R/W	GMAC PAD 驱动能力控制
25:24	pad_ctrl_miphy	R/W	MIPHY PAD 驱动能力控制
23:22	pad_ctrl_pcie	R/W	PCIe PAD 驱动能力控制
21:20	pad_ctrl_sata	R/W	SATA PAD 驱动能力控制
19:18	pad_ctrl_nand	R/W	NAND PAD 驱动能力控制
17:16	pad_ctrl_lio	R/W	LIO PAD 驱动能力控制
15:14	pad_ctrl_can	R/W	CAN PAD 驱动能力控制
13:12	pad_ctrl_spi	R/W	SPI PAD 驱动能力控制
11:10	pad_ctrl_jtag	R/W	JTAG PAD 驱动能力控制
9:8	pad_ctrl_uart	R/W	UART PAD 驱动能力控制
7:6	pad_ctrl_pwm	R/W	PWM PAD 驱动能力控制

5:4	pad_ctrl_i2c	R/W	I2C PAD 驱动能力控制
3:2	pad_ctrl_gpio	R/W	GPIO PAD 驱动能力控制
1:0	pad_ctrl_sdio	R/W	SDIO PAD 驱动能力控制

4.32 Compensation 状态寄存器

本寄存器用来读取补偿状态。

地址偏移：0x06e8

默认值：0000_0000h

表 4- 32 PAD 驱动配置寄存器

位域	名称	访问	描述
53:48	Compensation 模块状态指示	RO	53:compok_3v3_left 52:compok_3v3_top 51:compok_3v3_right 50:compok_3v3_rsm 49:compok_3v3_rsmacpi 48:compok_1v8
47:0	Compensation 模块锁定码	RO	[47:44]:nasrcp_3v3_left [43:40]:nasrcn_3v3_left [39:36]:nasrcp_3v3_top [35:32]:nasrcn_3v3_top [31:28]:nasrcp_3v3_right [27:24]:nasrcn_3v3_right [23:20]:nasrcp_3v3_rsm [19:16]:nasrcn_3v3_rsm [15:12]:nasrcp_3v3_rsmacpi [11:08]:nasrcn_3v3_rsmacpi [07:04]:nasrcp_1v8 , [03:00]:nasrcn_1v8

4.33 SATA PHY 配置寄存器

本寄存器用来配置 SATA PHY 的一些控制参数。

地址偏移：0x0740

默认值：0000_0000_0000_f412h

表 4- 33 SATA PHY 配置寄存器

位域	名称	访问	描述
63:62	phy_osc_mode	R/W	PHY OSC 模式
61	phy_hfc_ready	RO	-
60	phy_pl_osc_rdy	RO	-
59:40	phy_sata_bert_eb_fill	RO	-

39:36	phy_sigdet	RO	-
35:32	phy_syncchardet	RO	-
31:28	phy_rx_symbol_lock	RO	-
27:24	phy_rx_bit_lock	RO	-
23:20	phy_ready	RO	-
19:16	phy_cfg_soft_phy_rstn	R/W	PHY 软复位控制，每一位对应一个 PORT
15:12	phy_cfg_sel_soft_phy_rstn	R/W	PHY 软复位控制选择，每一位对应一个 PORT 0：由硬件控制 PHY 的复位信号； 1：由寄存器 phy_cfg_soft_phy_rstn 控制 PHY 的复位信号
11	phy_pl_osc_force_on	R/W	强制晶振启动信号
10	phy_pl_osc_bypass	R/W	OSC bypass
9:8	phy_osc_ref_div	R/W	OSC 参考时钟分频控制
7	phy_pl_rst_osc_n	R/W	OSC 复位信号
6	clksel	R/W	必须设置为 1
5	phy_pl_ssc_en	R/W	SSC 使能
4	phy_ref_diff	R/W	差分参考时钟选择，必须设置为 1
3	phy_ sel _sigdet_sync_logic	R/W	-
2	phy_ bypass_rx_data_gating_n	R/W	-
1	phy_soft_rst_n_sata	R/W	SATA 控制器软件复位控制，低有效
0	phy_cfg_reset_n	R/W	SATA PHY 全局复位控制，低有效

4.34 SATA PHY 配置访问寄存器

本组寄存器用来控制产生 SATA PHY 内部控制寄存器的配置访问操作。

地址偏移：0x0748

默认值：0000_0000h

表 4- 34 SATA PHY 配置访问寄存器

位域	名称	访问	描述
63:61	Reserved	R/W	保留
60	phy_cfg_done	RO	此位为 1 表示对的 PHY 一次配置访问完成。 写完成表示写的数据已经写入 PHY 内部寄存器，读完成表示读的数据已经返回到 phy_cfg_data 寄存器
59	phy_cfg_trigger	R/W	向此位先写入 1 再写入 0，启动一次对 PHY 的配置访问
58	phy_cfg_write	R/W	0：对 PHY 进行的是配置读访问 1：对 PHY 进行的是配置写访问
57	phy_cfg_clken	R/W	0：关闭 PHY 的配置访问端口时钟 1：开启 PHY 的配置访问端口时钟
56	phy_cfg_resetn	R/W	0：PHY 的配置访问端口处在复位状态 1：PHY 的配置访问端口退出复位状态
55:52	Reserved	R/W	保留

51:32	phy_cfg_addr	R/W	PHY 进行配置访问的地址
31:0	phy_cfg_data	R/W	PHY 配置读写数据。在写操作时，将数据先写入该寄存，然后再执行写操作；在读操作时，从 PHY 返回的读数据存储到该寄存器。

4.35 SATA PHY PLL 配置寄存器

本寄存器用来对 SATA PHY 内部 PLL 进行配置。

地址偏移：0x0750

默认值：00f0_0000h

表 4- 35 SATA PHY PLL 配置寄存器

位域	名称	访问	描述
23:0	pl_pll_ratio	R/W	SATA PHY PLL 配置

4.36 USB2.0 PHY 配置寄存器 0

地址偏移：0868h

默认值：0000_0000_0000_0000h

表 4- 36 USB2.0 PHY 配置寄存器 0

位域	名称	访问	描述
1:0	mode_p0	R/W	Code 校正模式选择
6:2	code_ext_p0	R/W	使用外部校正码，当 calib_bypass 置位时
7	calib_bypass_p0	R/W	绕过校正码，保持偏压活动状态，高有效
8	bgbypass_p0	R/W	绕过内部 Bandgap 输出而改用 BGEXT 引脚
9	tck_cal_p0	R/W	配置为 0
10	tq_cal_p0	R/W	IDDQ 模式使能，高有效
12:11	calib_tuning_p0	R/W	调整内部的 bandgap（具体功能见下图）
17:16	mode_p1	R/W	Code 校正模式选择
22:18	code_ext_p1	R/W	使用外部校正码，当 calib_bypass 置位时
23	calib_bypass_p1	R/W	绕过校正码，保持偏压活动状态，高有效
24	bgbypass_p1	R/W	绕过内部 Bandgap 输出而改用 BGEXT 引脚
25	tck_cal_p1	R/W	配置为 0
26	tq_cal_p1	R/W	IDDQ 模式使能，高有效
28:27	calib_tuning_p1	R/W	调整内部的 bandgap
33:32	mode_p2	R/W	Code 校正模式选择
38:34	code_ext_p2	R/W	使用外部校正码，当 calib_bypass 置位时
39	calib_bypass_p2	R/W	绕过校正码，保持偏压活动状态，高有效
40	bgbypass_p2	R/W	绕过内部 Bandgap 输出而改用 BGEXT 引脚
41	tck_cal_p2	R/W	配置为 0
42	tq_cal_p2	R/W	IDDQ 模式使能，高有效
44:43	calib_tuning_p2	R/W	调整内部的 bandgap

49:48	mode_p3	R/W	Code 校正模式选择
54:50	code_ext_p3	R/W	使用外部校正码，当 calib_bypass 置位时
55	calib_bypass_p3	R/W	绕过校正码，保持偏压活动状态，高有效
56	bgbypass_p3	R/W	绕过内部 Bandgap 输出而改用 BGEXT 引脚
57	tck_cal_p3	R/W	配置为 0
58	tq_cal_p3	R/W	IDDQ 模式使能，高有效
60:59	calib_tuning_p3	R/W	调整内部的 bandgap

注：以上后缀 p0~p3 代表四个 USB2.0 PHY 的编号

4.37 USB2.0 PHY 配置寄存器 1-5

以上表（0870h 偏移的寄存器）为基准，以下为功能重复定义的配置表，所以统一以变量形式自行对照相应的寄存器域，N 代表第 N 个端口，本芯片 USB2.0 端口有 5 个（含 1 个 OTG），编号从 0 开始，所以 N 的取值从 0~4。

地址偏移：0870h +（N*8）h

默认值：0000_0000_0000_0000h

表 4- 37 USB2.0 PHY 配置寄存器 1-5

位域	名称	访问	描述
63:60	usb2_debug_ob_N	RO	Debug 观测引脚引脚
48:44	usb2_debug_sel_N	R/W	Debug 观测引脚选择
43	Dppulldown_N	R/W	Dp 下拉电阻使能，低有效
42	Dmpulldown_N	R/W	Dm 下拉电阻使能，低有效
41	Det5v_N	RO	当 dp 或 dm 短接 5V 的 VBUS 时指示 1:短接 0:没短接
40	Dpdn_short_lfs_N	RO	当 dp 或 dm 互相短接，或者 5V 或者地时指示 1:短接 0:没短接
39	Codeminus_N	R/W	减少内部阻抗校正码 00010
38	Codeplus_N	R/W	增加内部阻抗校正码 00010
37:36	Zhsdrv_N	R/W	
35:32	Ihstx_N	R/W	HS 驱动电流调整引脚 0000-defaultVOH 0001- defaultVOH+5mv 0010- defaultVOH+2*5mv
31:0	Afetrim_N	R/W	

4.38 USB2.0 配置寄存器

地址偏移：08b8h

默认值：0000_0000_0000_0000h

表 4- 38 USB2.0 配置寄存器

位域	名称	访问	描述
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

5 地址空间分配

龙芯 2K1500 内部地址位宽为 40 位，路由主要通过系统的两级交叉开关以及 IO 子网络组成。交叉开关可以对每个 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，通过对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。

龙芯 2K1500 的地址空间与 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 2K1500 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。

全芯片的地址空间划分如下表所示：

表 5- 1 芯片地址空间划分

NAME	BASE	MASK	SIZE	备注
Memory	64’h0000_0000	64’hFFFF_FFFF_F000_0000	256M	内存空间
Memory mapped IO	64’h1000_0000	64’hFFFF_FFFF_F800_0000	128M	用于映射内部所使用的设备空间(BAR)
PCIe IO	64’h1800_0000	64’hFFFF_FFFF_FE00_0000	32M	用于映射 PCIe 控制器对外的 IO 空间
Type0	64’h1A00_0000	64’hFFFF_FFFF_FF00_0000	16M	Type0 配置空间
Type1	64’h1B00_0000	64’hFFFF_FFFF_FF00_0000	16M	Type1 配置空间
BOOT	64’h1C00_0000	64’hFFFF_FFFF_FF00_0000	16M	启动空间，可映射至 SPI 和 LIO 上
LIO Memory	64’h1D00_0000	64’hFFFF_FFFF_FF00_0000	16M	
Flash	64’h1FC0_0000	64’hFFFF_FFFF_FFF0_0000	1M	可映射至 SPI、NAND、SDIO 和 LIO 上
CONFIG	64’h1FE0_0000	64’hFFFF_FFFF_FFF0_0000	1M	芯片配置寄存器空间
SPI	64’h1FFF_0000	64’hFFFF_FFFF_FFFF_0000	64K	SPI 配置空间
Memory mapped IO	64’h4000_0000	64’hFFFF_FFFF_C000_0000	1G	PCIe MEM 空间
Memory	64’h8000_0000	64’hFFFF_FFFF_8000_0000	2G	内存空间
Memory	64’h1_0000_0000	64’hFFFF_FFFF_0000_0000	4G	内存空间
Memory	64’h2_0000_0000	64’hFFFF_FFFE_0000_0000	8G	内存空间
Memory mapped IO	64’h40_0000_0000	64’hFFFF_FFC0_0000_0000	256G	PCIe MEM 空间
Type0	64’hFE_0000_0000	64’hFFFF_FFFF_FF00_0000	256M	Type0 配置空间
Type1	64’hFE_1000_0000	64’hFFFF_FFFF_FF00_0000	256M	Type1 配置空间
Type0	64’hFE_2000_0000	64’hFFFF_FFFF_FF00_0000	256M	Type0 配置空间
Type1	64’hFE_3000_0000	64’hFFFF_FFFF_FF00_0000	256M	Type1 配置空间

5.1 一级交叉开关

一级交叉开关一共有 3 个主端口，分别连接 CORE0、CORE1 和 IO 桥。

一级开关的路由可以软件配置，也可以采用硬件路由，默认的地址空间分配如下表所示。其中包含了 32 位和 64 位地址空间访问模式。

表 5- 2 一级交叉开关路由规则

地址空间	大小	设备/路由目的空间	备注
0x1000_0000 - 0x17FF_FFFF	128M	I/O 设备的配置寄存器空间	需注意该段地址空间为 MEM 类型。各个 IO 设备的 BAR 地址在该地址空间。
0x1800_0000 - 0x19FF_FFFF	32M	PCIe 的 I/O 空间	32 位模式
0x1A00_0000 - 0x1BFF_FFFF	32M	各个 I/O 设备的配置头空间	32 位模式
0x1C00_0000 - 0x1CFF_FFFF	16M	启动空间	BOOT
0x1FC0_0000 - 0x1FCF_FFFF	1M	启动空间	BOOT
0x1FE0_0000 - 0x1FE0_FFFF	64K	Confbus	芯片配置空间
0x4000_0000 - 0x7FFF_FFFF	1G	32 位访问模式下 I/O 设备的 MEM 空间	32 位模式
0xFE_0000_0000 - 0xFE_FFFF_FFFF	4G	各个 I/O 设备的配置头空间	64 位模式
0xFD_FC00_0000 - 0xFD_FFFF_FFFF	64M	PCIe 的 I/O 空间	64 位模式
0xF0_0000_0000 - 0xFF_FFFF_FFFF	64G	I/O 设备的配置空间	64 位模式
其他地址或上述空间若为 CACHED		Bit6 为 0，路由到 scache0; Bit6 为 1，路由到 scache1	由地址的第 6 位决定路由目的

5.2 二级交叉开关

龙芯 2K1500 的二级交叉开关连接两个二级 Cache、内存控制器、启动模块（SPI 或者 LIO）、confbus 模块、UART 和 I2C。

地址窗口是供 SCACHE0、SCACHE1 两个具有主功能的 IP 进行路由选择和地址转换而设置的，路由可以软件配置，也可以采用硬件路由，默认的地址空间分配如下表所示。

表 5- 3 二级交叉开关路由规则

地址空间	大小	设备/路由目的空间	备注
0x1C00_0000 - 0x1CFF_FFFF	16M	启动空间	BOOT
0x1FC0_0000 - 0x1FCF_FFFF	1M	启动空间	BOOT
0x1FE0_01F0 - 0x1FE0_01FF	16B	SPI 配置空间	
0x1D00_0000 - 0x1DFF_FFFF	16M	LIO 空间	
0x1FE0_0000 - 0x1FE0_FFFF	64K	Confbus	芯片配置空间
0x3FF0_0000 - 0x3FF0_FFFF	64K	Confbus	芯片配置空间

0x1FE0_01E0 - 0x1FE0_01EF	16B	APB	
0x1FE0_0100 - 0x1FE0_017F	128B	APB	

5.3 交叉开关软件路由配置

龙芯 2K1500 交叉开关的路由可以通过软件实现。软件可以对每个 Master 端口接收到的请求进行路由配置，每个 Master 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 K 字节对齐；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 Slave 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。

表 5- 4 MMAP 字段对应的该空间访问属性

[7]	[6]	[5]	[4]
窗口使能	允许对 SCACHE/内存进行交错访问	允许块读	允许取指

窗口命中公式: (IN_ADDR & MASK) == BASE

由于龙芯 2K1500 缺省采用固定路由，在上电启动时，配置窗口都处于关闭状态，使用时需要系统软件对其进行使能配置。

地址窗口转换寄存器如下表所示。

表 5- 5 地址窗口寄存器表

地址	寄存器	地址	寄存器
0x3ff0_2000	CORE0_WIN0_BASE	0x3ff0_2100	CORE1_WIN0_BASE
0x3ff0_2008	CORE0_WIN1_BASE	0x3ff0_2108	CORE1_WIN1_BASE
0x3ff0_2010	CORE0_WIN2_BASE	0x3ff0_2110	CORE1_WIN2_BASE
0x3ff0_2018	CORE0_WIN3_BASE	0x3ff0_2118	CORE1_WIN3_BASE
0x3ff0_2020	CORE0_WIN4_BASE	0x3ff0_2120	CORE1_WIN4_BASE
0x3ff0_2028	CORE0_WIN5_BASE	0x3ff0_2128	CORE1_WIN5_BASE
0x3ff0_2030	CORE0_WIN6_BASE	0x3ff0_2130	CORE1_WIN6_BASE
0x3ff0_2038	CORE0_WIN7_BASE	0x3ff0_2138	CORE1_WIN7_BASE
0x3ff0_2040	CORE0_WIN0_MASK	0x3ff0_2140	CORE1_WIN0_MASK
0x3ff0_2048	CORE0_WIN1_MASK	0x3ff0_2148	CORE1_WIN1_MASK
0x3ff0_2050	CORE0_WIN2_MASK	0x3ff0_2150	CORE1_WIN2_MASK
0x3ff0_2058	CORE0_WIN3_MASK	0x3ff0_2158	CORE1_WIN3_MASK
0x3ff0_2060	CORE0_WIN4_MASK	0x3ff0_2160	CORE1_WIN4_MASK
0x3ff0_2068	CORE0_WIN5_MASK	0x3ff0_2168	CORE1_WIN5_MASK
0x3ff0_2070	CORE0_WIN6_MASK	0x3ff0_2170	CORE1_WIN6_MASK
0x3ff0_2078	CORE0_WIN7_MASK	0x3ff0_2178	CORE1_WIN7_MASK
0x3ff0_2080	CORE0_WIN0_MMAP	0x3ff0_2180	CORE1_WIN0_MMAP
0x3ff0_2088	CORE0_WIN1_MMAP	0x3ff0_2188	CORE1_WIN1_MMAP

0x3ff0_2090	CORE0_WIN2_MMAP	0x3ff0_2190	CORE1_WIN2_MMAP
0x3ff0_2098	CORE0_WIN3_MMAP	0x3ff0_2198	CORE1_WIN3_MMAP
0x3ff0_20a0	CORE0_WIN4_MMAP	0x3ff0_21a0	CORE1_WIN4_MMAP
0x3ff0_20a8	CORE0_WIN5_MMAP	0x3ff0_21a8	CORE1_WIN5_MMAP
0x3ff0_20b0	CORE0_WIN6_MMAP	0x3ff0_21b0	CORE1_WIN6_MMAP
0x3ff0_20b8	CORE0_WIN7_MMAP	0x3ff0_21b8	CORE1_WIN7_MMAP
0x3ff0_2200	IO_L1X_WIN0_BASE		
0x3ff0_2208	IO_L1X_WIN1_BASE		
0x3ff0_2210	IO_L1X_WIN2_BASE		
0x3ff0_2218	IO_L1X_WIN3_BASE		
0x3ff0_2220	IO_L1X_WIN4_BASE		
0x3ff0_2228	IO_L1X_WIN5_BASE		
0x3ff0_2230	IO_L1X_WIN6_BASE		
0x3ff0_2238	IO_L1X_WIN7_BASE		
0x3ff0_2240	IO_L1X_WIN0_MASK		
0x3ff0_2248	IO_L1X_WIN1_MASK		
0x3ff0_2250	IO_L1X_WIN2_MASK		
0x3ff0_2258	IO_L1X_WIN3_MASK		
0x3ff0_2260	IO_L1X_WIN4_MASK		
0x3ff0_2268	IO_L1X_WIN5_MASK		
0x3ff0_2270	IO_L1X_WIN6_MASK		
0x3ff0_2278	IO_L1X_WIN7_MASK		
0x3ff0_2280	IO_L1X_WIN0_MMAP		
0x3ff0_2288	IO_L1X_WIN1_MMAP		
0x3ff0_2290	IO_L1X_WIN2_MMAP		
0x3ff0_2298	IO_L1X_WIN3_MMAP		
0x3ff0_22a0	IO_L1X_WIN4_MMAP		
0x3ff0_22a8	IO_L1X_WIN5_MMAP		
0x3ff0_22b0	IO_L1X_WIN6_MMAP		
0x3ff0_22b8	IO_L1X_WIN7_MMAP		
0x3ff0_2400	SCACHE0_WIN0_BASE	0x3ff0_2500	SCACHE1_WIN0_BASE
0x3ff0_2408	SCACHE0_WIN1_BASE	0x3ff0_2508	SCACHE1_WIN1_BASE
0x3ff0_2410	SCACHE0_WIN2_BASE	0x3ff0_2510	SCACHE1_WIN2_BASE
0x3ff0_2418	SCACHE0_WIN3_BASE	0x3ff0_2518	SCACHE1_WIN3_BASE
0x3ff0_2420	SCACHE0_WIN4_BASE	0x3ff0_2520	SCACHE1_WIN4_BASE
0x3ff0_2428	SCACHE0_WIN5_BASE	0x3ff0_2528	SCACHE1_WIN5_BASE
0x3ff0_2430	SCACHE0_WIN6_BASE	0x3ff0_2530	SCACHE1_WIN6_BASE
0x3ff0_2438	SCACHE0_WIN7_BASE	0x3ff0_2538	SCACHE1_WIN7_BASE
0x3ff0_2440	SCACHE0_WIN0_MASK	0x3ff0_2540	SCACHE1_WIN0_MASK
0x3ff0_2448	SCACHE0_WIN1_MASK	0x3ff0_2548	SCACHE1_WIN1_MASK

0x3ff0_2450	SCACHE0_WIN2_MASK	0x3ff0_2550	SCACHE1_WIN2_MASK
0x3ff0_2458	SCACHE0_WIN3_MASK	0x3ff0_2558	SCACHE1_WIN3_MASK
0x3ff0_2460	SCACHE0_WIN4_MASK	0x3ff0_2560	SCACHE1_WIN4_MASK
0x3ff0_2468	SCACHE0_WIN5_MASK	0x3ff0_2568	SCACHE1_WIN5_MASK
0x3ff0_2470	SCACHE0_WIN6_MASK	0x3ff0_2570	SCACHE1_WIN6_MASK
0x3ff0_2478	SCACHE0_WIN7_MASK	0x3ff0_2578	SCACHE1_WIN7_MASK
0x3ff0_2480	SCACHE0_WIN0_MMAP	0x3ff0_2580	SCACHE1_WIN0_MMAP
0x3ff0_2488	SCACHE0_WIN1_MMAP	0x3ff0_2588	SCACHE1_WIN1_MMAP
0x3ff0_2490	SCACHE0_WIN2_MMAP	0x3ff0_2590	SCACHE1_WIN2_MMAP
0x3ff0_2498	SCACHE0_WIN3_MMAP	0x3ff0_2598	SCACHE1_WIN3_MMAP
0x3ff0_24a0	SCACHE0_WIN4_MMAP	0x3ff0_25a0	SCACHE1_WIN4_MMAP
0x3ff0_24a8	SCACHE0_WIN5_MMAP	0x3ff0_25a8	SCACHE1_WIN5_MMAP
0x3ff0_24b0	SCACHE0_WIN6_MMAP	0x3ff0_25b0	SCACHE1_WIN6_MMAP
0x3ff0_24b8	SCACHE0_WIN7_MMAP	0x3ff0_25b8	SCACHE1_WIN7_MMAP

一级 xbar 连接 2 个 Scache 和 IO 桥作为从设备，由 2 个 core 和 IO 桥作为主设备进行窗口映射。二级 xbar 主要连接内存控制器和 MISC 作为从设备，由 2 个 Scache 作为主设备进行窗口映射。

每个地址窗口由 BASE、MASK 和 MMAP 三个 64 位寄存器组成，BASE 以 K 字节对齐，MASK 采用类似网络掩码高位为 1 的格式，MMAP 中包含转换后地址、路由选择及使能控制等位，如下表所示：

表 5- 6MMAP 寄存器位域说明

[63:40]	[39：10]	[7：4]	[3：0]
保留	转换后地址	窗口使能	从设备号

其中，一级 xbar 从设备号对应的设备如下表所示：

表 5- 7 一级 xbar 从设备号与所述模块的对应关系

从设备号	目的设备
0	Scache0
1	Scache1
2	IO 桥

二级 xbar 从设备号对应的设备如下表所示：

表 5- 8 二级 xbar 从设备号与所述模块的对应关系

从设备号	目的设备
0	MC
4	SPI
5	LIO
6	APB
7	CONFIG

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[63:10],10'h0}

根据缺省的寄存器配置，芯片启动后，CPU 的 0x00000000 - 0x0fffffff 的地址区间（256M）映射到 DDR 的 0x00000000 - 0x0fffffff 的地址区间，0x10000000-0x17ffffff 映射到芯片的 PCI_MEM 空间，0x18000000-0x19ffffff 映射到芯片的 PCI_IO 空间，0x1a000000-0x1affffff 映射到芯片的 PCI 配置空间（Type0），0x1b000000-0x1bffffff 映射到芯片的 PCI 配置空间（Type1），0x40000000-0x7fffffff 映射到芯片的 PCI_MEM 空间。软件可以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 8 个地址窗口都不命中时，由内存控制器模块返回数据。

5.4 IO 互连网络

2K1500 中的 IO 互连网络采用南北桥结构。为了便于说明，这里把 IO 设备分成 PCIe 设备（包括 PCIe0 与 PCIe1）和其他 IO 设备（除 PCIe 之外的所有其他设备）两部分。

IO 互连网络主要由以下两部分通路组成：

1. CPU 访问 IO 设备的通路，其地址路由形式主要包括：
   - ●配置访问：CPU 通过配置请求访问 IO 设备的配置头；
   - ●IO 访问：IO 访问用于访问 PCIe 控制器下游设备的 IO 地址空间，根据 PCIe 配置头的 IO Base 和 IO Limit 配置决定地址路由方式；
   - ●内部寄存器访问：通过各个 IO 设备配置头的 BAR 地址访问 IO 设备的内部配置寄存器，MEM 访问类型。
   - ●PCIe 下游设备 MEM 访问：根据 PCIe 配置头的 Memory Base 和 Memory Limit、Prefetchable Memory Base 和 Prefetchable Memory Limit 进行地址路由，用于访问 PCIe 下游设备的 MEM 空间。
2. IO DMA 访问内存的通路，只有 CACHE 访问类型。

5.4.1 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 2K1500 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。

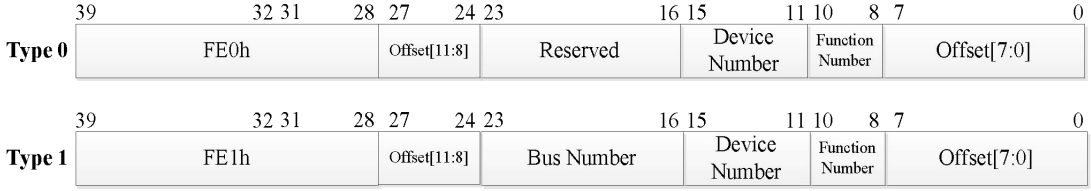


图 5- 164 位配置访问地址格式

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。

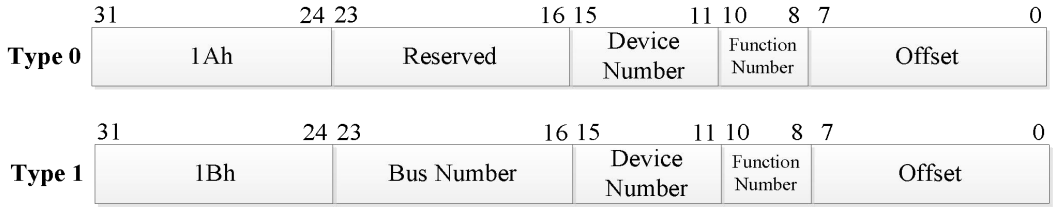


图 5- 232 位配置访问地址格式

5.4.2 PCI 设备和功能

龙芯 2K1500 芯片中，具备配置头空间的设备包括：APB 总线控制器、GMAC0/1 控制器、USB 控制器、SATA 控制器、PCIe0/1 控制器和 DMA 控制器。其中所有 APB 设备都作为一个整体挂在 APB 总线控制器上，由地址的[15:12]位进行译码，下文中将会做详细说明。各个设备的设备号、功能号以及配置头访问首地址如下表所示(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 控制器之外，其他设备的配置头空间大小都只有 256B。

表 5- 9 各个设备的配置头访问对应关系

Device	Device Number, Function Number	Vendor ID	Device ID	Size
APB	2, 0	0x0014	0x7A22	512K
GMAC0	3, 0	0x0014	0x7A03	32K
GMAC1	3, 1	0x0014	0x7A03	32K

USB2-XHCI	4, 0	0x0014	0x7A44	1M
USB2-OTG	5, 0	0X0014	0x7A54	256K
NAND	7, 0	0x0014	0x7A68	512
SPI2	7, 1	0x0014	0x7A2b	4K
SATA0	8, 0	0x0014	0x7A18	1K
PCIe0-Port0	9, 0	0x0014	0x7A79	8K
PCIe0-Port1	10, 0	0x0014	0x7A39	8K
PCIe0-Port2	11, 0	0x0014	0x7A39	8K
PCIe0-Port3	12, 0	0x0014	0x7A39	8K
PCIe1-Port0(RC)	15, 0	0x0014	0x7A79	8K
PCIe1-Port0(EP)	15, 0	0x0014	0x7AF9	4K(BAR0),8K(BAR1),256M(BAR2)
PCIe1-Port1	16, 0	0x0014	0x7A39	8K
CONFBUS	21, 0	0x0014	0x7a1a	64K
SPI3	22, 0	0x0014	0x7a1b	4K(REG) 16M(MEM)
SPI1	23, 0	0x0014	0x7a2b	4K
eMMC	28, 0	0x0014	0x7A88	4K
SD	28, 1	0x0014	0x7A48	4K
AES	29, 0	0x0014	0x7A0e	4K
DES	29, 1	0x0014	0x7A1e	4K
RSA	29, 2	0x0014	0x7A2e	4K
RNG	29, 3	0x0014	0x7A3e	4K
DMA	30, 0	0x0014	0x7A2F	4K

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。

表 5- 10 Type0 类型配置头

31	24	23	16	15	8	7	0	ADDR
Device ID				Vendor ID				00h
Status				Command				04h
Class Code						Revison ID		08h
BIST		Header Type		Lantency Timer		Cache Line Size		0Ch
Base Address 0								10h
Base Address 1								14h
Base Address 2								18h
Base Address 3								1Ch
Base Address 4								20h
Base Address 5								24h
CardBus CIS Pointer								28h
Subsystem ID				Subsystem Vendor ID				2Ch
Expension ROM Base Address								30h

Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

表 5- 11 Type0 的配置头寄存器

寄存器名称	偏移地址	读/写	功能描述	说明（除 PCIe 控制器）
Vendor ID 制造商 ID 号	0x0	RO	制造商 ID 号	缺省值都为 0x0014
Device ID 设备 ID 号	0x2	RO	设备 ID 号	
Command 命令寄存器	0x4	RW	Bit0：I/O 访问使能 Bit1：存储器访问使能 Bit2：主设备使能 Bit3：专用周期监视 Bit4：存储器写与使失效使能 Bit5：VGA 画板侦测 Bit6：奇偶校验错响应 Bit7：等待周期控制 Bit8：SERR#使能 Bit9：快速背靠背使能 其它：保留	扫描配置地址后，需要使能相应设备的存储器访问使能
Status 状态寄存器	0x6	RO/RW	Bit3-0：RO，保留 Bit4：RO，能力列表 Bit5：RO，66MHz 能力 Bit6：保留 Bit7：RO，快速背靠背能力 Bit8：RW，主设备数据奇偶校验错 Bit10-9：RO，设备选择定时 Bit11：RW，发出目标失败信号 Bit12：RW，收到目标失败 Bit13：RW，收到主设备失败 Bit14：RW，发出系统错信号 Bit15： RW，检测到奇偶错	
Revision ID 版本索引	0x8	RO	设备版本号	缺省值都为 0x0
Class Code 类代码	0x9	RO	Bit7-0：Prog. I/F 编程接口 Bit15-8：Subclass code，子类码 Bit23-16：Class code 类别码	
Cache Line Size 缓存行大小	0xC	RO	龙芯 2K1500 的缓存行大小为 64Byte	缺省值都为 0x10
Latency Timer 等待计时器	0xD	RW		保留
Header Type 配置头类型	0xE	RO	除 PCIe 控制器外，都是 Type0。 另外 bit7 表示多功能属性： 1-多功能设备 0-单功能设备	

BIST 内自建测试	0xF	RW	Bit3-0：完成代码，0-成功完成，非 0-设备指定的错误 Bit5-4：保留 Bit6：起动 BIST Bit7：BIST 能力，1-提供 BIST 功能，0-不提供	保留
Base Address Register 0 0 号基址寄存器	0x10	RW/RO	基地址寄存器 0- Bit0：RO，1-I/O 基地址寄存器，0-MEM 基地址寄存器 I/O 基地址寄存器： Bit1：保留 Bit31-2：RO，基地址单元 Bit63-32：保留 MEM 基地址存储器： Bit2-1：RO，MEM 基地址寄存器-译码器宽度单元，00-32 位，10-64 位 Bit3：RO，预提取属性 Bit64-4：基地址单元	只支持 64 位模式
Base Address Register 1 1 号基址寄存器	0x18	RW	基址寄存器 1	只支持 64 位模式
Base Address Register 2 2 号基址寄存器	0x20	RW	基址寄存器 2	只支持 64 位模式
Cardbus CIS Pointer CIS 卡总线指针	0x28	RW	CIS 卡总线指针	保留
Subsystem Vendor ID 子系统版本号	0x2C	RO	子系统供应商 ID 号	保留
Subsystem ID 子系统版本号	0x2D	RO	子系统版本 ID 号	保留
Expansion ROM Base Address 扩展 ROM 基地址	0x30	RW	扩展 ROM 基地址寄存器 Bit0：地址译码使能 Bit10-1：保留 Bit31-11：用于指定 ROM 的起始地址	保留
Capabilities Pointer 扩展能力指针	0x34	RW	扩展指针	保留
Reserved	0x35			保留
Reserved	0x38			保留
Interrupt Line 中断线寄存器	0x3C	RW	中断线寄存器	
Interrupt Pin 中断引脚寄存器	0x3D	RO	中断引脚寄存器	保留（除 PCIe 控制器）
Min Gnt 最小突发期时间	0x3E	RW		保留（除 PCIe 控制器）
Max Lat 获取 PCI 总线最大时间	0x3F	RW		保留（除 PCIe 控制器）

PCIe 控制器的配置头为 Type1 类型，具体如下：

表 5- 12 Type1 类型配置头

31	24	23	16	15	8	7	0	ADDR
Device ID				Vendor ID				00h
Status				Command				04h
Class Code						Revison ID		08h
BIST		Header Type		Lantency Timer		Cache Line Size		0Ch
Base Address 0								10h
Base Address 1								14h
Secondary Lantency Timer		Subordinate Bus #		Secondary Bus #		Primary Bus #		18h
Secondary Status				IO Limit		IO Base		1Ch
(Non-Prefetchable) Memory Limit				(Non-Prefetchable) Memory Base				20h
PrefetchableMemory Limit				PrefetchableMemory Base				24h
Prefetchable Memory Base Upper 32 Bits								28h
Prefetchable Memory Limit Upper 32 Bits								2Ch
IO Limit Upper 16 Bits				IO Base Upper 16 Bits				30h
Reserved						Capabilities Pointer		34h
Expansion ROM Base Address								38h
Bridge Control				Interrupt Pin		Interrupt Line		3Ch

表 5- 13 Type1 的配置头寄存器

寄存器名称	偏移地址	读/写	功能描述	说明（PCIe 控制器）
Vendor ID 制造商 ID 号	0x0	RO	制造商 ID 号	缺省值都为 0x0014
Device ID 设备 ID 号	0x2	RO	设备 ID 号	
Command 命令寄存器	0x4	RW	Bit0：I/O 访问使能 Bit1：存储器访问使能 Bit2：主设备使能 Bit3：专用周期监视 Bit4：存储器写与使失效使能 Bit5：VGA 画板侦测 Bit6：奇偶校验错响应 Bit7：等待周期控制 Bit8：SERR#使能 Bit9：快速背靠背使能 其它：保留	扫描配置地址后，需要使能相应设备的存储器访问使能
Status 状态寄存器	0x6	RO/RW	Bit3-0：RO，保留 Bit4：RO，能力列表 Bit5：RO，66MHz 能力 Bit6：保留 Bit7：RO，快速背靠背能力 Bit8：RW，主设备数据奇偶校验错 Bit10-9：RO，设备选择定时	

			Bit11：RW，发出目标失败信号 Bit12：RW，收到目标失败 Bit13：RW，收到主设备失败 Bit14：RW，发出系统错信号 Bit15： RW，检测到奇偶错	
Revision ID 版本索引	0x8	RO	设备版本号	缺省值都为 0x0
Class Code 类代码	0x9	RO	Bit7-0：Prog. I/F 编程接口 Bit15-8：Subclass code，子类码 Bit23-16：Class code 类别码	缺省值为 0x0b3000
Cache Line Size 缓存行大小	0xC	RO	缓存行大小	缺省值都为 0x00
Latency Timer 等待计时器	0xD	RW		保留
Header Type 配置头类型	0xE	RO	PCIe 控制器都是 Type1	缺省值为 0x01
BIST 内自建测试	0xF	RW	Bit3-0：完成代码，0-成功完成，非 0-设备指定的错误 Bit5-4：保留 Bit6：起动 BIST Bit7：BIST 能力，1-提供 BIST 功能，0-不提供	保留
Base Address Register 0	0x10	RW/RO	基地址寄存器 0- Bit0：RO，1-I/O 基地址寄存器，0-MEM 基地址寄存器 I/O 基地址寄存器： Bit1：保留 Bit31-2：RO，基地址单元 Bit63-32：保留 MEM 基地址存储器： Bit2-1：RO，MEM 基地址寄存器-译码器宽度单元， 00-32 位，10-64 位 Bit3：RO，预提取属性 Bit64-4：基地址单元	64 位 MEM 空间
Primary Bus #	0x18	RW	Primary Bus 编号	默认为 0
Secondary Bus #	0x19	RW	Secondary Bus 编号	默认为 0
Subordinate Bus #	0x1A	RW	Subordinate Bus 编号	默认为 0
IO Base	0x1C	RW/RO	[3:0] RO，0h-16bit,1h-32bit [7:4] RW	默认为 0x1
IO Limit	0X1D	RW/RO	[3:0] RO，0h-16bit,1h-32bit [7:4] RW	默认为 0x1
Secondary Lantency Timer	0x1E	RO	Secondary Bus 状态寄存器	默认为 0
(Non-Prefetchable) Memory Base	0x20	RW/RO	[3:0] RO，must be 0 [15:4] RW	默认为 0x0

(Non-Prefetchable) Memory Limit	0x22	RW/RO	[3:0] RO, must be 0 [15:4] RW	默认为 0x0
Prefetchable Memory Base	0x24	RW/RO	[3:0] RO, 0h-32bit, 1h-64bit [15:4] RW	默认为 0x1
Prefetchable Memory Limit	0x26	RW/RO	[3:0] RO, 0h-32bit, 1h-64bit [15:4] RW	默认为 0x1
Prefetchable Memory Base Upper 32bits	0x28	RW	可预取 MEM 地址空间高 32 位	默认为 0
Prefetchable Memory Limit Upper 32bits	0x2C	RW	可预取 MEM 地址空间高 32 位	默认为 0
IO Base Upper 16 Bits	0x30	RW	IO 地址 Base 高 16 位	默认为 0
IO Limit Upper 16 Bits	0x32	RW	IO 地址 Limit 高 16 位	默认为 0
Capabilities Pointer 扩展能力指针	0x34	RW	扩展指针	默认为 0x40
Expansion ROM Base Address 扩展 ROM 基地址	0x38	RW	扩展 ROM 基地址寄存器 Bit0：地址译码使能 Bit10-1：保留 Bit31-11：用于指定 ROM 的起始地址	保留
Interrupt Line 中断线寄存器	0x3C	RW	中断线寄存器	默认为 0xFF
Interrupt Pin 中断引脚寄存器	0x3D	RO	中断引脚寄存器	默认为 0x0
Bridge Control	0x3E	RW/RO	0(RW): Parity Error Response Enable 1(RW): SERR# Enable 2(RW): ISA Enable 3(RW): VGA Enable 4(RW): VGA 16bit Decode 5(RW): Master-Abort Mode 6(RW) Secondary Bus Reset 7(RW): Fast Back-to-Back Enable 8(RO): Primary Discard Timer 9: Secondary Discard Timer 10(RW1C): Discard Timer Status 11(RW): Discard Timer SERR# Enable 15:12 Reserved	默认为 0

5.4.3 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。内部 PCI 设备包括：PCIe、USB、SATA、GMAC、SPI。这些设备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 方式方位）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、MISC 低速设备块。

表 5- 14 和表 5- 15 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- 14 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

表 5- 15PCI 设备地址空间描述

模块	地址空间大小	访问类型
PCIe I/O	可配置	BHW
PCIe MEM	可配置	BHW
SPI MEM	16M	BHWDQC
USB-XHCI	1MB	W
SATA	1K	W
GMAC0	32K	W
GMAC1	32K	W
SPI REG	4K	B

上述设备中，除了 PCIe IO/MEM 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。

6 共享 Cache（SCache）

SCache 模块是龙芯 2K1500 处理器内部所有处理器核所共享的三级 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。
- 可用于片上多核结构，也可直接和单处理器 IP 对接。
- 采用 16 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 16 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（除非 16 路共享 Cache 中都是被锁住的块）。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，但必须保证 16 路共享 Cache 中有一路不被锁住。此外当共享 Cache 收到 DMA 写请求时，如果被写的区域在共享 Cache 中命中且被锁住，那么 DMA 写将直接写入到共享 Cache 而不是内存。

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	[47:0]	0 号锁窗口锁地址
Slock0_mask	0x3ff00240	[47:0]	0 号锁窗口掩码
Slock1_valid	0x3ff00208	[63:63]	1 号锁窗口有效位
Slock1_addr	0x3ff00208	[47:0]	1 号锁窗口锁地址

Slock1_mask	0x3ff00248	[47:0]	1 号锁窗口掩码
Slock2_valid	0x3ff00210	[63:63]	2 号锁窗口有效位
Slock2_addr	0x3ff00210	[47:0]	2 号锁窗口锁地址
Slock2_mask	0x3ff00250	[47:0]	2 号锁窗口掩码
Slock3_valid	0x3ff00218	[63:63]	3 号锁窗口有效位
Slock3_addr	0x3ff00218	[47:0]	3 号锁窗口锁地址
Slock3_mask	0x3ff00258	[47:0]	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。

2 个 scache 使用同一个配置寄存器，基地址为 0x1fe00000，偏移地址 0x0280。

表 6- 2 共享 Cache 配置寄存器（SC_CONFIG）

位域	字段名	访问	复位值	描述
0	LRU en	RW	1’b1	Scache LRU 替换算法使能
16	Prefetch En	RW	1’b1	Scache 预取功能使能
22:20	Prefetch config	RW	3’h1	当 scache 预取越过配置大小的地址范围时，停止预取 0 - 4KB 1 - 16KB 2 - 64KB 3 - 1MB 7 - 不受限 （注：SCID_SEL==0 时有效）
26:24	Prefetch lookahead	RW	3’h2	scache 预取步长 0 - 保留 1 - 0x100 2 - 0x200 3 - 0x300 4 - 0x400 5 - 0x500 6 - 0x600 7 - 0x700 （注：SCID_SEL==0 时有效）
30:28	Sc stall dirq cycle	RW	3’h2	SC 指令堵住 dirq 的时钟周期数 0 - 1 cycle（nonstall） 1 - 16-31 cycle random 2 - 32-63 cycle random 3- 64-127 cycle random 4 - 128-255 cycle random 其他 - 无效值

7 处理器核间中断与通信

龙芯 2K1500 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 2K1500 中支持两种不同的访问方式，一种是与 3A3000 等处理器兼容的地址访问模式，另一种是为了支持处理器寄存器空间的直接私有访问。后面章节进行分别说明。

7.1 按地址访问模式

对于龙芯 2K1500，下列寄存器可以使用基地址 0x3ff0_0000 或者 0x1fe0_0000 进行访问。其中，基地址 0x3ff0_0000 可以通过路由设置寄存器中的 disable_0x3ff0 控制位进行关闭。具体寄存器说明和地址见表 7- 1 到表 7- 3。

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。

在龙芯 2K1500 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

7.2 配置寄存器指令访问

在龙芯 2K1500 中，新增了处理器核直接的寄存器访问指令，可以通过私有空间对配置寄存器进行访问。为了更方便地使用核间中断寄存器，在这个模式下，对核间中断寄存器定义进行一些调整。

表 7- 4 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- 5 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 - MailBox4 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。 CSR[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

配置寄存器指令原则上在使用时不跨片访问，但为了满足对调试等的需求，在此使用多个寄存器地址对跨片访问进行支持。值得注意的是，这类寄存器只能写，不能读。

出了前一节提到的 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- 6 处理器核核间通信寄存器

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 2K1500 芯片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。

龙芯 2K1500 芯片中断源分为两部分，一部分来自南北桥上的设备，另一部分来自 NODE 节点的模块，两部分中断采用层次化管理，其中南北桥上的中断通过 INTO/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，如下图所示。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。

南北桥上的中断通过 INTO/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

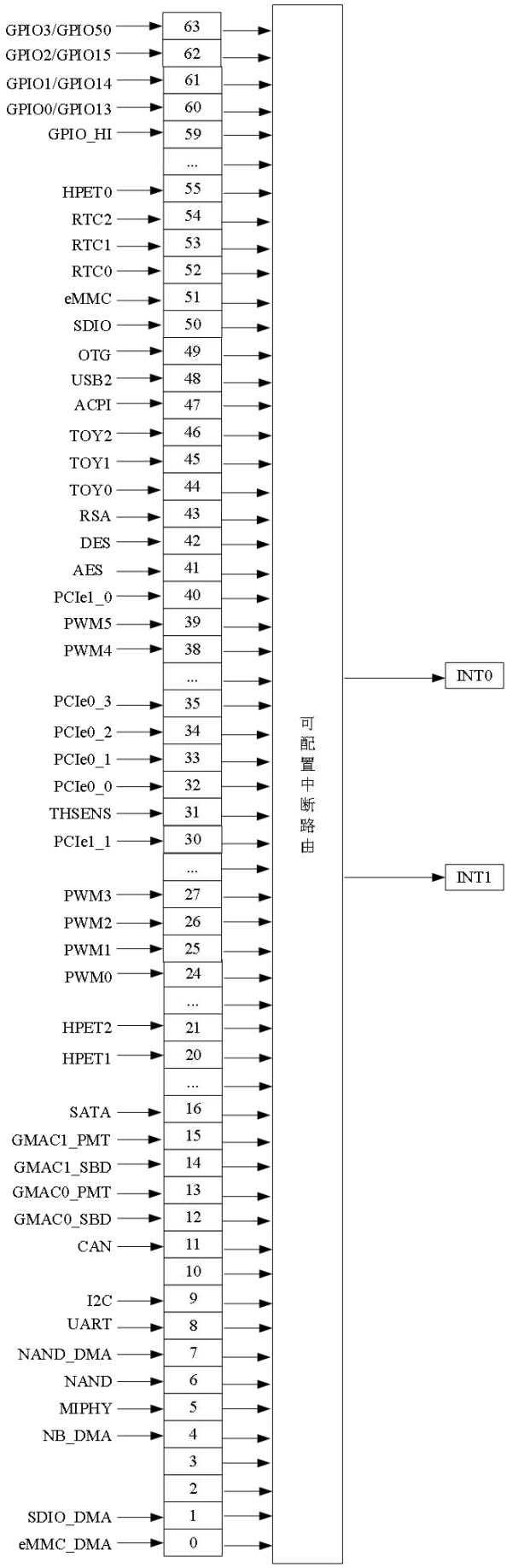

GPIO3/GPIO50
GPIO2/GPIO15
GPIO1/GPIO14
GPIO0/GPIO13
GPIO_HI
HPET0
RTC2
RTC1
RTC0
eMMC
SDIO
OTG
USB2
ACPI
TOY2
TOY1
TOY0
RSA
DES
AES
PCIe1_0
PWM5
PWM4
PCIe0_3
PCIe0_2
PCIe0_1
PCIe0_0
THSENS
PCIe1_1
PWM3
PWM2
PWM1
PWM0
HPET2
HPET1
SATA
GMAC1_PMT
GMAC1_SBD
GMAC0_PMT
GMAC0_SBD
CAN
I2C
UART
NAND_DMA
NAND
MIPHY
NB_DMA
SDIO_DMA
eMMC_DMA
63
62
61
60
59
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
35
34
33
32
31
30
27
26
25
24
21
20
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
可配置中断路由
INT0
INT1

图 8- 1 龙芯 2K1500 处理器南北桥中断路由示意图

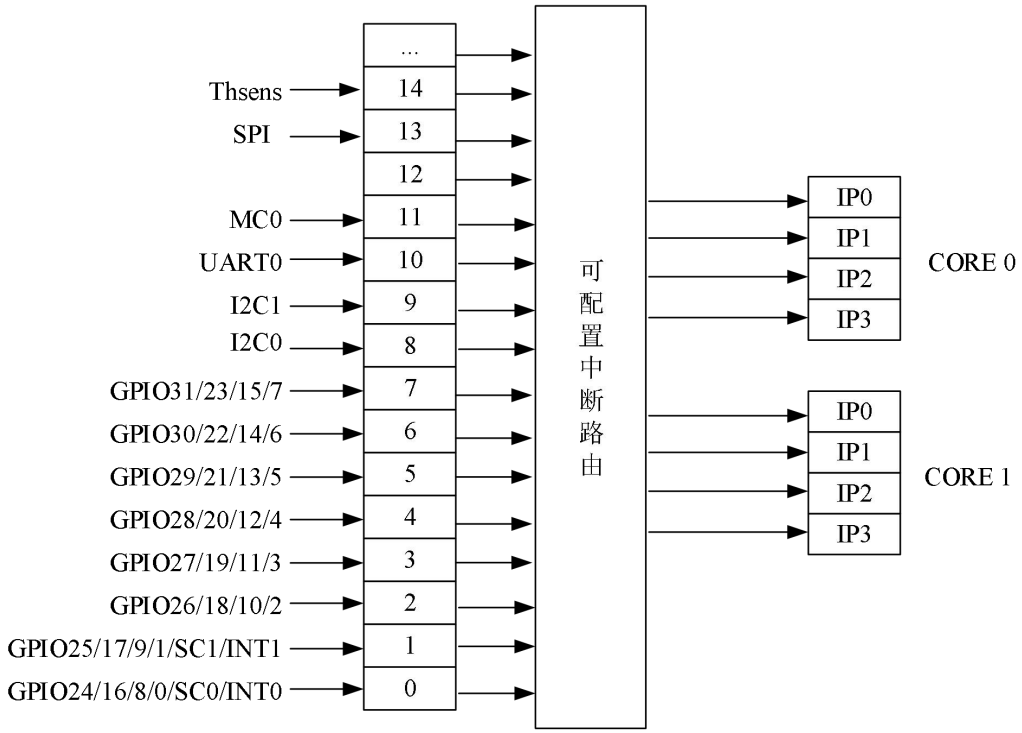


图 8- 2 龙芯 2K1500 处理器 NODE 中断路由示意图

8.1 南北桥中断控制

8.1.1 中断相关寄存器描述

南北桥的中断控制器针对每一个中断源，有一套控制和状态寄存器。

表 8- 2 中断相关寄存器描述

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；	0

			11b：忙碌分发模式。	
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

南北桥中断控制器的访问地址是固定的，访问基地址为 0x5fff,f000，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 8- 2。

表 8- 3 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：100-103h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:16	Reserved	R/W	保留
9:8	SDIO_DMA	R/W	SDIO_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

地址偏移：104-107h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	NAND_DMA	R/W	NAND_DMA 中断路由配置寄存器
17:16	NAND	R/W	NAND 中断路由配置寄存器
9:8	sataphy	R/W	SATAPHY 中断路由配置寄存器
1:0	NB_DMA	R/W	NB_DMA 中断路由配置寄存器

地址偏移：108-10Bh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	can_int_route	R/W	CAN 中断路由配置寄存器
17:16	Reserved	R/W	保留
9:8	i2c_int_route	R/W	I2C 中断路由配置寄存器
1:0	uart_int_route	R/W	UART 中断路由配置寄存器

地址偏移：10C-10Fh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	gmac1_pmt_int_route	R/W	GMAC1_PMT 中断路由配置寄存器
17:16	gmac1_sbd_int_route	R/W	GMAC1_SBD 中断路由配置寄存器
9:8	gmac0_pmt_int_route	R/W	GMAC0_PMT 中断路由配置寄存器
1:0	gmac0_sbd_int_route	R/W	GMAC0_SBD 中断路由配置寄存器

地址偏移：110-113h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:8	Reserved	R/W	保留
1:0	SATA_int_route	R/W	SATA 中断路由配置寄存器

地址偏移：114-117h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
31:16	Reserved	R/W	保留
9:8	hpet2_int_route	R/W	HPET2 中断路由配置寄存器
1:0	hpet1_int_route	R/W	HPET1 中断路由配置寄存器

地址偏移：118-11Bh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	pwm3_int_route	R/W	PWM3 中断路由配置寄存器

17:16	pwm2_int_route	R/W	PWM2 中断路由配置寄存器
9:8	pwm1_int_route	R/W	PWM1 中断路由配置寄存器
1:0	pwm0_int_route	R/W	PWM0 中断路由配置寄存器

地址偏移：11C-11Fh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	thsens_int_route	R/W	Thsensor 中断路由配置寄存器
17:0	Reserved	R/W	保留

地址偏移：120-123h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	pcie_f0_p3_int_route	R/W	PCIe_F0 控制器 3 中断路由配置寄存器
17:16	pcie_f0_p2_int_route	R/W	PCIe_F0 控制器 2 中断路由配置寄存器
9:8	pcie_f0_p1_int_route	R/W	PCIe_F0 控制器 1 中断路由配置寄存器
1:0	pcie_f0_p0_int_route	R/W	PCIe_F0 控制器 0 中断路由配置寄存器

地址偏移：124-127h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	pwm5_int_route	R/W	PWM5 中断路由配置寄存器
17:16	pwm4_int_route	R/W	PWM4 中断路由配置寄存器
9:0	Reserved	R/W	保留

地址偏移：128-12Bh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	rsa_int_route	R/W	RSA 中断路由配置寄存器
17:16	des_int_route	R/W	DES 中断路由配置寄存器
9:8	aes_int_route	R/W	AES 中断路由配置寄存器
1:0	pcie_g0_p0_int_route	R/W	PCIe_G0 控制器 0 中断路由配置寄存器

地址偏移：12C-12Fh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	acpi_int_route	R/W	ACPI 中断路由配置寄存器
17:16	toy2_int_route	R/W	TOY2 中断路由配置寄存器
9:8	toy1_int_route	R/W	TOY1 中断路由配置寄存器
1:0	toy0_int_route	R/W	TOY0 中断路由配置寄存器

地址偏移：130-133h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	emmc_int_route	R/W	eMMC 控制器中断路由配置寄存器
17:16	sdio_int_route	R/W	SDIO 控制器中断路由配置寄存器
9:8	otg_int_route	R/W	OTG 控制器中断路由配置寄存器
1:0	usb2_int_route	R/W	USB2 控制器中断路由配置寄存器

地址偏移：134-137h　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	hpet_int_route	R/W	HPET 中断路由配置寄存器
17:16	rtc2_int_route	R/W	RTC2 中断路由配置寄存器

9:8	rtc1_int_route	R/W	RTC1 中断路由配置寄存器
1:0	rtc0_int_route	R/W	RTC0 中断路由配置寄存器

地址偏移：138-13Bh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	gpio_hi_int_route	R/W	GPIO 高位（bit[56:4]）中断路由配置寄存器
17:0	Reserved	R/W	保留

地址偏移：13C-13Fh　　属性：R/W
默认值：00000000h　　大小：32 位

位域	名称	访问	描述
25:24	gpio3_int_route/gpio50_int_route	R/W	GPIO3/50 中断路由配置寄存器
17:16	gpio2_int_route/gpio15_int_route	R/W	GPIO2/15 中断路由配置寄存器
9:8	gpio1_int_route/gpio14_int_route	R/W	GPIO1/14 中断路由配置寄存器
1:0	gpio0_int_route/gpio13_int_route	R/W	GPIO0/13 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：200-203h　　属性：R/W
默认值：03020100h　　大小：32 位

位域	名称	访问	描述
31:8	Reserved	R/W	保留
15:8	SDIO_DMA_int_vector	R/W	SDIO_DMA MSI 中断向量配置寄存器
7：0	eMMC_DMA_int_vector	R/W	eMMC_DMA MSI 中断向量配置寄存器

地址偏移：204-207h　　属性：R/W
默认值：070605040h　　大小：32 位

位域	名称	访问	描述
31:24	NAND_DMA_int_vector	R/W	NAND_DMA MSI 中断向量配置寄存器
23:16	NAND_int_vector	R/W	NAND MSI 中断向量配置寄存器
15:8	sataphy_int_vector	R/W	SATAPHY MSI 中断向量配置寄存器
7:0	NB_DMA_int_vector	R/W	NB_DMA MSI 中断向量配置寄存器

地址偏移：208-20Bh　　属性：R/W
默认值：0B0A0908h　　大小：32 位

位域	名称	访问	描述
31:24	can_int_vector	R/W	CAN MSI 中断向量配置寄存器
23:16	Reserved	R/W	保留
15:8	i2c_int_vector	R/W	I2C MSI 中断向量配置寄存器
7:0	uart_int_vector	R/W	UART MSI 中断向量配置寄存器

地址偏移：20C-20Fh　　属性：R/W
默认值：0E0F0D0Ch　　大小：32 位

位域	名称	访问	描述
31:24	gmac1_pmt_int_vector	R/W	GMAC1_PMT MSI 中断向量配置寄存器
23:16	gmac1_sbd_int_vector	R/W	GMAC1_SBD MSI 中断向量配置寄存器
15:8	gmac0_pmt_int_vector	R/W	GMAC0_PMT MSI 中断向量配置寄存器

7:0	gmac0_sbd_int_vector	R/W	GMAC0_SBD MSI 中断向量配置寄存器

地址偏移：210-213h　　属性：R/W
默认值：13121110h　　大小：32 位

位域	名称	访问	描述
31:24	Reserved	R/W	保留
7:0	SATA_int_vector	R/W	SATA MSI 中断向量配置寄存器

地址偏移：214-217h　　属性：R/W
默认值：17161514h　　大小：32 位

位域	名称	访问	描述
31:24	can_int_vector	R/W	CAN MSI 中断向量配置寄存器
31:16	Reserved	R/W	保留
15:8	hpet2_int_route	R/W	HPET2 MSI 中断向量配置寄存器
7:0	hpet1_int_route	R/W	HPET1 MSI 中断向量配置寄存器

地址偏移：218-21Bh　　属性：R/W
默认值：1B1A1918h　　大小：32 位

位域	名称	访问	描述
31:24	pwm3_int_vector	R/W	PWM3 MSI 中断向量配置寄存器
23:16	pwm2_int_vector	R/W	PWM2 MSI 中断向量配置寄存器
15:8	pwm1_int_vector	R/W	PWM1 MSI 中断向量配置寄存器
7:0	pwm0_int_vector	R/W	PWM0 MSI 中断向量配置寄存器

地址偏移：21C-21Fh　　属性：R/W
默认值：1E1F1D1Ch　　大小：32 位

位域	名称	访问	描述
31:24	thsens_int_vector	R/W	Thsensor MSI 中断向量配置寄存器
23:0	Reserved	R/W	保留

地址偏移：220-223h　　属性：R/W
默认值：43424140h　　大小：32 位

位域	名称	访问	描述
31:24	pcie_f0_p3_int_vector	R/W	PCIe_F0 控制器 3 MSI 中断向量配置寄存器
23:16	pcie_f0_p2_int_vector	R/W	PCIe_F0 控制器 2 MSI 中断向量配置寄存器
15:8	pcie_f0_p1_int_vector	R/W	PCIe_F0 控制器 1 MSI 中断向量配置寄存器
7:0	pcie_f0_p0_int_vector	R/W	PCIe_F0 控制器 0 MSI 中断向量配置寄存器

地址偏移：224-227h　　属性：R/W
默认值：47464544h　　大小：32 位

位域	名称	访问	描述
31:24	pwm5_int_vector	R/W	PWM5 MSI 中断向量配置寄存器
23:16	pwm4_int_vector	R/W	PWM4 MSI 中断向量配置寄存器
15:0	Reserved	R/W	保留

地址偏移：228-22Bh　　属性：R/W
默认值：4B4A4948h　　大小：32 位

位域	名称	访问	描述
31:24	rsa_int_vector	R/W	RSA MSI 中断向量配置寄存器
23:16	des_int_vector	R/W	DES MSI 中断向量配置寄存器
15:8	aes_int_vector	R/W	AES MSI 中断向量配置寄存器
7:0	pcie_g0_p0_int_vector	R/W	PCIe_G0 控制器 0 MSI 中断向量配置寄存器

地址偏移：22C-22Fh　　属性：R/W

默认值：4F4E4D4Ch　　大小：32 位

位域	名称	访问	描述
31:24	acpi_int_vector	R/W	ACPI MSI 中断向量配置寄存器
23:16	toy2_int_vector	R/W	TOY2 MSI 中断向量配置寄存器
15:8	toy1_int_vector	R/W	TOY1 MSI 中断向量配置寄存器
7:0	toy0_int_vector	R/W	TOY0 MSI 中断向量配置寄存器

地址偏移：230-233h　　属性：R/W
默认值：53525150h　　大小：32 位

位域	名称	访问	描述
31:24	emmc_int_vector	R/W	eMMC 控制器 MSI 中断向量配置寄存器
23:16	sdio_int_vector	R/W	SDIO 控制器 MSI 中断向量配置寄存器
15:8	otg_int_vector	R/W	OTG 控制器 MSI 中断向量配置寄存器
7:0	usb2_int_vector	R/W	USB2 控制器 MSI 中断向量配置寄存器

地址偏移：234-237h　　属性：R/W
默认值：57565554h　　大小：32 位

位域	名称	访问	描述
31:24	hpet_int_vector	R/W	HPET MSI 中断向量配置寄存器
23:16	rtc2_int_vector	R/W	RTC2 MSI 中断向量配置寄存器
15:8	rtc1_int_vector	R/W	RTC1 MSI 中断向量配置寄存器
7:0	rtc0_int_vector	R/W	RTC0 MSI 中断向量配置寄存器

地址偏移：238-23Bh　　属性：R/W
默认值：5B5A5958h　　大小：32 位

位域	名称	访问	描述
31:24	gpio_hi_int_vector	R/W	GPIO 高位（bit[56:4]） MSI 中断向量配置寄存器
23:0	Reserved	R/W	保留

地址偏移：23C-23Fh　　属性：R/W
默认值：5F5E5D5Ch　　大小：32 位

位域	名称	访问	描述
31:24	gpio3_int_vector/gpio50_int_vector	R/W	GPIO3/50 MSI 中断向量配置寄存器
23:16	gpio2_int_vector/gpio15_int_vector	R/W	GPIO2/15 MSI 中断向量配置寄存器
15:8	gpio1_int_vector/gpio14_int_vector	R/W	GPIO1/14 MSI 中断向量配置寄存器
7:0	gpio0_int_vector/gpio13_int_vector	R/W	GPIO0/13 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

南北桥内部设备中断都连接到了南北桥的中断控制器上。对于南北桥来说，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 控制器的中断送给南北桥的中断控制器；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，南北桥内部的 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

南北桥支持双路中断输出，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 2K1500 芯片的传统中断支持 32 个中断源，以统一方式进行管理。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。

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- 4 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	GPIO24/16/8/0/SC0/INT0
1	RW / 0	R / 0	RW / 0	RW / 0	GPIO25/17/9/1/SC1/INT1
2	RW / 0	R / 0	RW / 0	RW / 0	GPIO26/18/10/2/SC2
3	RW / 0	R / 0	RW / 0	RW / 0	GPIO27/19/11/3/SC3
4	RW / 0	R / 0	RW / 0	RW / 0	GPIO28/20/12/4
5	RW / 0	R / 0	RW / 0	RW / 0	GPIO29/21/13/5
6	RW / 0	R / 0	RW / 0	RW / 0	GPIO30/22/14/6
7	RW / 0	R / 0	RW / 0	RW / 0	GPIO31/23/15/7
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	MC0
12	RW / 0	R / 0	RW / 0	RW / 0	
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	
23 : 16	RW / 0	R / 0	RW / 0	RW / 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 或 0x3ff00000 进行访问，也可以通过处理器核的专用寄存器配置指令进行访问。

8.2.1 按地址访问

这种访问方式基地址可以使用 0x1fe00000 或 0x3ff00000。0x3ff00000 的基地址可以通过路由配置寄存器中的 disable_0x3ff0 控制位进行禁用。

表 8- 5 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
CORE0_INTISR	0x1440	路由给 CORE0 的 32 位中断状态
CORE1_INTISR	0x1448	路由给 CORE1 的 32 位中断状态

龙芯 2K1500 中集成了 2 个处理器核，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处理器核。进一步，中断源可以选择路由到处理器核中断 INT0 到 INT3 中的任意一个，即对应 CP0_Status 的 IP2 到 IP5。32 个 I/O 中断源中每一个都对应一个 8 位的路由控制器，其格式和地址如表 8- 5 和表 8- 6 所示。路由寄存器采用向量的方式进行路由选择，如 0x42 表示路由到 1 号处理器的 INT2 上。

龙芯 2K1500 中断引脚路由位增加了编码方式，由 CSR[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，0x22 表示路由到 1 号处理器的 INT2 上。

表 8- 6 中断路由寄存器的说明

位域	说　明
3:0	路由的处理器核向量号
7:4	路由的处理器核中断引脚向量号

表 8- 7 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	GPIO24/16/8/0	Entry16	0x1410	
Entry1	0x1401	GPIO25/17/9/1	Entry17	0x1411	
Entry2	0x1402	GPIO26/18/10/2	Entry18	0x1412	
Entry3	0x1403	GPIO27/19/11/3	Entry19	0x1413	
Entry4	0x1404	GPIO28/20/12/4	Entry20	0x1414	
Entry5	0x1405	GPIO29/21/13/5	Entry21	0x1415	
Entry6	0x1406	GPIO30/22/14/6	Entry22	0x1416	
Entry7	0x1407	GPIO31/23/15/7	Entry23	0x1417	
Entry8	0x1408	I2C0	Entry24	0x1418	
Entry9	0x1409	I2C1	Entry25	0x1419	
Entry10	0x140a	UART0	Entry26	0x141a	
Entry11	0x140b	MC0	Entry27	0x141b	
Entry12	0x140c		Entry28	0x141c	
Entry13	0x140d	SPI	Entry29	0x141d	
Entry14	0x140e	Thsens	Entry30	0x141e	
Entry15	0x140f		Entry31	0x141f	

8.2.2 配置寄存器指令访问

在龙芯 2K1500 中，同样可以通过配置寄存器指令的访问方法，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- 8 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 扩展 I/O 中断

除了兼容原有的传统 IO 中断方式，2K1500 支持扩展 I/O 中断，用于将中断直接分发给各个处理器核，而不再通过中断线进行转发，提升 IO 中断使用的灵活性。

在扩展 IO 中断模式下，中断可以直接进行轮转分发操作。当前版本，最多可以支持 256 个扩展中断向量。

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 或 0x3ff00000，也可以通过处理器核的专用寄存器配置指令进行访问。

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

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

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 INT3 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 CP0_Status 的 IP2 到 IP5。下面是按组进行配置的中断引脚路由寄存器。

龙芯 2K1500 中断引脚路由位增加了编码方式，由 CSR[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，其格式和地址如下表 8- 14 和表 8- 15 所示。路由寄存器采用向量的方式进行路由选择，如 0x2 表示路由到 1 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	保留

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

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- 17 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- 18 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 2K1500 内部集成一个温度传感器，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

传感器监测点如下：

采样点编号	实际监测点
1	USB
2	PCIe_F0
3	PCIe_G0
4	CORE0
5	CORE1
6	Scache1
7	Scache0

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

对于高低温中断报警功能，分别有 4 组控制寄存器对其阈值进行设置。每组寄存器包含以下三个控制位：

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，将会产生中断。需要注意的是，Gate 值的设置应该是与 0x428 寄存器相对应的 16 位数值，而不是摄氏温度；

EN：中断使能控制。置 1 之后该组寄存器的设置才有效；

SEL：输入温度选择。当前 2K1500 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。

高温中断控制寄存器中包含 4 组用于控制高温中断触发的设置位；低温中断控制寄存器中包含 4 组用于控制低温中断触发的设置位。另外还有一组寄存器用于显示中断状态，分别对应于高温中断和低温中断，对该寄存器进行任意写操作将清除中断状态。

这几个寄存器的具体描述如下，其基地址为 0x1fe00000 或 0x3ff00000：

表 9- 1 高低温中断寄存器说明

寄存器	地址	控制	说明
高温中断控制寄存器 Thsens_int_ctrl_Hi	0x1460	RW	[7:0]：Hi_gate0：高温阈值 0，超过这个温度将产生中断 [8:8]：Hi_en0：高温中断使能 0 [11:10]：Hi_Sel0：选择高温中断 0 的温度传感器输入源 [23:16]：Hi_gate1：高温阈值 1，超过这个温度将产生中断 [24:24]：Hi_en1：高温中断使能 1 [27:26]：Hi_Sel1：选择高温中断 1 的温度传感器输入源 [39:32]：Hi_gate2：高温阈值 2，超过这个温度将产生中断 [40:40]：Hi_en2：高温中断使能 2 [43:42]：Hi_Sel2：选择高温中断 2 的温度传感器输入源 [55:48]：Hi_gate3：高温阈值 3，超过这个温度将产生中断 [56:56]：Hi_en3：高温中断使能 3 [59:58]：Hi_Sel3：选择高温中断 3 的温度传感器输入源
低温中断控制寄存器 Thsens_int_ctrl_Lo	0x1468	RW	[7:0]：Lo_gate0：低温阈值 0，低于这个温度将产生中断 [8:8]：Lo_en0：低温中断使能 0 [11:10]：Lo_Sel0：选择低温中断 0 的温度传感器输入源 [23:16]：Lo_gate1：低温阈值 1，低于这个温度将产生中断 [24:24]：Lo_en1：低温中断使能 1 [27:26]：Lo_Sel1：选择低温中断 1 的温度传感器输入源 [39:32]：Lo_gate2：低温阈值 2，低于这个温度将产生中断 [40:40]：Lo_en2：低温中断使能 2 [43:42]：Lo_Sel2：选择低温中断 2 的温度传感器输入源 [55:48]：Lo_gate3：低温阈值 3，低于这个温度将产生中断 [56:56]：Lo_en3：低温中断使能 3 [59:58]：Lo_Sel3：选择低温中断 3 的温度传感器输入源
中断状态寄存器 Thsens_int_status/clr	0x1470	RW	中断状态寄存器，写 1 清除中断 [0]：高温中断触发 [1]：低温中断触发
高低温中断控制高位 Thsens_int_up	0x1478	RW	[7:0] Hi_gate0 高 8 位 [15:8] Hi_gate1 高 8 位 [23:16] Hi_gate2 高 8 位 [31:24] Hi_gate3 高 8 位

			[39:32] Lo_gate0 高 8 位 [47:40] Lo_gate1 高 8 位 [55:48] Lo_gate2 高 8 位 [63:56] Lo_gate3 高 8 位

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
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

此外，还可以使用新增的摄氏温度寄存器直接读取当前的摄氏温度。这个寄存器可以使用 0x1FE00000 或者 0x3FF00000 为基地址的读操作进行访问，也可以使用配置寄存器指令进行直接访问，偏移地址为 0x0428。该寄存器描述如下：

表 9- 2 扩展 IO 中断触发寄存器

名称	偏移地址	权限	描述
Thsens_Temperature	0x0428	R	温度传感器摄氏温度

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 2K1500 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 或 0x3ff00000。

表 9- 3 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

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 2K1500 处理器集成了 4 个 SPI 控制器，其中 SPI0 和 SPI3 可以配置为 4 线模式，SPI0 控制器的地址空间分配如表 10- 1 所示，其余 3 个 SPI 通过 PCI 设备的方式访问。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 Memory0	0X1FC0_0000-0X1FD0_0000	1MByte
SPI Memory1	0X1C00_0000-0X1CFF_FFFF	16MByte
SPI Register	0X1FE0_01F0-0X1FE0_01FF	16Byte

SPI 控制器寄存器物理地址基址为 0x1FE001F0。

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 SPI。

SPI Memory 空间可以通过 CPU 的读请求直接访问，需要注意的是 SPI Memory1 的最低 1M 字节与 SPI Memory0 空间重叠，仅仅是采用了不同的映射方式。

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
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 spr	00 00	00 01	00 10	00 11	01 00	01 01	01 10	01 11	10 00	10 01	10 10	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
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

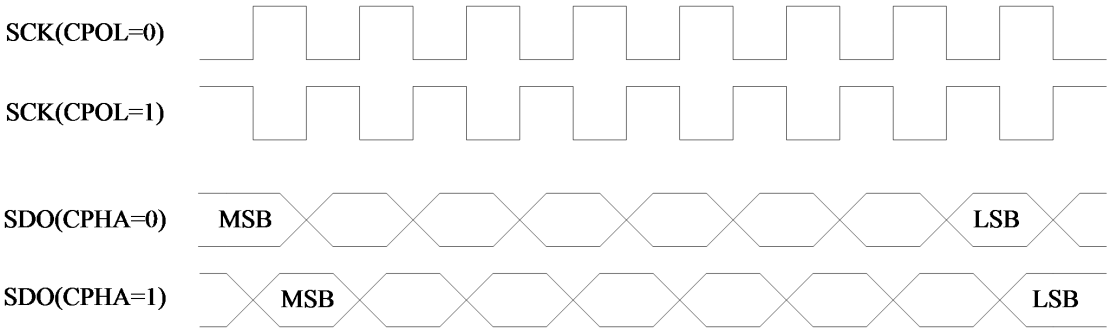


图 10- 1 SPI 主控制器接口时序

10.4.2 SPI Flash 访问时序

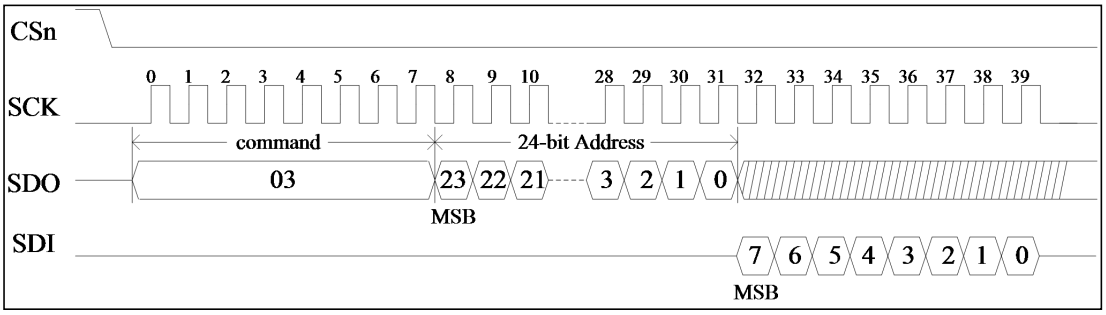


图 10- 2 SPI Flash 标准读时序

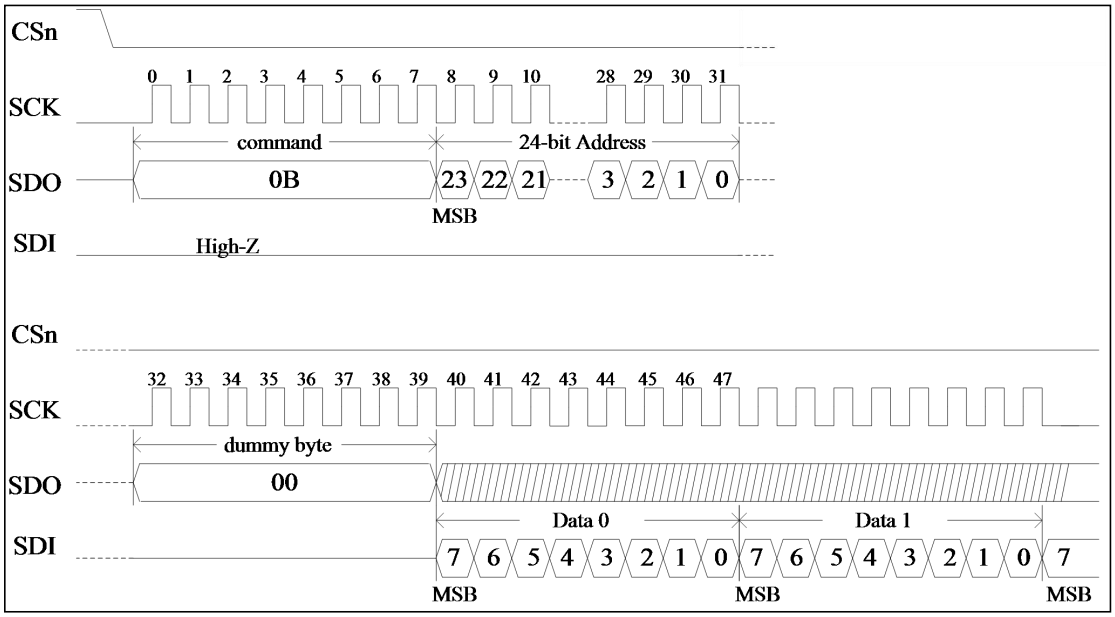


图 10- 3 SPI Flash 快速读时序

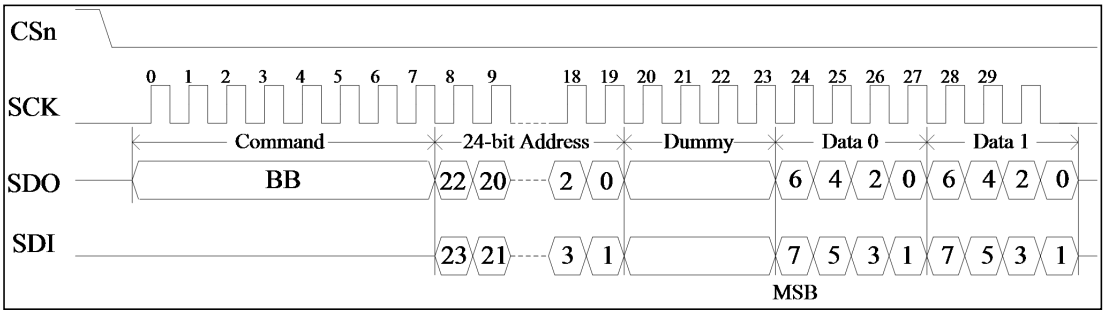


图 10- 4 SPI Flash 双向 I/O 读时序

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
0x1C00_0000	0x1CFF_FFFF	启动空间	当引脚设置为 LIO 启动时可访问，其它情况下不可访问
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。

与 LocalIO 相关的引脚设置寄存器为 4.16 节中的 lio_sel。

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，主要用于连接系统启动 ROM。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。

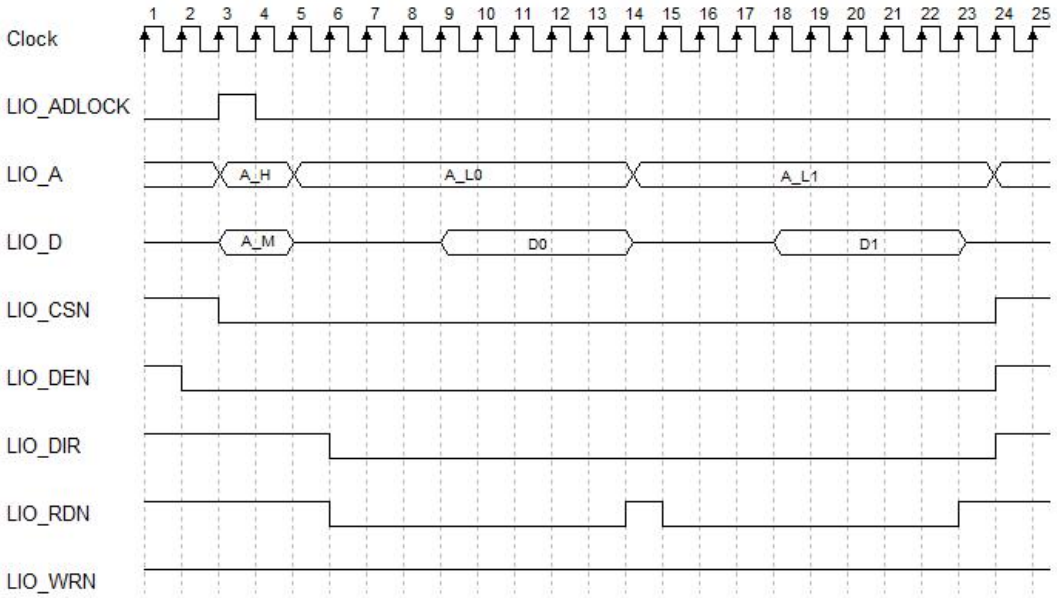


图 11- 1 LocalIO 读时序

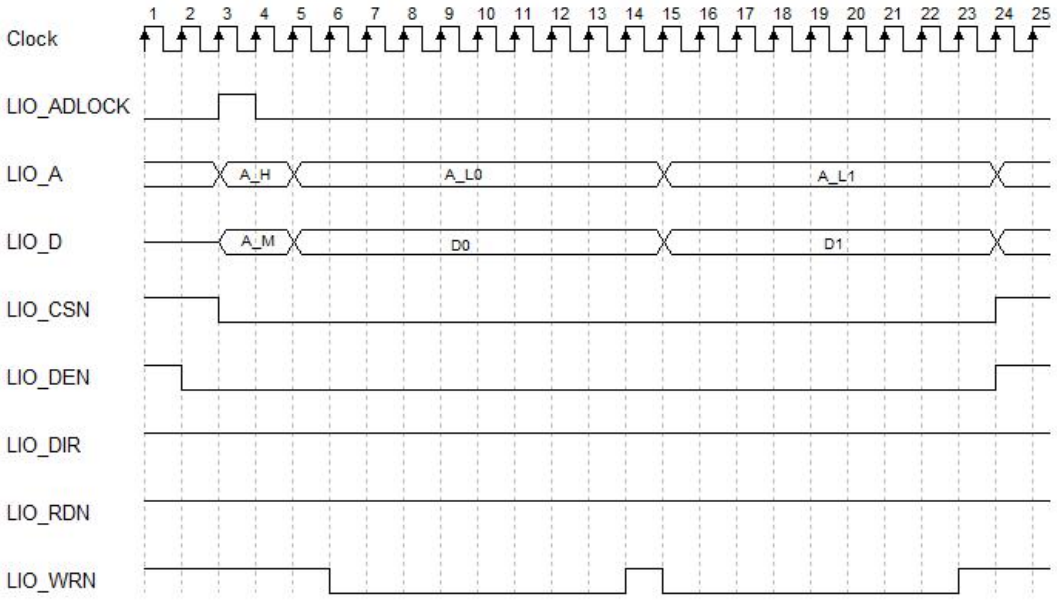


图 11- 2 LocalIO 写时序

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 DDR3 控制器

12.1 DDR3 内存接口

芯片集成的内存接口遵守 DDR3 SDRAM 行业标准（JESD79-3）。

访问地址

DDR3 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。

12.2 DDR3 SDRAM 读操作协议

该控制器兼容 DDR3，DDR3 SDRAM 读操作的协议如图 12- 1 所示。在图中命令（Command，简称 CMD）由 RAS_n，CAS_n 和 WE_n，共三个信号组成。对于读操作，RAS_n=1，CAS_n＝0，WE_n＝1。

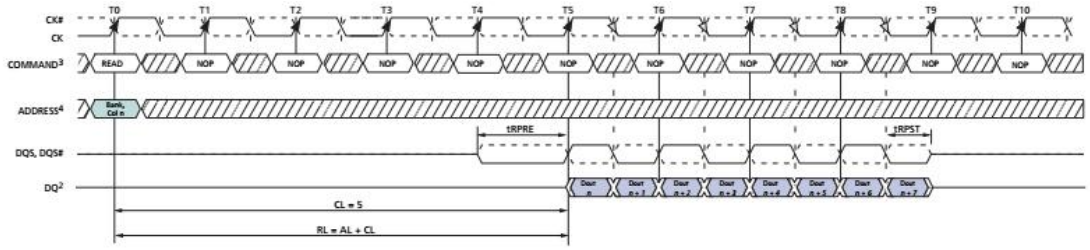


图 12- 1 DDR3 SDRAM 读操作协议

上图中，Cas Latency (CL) = 5，Read Latency (RL) = 5，Burst Length = 8。

12.3 DDR3 SDRAM 写操作协议

DDR3 SDRAM 写操作的协议如图 12- 2 所示。在图中命令 CMD 是由 RAS_n，CAS_n 和 WE_n，共三个信号组成的。对于写操作，RAS_n=1，CAS_n＝0，WE_n＝0。另外，与读操作不同，写操作需要 DQM 来标识写操作的掩码，即需要写入的字节数。DQM 与图中 DQs 信号同步。

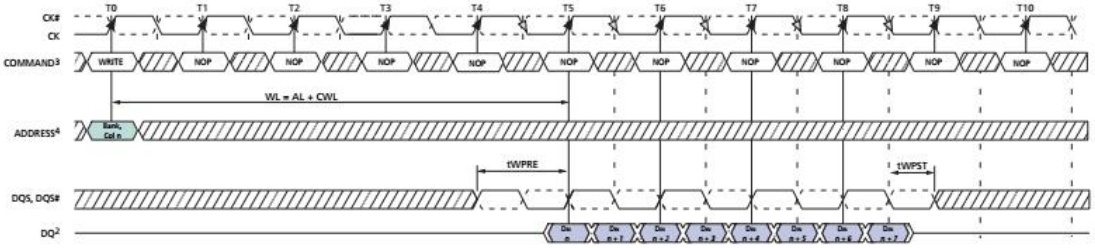


图 12- 2 DDR3 SDRAM 写操作协议

上图中，Cas Latency (CL) = 5，Write Latency (WL) = 5，Burst Length = 8。

12.4 DDR3 SDRAM 参数配置格式

内存控制器的参数列表

表 12- 1 内存控制器软件可见参数列表

Offset	63:56	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_dis able	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_enab le
……								
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					dly_2x_0		redge_sel_0	rddqs_phase_0(R D)
0x0120	w_bdly0_0[31:28]	w_bdly0_0[27: 24]	w_bdly0_0[23: 20]	w_bdly0_0[19:16]	w_bdly0_0[15:12]	w_bdly0_0[11:8]	w_bdly0_0[7:4]	w_bdly0_0[3:0]
0x0128		w_bdly0_0[59: 56]	w_bdly0_0[55: 52]	w_bdly0_0[51:48]	w_bdly0_0[47:44]	w_bdly0_0[43:4 0]	w_bdly0_0[39: 36]	w_bdly0_0[35:32]
0x0130	w_bdly1_0[24:21]	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								w_bdly1_0[27:26]
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[1 1:8]	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[1 1:8]	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					dly_2x_1		redge_sel_1	rddqs_phase_1(RD)
0x01a0	w_bdly0_1[31:28]	w_bdly0_1[27:24]	w_bdly0_1[23:20]	w_bdly0_1[19:16]	w_bdly0_1[15:12]	w_bdly0_1[11:8]	w_bdly0_1[7:4]	w_bdly0_1[3:0]
0x01a8		w_bdly0_1[59:56]	w_bdly0_1[55:52]	w_bdly0_1[51:48]	w_bdly0_1[47:44]	w_bdly0_1[43:40]	w_bdly0_1[39:36]	w_bdly0_1[35:32]
0x01b0	w_bdly1_1[24:21]	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								w_bdly1_1[27:26]
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					dly_2x_2		redge_sel_2	rddqs_phase_2(RD)
0x0220	w_bdly0_2[31:28]	w_bdly0_2[27:24]	w_bdly0_2[23:20]	w_bdly0_2[19:16]	w_bdly0_2[15:12]	w_bdly0_2[11:8]	w_bdly0_2[7:4]	w_bdly0_2[3:0]
0x0228		w_bdly0_2[59:56]	w_bdly0_2[55:52]	w_bdly0_2[51:48]	w_bdly0_2[47:44]	w_bdly0_2[43:40]	w_bdly0_2[39:36]	w_bdly0_2[35:32]
0x0230	w_bdly1_2[24:21]	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								w_bdly1_2[27:26]
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					dly_2x_3		redge_sel_3	rddqs_phase_3(RD)
0x02a0	w_bdly0_3[31:28]	w_bdly0_3[27:24]	w_bdly0_3[23:20]	w_bdly0_3[19:16]	w_bdly0_3[15:12]	w_bdly0_3[11:8]	w_bdly0_3[7:4]	w_bdly0_3[3:0]
0x02a8		w_bdly0_3[59:56]	w_bdly0_3[55:52]	w_bdly0_3[51:48]	w_bdly0_3[47:44]	w_bdly0_3[43:40]	w_bdly0_3[39:36]	w_bdly0_3[35:32]
0x02b0	w_bdly1_3[24:21]	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								w_bdly1_3[27:26]
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					dly_2x_4		redge_sel_4	rddqs_phase_4(RD)
0x0320	w_bdly0_4[31:28]	w_bdly0_4[27:24]	w_bdly0_4[23:20]	w_bdly0_4[19:16]	w_bdly0_4[15:12]	w_bdly0_4[11:8]	w_bdly0_4[7:4]	w_bdly0_4[3:0]
0x0328		w_bdly0_4[59:56]	w_bdly0_4[55:52]	w_bdly0_4[51:48]	w_bdly0_4[47:44]	w_bdly0_4[43:40]	w_bdly0_4[39:36]	w_bdly0_4[35:32]
0x0330	w_bdly1_4[24:21]	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								w_bdly1_4[27:26]
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					dly_2x_5		redge_sel_5	rddqs_phase_5(RD)
0x03a0	w_bdly0_5[31:28]	w_bdly0_5[27:24]	w_bdly0_5[23:20]	w_bdly0_5[19:16]	w_bdly0_5[15:12]	w_bdly0_5[11:8]	w_bdly0_5[7:4]	w_bdly0_5[3:0]
0x03a8		w_bdly0_5[59:56]	w_bdly0_5[55:52]	w_bdly0_5[51:48]	w_bdly0_5[47:44]	w_bdly0_5[43:40]	w_bdly0_5[39:36]	w_bdly0_5[35:32]
0x03b0	w_bdly1_5[24:21]	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								w_bdly1_5[27:26]
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					dly_2x_6		redge_sel_6	rddqs_phase_6(RD)
0x0420	w_bdly0_6[31:28]	w_bdly0_6[27:24]	w_bdly0_6[23:20]	w_bdly0_6[19:16]	w_bdly0_6[15:12]	w_bdly0_6[11:8]	w_bdly0_6[7:4]	w_bdly0_6[3:0]
0x0428		w_bdly0_6[59:56]	w_bdly0_6[55:52]	w_bdly0_6[51:48]	w_bdly0_6[47:44]	w_bdly0_6[43:40]	w_bdly0_6[39:36]	w_bdly0_6[35:32]
0x0430	w_bdly1_6[24:21]	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								w_bdly1_6[27:26]
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					dly_2x_7		redge_sel_7	rddqs_phase_7(RD)
0x04a0	w_bdly0_7[31:28]	w_bdly0_7[27:24]	w_bdly0_7[23:20]	w_bdly0_7[19:16]	w_bdly0_7[15:12]	w_bdly0_7[11:8]	w_bdly0_7[7:4]	w_bdly0_7[3:0]
0x04a8		w_bdly0_7[59:56]	w_bdly0_7[55:52]	w_bdly0_7[51:48]	w_bdly0_7[47:44]	w_bdly0_7[43:40]	w_bdly0_7[39:36]	w_bdly0_7[35:32]
0x04b0	w_bdly1_7[24:21]	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								w_bdly1_7[27:26]
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					dly_2x_8		redge_sel_8	rddqs_phase_8(RD)
0x0520	w_bdly0_8[31:28]	w_bdly0_8[27:24]	w_bdly0_8[23:20]	w_bdly0_8[19:16]	w_bdly0_8[15:12]	w_bdly0_8[11:8]	w_bdly0_8[7:4]	w_bdly0_8[3:0]
0x0528		w_bdly0_8[59:56]	w_bdly0_8[55:52]	w_bdly0_8[51:48]	w_bdly0_8[47:44]	w_bdly0_8[43:40]	w_bdly0_8[39:36]	w_bdly0_8[35:32]
0x0530	w_bdly1_8[24:21]	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								w_bdly1_8[27:26]
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	
CTL								
0x1000	tMRD	tRP	tWLDQSEN	tMOD	tXPR		tCKE	tRESET
0x1008								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								
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					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_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_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			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			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			nc16_map	nc	channel_width	ba_xor_row_offset	addr_new	cs_place
0x1208						bg_xor_row_offset		addr_mirror
0x1210	addr_base_1				addr_base_0			
0x1218								
0x1220	addr_mask_1				addr_mask_0			
0x1228								
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								tRWGNTidle
0x1250							rfifo_age	
0x1258	prior_age3		prior_age2		prior_age1		prior_age0	

0x1260	retry_cnt(RD)					rbuffer_max(RD)	rdfifo_depth	stat_en
0x1268								
……								
0x1280	aw_512_align		rd_before_wr	ecc_enable		int_vector(RD)	int_trigger(RD)	int_enable
0x1288								
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_code_256(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)							
……								
0x1300							ref_num	ref_sch_en
0x1308							Status_sref(RD)	srefresh_req
……								
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			
……								
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				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)							
……								
0x14b8	mpr_train_ecc							
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
……								
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								
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							

12.5 软件编程指南

12.5.1 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。

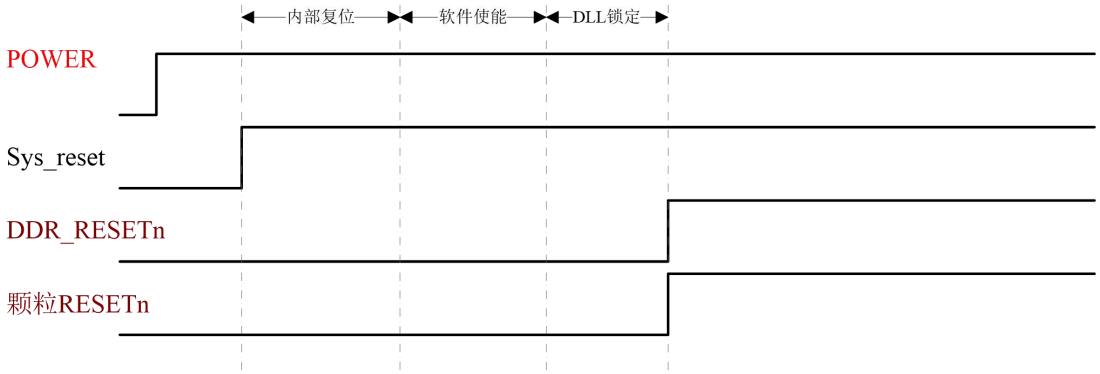
12.5.2 复位引脚的控制

为了在 STR 等状态下更加简单地控制复位引脚，可以通过 pad_reset_po（0x808）寄存器进行特别的复位引脚（DDR_RESETn）控制，主要的控制模式有两种：

(1) 一般模式，reset_ctrl[1:0] == 2’b00。这种模式下，复位信号引脚的行为与一般的控制模式相兼容。主板上直接将 DDR_RESETn 与内存槽上的对应引脚相连。引脚的行为是：

- 未上电时：引脚状态为低；
- 上电时：引脚状态为低；
- 控制器开始初始化时，引脚状态为高；
- 正常工作时，引脚状态为高。

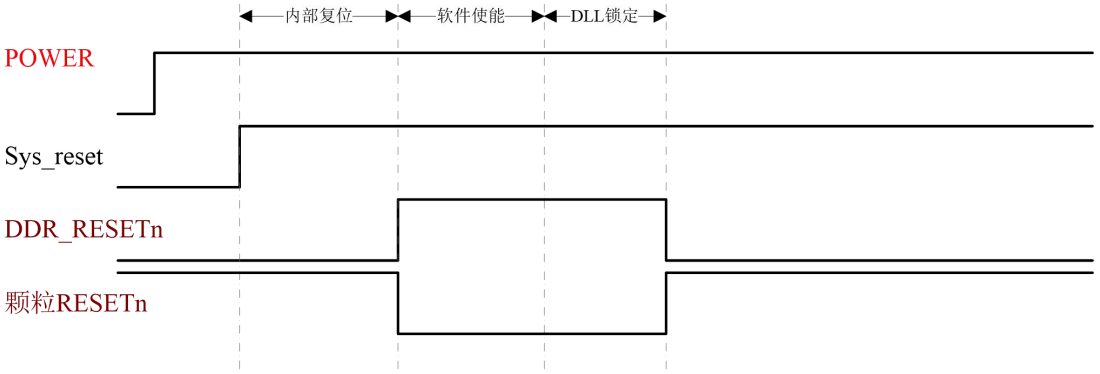
时序如下图所示：



(2) 反向模式，reset_ctrl[1:0] == 2’b10。这种模式下，复位信号引脚在进行内存实际控制的时候，有效电平与一般的控制模式相反。所以主板上需要将 DDR_RESETn 通过反向器与内存槽上的对应引脚相连。引脚的行为是：

- 未上电时：引脚状态为低；
- 上电时：引脚状态为低；
- 控制器开始配置时：引脚状态为高；
- 控制器开始初始化时：引脚状态为低；
- 正常工作时：引脚状态为低。

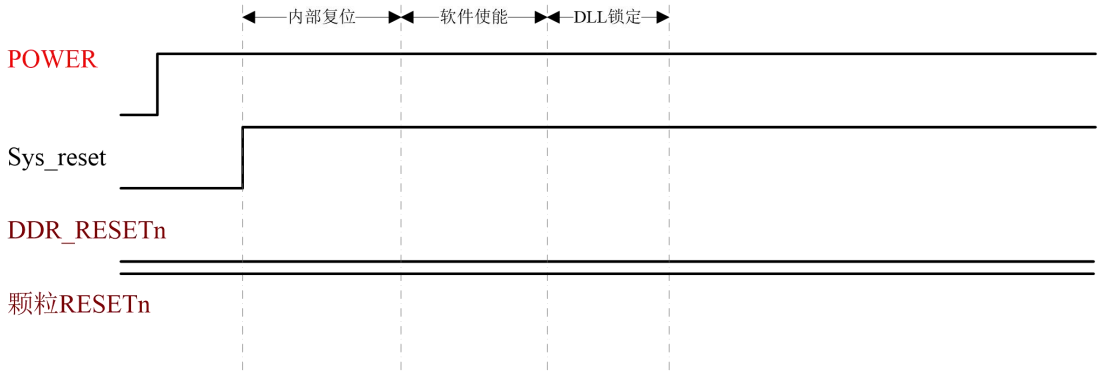
时序如下图所示：



(3) 复位禁止模式，pm_pad_reset_o[1:0] == 2’b01。这种模式下，复位信号引脚在整个内存工作期间，保持低电平。所以主板上需要将 DDR_RESETn 通过反向器与内存槽上的对应引脚相连。引脚的行为是：

- 始终为低；

时序如下图所示：



由后两种复位模式相配合，就可以直接在使用内存控制器的复位信号的情况下实现 STR 控制。当整个系统从关闭状态下启动时，使用（2）中的方法来使用内存条正常复位并开始工作。当系统从 STR 中恢复的时候，使用（3）中的方法来重新配置内存条，使得在不破坏内存条原有状态的条件上使其重新开始正常工作。

12.5.3 Leveling

Leveling 操作是在 DDR3 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。

12.5.4 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。

12.5.5 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。

12.5.6 单独发起 MRS 命令

对于 DDR3 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。

12.5.7 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。

12.5.8 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。

12.6 ECC 功能使用控制

ECC 功能只有在 32 位模式下可以使用。

Ecc_enable（0x1280）包括以下 2 个控制位：

Ecc_enable[0]控制是否使能 ECC 功能，只有设置了这个有效位，才会使能 ECC 功能。

Ecc_enable[1]控制是否通过处理器内部的读响应通路进行报错，以使得出现 ECC 两位错的读访问能立即导致处理器核的异常发生。

除此之外，ECC 出错还可以通过中断方式通知处理器核。这个中断通过 Int_enable 进行控制。中断包括两个向量，Int_vector[0]表示出现 ECC 错误（包括 1 位错与 2 位错），Int_vecotr[1]表示出现 ECC 两位错。Int_vector 的清除通过向对应位写 1 实现。

12.7 出错状态观测

内存控制器出错后，可通过访问相应的系统配置寄存器来获取相应的出错信息，并进行简单的调试操作。寄存器基地址为 0x1fe00000 或 0x3ff00000，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。

表 12- 2 内存控制器出错状态观测寄存器

寄存器	偏移地址	控制	说明
内存控制器 ECC 设置寄存器 mc_ecc_set	0x0600	RW	内存控制器 ECC 设置寄存器 [5:0]：mc int_enable，中断使能 [8]：mc int_trigger，中断触发配置 [21:16]：mc int_vector(RO)，中断向量（只读） [33:32]：mc ecc_enable，ECC 相关功能使能 [40]：mc rd_before_wr，读后写功能使能

	0x0608	RW	保留
内存控制器 ECC 计数寄存器 mc_ecc_cnt	0x0610	RW	内存控制器 ECC 计数寄存器 [7:0]: mc int_cnt，配置 ECC 校验触发中断次数阈值 [15:8]: mc int_cnt_err(RO)，ECC 校验一位出错次数统计（只读） [23:16]: mc int_cnt_fatal(RO)，ECC 校验两位出错次数统计（只读）
内存控制器 ECC 出错次数统计寄存器 mc_ecc_cs_cnt	0x0618	RO	内存控制器 ECC 出错次数统计寄存器 [7:0]: mc ecc_cnt_cs_0，CS0 出现 ECC 校验错次数统计 [15:8]: mc ecc_cnt_cs_1，CS1 出现 ECC 校验错次数统计
内存控制器 ECC 校验码寄存器 mc_ecc_code	0x0620	RO	内存控制器 ECC 校验码寄存器 [7:0]: mc ecc_code_64，64 位 ECC 校验时的 ECC 校验码，使能内存目录功能时无效
内存控制器 ECC 出错地址寄存器 mc_ecc_addr	0x0628	RO	内存控制器 ECC 出错地址寄存器 [63:0]: mc ecc_addr，ECC 校验出错的地址信息
内存控制器 ECC 出错数据寄存器 0 mc_ecc_data0	0x0630	RO	内存控制器 ECC 出错数据寄存器 0 [63:0]:mc_ecc_data0，ECC 校验出错时数据的[63:0]位
内存控制器 ECC 出错数据寄存器 1 mc_ecc_data1	0x0638	RO	内存控制器 ECC 出错数据寄存器 1 [63:0]:mc_ecc_data1，ECC 校验出错时数据的[127:64]位
内存控制器 ECC 出错数据寄存器 2 mc_ecc_data2	0x0640	RO	内存控制器 ECC 出错数据寄存器 2 [63:0]:mc_ecc_data2，ECC 校验出错时数据的[191:128]位
内存控制器 ECC 出错数据寄存器 3 mc_ecc_data3	0x0648	RO	内存控制器 ECC 出错数据寄存器 3 [63:0]:mc_ecc_data3，ECC 校验出错时数据的[255:192]位

13 MISC 低速设备（D2:F0）

13.1 MISC低速设备配置寄存器（MISC-D2:F0）

MISC 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 13- 1 MISC 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	7A22h	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 MISC 低速设备块内部设备的访问。 0：禁止访问； 1：使能对 MISC 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

13.2 内部设备地址路由

MISC 低速设备块下挂载了多个低速设备，包括：UART/CAN、I2C、PWM、HPET、RTC 和 GPIO。这些低速设备根据地址位的 bit[18:16]进行区分，对应关系为：

表 13- 2 MISC 低速设备地址路由

bit[18:16]	0	1	2	4	5	6	7
设备	UART/CAN	I2C	PWM	HPET	RTC	GPIO	-
访问类型	B	B	W	W	W	B	保留

对于 UART、CAN、I2C、PWM 来说，由于包含多个控制器，它们需要进一步的路由。这些设备的内部路由将在各自章节分别介绍。

后续章节将分别对这些低速设备进行介绍。

14 UART 控制器

14.1 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。

2K1500 集成了 12 个 UART 接口和 13 个 UART 控制器，其中，UART0、UART3、UART4、UART5 复用 UART0 接口；UART1、UART6、UART7、UART8 复用 UART1 接口；UART2、UART9、UART10、UART11 复用 UART2 接口。UART0 默认为 node 上的控制器接口，当配置为 2 线模式时，UART3、UART4、UART5 可以选择南桥上的 UART 控制器。UART1 和 UART2 有独立引脚，也可以通过软件调整为与 LIO 复用引脚。

14.2 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 MISC 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 14- 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 - 0xB，分别表示各个 UART 号
[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。

与 UART 相关的引脚设置寄存器为 4.15 节中的 uart1_sel、uart2_sel、uart0_enable、uart1_enable 及 uart2_enable。

14.3 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。

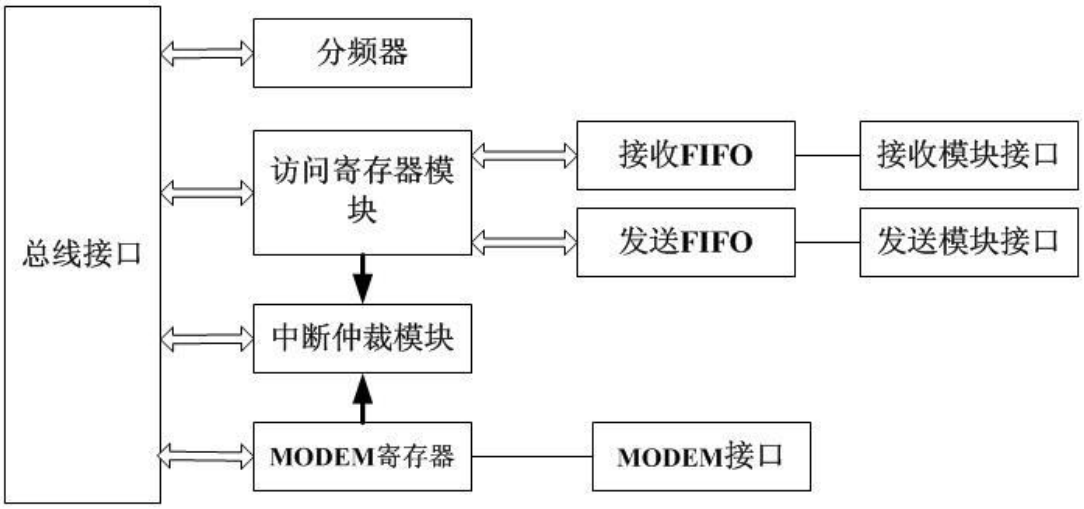


图 14- 1 UART 控制器结构

14.4 寄存器描述

14.4.1 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 14- 2 UART 数据寄存器

位域	位域名称	位宽	访问	描述
7:0	Tx FIFO	8	RW	数据传输寄存器

14.4.2 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 14- 3 UART 中断使能寄存器

位域	位域名称	位宽	访问	描述
7:4	Reserved	4	RW	保留
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

14.4.3 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 14- 4 UART 中断标识寄存器

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

中断控制功能表：

表 14- 5 UART 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

14.4.4 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 14- 6 UART 的 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’ - 4 字节 ‘10’ - 8 字节 ‘11’ - 14 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

14.4.5 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 14- 7 UART 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

14.4.6 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 14- 8 UART 的 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

14.4.7 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 14- 9 UART 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
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。

14.4.8 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 14- 10 UART 的 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

14.4.9 分频锁存器

中文名：分频锁存器 1

寄存器位宽： [7：0]

偏移量： 0x00

复位值： 0x00

表 14- 11 UART 分频锁存器 1

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

中文名：分频锁存器 2

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x00

表 14- 12 UART 分频锁存器 2

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

中文名：分频锁存器 3

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x00

表 14- 13 UART 分频锁存器 3

位域	位域名称	位宽	访问	描述
7:0	D_DIV	8	RW	存放分频锁存器的小数分频值

14.4.10 新增寄存器的使用

新增的接收 FIFO 计数器（RFC）可供 CPU 检测接收 FIFO 中有效数据的个数，据此 CPU 可在收到一次中断后连续读取多个数据，提高 CPU 处理 UART 接收数据的能力；

发送 FIFO 计数器（TFC）可供 CPU 检测发送 FIFO 中有效数据的个数，据此 CPU 可在保证发送 FIFO 不溢出的前提下连续发送多个数据，提高 CPU 处理 UART 发送数据的能力；

分频锁存器 3（即小数分频寄存器）用于解决仅用整数除法无法精确得到所需波特率的问题。以参考时钟 100MHz 除以 16，再除以波特率，所得商整数部分赋值给由分频器锁存器 MSB 和 LSB，小数部分乘以 256 赋值给分频锁存器 D_DIV。

15 CAN

龙芯 2K1500 集成了 6 路 CAN 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。器件与器件之间进行双向传送，最高传送速率 1Mbps。

15.1 访问地址及引脚复用

6 个 CAN 控制器访问基址参考第 13 章，内部寄存器的物理地址构成如下：

表 15- 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 0x10-0x15
[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。

与 CAN 相关的引脚设置寄存器为 4.16 节中的 can_sel。

15.2 标准模式

地址区包括控制段和信息缓冲区，控制段在初始化载入是可被编程来配置通讯参数的，应发送的信息会被写入发送缓冲器，成功接收信息后，微控制器从接收缓冲器中读取接收的信息，然后释放空间以做下一步应用。

初始载入后，寄存器的验收代码，验收屏蔽，总线定时寄存器 0 和 1 以及输出控制就不能改变了。只有控制寄存器的复位位被置高时，才可以访问这些寄存器。在复位模式和工作模式两种不同的模式中，访问寄存器是不同的。当硬件复位或控制器掉线，状态寄存器的总线状态位时会自动进入复位模式。工作模式是通过置位控制寄存器的复位请求位激活的。

在标准模式下，CAN 控制器将 ID[10:3]的值和验收代码的值相同的消息包存入接收缓冲区。如果验收屏蔽码的某位为 1，则验收码对应的位不参与对 ID 的检查。

表 15- 2 CAN 控制器标准模式下寄存器定义

CAN 地址	段	工作模式		复位模式	
		读	写	读	写
0	控制	控制	控制	控制	控制
1		FF	命令	FF	命令
2		状态	—	状态	—
3		FF	—	中断	—
4		FF	—	验收代码	验收代码
5		FF	—	验收屏蔽	验收屏蔽
6		FF	—	总线定时 0	总线定时 0
7		FF	—	总线定时 1	总线定时 1

8		保留	保留	保留	保留
9		—	FIFO 预警深度	—	FIFO 预警深度
10	发送缓冲器	ID(10-3)	ID(10-3)	FF	—
11		ID(2-0), RTR, DLC	ID(2-0), RTR, DLC	FF	—
12		数据字节 1	数据字节 1	FF	—
13		数据字节 2	数据字节 2	FF	—
14		数据字节 3	数据字节 3	FF	—
15		数据字节 4	数据字节 4	FF	—
16		数据字节 5	数据字节 5	FF	—
17		数据字节 6	数据字节 6	FF	—
18		数据字节 7	数据字节 7	FF	—
19		数据字节 8	数据字节 8	FF	—
20	接收缓冲器	ID(10-3)	ID(10-3)	FF	—
21		ID(2-0), RTR, DLC	ID(2-0), RTR, DLC	FF	—
22		数据字节 1	数据字节 1	FF	—
23		数据字节 2	数据字节 2	FF	—
24		数据字节 3	数据字节 3	FF	—
25		数据字节 4	数据字节 4	FF	—
26		数据字节 5	数据字节 5	FF	—
27		数据字节 6	数据字节 6	FF	—
28		数据字节 7	数据字节 7	FF	—
29		数据字节 8	数据字节 8	FF	—
30	小数分频控制	系数小数部分	系数小数部分	系数小数部分	系数小数部分
31		系数整数低位	系数整数低位	系数整数低位	系数整数低位
32		系数整数高位	系数整数高位	系数整数高位	系数整数高位

15.2.1 控制寄存器（CR）

中文名：控制寄存器

寄存器位宽：［7：0］

偏移量：　　0x00

复位值：　　0x01

读此位的值总是逻辑 1。在硬启动或总线状态位设置为 1（总线关闭）时，复位请求位被置为 1。如果这些位被软件访问，其值将发生变化而且会影响内部时钟的下一个上升沿，在外部复位期间微控制器不能把复位请求位置为 0。如果把复位请求位设为 0，微控制器就必须检查这一位以保证外部复位引脚不保持为低。复位请求位的变化是同内部分频时钟同步的。读复位请求位能够反映出这种同步状态。

复位请求位被设为 0 后控制器将会等待

a）一个总线空闲信号（11 个弱势位），如果前一次复位请求是硬件复位或 CPU 初始复位。

b）128 个总线空闲，如果前一次复位请求是 CAN 控制器在重新进入总线开启模式前初

始化总线造成的。

表 15- 3 CAN 控制器标准模式下的控制寄存器格式

位域	位域名称	位宽	访问	描述
7：4	Reserve	2	—	保留
5	FAIE	1	RW	接收缓冲将满中断使能
4	OIE	1	RW	溢出中断使能
3	EIE	1	RW	错误中断使能
2	TIE	1	RW	发送中断使能
1	RIE	1	RW	接收中断使能
0	RR	1	RW	复位请求

15.2.2 命令寄存器（CMR）

中文名：命令寄存器

寄存器位宽：［7：0］

偏移量：　　0x01

复位值：　　0x00

命令寄存器对微控制器来说是只写存储器如果去读这个地址返回值是 0xff

表 15- 4 CAN 控制器标准模式下命令寄存器格式

位域	位域名称	位宽	访问	描述
7	EFF	1	W	扩展模式
6：5	Reserve	2	—	保留
4	GTS	1	W	睡眠
3	CDO	1	W	清除数据溢出
2	RRB	1	W	释放接收缓冲器
1	AT	1	W	中止发送
0	TR	1	W	发送请求

15.2.3 状态寄存器（SR）

中文名：状态寄存器

寄存器位宽：［7：0］

偏移量：　　0x02

复位值：　　0x00

表 15- 5 CAN 控制器标准模式下状态寄存器格式

位域	位域名称	位宽	访问	描述
7	BS	1	R	总线状态
6	ES	1	R	出错状态
5	TS	1	R	发送状态
4	RS	1	R	接收状态
3	TCS	1	R	发送完毕状态
2	TBS	1	R	发送缓存器状态
1	DOS	1	R	数据溢出状态
0	RBS	1	R	接收缓存器状态

15.2.4 中断寄存器（IR）

中文名：中断寄存器

寄存器位宽：［7：0］

偏移量：　　0x03

复位值：　　0x00

表 15- 6 CAN 控制器标准模式下中断寄存器格式

位域	位域名称	位宽	访问	描述
7：5	Reserved	1	R	保留
4	FAI	1	R	接收缓冲将满中断
3	DOI	1	R	数据溢出中断
2	EI	1	R	错误中断
1	TI	1	R	发送中断
0	RI	1	R	接收中断

15.2.5 验收代码寄存器（ACR）

中文名：验收代码寄存器

寄存器位宽：［7：0］

偏移量：　　0x04

复位值：　　0x00

在复位情况下，该寄存器是可以读写的。

表 15- 7 CAN 验收代码寄存器

位域	位域名称	位宽	访问	描述
7:0	AC	8	RW	ID 验收代码

15.2.6 验收屏蔽寄存器（AMR）

中文名：验收屏蔽寄存器

寄存器位宽：［7：0］

偏移量：　　0x05

复位值：　　0x00

验收代码位 AC 和信息识别码的高 8 位 ID.10-ID.3 相等且与验收屏蔽位 AM 的相应位相或为 1 时数据可以接收。在复位情况下，该寄存器是可以读写的。

表 15- 8 CAN 验收屏蔽寄存器

位域	位域名称	位宽	访问	描述
7:0	AM	8	RW	ID 屏蔽位

15.2.7 发送缓冲区列表

缓冲器是用来存储微控制器要 CAN 控制器发送的信息，它被分为描述符区和数据区。发送缓冲器的读/写只能由微控制器在工作模式下完成，在复位模式下读出的值总是 0xff。

表 15- 9 CAN 控制器标准模式下发送缓冲区格式

地址	区	名称	数据位
10	发送缓冲器	识别码字节 1	ID(10-3)
11		识别码字节 2	ID(2-0)， RTR,DLC
12		TX 数据 1	TX 数据 1
13		TX 数据 2	TX 数据 2
14		TX 数据 3	TX 数据 3
15		TX 数据 4	TX 数据 4
16		TX 数据 5	TX 数据 5
17		TX 数据 6	TX 数据 6
18		TX 数据 7	TX 数据 7
19		TX 数据 8	TX 数据 8

15.2.8 接收缓冲区列表

接收缓冲区的配置和发送缓冲区的一样，只是地址变为 20－29。

15.3 扩展模式

在扩展模式下，允许使用 11 位 ID 的标准帧和 29 位 ID 的扩展帧（是标准帧还是扩展帧由 TX 帧信息的最高位 IDE 位确定）。

在扩展模式下，CAN 控制器可以接收 11 位 ID 的标准帧也可以接收 29 位 ID 的扩展帧。在接收不同格式的帧的时候，验收代码 0～验收代码 3（code0 ~ code3）所检查的内容有所不同。验收屏蔽码 0~验收屏蔽码 3（mask0 ~ mask3）对应验收代码 0～验收代码 3。如果验收屏蔽码的某 1 位为 1，则对应的验收代码的那一位就不参与对接收包的 ID 的检查。

在扩展模式下，CAN 控制器可以对接收到的消息包进行单滤波也可以进行双滤波。在进行单滤波时，验收代码 0～验收代码 3 仅对单一 ID 进行过滤。在进行双滤波时，验收代码 0～验收代码 3 可以对两个不同的 ID 进行。CAN 控制器只将 ID 符合滤波条件的消息包存入接收缓冲区。

对 11 位 ID 包的单滤波：

允许将 rdata0 和 rdata1 用来扩展对 ID 的验收长度

(rx_id[10:3] == code0)&&(rtr == code1[4])&&(rx_id[2:0] == code1[7:5])&&(rx_data0 == code2)&&(rx_data1 == code3)

对 11 位 ID 包的双滤波：

(rx_id[10:3] == code0)&&(rtr == code1[4])&&(rx_id[2:0] == code1[7:5])&&(rx_data0 == {code1[3:0],code3[3:0]})

或者

(rx_id[10:3] == code2)&&(rtr == code3[4])&&(rx_id[2:0] == code3[7:5])

对 29 位 ID 包的单滤波：

(rx_id[28:21] == code0) && (rx_id[20:13] == code1)&&(rx_id[12:5] == code2)&&(rtr == code3[2])&&(rx_id[4:0] == code3[7:3])

对 29 位 ID 的双滤波：

(rx_id[28:21] == code0) && (rx_id[20:13] == code1)

或者

(rx_id[28:21] == code2) && (rx_id[20:13] == code3)

表 15- 10 扩展模式下 CAN 控制器的地址列表

CAN 地址	工作模式		复位模式	
	读	写	读	写
0	控制	控制	控制	控制
1	0	命令	0	命令
2	状态	—	状态	—
3	中断	—	中断	—
4	中断使能	中断使能	中断使能	中断使能
5	—	—	验收屏蔽	验收屏蔽
6	总线定时 0	—	总线定时 0	总线定时 0
7	总线定时 1	—	总线定时 1	总线定时 1
8	保留	保留	保留	保留
9	—	FIFO 预警深度	—	FIFO 预警深度
10	保留	保留	保留	保留
11	仲裁丢失捕捉	—	仲裁丢失捕捉	—
12	错误代码捕捉	—	错误代码捕捉	—
13	错误警报限制	—	错误警报限制	—
14	RX 错误计数器	—	RX 错误计数器	—
15	TX 错误计数器	—	TX 错误计数器	—
16	RX 帧信息 {IDE,RTR,2’ h0,DLC[3:0]}	TX 帧信息 {IDE,RTR,2’ h0,DLC[3:0]}	验收代码 0 Id[28:21]（扩展帧） Id[10:3](非扩展帧)	验收代码 0
17	RX_Id[28:21] （扩展帧） RX_Id[10:3] （非扩展帧）	TX_Id[28:21] （扩展帧） TX_Id[10:3] （非扩展帧）	验收代码 1 Id[20:13]（扩展帧） {Id[2:0],RTR,4’ h0}(非扩展帧，单滤波) {Id[2:0],RTR,data0[7:4] }(非扩展帧，双滤波) {Id[2:0],RTR,4’ h0] }(非扩展帧，单滤波)	验收代码 1
18	RX_Id[20:13]（扩展帧） {RX_Id[2:0],5’ h0}(非扩展帧)	TX_Id[20:13]（扩展帧） {TX_Id[2:0],5’ h0}(非扩展帧)	验收代码 2 Id2[28:21]（扩展帧，双滤波） Id[12:5]（扩展帧，单滤波） Id2[10:3](非扩展帧，双滤波) Data0(非扩展帧,单滤波）	验收代码 2
19	RX Id[12:5]（扩展帧） RX data0(非扩展帧)	TX Id[12:5]（扩展帧） TX data0(非扩展帧)	验收代码 3 Id2[20:13]（扩展帧，双滤波） {id[4:0],RTR,2’ h0}（扩展帧，单滤波）	验收代码 3

			{id2[2:0],RTR,data0[3:0]}(非扩展帧，双滤波) Data1(非扩展帧,单滤波)	
20	{RX_id[4:0],3' h0} （扩展帧） RX data1(非扩展帧)	{TX_id[4:0],3' h0} （扩展帧） TX data1(非扩展帧)	验收屏蔽 0 （不判断为 1 的那位的 id 值）	验收屏蔽 0
21	RX data0(扩展帧) RX data2(非扩展帧)	TX data0(扩展帧) TX data2(非扩展帧)	验收屏蔽 1	验收屏蔽 1
22	RX data1(扩展帧) RX data3(非扩展帧)	TX data1(扩展帧) TX data3(非扩展帧)	验收屏蔽 2	验收屏蔽 2
23	RX data2(扩展帧) RX Data4(非扩展帧)	TX data2(扩展帧) TX Data4(非扩展帧)	验收屏蔽 3	验收屏蔽 3
24	RX data3(扩展帧) RX data5(非扩展帧)	TX data3(扩展帧) TX data5(非扩展帧)	—	—
25	RX data4(扩展帧) RX data6(非扩展帧)	TX data4(扩展帧) TX data6(非扩展帧)	—	—
26	RX data5(扩展帧) RX data7(非扩展帧)	TX data5(扩展帧) TX data7(非扩展帧)	—	—
27	RX data6(扩展帧)	TX data6(扩展帧)	—	—
28	RX data7(扩展帧)	TX data7(扩展帧)	—	—
29	RX 信息计数器	—	RX 信息计数器	—
30	系数小数部分	系数小数部分	系数小数部分	系数小数部分
31	系数整数低位	系数整数低位	系数整数低位	系数整数低位
32	系数整数高位	系数整数高位	系数整数高位	系数整数高位

15.3.1 模式寄存器（MOD）

中文名：模式寄存器

寄存器位宽：［7：0］

偏移量：　　0x00

复位值：　　0x01

读此位的值总是逻辑 1。在硬启动或总线状态位设置为 1（总线关闭）时，复位请求位被置为 1。如果这些位被软件访问，其值将发生变化而且会影响内部时钟的下一个上升沿，在外部复位期间微控制器不能把复位请求位置为 0。如果把复位请求位设为 0，微控制器就必须检查这一位以保证外部复位引脚不保持为低。复位请求位的变化是同内部分频时钟同步的。读复位请求位能够反映出这种同步状态。

复位请求位被设为 0 后控制器将会等待

a）一个总线空闲信号（11 个弱势位），如果前一次复位请求是硬件复位或 CPU 初始复位。

b）128 个总线空闲，如果前一次复位请求是 CAN 控制器在重新进入总线开启模式前初始化总线造成的。

表 15- 11 CAN 控制器扩展模式下的模式寄存器格式

位域	位域名称	位宽	访问	描述
7	DM	1	RW	接收缓冲的 DMA 模式
6	FDE	1	RW	小数分频使能

5	Reserve	1	—	—
4	SM	1	RW	睡眠模式
3	AFM	1	RW	单/双滤波模式(0 为双滤波,仅复位模式可写)
2	STM	1	RW	正常工作模式(1 为自测试模式,仅复位模式可写)
1	LOM	1	RW	只听模式（仅复位模式可写）
0	RM	1	RW	复位模式

15.3.2 命令寄存器（CMR）

中文名：命令寄存器

寄存器位宽： [7：0]

偏移量： 0x01

复位值： 0x00

命令寄存器对微控制器来说是只写存储器如果去读这个地址返回值是 0xff

表 15- 12 CAN 控制器扩展模式下命令寄存器格式

位域	位域名称	位宽	访问	描述
7	EFF	1	W	扩展模式（仅在复位模式下可写）
6：5	Reserve	2	—	保留
4	SRR	1	W	自接收请求（和 TR 不能同时为 1）
3	CDO	1	W	清除数据溢出
2	RRB	1	W	释放接收缓冲器
1	AT	1	W	中止发送
0	TR	1	W	发送请求（和 SRR 不能同时为 1）

15.3.3 状态寄存器（SR）

中文名：状态寄存器

寄存器位宽： [7：0]

偏移量： 0x02

复位值： 0x00

表 15- 13 CAN 控制器扩展模式下状态寄存器格式

位域	位域名称	位宽	访问	描述
7	BS	1	R	总线状态
6	ES	1	R	出错状态
5	TS	1	R	发送状态
4	RS	1	R	接收状态
3	TCS	1	R	发送完毕状态
2	TBS	1	R	发送缓存器状态
1	DOS	1	R	数据溢出状态
0	RBS	1	R	接收缓存器状态

15.3.4 中断寄存器（IR）

中文名：中断寄存器

寄存器位宽： [7：0]

偏移量： 0x03

复位值： 0x00

表 15- 14 CAN 控制器扩展模式下中断寄存器格式

位域	位域名称	位宽	访问	描述
7	BEI	1	R	总线错误中断
6	ALI	1	R	仲裁丢失中断
5	EPI	1	R	错误消极中断
4	FAI	1	R	接收缓冲将满中断
3	DOI	1	R	数据溢出中断
2	EI	1	R	错误中断
1	TI	1	R	发送中断
0	RI	1	R	接收中断

15.3.5 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： 0x04

复位值： 0x00

表 15- 15 CAN 控制器扩展模式下中断使能寄存器格式

位域	位域名称	位宽	访问	描述
7	BEIE	1	RW	总线错误中断使能
6	ALIE	1	RW	仲裁丢失中断使能
5	EPIE	1	RW	错误消极中断使能
4	FAIE	1	RW	接收缓冲将满中断使能
3	DOIE	1	RW	数据溢出中断使能
2	EIE	1	RW	错误中断使能
1	TIE	1	RW	发送中断使能
0	RIE	1	RW	接收中断使能

15.3.6 仲裁丢失捕捉寄存器

中文名：仲裁丢失捕捉寄存器

寄存器位宽： [7：0]

偏移量： 0xB

复位值： 0x00

表 15- 16 CAN 控制器扩展模式下仲裁丢失捕捉寄存器格式

位域	位域名称	位宽	访问	描述
7：5	—	3	R	保留
4	BITNO4	1	R	第四位
3	BITNO3	1	R	第三位
2	BITNO2	1	R	第二位
1	BITNO1	1	R	第一位

0	BITN00	1	R	第零位

位					十进制值	功能
ALC. 4	ALC. 3	ALC. 2	ALC. 1	ALC. 0		
0	0	0	0	0	0	仲裁丢失在识别码的bit1
0	0	0	0	1	1	仲裁丢失在识别码的bit2
0	0	0	1	0	2	仲裁丢失在识别码的bit3
0	0	0	1	1	3	仲裁丢失在识别码的bit4
0	0	1	0	0	4	仲裁丢失在识别码的bit5
0	0	1	0	1	5	仲裁丢失在识别码的bit6
0	0	1	1	0	6	仲裁丢失在识别码的bit7
0	0	1	1	1	7	仲裁丢失在识别码的bit8
0	1	0	0	0	8	仲裁丢失在识别码的bit9
0	1	0	0	1	9	仲裁丢失在识别码的bit10
0	1	0	1	0	10	仲裁丢失在识别码的bit11
0	1	0	1	1	11	仲裁丢失在SRTR位
0	1	1	0	0	12	仲裁丢失在IDE位
0	1	1	0	1	13	仲裁丢失在识别码的bit12
0	1	1	1	0	14	仲裁丢失在识别码的bit13
0	1	1	1	1	15	仲裁丢失在识别码的bit14
1	0	0	0	0	16	仲裁丢失在识别码的bit15
1	0	0	0	1	17	仲裁丢失在识别码的bit16
1	0	0	1	0	18	仲裁丢失在识别码的bit17
1	0	0	1	1	19	仲裁丢失在识别码的bit18
1	0	1	0	0	20	仲裁丢失在识别码的bit19
1	0	1	0	1	21	仲裁丢失在识别码的bit20
1	0	1	1	0	22	仲裁丢失在识别码的bit21
1	0	1	1	1	23	仲裁丢失在识别码的bit22
1	1	0	0	0	24	仲裁丢失在识别码的bit23
1	1	0	0	1	25	仲裁丢失在识别码的bit24
1	1	0	1	0	26	仲裁丢失在识别码的bit25
1	1	0	1	1	27	仲裁丢失在识别码的bit26
1	1	1	0	0	28	仲裁丢失在识别码的bit27
1	1	1	0	1	29	仲裁丢失在识别码的bit28
1	1	1	1	0	30	仲裁丢失在识别码的bit29
1	1	1	1	1	31	仲裁丢失在RTR位

15.3.7 错误警报限制寄存器（EMLR）

中文名：错误警报限制寄存器

寄存器位宽：［7：0］

偏移量：　　0xD

复位值：　　0x60

表 15- 17 CAN 错误劲爆限制寄存器

位域	位域名称	位宽	访问	描述
7：0	EML	8	RW	错误警报阀值

15.3.8 RX 错误计数寄存器（RXERR）

中文名： RX 错误计数寄存器

寄存器位宽： [7：0]

偏移量： 0xE

复位值： 0x60

表 15- 18 CAN 的 RX 错误计数寄存器

位域	位域名称	位宽	访问	描述
7：0	RXERR	8	R	接收错误计数

15.3.9 TX 错误计数寄存器（TXERR）

中文名： TX 错误计数寄存器

寄存器位宽： [7：0]

偏移量： 0xF

复位值： 0x60

表 15- 19 CAN 的 TX 错误计数寄存器

位域	位域名称	位宽	访问	描述
7：0	TXERR	8	R	发送错误计数

15.3.10 验收滤波器

在验收滤波器的帮助下，只有当接收信息中的识别位和验收滤波器预定义的值相等时，CAN 控制器才允许将已接收信息存入 RXFIFO。验收滤波器由验收代码寄存器和验收屏蔽寄存器定义。在模式寄存器中选择单滤波器模式或者双滤波器模式。具体的配置可以参考 SJA1000 的数据手册。

15.3.11 RX 信息计数寄存器（RMCR）

中文名： RX 信息计数寄存器

寄存器位宽： [7：0]

偏移量： 0x1D

复位值： 0x00

表 15- 20 CAN 的 RX 错信息计数寄存器

位域	位域名称	位宽	访问	描述
7：0	RMCR	8	R	接收的数据帧计数器

15.4 公共寄存器

1bit time = internal_clock_time * ((BRP + 1) * 2) *(1+ (TESG2 + 1) + (TESG1 + 1))

15.4.1 总线定时寄存器 0（BTR0）

中文名：总线定时寄存器

寄存器位宽：［7：0］

偏移量：　　0x06

复位值：　　0x00

注：在复位模式是可以读写的，工作模式是只读的。

表 15- 21 CAN 总线定时寄存器 0

位域	位域名称	位宽	访问	描述
7：6	SJW	8	RW	同步跳转宽度
5：0	BRP	8	RW	波特率分频系数

15.4.2 总线定时寄存器 1（BTR1）

中文名：总线定时寄存器 1

寄存器位宽：［7：0］

偏移量：　　0x07

复位值：　　0x00

表 15- 22 CAN 总线定时寄存器 1

位域	位域名称	位宽	访问	描述
7	SAM	1	RW	为 1 时三次采样，否则是一次采用
6：4	TESG2	3	RW	一个 bit 中的时间段 2 的计数值
3：0	TSEG1	4	RW	一个 bit 中的时间段 1 的计数值

15.4.3 输出控制寄存器（OCR）

中文名：输出控制寄存器

寄存器位宽：［7：0］

偏移量：　　0x08

复位值：　　0x00

表 15- 23 CAN 输出控制寄存器

位域	位域名称	位宽	访问	描述
7：0	OCR	8	RW	保留（未使用）

16 I2C 控制器

16.1 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。其中 I2C2 还可以作为从设备，包含 6 个 8bit 数据寄存器可用于 LA132 的通信接口，主设备还可以访问温度传感器、RTC 计数值，并通过中断寄存器产生中断请求。

16.2 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。

与 I2C 相关的引脚设置寄存器为 4.1 节中的 i2c_sel。

16.3 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。

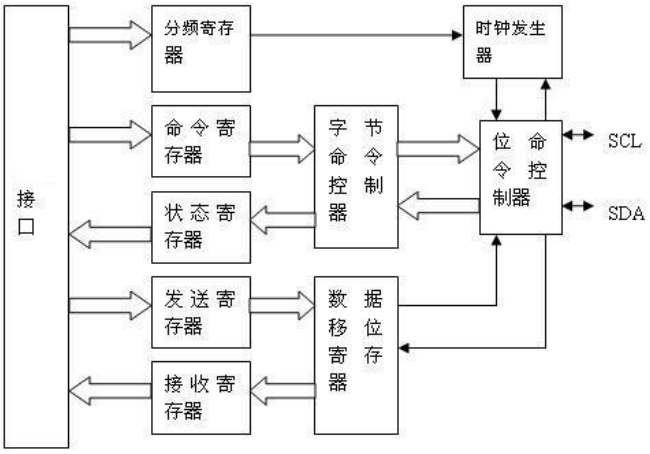


图 16- 1 I2C 主控制器结构

16.4 I2C 主控制器寄存器说明

16.4.1 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

16.4.2 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，从 LPB 总线 PCLK 时钟输入的频率为 clock_a，SCL 总线的输出频率为 clock_s，则应满足如下关系：

Prcescale = clock_a/(4*clock_s)-1

16.4.3 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	Slave_mode	1	RW	从模式设置（仅对 I2C2 有效） 0：主模式 1：从模式
4:0	Reserved	6	RW	保留

16.4.4 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

16.4.5 接收数据寄存器（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

16.4.6 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	产生应答信号
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。对这些位读操作时候总是读回‘0’。bit 3 为 1 时表示此次传输结束时控制器不发送 ack，反之结束时发送 ack。

16.4.7 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

16.4.8 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

*该寄存器仅用于 I2C2

16.5 I2C2 从模式地址空间

当 I2C2 设备配置为从模式时（将主控制器寄存器的 0x7 使能从模式，并配置从模式地址），I2C2 作为从设备，额外的寄存器空间被使能，该部分额外的寄存器空间即可被 cpu 访问，也可以通过外部 I2C 主设备访问。

I2C 主设备可访问 0x0~0x16 的 I2C 总线地址空间，其中前 16 字节为只读空间，为芯片

内部设备的数据，这部分寄存器无法通过 cpu 访问。I2C 地址 0x10~0x16 为可读写的中断寄存器和 6 个数据寄存器。从 cpu 的地址空间上看，这部分寄存器被映射到了在主模式下原本为 I2C 主控制器的控制寄存器的地址空间，实现主设备和 cpu 访问的通信接口。值得注意的是从模式下 cpu 仍然可以访问 0x7 从模式控制寄存器以实现从模式的使能控制。

从模式下的寄存器空间如下表所示。

地址	I2C 总线访问	CPU 访问
0x0	温度（R）	Slave 中断（R/W）
0x1	温度（R）	数据寄存器 1（R/W）
0x2	0（R）	数据寄存器 2（R/W）
0x3	0（R）	数据寄存器 3（R/W）
0x4	0（R）	数据寄存器 4（R/W）
0x5	0（R）	数据寄存器 5（R/W）
0x6	0（R）	数据寄存器 6（R/W）
0x7	0（R）	从模式控制寄存器（R/W）
0x8	RTC（R）	
0x9	RTC（R）	
0xa	RTC（R）	
0xb	RTC（R）	
0xc	RTC（R）	
0xd	RTC（R）	
0xe	RTC（R）	
0xf	RTC（R）	
0x10	Slave 中断（R/W）	
0x11	数据寄存器 1（R/W）	
0x12	数据寄存器 2（R/W）	
0x13	数据寄存器 3（R/W）	
0x14	数据寄存器 4（R/W）	
0x15	数据寄存器 5（R/W）	
0x16	数据寄存器 6（R/W）	

17 PWM 控制器

17.1 概述

2K1500 芯片里实现了 6 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 50MHz，计数寄存器和参考寄存器均 32 位数据宽度。

17.2 访问地址及引脚复用

PWM 控制器访问基址参考第 13 章，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 0x0-0x5 分别代表 PWM0-PWM5
[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。

17.3 寄存器描述

每路控制器共有五个寄存器，具体描述如下：

表 17- 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 17- 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

17.4 功能说明

17.4.1 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。

17.4.2 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。

17.4.3 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）：

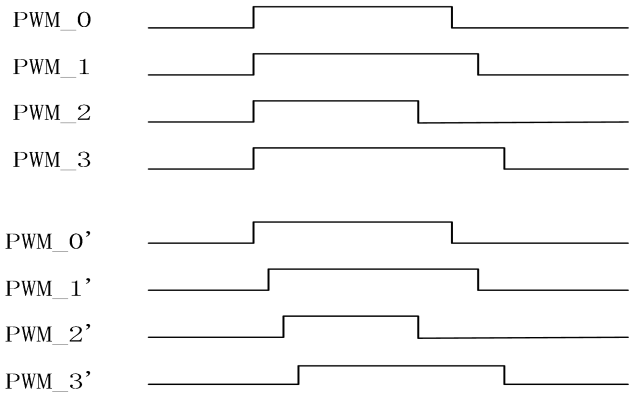


图 17- 1 防死区功能

18 HPET 控制器

18.1 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（50MHz）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。

18.2 访问地址

HPET 模块的访问基地址为 MISC 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

18.3 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。

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。这个值必须大于 0，且小于或等于 0x05F5E100（100ns，即 10MHz）	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，2K1500 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

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

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。

Main Counter Value Register

位	名称	描述	读写特性
63：32	Reserved		
31：0	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

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	R/W

		作使能相应的定时器产生周期性中断。 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		

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

19 GPIO

龙芯 2K1500 共有 96 个 GPIO 引脚，12 个为专用 GPIO，其余 84 个与其他功能复用。96 个 GPIO 中 32 个为 node 上的 GPIO，其余 64 个为南桥上的 GPIO，两种 GPIO 的访问地址有所不同，下面分别介绍。

19.1 NODE GPIO 控制

node 提供最多 32 个 GPIO 供系统使用，绝大部分都与其它功能复用。通过寄存器设置，还可以将 GPIO 配置为中断输入功能，并可以设置其中断电平。

本节各个配置寄存器的基地址为 0x1fe00000。

19.1.1 输出使能寄存器（0x0500）

基地址为 0x1fe00000，偏移地址 0x0500。

表 19- 1 输出使能寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_OEn	RW	32’ hffffffff	GPIO 输出使能（低有效）
63:32	GPIO_FUNC_En	RW	32’ hffff0000	GPIO 功能使能（低有效）

19.1.2 输入输出寄存器（0x0508）

基地址为 0x1fe00000，偏移地址 0x0508。

表 19- 2 输入输出寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_O	RW	32’ h0	GPIO 输出设置
63:32	GPIO_I	RO	32’ h0	GPIO 输入状态

19.1.3 中断控制寄存器（0x0510）

基地址为 0x1fe00000，偏移地址 0x0510。

表 19- 3 中断控制寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_INT_Pol	RW	32’ h0	GPIO 中断有效电平设置 0 - 低电平有效 1 - 高电平有效
63:32	GPIO_INT_en	RW	32’ h0	GPIO 中断使能控制，高有效

19.1.4 GPIO 中断控制

node 中 GPIO 引脚都可以作为中断输入使用。

GPIO00、GPIO08、GPIO16、GPIO24 共享中断控制器的 0 号中断线。

GPIO01、GPIO09、GPIO17、GPIO25 共享中断控制器的 1 号中断线。

GPIO02、GPIO10、GPIO18、GPIO26 共享中断控制器的 2 号中断线。

GPIO03、GPIO11、GPIO19、GPIO27 共享中断控制器的 3 号中断线。

GPIO04、GPIO12、GPIO20、GPIO28 共享中断控制器的 4 号中断线。

GPIO05、GPIO13、GPIO21、GPIO29 共享中断控制器的 5 号中断线。

GPIO06、GPIO14、GPIO22、GPIO30 共享中断控制器的 6 号中断线。

GPIO07、GPIO15、GPIO23、GPIO31 共享中断控制器的 7 号中断线。

每个 GPIO 的中断使能由配置寄存器 GPIO_INT_en 控制，中断电平由 GPIO_INT_POL 控制，寄存器如下：

基地址为 0x1fe00000，偏移地址 0x0510。

表 19- 4 中断控制寄存器

位域	字段名	访问	复位值	描述
31:0	GPIO_INT_Pol	RW	32’h0	GPIO 中断有效电平设置 0 - 低电平有效 1 - 高电平有效
63:32	GPIO_INT_en	RW	32’h0	GPIO 中断使能控制，高有效

当中断控制器上的每个中断线只使能其中一位 GPIO 时，可以使用边沿触发方式，固定在某个沿（POL 设为 0 时下降沿，为 1 时上升沿）触发中断并在中断控制器中记录。

19.2 南桥 GPIO 控制

南桥提供最多 64 个 GPIO 供系统使用，绝大部分都与其它功能复用。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 19- 5GPIO 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><th>POL</th><th>EDGE</th><th>描述</th></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>
</table>

GPIO_INT_STS	1	GPIO 中断状态

19.2.1 访问地址

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

南桥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 19- 6 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 19- 7 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。

19.2.2 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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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：4

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

20 watch dog

2K1500 集成一个看门狗，最大定时时间约 82s。

20.1 访问地址

看门狗的访问基地址为 MISC 低速设备块的基地址加偏移 0x50000。

注意：支持按 4/1 字节访问。

看门狗内部寄存器物理地址构成如下：

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

20.2 寄存器描述

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 。		

21 RTC

21.1 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。

21.2 访问地址

RTC 模块的访问基地址为 MISC 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

21.3 寄存器描述

21.3.1 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

21.3.2 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

21.3.3 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

21.3.4 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

21.3.5 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

21.3.6 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 0～16383
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

21.3.7 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

21.3.8 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		

21.3.9 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		

21.3.10 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		

22 NAND 控制器（D7:F0）

22.1 NAND 控制器结构描述

NAND FLASH 控制器最大支持单片 16GB FLASH 的容量，最大页大小 8KB,芯片最多支持 4 个片选和 4 个 RDY 信号，控制器支持 SLC 和 MLC 两种类型 FLASH 的操作。

22.2 访问地址及引脚复用

NAND 控制器内部寄存器的物理地址构成如下：

地址位	构成	备注
[27:09]	BAR_BASE	Device7、FUNC 0 的基地址寄存器值
[08:00]	REG	内部寄存器地址

对于 NAND 模块，使用时要注意将对应的引脚设置为相应的功能。

与 NAND 相关的引脚设置寄存器为 4.1 节中的 nand_sel。

22.3 NAND 寄存器配置描述

NAND 内部的寄存器设置如下：

偏移地址	寄存器名称
0x00	NAND_CMD
0x04	ADDR_C
0x08	ADDR_R
0x0C	NAND_TIMING
0x10	ID_L
0x14	STATUS & ID_H
0x18	NAND_PARAMETER
0x1C	NAND_OP_NUM
0x20	CS_RDY_MAP
0x40	DMA access address

22.3.1 命令寄存器 NAND_CMD（偏移地址 0x00）

位	位域名	读写	描述
31	DMA_REQ	R/-	非 ECC 模式下 NAND 发出 DMA 请求
30	ECC_DMA_REQ	R/-	ECC 模式下 NAND 发出 DMA 请求
29:25	STATUS	R/-	内部状态机（供测试用）
24		R/-	Reserved
23：20	NAND_CE	R/-	外部 NAND 芯片片选情况，四位分别对应片外四个片选，0 表示选中
19：16	NAND_RDY	R/-	外部 NAND 芯片 RDY 情况，对应关系和 NAND_CE 一致，1 表示准备好
15			Reserved
14	wait_ecc	R/W	为 1 表示等待 ECC 读完成（用于 ECC 读）
13	INT_EN	R/W	NAND 中断使能信号，为 1 表示使能中断
12	RS_WR	R/W	为 1 表示写操作时候 ECC 功能开启
11	RS_RD	R/W	为 1 表示读操作时候 ECC 功能开启

10	done	R/W	为 1 表示操作完成，需要软件清零
9	Spare	R/W	为 1 表示操作发生在 NAND 的 SPARE 区
8	Main	R/W	为 1 表示操作发生在 NAND 的 MAIN 区
7	Read status	R/W	为 1 表示读 NAND 的状态操作
6	Reset	R/W	为 1 表示 Nand 复位操作
5	read id	R/W	为 1 表示读 ID 操作
4	blocks erase	R/W	连续擦除标志，缺省 0；1 有效，连续擦擦块的数目由 nand_op_num 决定
3	erase operation	R/W	为 1 表示擦除操作
2	write operation	R/W	为 1 表示写操作
1	read operation	R/W	为 1 表示读操作
0	command valid	R/W	为 1 表示命令有效，操作完成后硬件自动清零

22.3.2 页内偏移地址寄存器 ADDR_C（偏移地址 0x04）

位	位域名	读写	描述
31:14		R/-	Reserved
13:0	Nand_Col	R/W	读、写、擦除操作起始地址页内地址（必须以字对齐，为 4 的倍数），和页大小对应关系如下： 512Bytes：只需要填充[8:0] 2K：需要填充[11:0]，[11]表示 spare 区，[10:0]表示页内偏移地址 4K：需要填充[12:0]，[12]表示 spare 区，[11:0]表示页内偏移地址 8K：需要填充[13:0]，[13]表示 spare 区，[12:0]表示页内偏移地址

22.3.3 页地址寄存器 ADDR_R（偏移地址 0x08）

位	位域名	读写	描述
31:25		R/-	Reserved
24:0	Nand_Row	R/W	读、写、擦除操作起始地址页地址，地址组成如下： {片选，页数} 其中片选固定为 2 位，页数根据实际的单片颗粒容量确定，如 1M 页则为[19:0]，[21:20]用于选择 4 片中的第几片

22.3.4 时序寄存器 NAND_TIMING（偏移地址 0x0C）

位	位域名	读写	描述
31:16		R/-	Reserved
15：8	Hold cycle	R/W	NAND 命令有效需等待的周期数，缺省 4
7：0	Wait cycle	R/W	NAND 一次读写所需总时钟周期数，缺省 18 ECC 模式下配置为 8’hb

22.3.5 ID 寄存器 ID_L（偏移地址 0x10）

位	位域名	读写	描述
31：0	ID[31:0]	R/-	ID[31:0]

22.3.6 ID 和状态寄存器 STATUS & ID_H（偏移地址 0x14）

位	位域名称	访问	
31:24		R/-	Reserved
23:16	STATUS	R/-	NAND 设备当前的读写完成状态

15:0	ID[47:32]	R/-	ID 高 16 位

22.3.7 参数配置寄存器 NAND_PARAMETER（偏移地址 0x18）

位	位域名称	访问	描述
31:30		-	Reserved
29:16	op_scope	R/W	每次能操作的范围，配置如下： 1.操作 main 区，配置为一页的 main 区大小 2.操作 spare 区，配置为一页的 spare 区大小 3.操作 main 加 spare 区,配置为一页的 main 区加上 spare 区大小
15		R/-	Reserved
14:12	ID_number	R/W	ID 号的字节数
11:8	外部颗粒容量大小	R/W	0：1Gb（2K 页） 1：2Gb（2K 页） 2：4Gb（2K 页） 3：8Gb（2K 页） 4：16Gb（4K 页） 5：32Gb（8K 页） 6：64Gb(8K 页) 7：128Gb(8K 页) 9：64Mb(512B 页) a:128Mb(512B 页) b:256Mb(512B 页) c:512Mb(512B 页) d:1Gb(512B 页) (bit)
7:0		R/-	Reserved

22.3.8 操作数量寄存器 NAND_OP_NUM（偏移地址 0x1C）

位	位域名	读写	描述
31:0	NAND_OP_NUM	R/W	NAND 读写操作 Byte 数(非 ECC 模式下必须以字对齐，为 4 的倍数；ECC 模式下，配置成 527*N+2，其中 512*N 为 main 区大小)；擦除为块数

22.3.9 映射寄存器 CS_RDY_MAP（偏移地址 0x20）

NAND 的 4 个 CS 由所访问的地址硬件自动生成，CS0/RDY0 对应最低一块空间，CS1/RDY1 对应次低一块空间，其它以此类推。如果需要调整外部芯片和 NAND 地址的关系，可通过设置本寄存器，对 cs_rdy1/2/3 进行重新映射。

位	位域名	读写	描述
31:28	rdy3_sel	R/W	rdy3 信号从芯片引脚到 NAND 控制器的映射 4’b0001:NAND_RDY[0] 4’b0010:NAND_RDY[1] 4’b0100:NAND_RDY[2] 4’b1000:NAND_RDY[3]
27:24	cs3_sel	R/W	cs3 信号从 NAND 控制器到芯片引脚的映射 4’b0001:NAND_CS[0] 4’b0010:NAND_CS[1] 4’b0100:NAND_CS[2]

			4’b1000:NAND_CS[3]
23:20	rdy2_sel	R/W	rdy2 信号从芯片引脚到 NAND 控制器的映射 4’b0001:NAND_RDY[0] 4’b0010:NAND_RDY[1] 4’b0100:NAND_RDY[2] 4’b1000:NAND_RDY[3]
19:16	cs2_sel	R/W	cs2 信号从 NAND 控制器到芯片引脚的映射 4’b0001:NAND_CS[0] 4’b0010:NAND_CS[1] 4’b0100:NAND_CS[2] 4’b1000:NAND_CS[3]
15:12	rdy1_sel	R/W	rdy1 信号从芯片引脚到 NAND 控制器的映射 4’b0001:NAND_RDY[0] 4’b0010:NAND_RDY[1] 4’b0100:NAND_RDY[2] 4’b1000:NAND_RDY[3]
11:8	cs1_sel	R/W	cs1 信号从 NAND 控制器到芯片引脚的映射 4’b0001:NAND_CS[0] 4’b0010:NAND_CS[1] 4’b0100:NAND_CS[2] 4’b1000:NAND_CS[3]
7:0		R/-	reserved

22.3.10 DMA 读写数据寄存器 DMA_ADDRESS（偏移地址 0x40）

位	位域名	读写	描述
31:0	DMA_ADDRESS	R/W	NAND 读写 NAND flash 数据（ID/STATUS 除外）时候的访问地址，读/写地址相同，读写方向通过 DMA 配置实现

22.4 NAND ADDR 说明

以 2K 页的 NAND flash 为例，定义如下:

每一页的 main 区大小为 2KB，spare 区大小为 64B

main_op = NAND_CMD[8];

spare_op = NAND_CMD[9];

addr_in_page ={ A11, A10.. A2, A1, A0}=ADDR_C

page_number ={ …A30,A29,A28,A27…A13,A12}= ADDR_R

NAND flash 的 main 区总容量计算公式为

容量=$2^{(ADDR\_C-1)}$ * $2^{(ADDR\_R)}$*8bit = 2K*$2^{(ADDR\_R)}$*8bit

NAND 地址空间示例如下表：

	I/O	0	1	2	3	4	5	6	7
Column1	1st Cycle	A0	A1	A2	A3	A4	A5	A6	A7
Column2	2nd Cycle	A8	A9	A10	A11	*L	*L	*L	*L
Row1	3rd Cycle	A12	A13	A14	A15	A16	A17	A18	A19
Row2	4th Cycle	A20	A21	A22	A23	A24	A25	A26	A27
Row3	5th Cycle	A28	A29	A30	A31	A32	A33	….	….

(注：2K 页的 1Gb 容量 NAND flash 对应的 Row 最大值为 A27，发送地址给 NAND flash

时只用发 Column1~2 和 Row1~2，不用发 Row3。配置 NAND 参数时，注意不要配错型号，否则可能会读不出数据并且控制器会死等)

对系统板上 NAND 颗粒来说，如果仅仅操作 spare 区，A11=1 是唯一标志。所以软件配置内部寄存器时，需要配置 A11 和 spare_op 均为 1(见 Examples5)，错误的示例见 Examples2。

对系统板上 NAND 颗粒来说，如果仅仅操作 main 区，A11=0 是唯一标志.；所以软件配置内部寄存器时，需要配置 A11 和 spare_op 均为 0（见 Examples1），错误的示例见 Examples4。

对系统板上 NAND 颗粒来说，如果操作 main+spare 区，A11 可以为 0(见 Examples3)；也可以为 1（见 Examples6）。

Examples1:（非 ECC 模式下。NAND 颗粒中一个 page 的数据只能位于 0x0-0x83f，第一个 op 表示读写开始的数据，接下来的 op 表示随后的读写数据；NO_op 表示不能被本次 NAND 配置读写的数据）

(spare_op =0 & main_op =0) equal to (spare_op =0 & main_op =1); ADDR_C =0x30

Data in a page	0	0x30	….	0x7ff	0x800	0x830	0x83f
Page 0	NO_op	op	op	op	NO_op	NO_op	NO_op
Page 1	op	op	op	op	NO_op	NO_op	NO_op
Page 2	op	op	op	op	NO_op	NO_op	NO_op

Examples2:

spare_op=1 & main_op=0; ADDR_C = 0x30 (配置出错！！开始操作不在 spare 区，下图是可能的错误访问顺序)

Data in a page	0	0x30	….	0x7ff	0x800	0x830	0x83f
Page 0	NO_op	op	op	op	Op	op	op
Page 1	NO_op	NO_op	NO_op	NO_op	Op	op	op
Page 2	NO_op	NO_op	NO_op	NO_op	Op	op	op
Page 3	NO_op	NO_op	NO_op	NO_op	Op	op	op

Examples3:

spare_op = 1 & main_op =1; ADDR_C = 0x30

Data in a page	0	0x30	….	0x7ff	0x800	0x830	0x83f
Page 0	NO_op	op	op	op	op	op	op
Page 1	op	op	op	op	op	op	op
Page 2	op	op	op	op	op	op	op

Examples4:

(spare_op=0 & main_op=0), (equal to spare_op=0 & main_op=1); ADDR_C =0x830:

(配置出错！！开始操作在 spare 区，下图是可能的错误访问顺序)

Data in a page	0	0x30	….	0x7ff	0x800	0x830	0x83f
Page 0	NO_op	NO_op	NO_op	NO_op	NO_op	NO_op	NO_op
Page 1	NO_op	op	op	op	op	NO_op	NO_op
Page 2	op	op	op	op	op	NO_op	NO_op
Page 3	op	op	op	op	op	NO_op	NO_op

Examples5:

spare_op = 1 and main_op =0; ADDR_C = 0x830

Data in a page	0	0x30	….	0x7ff	0x800	0x830	0x83f
Page 0	NO_op	NO_op	NO_op	NO_op	NO_op	op	op
Page 1	NO_op	NO_op	NO_op	NO_op	op	op	op
Page 2	NO_op	NO_op	NO_op	NO_op	op	op	op

Examples6:

spare_op = 1 & main_op =1; ADDR_C = 0x830

Data in a page	0	0x30	….	0x7ff	0x800	0x830	0x83f
Page 0	NO_op	NO_op	NO_op	NO_op	NO_op	op	op
Page 1	op	op	op	op	op	op	op
Page 2	op	op	op	op	op	op	op
Page 3	op	op	op	op	op	op	op

512B 页大小的 NAND flash 和 2KB 页大小配置类似，但在以下几个地方会有不同，需要注意：

每一页的 main 区大小为 512B，spare 区大小为 16B。其中 main 区分为两个 256B 区，每个 256B 区通过 A0~A7 来寻址。读写操作时，通过发送命令 0x00、0x01 和 0x50 来选择是在哪个 256B 区或者 spare 区（软件不必关心，硬件自动选择，如配置 NAND 控制器写 0x100 时，硬件会自动发送到高 256B 区）。

发送地址命令顺序如下：

	I/O	0	1	2	3	4	5	6	7
Column1	1st Cycle	A0	A1	A2	A3	A4	A5	A6	A7
Row1	2nd Cycle	A9	A10	A11	A12	A13	A14	A15	A16
Row2	3rd Cycle	A17	A18	A19	A20	A21	A22	A23	A24
Row3（注）	4th Cycle	A25	A26	*L	*L	*L	*L	*L	*L

（注：Nand flash 容量为 64Mb、128Mb 和 256Mb 时，对应的最大列地址 ADDR_R 分别为 A22、A23 和 A24，只用发送三次地址命令 Column1 和 Row1~2，不用发送 Row3；容量为 512Mb 和 1Gb 时，需要发送 Row3）

4K/8K 页大小的配置和 2K 页配置一样，4K 页的 main 区大小为 4KB，spare 区大小为 128B；8K 页的 main 区大小为 8KB，spare 区大小为 640B。都需要发送五次地址命令。

22.5 NAND-flash 读写操作举例

命令寄存器的‘command valid ’位不能与其他读写使能位同时置位，要先设置好要进行的操作，最后才置‘command valid ’位。

例如：现在要读 Main 区的数据，那么操作分成以下两步：

a、先 NAND_CMD = 0x102

b、再 NAND_CMD = 0x103

22.6 NAND ECC 说明

硬件集成 ECC 功能，ECC 采用 RS(527,512)方法进行编码和解码，即每 512Byte 的有效数据对应 15Byte 的 ECC 编码。在配置软件过程中需要注意以下几点：

1. 每次读写 NAND 的时候，需要每次操作一个 Page，即使原始数据不够一个 Page；
2. 原始数据存储在 main 区，ECC 编码存储在 spare 区。ECC 码从 spare 区的第三个字节开始存储，防止缺陷标记区 defect area 被覆盖，该功能由硬件完成，软件只需按下文要求配置参数即可；
3. NAND 的 op_num 参数与 DMA 控制器的操作数的关系在不同情况下处理方式不一样，具体见下文；
4. ECC 操作和 OOB 操作可以分开，比如对一个页完成 ECC 读/写后可以对其 OOB 进行操作；
5. ECC 模式下，不支持 512B 页大小。

在软件控制上，写操作与读操作流程有些差别，下面给出具体操作步骤：

写操作：

1. 设置 NAND_CMD[12]为 1，表示接下来写操作为 ECC 写
2. 设置 NAND_CMD[8]和 NAND_CMD[9]为 1，表示同时操作 main 区和 spare 区
3. 设置 NAND_OP_NUM 为（main 区大小+main 区数据对应的 ECC+2）
4. 设置 ADDR_C 为 0
5. 设置其他寄存器
6. 设置 NAND_CMD[2]表示写操作
7. 设置 NAND_CMD[0]开始写操作
8. 等待 NAND_CMD[10]完成

以上每步的设置需要保证对应寄存器的其他位保持不变。对应的 DMA 控制器的操作数配置为 main 区大小/4。

读操作：

读操作比写操作复杂一些，其需要两个步骤来完成。第一个步骤先将 spare 区的 ECC

码读到 NAND 控制器的 FIFO 中暂存，具体步骤为：

1. 设置 NAND_CMD[9]和 NAND_CMD[11],表示接下来的操作在 spare 区进行，而且读回来的数据是为 ECC 解码做准备
2. 设置 NAND_OP_NUM 为 ECC 码 byte 数
3. 设置 ADDR_C 为 main 区大小
4. 设置 NAND_CMD[14]
5. 设置其他寄存器
6. 设置 NAND_CMD[1]表示读操作
7. 设置 NAND_CMD[0]开始读操作
8. 等待 NAND_CMD[10]完成

该步骤不需要 DMA 控制器，因此不必启动 DMA 控制器。

第二个步骤，读 main 区数据，NAND 控制器会同步将译码后的数据返回给 DMA 控制器，其步骤为：

1. 设置 NAND_CMD[8],同时设置 NAND_CMD[9]为无效，表示接下来操作在 main 区，同时需保证 NAND_CMD[11]为 1
2. 设置 NAND_OP_NUM 为 main 区大小
3. 设置 ADDR_C 为 0
4. 设置 NAND_CMD[1]表示读操作
5. 设置 NAND_CMD[0]开始读操作
6. 等待 NAND_CMD[10]完成

该步骤中 DMA 控制器的操作数配置为 main 区大小/4。

需注意在整个读操作过程中保持 NAND_CMD[11](rs_rd)为 1，否则 NAND 控制器的 FIFO 会被复位。

可以在 ECC 操作完成后通过普通方式读回所有内容，包括原始数据和 ECC 校验增加的数据（此时配置操作数 op_num 和 DMA 相同）。

校验能力说明：最多可以纠错 6 个 10bit 单元，这些 10bit 单元内部出错的位数可以是 1-10 个。10bit 单元指的是对于每个 512Byte 原始数据+15Byte ECC 码数据，从 bit0 开始把每个连续的 10bit 数据作为一个单元来看待。

22.7 支持 NAND 型号

本节列出经过验证的可支持的 NAND 型号，其它类型的 NAND 颗粒未经验证，不保证与本控制器兼容。

- 镁光 L84A_64Gb_128Gb_256Gb_AsyncSync_NAND：可用于存储
- 三星 K9F1G08U0C：可用于存储

23 加解密

23.1 DES（D29:F1）

23.1.1 DES 功能概述

DES 控制器采用 32 位的 APB 接口，支持使用 DES/TDEA 算法进行加/解密，并支持使用 APB 接口进行 DMA 操作。DES 控制器采用了 OpenCore 的面积节约方案的 DES3 加/解密单元作为实现加/解密的运算单元。这个运算单元每 16 个时钟周期完成一次 DES 加/解密，每 48 个时钟周期完成一次 TDEA 加/解密。为了减少加/解密的等待时间，DES 控制器使用 2 个时钟域分别处理 APB 接口的操作和进行 DES 加/解密计算(DES 加解密使用的时钟的频率更高)。两个不同的时钟域通过 2 个 4 项的 64bit 位宽异步 FIFO 交换加/解密运算前后的数据。

DES 控制器在使用前需要先配置密钥以及 Command 寄存器。控制器有 3 个使用 64bit 格式存储的密钥：Key0、Key1、Key2。其中，Key1、Key2 仅在 TDEA 算法是需要进行配置。控制器不检查密钥中的校验位。待运算（加密还是解密由 Command[1]的值确定）的数据通过 Data_low 和 Data_high 写入运算数据 FIFO。DES 运算模块从运算数据 FIFO 读取数据后进行加/解密运算迭代，迭代的结果被送入运算结果 FIFO。通过 APB 接口查询 Command[4]可以获知运算结果是否就绪。当 Command[4]的值为 0 时，可以从 APB 接口通过 Data_low 和 Data_high 读取运算结果 FIFO 中的运算结果。

23.1.2 DES 访问地址：

DES 的 PCI 设备编号为(dev29,func1)，可以通过配置总线设置 DES 寄存器的基地址。物理地址的低 6 位作为偏移访问配置寄存器。

23.1.3 DES 寄存器描述

DES 的寄存器列表及寄存器说明如下：

地址偏移	名称	说明
0x0	Key0_low	DES 密钥的低 32 位/TDEA 密钥 0 的低 32 位
0x4	Key0_high	DES 密钥的高 32 位/TDEA 密钥 0 的高 32 位
0x8	Key1_low	TDEA 密钥 1 的低 32 位
0xc	Key1_high	TDEA 密钥 1 的高 32 位
0x10	Key2_low	TDEA 密钥 2 的低 32 位
0x14	Key2_high	TDEA 密钥 2 的高 32 位
0x18	Data_low	非 DMA 模式（Command[2] = 1’b0）：待加/解密数据低 32 位的写入端口；加/解密后数据低 32 位的读出端口。 DMA 模式（Command[2] = 1’b1）：Data_low 和 Data_high 不做

		区分。先写入/读取的是数据的低 32 位，后写入/读取的是数据的高 32 位。
0x1c	Data_high	非 DMA 模式（Command[2] = 1’b0）：待加/解密数据高 32 位的写入端口；加/解密后数据高 32 位的读出端口。 DMA 模式（Command[2] = 1’b1）：Data_low 和 Data_high 不做区分。先写入/读取的是数据的低 32 位，后写入/读取的是数据的高 32 位。
0x20	Command	命令和状态控制寄存器
0x24	Rev	保留
0x28		
0x2c		

Comand 寄存器位域说明：

位域	复位值	名称	属性	说明
0	0	des3	RW	0：使用 DES 算法 1：使用 TDEA 算法
1	0	decrypt	RW	0：加密操作 1：解密操作
2	0	dma_start	RW	写入 1 启动 DMA 操作，写入 0 无影响 在 DMA 操作完成前此位保持为 1 当 DMA 操作完成时此位自动清零
3	0	dma_done	RW1C	当 DMA 操作完成时，此位置 1 向此位写入 1，则清零
4	1	dout_empty	RO	0：数据读 FIFO 非空 1：数据读 FIFO 为空
5	0	din_full	RO	0：数据写 FIFO 未满 1：数据写 FIFO 已满
7:6	0	Rev	RO	保留
31:8	0	dma_count	RW	需要进行 DMA 加/解密的 64 位数的个数

23.2 AES（D29:F0）

23.2.1 AES 功能概述

AES 控制器采用 32 位的 APB 接口，支持使用 128-bit KEY、192-bit KEY、256-bit KEY 进行 AES 算法加/解密，并支持使用 APB 接口进行 DMA 操作。在使用 DMA 功能时，支持 CTR 模式进行加/解密。AES 控制器有 3 个主要模块：KEY 扩展模块、加密模块、解密模块。加/解密运算模块每 11 个时钟周期完成一次 128-bit KEY 的 AES 加/解密，每 13 个时钟周期完成一次 192-bit KEY 的 AES 加/解密，每 15 个时钟周期完成一次 256-bit KEY 的 AES 加/解

密。为了减少加/解密的等待时间，AES 控制器使用 2 个时钟域分别处理 APB 接口的操作和进行 AES 加/解密计算（AES 加解密使用的时钟的频率更高）。KEY 扩展模块工作在速度较慢的 APB 时钟域。两个不同的时钟域通过 2 个 4 项的 128bit 位宽异步 FIFO 交换加/解密运算前后的数据。

AES 控制器在使用前需要先配置密钥以及 Command 寄存器。AES 的密钥、明文和密文均以小尾端的格式进行存储。控制器使用 8 个 32bit 寄存器以小尾端的格式存储密钥：Key0、Key1、Key2、key3、key4、key5、key6、key7。其中，Key4、Key5 仅在 192-bit KEY 和 256-bit KEY 时需要进行配置；Key6、Key7 仅在 256-bit KEY 时需要进行配置。待运算（加密还是解密由 Command[1]的值确定）的数据通过 4 个 32 位寄存器地址（Data0、Data1、Data2、Data3）写入运算数据 FIFO。AES 运算模块从运算数据 FIFO 读取数据后进行加/解密运算迭代，迭代的结果被送入运算结果 FIFO。通过 APB 接口查询 Command[4]可以获知运算结果是否就绪。当 Command[4]的值为 0 时，可以从 APB 接口通过 Data0、Data1、Data2、Data3 读取运算结果 FIFO 中的运算结果。

23.2.2 AES 访问地址：

AES 的 PCI 设备编号为(dev29,func0)，可以通过配置总线设置 AES 寄存器的基地址。物理地址的低 6 位作为偏移访问配置寄存器。

23.2.3 AES 寄存器描述

AES 的寄存器列表及寄存器说明如下：

地址偏移	名称	说明
0x0	Key0	AES 密钥的 Key[31:0]，以小尾端形式存储的 AES 密钥的最低 32 位
0x4	Key1	AES 密钥的 Key[63:32]
0x8	Key2	AES 密钥的 Key[95:64]
0xc	Key3	AES 密钥的 Key[127:96]，在使用 128-bit Key 时为以小尾端形式存储的 AES 密钥的最高 32 位
0x10	Key4	AES 密钥的 Key[159:128]，仅在使用 192/256-bit 的 Key 时使用
0x14	Key5	AES 密钥的 Key[191:160] ，仅在使用 192/256-bit 的 Key 时使用，在使用 192-bit Key 时为以小尾端形式存储的 AES 密钥的最高 32 位
0x18	Key6	AES 密钥的 Key[223:192] ，仅在使用 256-bit 的 Key 时使用
0x1c	Key7	AES 密钥的 Key[255:224] ，仅在使用 256-bit 的 Key 时使用，在使用 256-bit Key 时为以小尾端形式存储的 AES 密钥的最高 32 位
0x20	Data0	非 DMA 模式(Command[2]=1’b0):待加/解密数据最低 32 位的写入端口和加/解密后数据最低 32 位的读出端口 DMA 模式(Command[2]=1’b1):Data0-Data3 不做区分，数据按照从低到高的小尾端顺序写入或读出
0x24	Data1	非 DMA 模式(Command[2]=1’b0):待加/解密数据 63～32 位的写入端口

		和加/解密后数据高 63～32 位的读出端口 DMA 模式(Command[2]=1’b1):Data0-Data3 不做区分，数据按照从低到高的小尾端顺序写入或读出
0x28	Data2	非 DMA 模式(Command[2]=1’b0):待加/解密数据 95～64 位的写入端口和加/解密后数据高 95～64 位的读出端口 DMA 模式(Command[2]=1’b1):Data0-Data3 不做区分，数据按照从低到高的小尾端顺序写入或读出
0x2c	Data3	非 DMA 模式(Command[2]=1’b0):待加/解密数据的最高 32 位（127～96）的写入端口和加/解密后数据最高 32 位（127～96）的读出端口 DMA 模式(Command[2]=1’b1):Data0-Data3 不做区分，数据按照从低到高的小尾端顺序写入或读出
0x30	Ctr_init_val	使用 CTR 模式时 CTR 计数器的初始值。要使此寄存器的值发生作用需要先配置此寄存器的值，再向 cammand[0]写入 0。
0x34	Command	命令和状态控制寄存器
0x38 0x3c	Rev	保留

Comand 寄存器位域说明：

位域	复位值	名称	属性	说明
0	0	Ctr_mode	RW	0：使用普通的 AES 算法进行加/解密 1：使用 CTR 模式进行 AES 算法的加/解密
1	0	decrypt	RW	0：加密操作 1：解密操作
2	0	dma_start	RW	写入 1 启动 DMA 操作，写入 0 无影响 在 DMA 操作完成前此位保持为 1 当 DMA 操作完成时此位自动清零
3	0	dma_done	RW1C	当 DMA 操作完成时，此位置 1，中断输出信号变为高电平 向此位写入 1，则清零，中断输出信号变为低电平
4	1	dout_empty	RO	0：数据读 FIFO 非空 1：数据读 FIFO 为空
5	0	din_full	RO	0：数据写 FIFO 未满 1：数据写 FIFO 已满
7:6	0	Kr_mode	RW	0：128-bit KEY 1：192-bit KEY 2：256-bit KEY 3：保留
31:8	0	dma_count	RW	需要进行 DMA 加/解密的 128 位数的个数

23.3 RSA（D29:F2）

23.3.1 RSA 访问地址：

RSA 的 PCI 设备编号为(dev29,func2)，可以通过配置总线设置 RSA 寄存器的基地址。物理地址的低 11 位作为偏移访问内部空间。

23.4 RNG（D29:F3）

23.4.1 RNG 访问地址：

RNG 的 PCI 设备编号为(dev29,func33)，可以通过配置总线设置 RNG 的基地址。

直接通过以上地址访问 RNG 会返回一个 32 位随机数。

24 EMMC 控制器（D28:F0）

24.1 功能特性

- 兼容eMMC5.1版本
- 支持eMMC启动
- 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

24.2 寄存器描述

EMMC 控制器的寄存器详细说明如下:

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_CON	0x00	R/W	EMMC 控制寄存器	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
EMMC_PRE	0x04	R/W	EMMC 预分频寄存器	0x1

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_CMD_ARG	0x08	R/W	EMMC 命令参数寄存器	0x0

EMMC_CMD_ARG	位	缺省值	描述
sdi_cmd_arg	31:0	0x0	命令参数

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_CMD_CON	0x0c	R/W	EMMC 命令控制寄存器	0x0

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，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）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_CMD_STA	0x10	RO	EMMC 命令状态寄存器	0x0

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）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_RSP0	0x14	RO	EMMC 命令响应寄存器 0	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_RSP1	0x18	RO	EMMC 命令响应寄存器 1	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_RSP2	0x1c	RO	EMMC 命令响应寄存器 2	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_RSP3	0x20	RO	EMMC 命令响应寄存器 3	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_DTIMER	0x24	R/W	EMMC 命令数据超时寄存器	0x0

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_BSIZE	0x28	R/W	EMMC 块大小寄存器	0x0

EMMC_BSIZE	位	缺省值	描述
Reserved	31:12	0x0	
sdi_bsize	11:0	0x0	块大小值（0~4095）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_DAT_CON	0x2c	R/W	EMMC 数据控制寄存器	0x0

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。

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_DAT_CNT	0x30	R/W	EMMC 数据计数寄存器	0x0

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_DAT_STA	0x34	RO	EMMC 数据状态寄存器	0x0

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。

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_FIFO_STA	0x38	RO	EMMC FIFO 状态寄存器	0x0

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	

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_INT_MASK	0x3c	R/W	EMMC 中断寄存器	0x0

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_DAT	0x40	RO	EMMC 命令数据寄存器	0x0

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）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
EMMC_INT_EN	0x64	R/W	EMMC 中断寄使能存器	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
dll_master_val	0xf0	r	DLL master 锁定值	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
dll_con	0xf4	r/w	DLL 控制寄存器	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
param_delay	0xf8	r/w	DLL 延迟参数寄存器。理想情况下，DLL 一级延迟为 100ps，共 256 级	0x0

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）。

寄存器名称	地址	读/写(R/W)	功能描述	复位值
sdio_emmc_sel	0xfc	r/w	总线模式选择	0x0

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

24.3 DMA 控制器

24.3.1 DMA 控制器结构描述

芯片中包含 1 个 DMA 控制器，用来实现内存与 EMMC 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。

24.3.2 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 EMMC 设备，该寄存器指定了读内存的地址；内存写：从 EMMC 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 EMMC 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

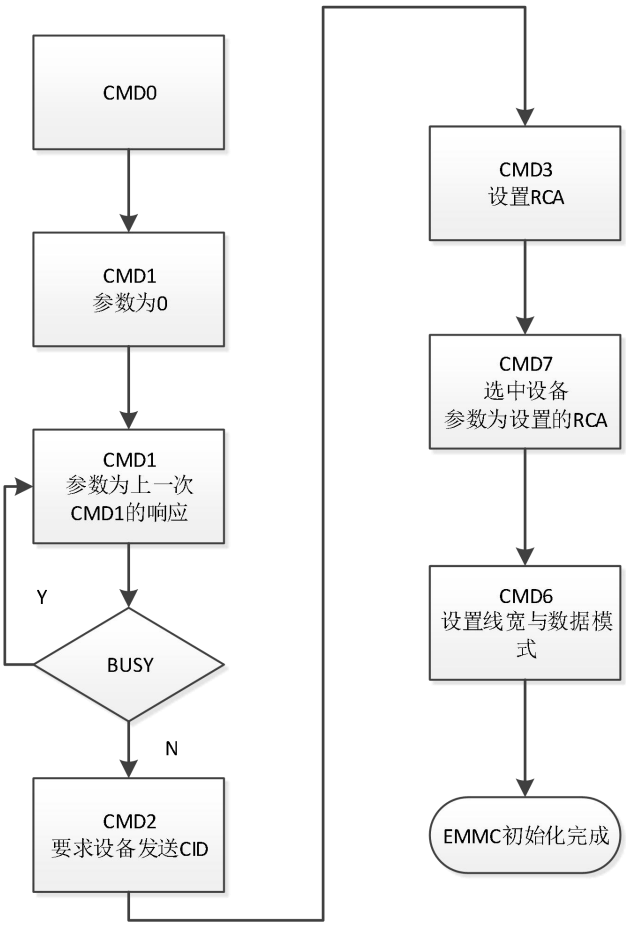
位域	位域名称	位宽	访问	描述
31:0	dma_saddr	32	R/W	DMA 操作的内存地址(高 32 位)

24.4 软件配置流程

24.4.1 EMMC 正常读写软件配置流程

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：EMMC 控制器基址加上 0x800 为 DMA 读，基址加上 0x400 为 DMA 写）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。

24.4.2 EMMC 初始化流程



24.4.3 DDR 模式

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

25 SDIO 控制器（D28:F1）

25.1 功能概述

龙芯 2K1500 集成了一个 SDIO 控制器，用于 SD Memory 和 SDIO 卡的读写。SDIO 控制器特性如下：

- 兼容SD 存储卡规格（4.0版本）
- 兼容SDIO卡规格（4.0版本）
- 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

25.2 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。

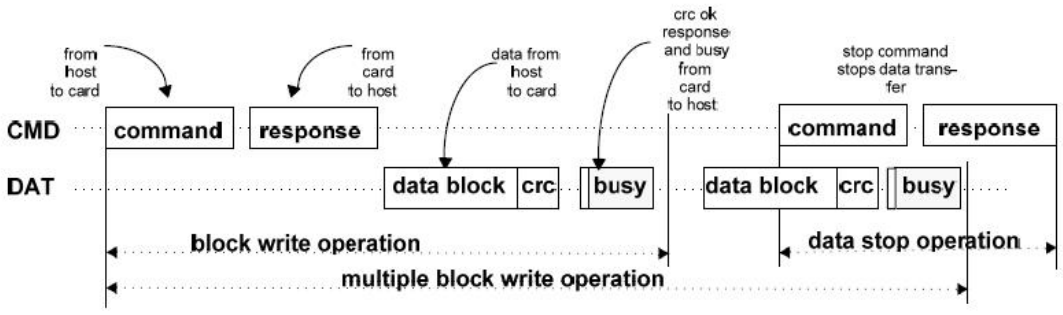


图 25- 1 SD 卡多块写操作示意图

多块读操作的过程和多块写操作的过程类似（等补充说明）。

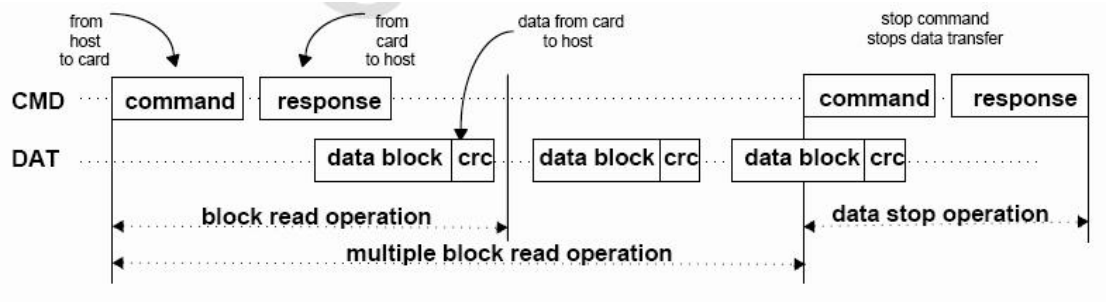


图 25- 2 SD 卡多块读操作示意图

不同的命令都有统一的格式。一般命令的格式如下表，1bit 起始位，1bit 传输方向位，6bit 命令序号，32bit 命令参数，7bit CRC 检验位再加上 1bit 结束位。

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	'0'	'1'	x	x	x	'1'
Description	start bit	transmission bit	command index	argument	CRC7	end bit

其中 command index 对应命令的序号，比如命令 0，cmdindex 为 0，命令 55，cmdindex 为 37。不同命令的参数可能不同，具体参考 SD 协议规范。

25.3 寄存器描述

SDIO 控制器的寄存器详细说明如下:

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_CON	0x00	R/W	SDIO 控制寄存器	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
SDI_PRE	0x04	R/W	SDIO 预分频寄存器	0x1

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_CMD_ARG	0x08	R/W	SDIO 命令参数寄存器	0x0

SDI_CMD_ARG	位	缺省值	描述
sdi_cmd_arg	31:0	0x0	命令参数

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_CMD_CON	0x0c	R/W	SDIO 命令控制寄存器	0x0

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）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_CMD_STA	0x10	RO	SDIO 命令状态寄存器	0x0

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）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_RSP0	0x14	RO	SDIO 命令响应寄存器 0	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_RSP1	0x18	RO	SDIO 命令响应寄存器 1	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_RSP2	0x1c	RO	SDIO 命令响应寄存器 2	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_RSP3	0x20	RO	SDIO 命令响应寄存器 3	0x0

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]

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_DTIMER	0x24	R/W	SDIO 命令数据超时寄存器	0x0

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_BSIZE	0x28	R/W	SDIO 块大小寄存器	0x0

SDI_BSIZE	位	缺省值	描述
Reserved	31:12	0x0	
sdi_bsize	11:0	0x0	块大小值（0~4095）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_DAT_CON	0x2c	R/W	SDIO 数据控制寄存器	0x0

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。

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_DAT_CNT	0x30	R/W	SDIO 数据计数寄存器	0x0

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_DAT_STA	0x34	RO	SDIO 数据状态寄存器	0x0

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。

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_FIFO_STA	0x38	RO	SDIO FIFO 状态寄存器	0x0

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	

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

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_DAT	0x40	RO	SDIO 命令数据寄存器	0x0

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）

寄存器名称	偏移地址	读/写(R/W)	功能描述	复位值
SDI_INT_EN	0x64	R/W	SDIO 中断寄使能存器	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
dll_master_val	0xf0	r	DLL master 锁定值	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
dll_con	0xf4	r/w	DLL 控制寄存器	0x0

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

寄存器名称	地址	读/写(R/W)	功能描述	复位值
param_delay	0xf8	r/w	DLL 延迟参数寄存器。理想情况下，DLL 一级延迟为 100ps，共 256 级	0x0

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）。

寄存器名称	地址	读/写(R/W)	功能描述	复位值
sdio_emmc_sel	0xfc	r/w	总线模式选择	0x0

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

25.4 软件编程指南

25.4.1 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。初始化的流程示意图如下：

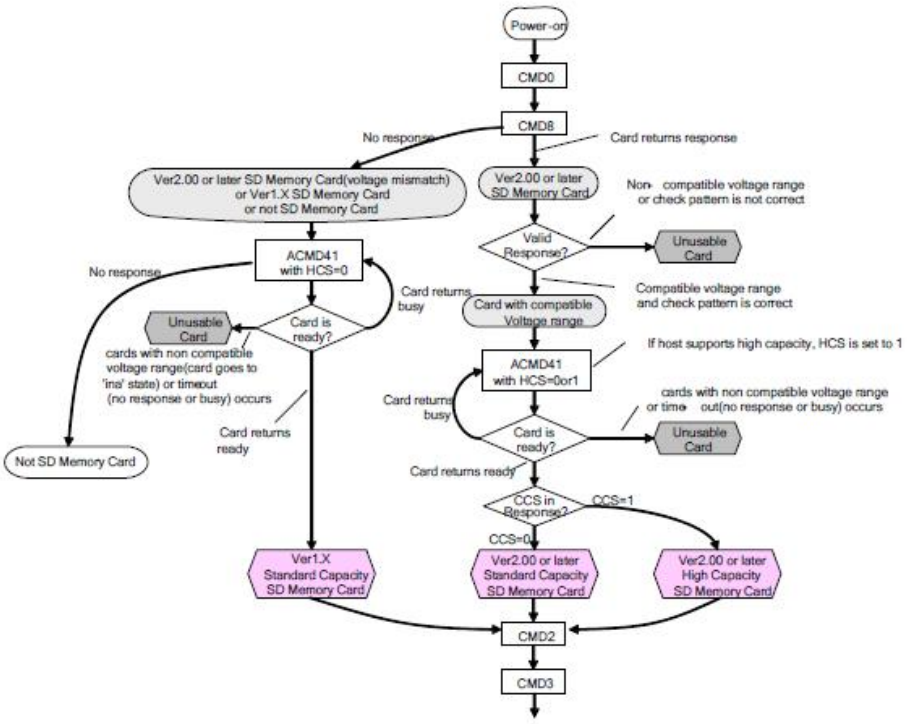


图 25- 3 SD Memory 卡初始化流程示意图

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. 按照上图初始化流程初始化控制器

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参考第 24.3 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

25.4.2 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。

25.4.3 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

25.5 支持 SDIO 型号

本节列出经过验证的可支持的 SDIO 卡型号，其它类型的 SDIO 卡未经验证，不保证与本控制器兼容。

- Mem 卡：Kingston SD-CO2G SDC/2GB
- IO 卡（wifi）：maxwell sd8686

26 GMAC 控制器（D3:F0, D3:F1）

桥片集成了两个 GMAC 控制器，即 GMAC0 和 GMAC1，通过 RGMII 接口连接外置 PHY，分别为 Device 3 的功能 0 和功能 1。

26.1 GMAC配置寄存器（D3:F0, D3:F1）

表 26- 1GMAC 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	7A13h	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

26.2 GMAC 软件编程指南

DMA 初始化

1. 软件重置(reset)GMAC

2. 等待重置完成(查询 DMA reg0[0])
3. 对 DMA reg0 的以下域进行编程
   - MIX-BURST 和 AAL(DMA reg0[26]、[25])
   - Fixed-burst 或者 undefined-burst(DMA reg0[16])
   - Burst-length 和 Burst-mode
   - Descriptor Length(只有当环形格式时有效)
   - Tx 和 Rx 仲裁调度
4. 对 Bus Mode Reg 进行编程。如果选择了 Fixed-burst，则需要在该寄存器内设置最大 burst length
5. 分别创建发送、接收描述符链，可以分别选择环形模式或者链型模式进行连接，并将接收描述符的 OWN 位设为 1(DMA 拥有)
6. 在软件启用 DMA 描述符之前，必须保证至少发送/接收描述符链中有三个描述符
7. 将发送、接收描述符链表的首地址写入 DMA reg3、4
8. 对 DMA reg6(DMA mode operation)中的以下位进行配置
   - 接收/发送的 Store and FoR/Ward
   - 接收/发送的阈值因子(Threshold Control)
   - 启用流控制(hardware flow control enable)
   - 错误帧和未识别的正确帧略过(foR/Warding enable)
   - OSF 模式
9. 向 DMA reg6(Status reg)写 1，清除所有中断请求
10. 向 DMA reg7(interrupt enable reg)写 1，启用所有中断
11. 向 DMA reg6[1]、[13]中写 1，启用发送和接收 DMA

MAC 初始化

1. 正确配置配套 PHY 芯片
2. 对 GMAC reg4(GMII Address Register)进行正确配置，使其能够正常访问 PHY 相关寄存器
3. 读取 GMAC reg5(GMII Data Register)获取当前 PHY 的链接(link)、速度(speed)、模式(双工)等信息
4. 配置 MAC 地址
5. 如果启用了 hash filtering，则需要对 hash filtering 进行配置
6. 对 GMAC reg1(Mac Frame filter)以下域进行配置，来进行帧过滤

- 接收所有
- 混杂模式(promiscuous mode)
- 哈希或完美过滤(hash or perfect filter)

- 组播、多播过滤设置等等

7. 对 GMAC reg6(Flow control register)以下域进行配置

- 暂停时间和其他暂停控制位
- 接收和发送流控制位
- 流控制忙/后压力启用

8. 对中断掩码寄存器(Mac reg15)进行配置
9. 基于之前得到的线路信息(link,speed,mode)对 GMAC reg0 进行正确的配置
10. 设置 GMAC reg0[2]、[3]来启用 GMAC 中的发送、接收模块

发送和接收的一般过程

1. 检测到发送或接收中断后，查寻相应描述符来判断其是否属于主机，并读取描述符中的数据
2. 完成对描述符中数据的读取后，将描述符各位清 0 并设置其 OWN 位，使其继续发送/接收数据
3. 如果当前发送或接收描述符不属于 DMA(OWN=0)，则 DMA 模块会进入挂起状态。当有数据需要被发送或接收时，向 DMA Tx/Rx POLL 寄存器写 1 重新使能 DMA 模块。需要注意的是接收描述符在空闲时应该总是属于 DMA(OWN=1)
4. 发送和接收描述符及对应 buffer 地址的实时信息可以通过查寻 DMA reg18、19、20、21 获得

27 OTG 控制器（D5:F0）

27.1 概述

2K1500 的 OTG 对应 USB2 PHY 的 PORT0，支持特性如下:

支持 HNP 与 SRP 协议；

内嵌 DMA，无需占用处理器带宽即可在 OTG 与外部存储之间移动数据；

在 device 模式下，为高速设备（480Mbps）；

在 host 模式下，支持高速设备（480Mbps）；

在 device 模式下，支持 6 个双向的 endpoint，其中仅有默认的 endpoint0 支持控制传输；

在 device 模式下，最多同时支持 4 个 IN 方向的传输；

在 host 模式下，支持 12 个 channel，且软件可配置每个 channel 的方向；

在 host 模式下，支持 periodic OUT 传输；

27.2 访问地址

OTG 控制器内部寄存器的物理地址构成如下：

地址位	构成	备注
[63:18]	BAR_BASE	Device 5、FUNC 0 的基地址寄存器值
[17:00]	REG	内部寄存器地址

28 USB 控制器（D4:F0）

28.1 总体概述

2K1500 的 USB 端口特性如下：

- USB 2.0 协议
- 兼容 USB 1.1 协议
- 兼容 XHCI 1.1 协议
- 支持 4 个 USB2.0 端口，每个端口都可挂 LS、FS 或 HS 设备
- USB 2.0 为 XHCI 控制器，对应 USB2 PHY 的 PORT1/2/3/4

28.2 XHCI 控制器

28.2.1 XHCI 配置寄存器（D4:F0）

表 28- 1USB-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	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。

PCICMD-PCI 命令寄存器（USB XHCI-D4: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

28.2.2 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

29 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

29.1 SATA 配置寄存器（D8:F0）

表 29- 1SATA 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。

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

29.2 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
0x180	32	P1_CLB	命令列表基地址低 32 位
0x184	32	P1_CLBU	命令列表基地址高 32 位

0x188	32	P1_FB	FIS 基地址低 32 位
0x18c	32	P1_FBU	FIS 基地址高 32 位
0x190	32	P1_IS	中断状态寄存器
0x194	32	P1_IE	中断使能寄存器
0x108	32	P1_CMD	命令寄存器
0x1a0	32	P1_TFD	任务文件数据寄存器
0x1a4	32	P1_SIG	签名寄存器
0x1a8	32	P1_SSTS	SATA 状态寄存器
0x1aC	32	P1_SCTL	SATA 控制寄存器
0x1b0	32	P1_SERR	SATA 错误寄存器
0x1b4	32	P1_SACT	SATA 激活寄存器
0x1b8	32	P1_CI	命令发送寄存器
0x1bC	32	P1_SNTF	SATA 命令通知寄存器
0x1f0	32	P1_DMACR	DMA 控制寄存器
0x1f8	32	P1_PHYCR	PHY 控制寄存器
0x1fC	32	P1s_PHYSR	PHY 状态寄存器

29.3 SATA PHY 初始化流程

SATA PHY 初始化

1. 如果使用片内生成的 25MHz 时钟,则将寄存器 sata_phy_p1_osc_force_ext 设置为 1
2. 撤销 SATA PHY 的复位信号 sata_phy_cfg_reset_n
3. 对 SATA PHY 的内部寄存器进行配置
4. 撤销 SATA PHY 的复位信号 sata_phy_cfg_soft_phy_rstn
5. 等待 sata_phy_ready 变为 0xf
6. 撤销 SATA 控制器的软复位(默认情况下 SATA 控制器处于复位状态)

30 PCIe 控制器（Dev 9/A/B/C/F/10）

龙芯 2K1500 有两个 PCIe 模块：F0、G0。F0 模块既可以作为一个 X4/X2/X1 的 PCIe 端口也可以作为 4 个独立的 X1 PCIe 端口；G0 模块既可以作为一个 X4/X2/X1 的 PCIe 端口也可以作为 2 个独立的 X1 PCIe 端口，作为 X1 端口时，仅 LANE0 和 LANE1 可用，LANE2 和 LANE3 不可用。

F0 模块包含 0～3 号，共 4 个 PCIe 端口。0 号端口可以以 X4/X2/X1 的方式工作，1～3 号端口仅能以 X1 的方式工作。4X1 模式时，F0 的所有 PCIe 端口最高工作速率为 gen2（5Gbps）。F0 只能工作在 RC 模式。

G0 模块包含 0 和 1 号 PCIe 端口。0 号端口可以以 X4/X2/X1 的方式工作，1 号端口仅能以 X1 的方式工作。2X1 模式时，G0 的所有 PCIe 端口最高工作速率为 gen2（5Gbps）。G0 的 0 号端口允许工作在 RC 或 EP 模式。

F0 与 G0 的 0 号端口有内置的 DMA 控制器，可以在内部总线与 PCIe 总线间进行数据搬运。

30.1 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 30- 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 30- 2.

表 30- 2　　PCIe 端口 DID 表

PCI 设备号	描述	设备标识寄存器
D9:F0	PCIe_F0 端口 0	7A49h
D10:F0	PCIe_F0 端口 1	7A39h
D11:F0	PCIe_F0 端口 2	7A39h
D12:F0	PCIe_F0 端口 3	7A39h
D15:F0	PCIe_G0 端口 0（RC）	7A79h
D15:F0	PCIe_G0 端口 0（EP）	7AF9h
D16:F0	PCIe_G0 端口 1	7A39h

30.2　地址空间划分

芯片中的 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 SOC 中的独立设备，每个端口都包含一个 PCIe 配置头。当 F0、G0 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 8KB。其中低 4KB 通过将配置请求的 func 设为 0 来访问，用于访问标准配置头；高 4KB 通过将配置请求的 func 设为 1 来访问，用于改写标准配置头中的一些只读寄存器。当控制器作为 EP 使用时，来自外部 PCIe 总线的所有设备号为 0 的 TYPE0 访问被用于该 EP 端口的 PCIe header。

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。当控制器作为 EP 使用时，来自外部 PCIe 总线的所有设备号非 0 的 TYPE0 访问被用于通过芯片内部的互连网络访问芯片内部的各功能接口的 PCI header；所有来自外部 PCIe 总线的 TYPE1 访问则依据其总线号通过内部互连来访问芯片的其它作为 RC 的 PCIe 端口。

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。当控制器作为 EP 使用时，来自外部 PCIe 总线的所有 MEM 访问将通过芯片内部的互连网络直接访问芯片内部的资源。

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。当控制器作为 EP 使用时，来自外部 PCIe 总线的所有 IO 访问将通过芯片内部的互连网络直接访问芯片内部的资源。

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。

30.3 内部寄存器定义

芯片的每个 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：0xc8ff0044

Bits	复位值	属性	名字	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	复位后应置为 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
23:16	0xff	R/W	Reserved	保留
24	0	R/W	at_en	PCIe 端口入站（PCIe 到内部总线）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
26	0	R/W	read_pass_write_en	内部总线读写顺序控制 0：PCIe 控制器在内部总线上发起的请求中，读请求不允许越过写请求 1：PCIe 控制器在内部总线上发起的请求中，读请求允许越过写请求
27	1	RO	Reserved	此位的值禁止软件进行改写
31:28	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	名字	描述
0	0	R/W1P	App_unlock_msg	在 PCIe 端口上发送解锁消息
1	0	R/W1P	Apps_pm_xmt_turnoff	在 PCIe 端口上发送 PM_Turn_Off 消息
2	0	R/W1P	App_init_rst	在 PCIe 端口上发送热复位消息
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
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	名字	描述
1:0	Cfg_pwr_ind	系统电源状态指示 00b: Reserved 01b: On 10b: Blink 11b: Off
3:2	Cfg_atten_ind	Attention 按钮状态指示 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
13:11	Cfg_max_payload_size	最大数据负载
14	Cfg_rcb	RCB 位
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 31:21 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	名字	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	Rtlh_rfc_upd	数据链路层收到流控包指示
8	Cfg_eml_control	
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
31:30	Reserved	保留

用户定义消息 ID 寄存器

偏移地址：0x10

Bits	名字	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	名字	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	名字	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		Reserved
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
29	edma_int	内置 DMA 控制器中断 仅端口 0（X4 控制器）使用
31：30		Reserved

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

默认值：0x2bffffff

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：　0x2

Bits	名字	属性	描述
31:4	Reserved	RO	保留
3	Ep_otrans_en	RW	此位为 1 表示在 EP 模式下开启内部总线地址 PCIe 到地址的转换功能
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	Reserved	RW	保留

PCIe 端口控制和状态寄存器

偏移地址：0x28

Bits	复位值	属性	名字	描述
0	0	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）
7:1				Reserved
8	0	R/W	ssc_en	让 PHY 进入 SSC 模式
15:9				Reserved
16	0	R/W	aux_clk_en	让 PHY 进入 SSC 模式 PCIe 端口辅助时钟使能
25:17				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	名字	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	名字	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	名字	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。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	名字	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。

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 PCIe 入站（PCIe 到内部总线）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 PCIe 入站（PCIe 到内部总线）地址转换的转换地址的高 32 位。

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。

EP 模式地址译码掩码寄存器 0

偏移地址：0x80

用于存储此 PCIe 端口作为 EP 时从内部总线地址到 PCIe 地址转换的地址掩码的低 32 位。

EP 模式地址译码掩码寄存器 1

偏移地址：0x84
R/W

用于存储此 PCIe 端口作为 EP 时从内部总线地址到 PCIe 地址转换的地址掩码的高 32 位。

EP 模式地址译码转换地址寄存器 0

偏移地址：0x88
R/W

用于存储此 PCIe 端口作为 EP 时从内部总线地址到 PCIe 地址转换的地址的低 32 位。

EP 模式地址译码转换地址寄存器 1

偏移地址：0x8c
R/W

用于存储此 PCIe 端口作为 EP 时从内部总线地址到 PCIe 地址转换的地址的高 32 位。

30.4 PCIe 配置头空间

芯片的每个 PCIe 端口都有自己的 PCIe 配置头空间。每个端口的 PCIe 配置头空间都有自己的基址，该基址通过芯片配置空间基址、BUS 号（内部总线 BUS 号为 0）以及设备号来确定。

30.5 软件编程指南

PCIe 控制器在作为 RC 使用时，在使用前需要经过以下几个步骤的初始化过程：

1. 参考时钟准备
2. 控制器使能
3. PHY 参数初始化
4. 链路建立初始化

参考时钟准备

芯片的 PCIe 控制器使用了由芯片内部系统时钟为芯片内部 PCIe 控制器以及外部的 PCIe 设备提供同源参考时钟的参考时钟方案。在使用 PCIe 控制器前，需要首先确保参考时钟的选择与所用的板卡环境一致。F0 的参考时钟源由 5.1 小节的通用配置寄存器 0 的 bit2 控制；G0 的参考时钟源由 5.1 小节的通用配置寄存器 0 的 bit15 控制。

PCIe 控制器使能

使能 PCIe 控制器的的操作包含下列步骤：

1. 设置 PCIe 控制器所属模块的工作模式
2. 操作芯片配置寄存器 0，对 PCIe 控制器模块进行复位和使能

下面的伪代码给出了完成上述 2 个步骤的操作过程

```
void pcie_enable(unsigned int port_num, unsigned int cfg_reg1_offset, unsigned int soft_reset_offset, unsigned int enable_offset)
{
    volatile unsigned int read_data;
read_data = *((volatile unsigned long *)(CONFBUS_BASE_ADDR + cfg_reg1_offset));
    if(port_num == 1){//set to max lane num mode
       *((volatile unsigned long *)(CONFBUS_BASE_ADDR + cfg_reg1_offset)) = read_data | (0x3 << 26);
    } else {
       *((volatile unsigned long *)(CONFBUS_BASE_ADDR + cfg_reg1_offset)) = read_data | (0x1 << 27);
    }
    printf("set soft reset\n");
    read_data = *((volatile unsigned int *)(CONFBUS_BASE_ADDR + 0x420));
    *((volatile unsigned int *)(CONFBUS_BASE_ADDR + 0x420)) = read_data | (0x1 << soft_reset_offset);
    printf("release soft reset\n");
    read_data = *((volatile unsigned int *)(CONFBUS_BASE_ADDR + 0x420));
```

```
        *((volatile unsigned int *)(CONFBUS_BASE_ADDR + 0x420)) = read_data & (~(0x1 <<
soft_reset_offset));
        printf("set pcie enable\n");
        read_data = *((volatile unsigned int *)(CONFBUS_BASE_ADDR + 0x420));
        *((volatile unsigned int *)(CONFBUS_BASE_ADDR + 0x420)) = read_data | (0x1 <<
enable_offset);
    }
```

上面伪代码中函数包含 4 个参数。参数 port_num 为 1，表示这个 PCIe 模块只使用 port0，且 port0 工作在最大 lane 宽度上；否则这个 PCIe 模块工作在多个 port 同时使用的模式下（F0 为 4X1、G0 为 2X1）。参数 cfg_reg1_offset 为这个 PCIe 模块的配置寄存器 1 在芯片配置寄存器中的地址偏移。参数 soft_reset_offset 为这个 PCIe 模块的软件复位控制位在桥芯片的通用配置寄存器 0 中的位偏移。参数 enable_offset 为这个 PCIe 模块的使能控制位在桥芯片的通用配置寄存器 0 中的位偏移。

PHY 参数初始化

在 RC 模式下，PCIe 控制器在使用前需要完成对 PHY 的参数配置。完成对 PHY 进行参数配置后才能释放对 PHY 的 PLL 的复位信号，让 PCIe 控制器的复位能够完成。PHY 的参数初始化流程如下：

1. 等待 PHY 的内建初始化配置过程结束。
2. 配置 PHY 的内部寄存器，按照需要，修订 PHY 和 PIPE 接口的工作参数。
3. 将控制器所属的 PHY 控制寄存器中的 phy_soft_cfg_done 置 1，释放 PHY 的 PLL 的复位信号。
4. 查询芯片配置寄存器中的通用配置寄存器 0 中控制器对应的时钟就绪状态，直到对应时钟就绪。
5. PHY 的初始化配置结束。

PCIe 配置头访问

基于前文对配置请求的介绍，对 PCIe 配置头的访问为 Type0 的访问，其格式为：

	39 32	31 28	27 24	23 16	15 11	10 8	7 0
Type 0	FE0h	Offset[11:8]	Reserved	Device Number	Function Number		Offset[7:0]

PCIe 各个 Port 的设备号（device number）分别为 0x9，0xa，0xb，0xc， 0xf，0x10。根据设备号及所要访问的寄存器偏移地址(offset)即可得到对应寄存器的物理地址。功能号 Function Number 为 0 时访问配置头内寄存器，功能号 Function Number 为 1 时可以访问配置头空间内影子寄存器，用于改写配置头内的只读寄存器。

PCIe 链路建立(Linkup)

链路建立流程如下：

1. 设置链路目标速度。
2. 依据是否要进行 gen3 的链路均衡参数训练，设置是否关闭 gen3 的参数训练。
3. 配置访问设置 Gen2 Control Register 寄存器中（0x80c）Directed Speed Change 为 1，PHY Tx Swing 为 0。注意配置地址格式，因为 offset 大于 8 位地址，需将高 4 位填到地址的 bit24-bit27。
4. 如果使用 gen3 速率，在 SPCIe 寄存器中为每个 lane 设置初始的 PRESET 值（这些 PRESET 值从 header 的地址偏移 0x14c 开始存储）。
5. 根据 BAR0 中配的地址通过 MEM 访问设置 PCIe 控制器内部寄存器 app_ltssm_enable(0x0)为 1，开始 link training 过程。
6. 等待链路速率达到设定的速率。
7. 等待内部寄存器 Xmlh_ltssm_state(0xc)为 0x11。
8. Linkup 成功。

上述步骤 1~4 之间没有顺序要求。

TYPE1 类型配置访问

TYPE1 类型配置请求地址格式如下：

	39 32	31 28	27 24	23 16	15 11	10 8	7 0
Type 1	FE1h		Offset[11:8]	Bus Number	Device Number	Function Number	Offset[7:0]

发送 TYPE1 类型配置访问之前需要设置配置头的 Primary Bus Number、Secondary Bus Numer 和 Subordinate Bus Number。然后直接按照 TYPE1 地址格式发送读写请求即可，PCIe 控制器根据地址中的 Bus Number 与所配置的 Secondary Bus Numer 和 Subordinate Bus Number 决定发出 TYPE0 类型还是 TYPE1 类型的配置请求。

如果 Bus Number==Secondary Bus Numer，则发出 TYPE0 类型的配置请求。

如果 Bus Number> Secondary Bus Numer 并且 Bus Number <= Subordinate Bus Number，则发出 TYPE1 类型的配置请求。

30.6 常用例程

本节给出龙芯 2K1500 的 PCIe 控制器的常用例程。需要先执行下面示例中的 pcie_link_init，再执行下面示例中的 pcie_header_init。随后，芯片可以通过 cfg_device_read 和 cfg_device_write 这两个函数对下游设备的 PCI Header 进行初始化。

miphy_read_reg 、miphy_write_reg 是用于对 PHY 内部的寄存器进行读、写操作的函数。其第一个参数 base 是要访问的 PHY 的内部访问端口在芯片配置寄存器中的地址偏移；其第二个参数是要访问的 PHY 内部寄存器在 PHY 内部的地址偏移。

```
#define FUNC_OFFSET 8
#define DEV_OFFSET  11
```

```
#define BUS_OFFSET  16
#define REG_HBIT_OFFSET 24
#define PBUS_NUM 0
#define HT_BASE                 0x9000000000000000
#define CONFBUS_BASE_ADDR 0x90000fdfe010800
#define TYPE0_CFGBASE  0xfe00000000
#define TYPE1_CFGBASE  0xfe10000000
#define IO_BASE          0xfdfc000000
#define REG_HEADER(BASE, BUS_NUM, DEV_NUM, FUNC_NUM, REG_HBIT, REG_LBIT) \
        *((volatile unsigned int *)(HT_BASE + BASE + \
                                    (BUS_NUM  << BUS_OFFSET)  + \
                                    (DEV_NUM  << DEV_OFFSET)  + \
                                    (FUNC_NUM << FUNC_OFFSET) + \
                                    (REG_HBIT << REG_HBIT_OFFSET) + \
                                     REG_LBIT))
void miphy_write_reg(unsigned int base, unsigned int offset, unsigned int data)
{
    *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base)) = data;
    *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base + 0x4)) = (offset&0xfffff) |
(0x1<<25) | (0x1<<24) | (0x1<<26) | (0x1<<27);
    *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base + 0x4)) = (offset&0xfffff) |
(0x1<<25) | (0x1<<24) | (0x1<<26) | (0x0<<27);
    unsigned int cfg_read_data;
    do{
        cfg_read_data = *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base + 0x4));
    }while(!((cfg_read_data>>28)&0x1));
}

unsigned int miphy_read_reg(unsigned int base, unsigned int offset)
{
    *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base + 0x4)) = (offset&0xfffff) |
(0x1<<25) | (0x1<<24) | (0x0<<26) | (0x1<<27);
    *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base + 0x4)) = (offset&0xfffff) |
(0x1<<25) | (0x1<<24) | (0x0<<26) | (0x0<<27);
    unsigned int cfg_read_data;
    do{
        cfg_read_data = *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base + 0x4));
    }while(!((cfg_read_data>>28)&0x1));
    cfg_read_data = *((volatile unsigned int *)(CONFBUS_BASE_ADDR + base));
    return cfg_read_data;
```

```
    }
void wait_speedchange(unsigned int dev_num, unsigned int tgt_spd){
    volatile unsigned int read_data;
    unsigned int spd_val;
    if(tgt_spd == 0x1)//gen1 speed
        spd_val = 0x10000;
    else if(tgt_spd == 0x2)//gen2 speed
        spd_val = 0x20000;
    else if(tgt_spd == 0x3)//gen3 speed
        spd_val = 0x30000;
    else
        spd_val = 0x0;
    //linkstatus register
    read_data = REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 0, 0x80);
    while((read_data&0xf0000)!=spd_val)
    {//link status register
        read_data = REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 0, 0x80);
    }
}

void wait_linkup(unsigned int bar0base){
    volatile unsigned int read_data;
    read_data = *((volatile unsigned int *)(HT_BASE + bar0base +
0xc));//xmlh_ltssm_state
    while((read_data&0x1f)!=0x11)
    {
        read_data = *((volatile unsigned int *)(HT_BASE + bar0base + 0xc));
    }
}

void pcie_link_init(unsigned long bar0base,unsigned int dev_num, unsigned int
tgt_spd , unsigned int start_preset, unsigned int do_eq23, unsigned int lane_cnt)
{
    volatile unsigned int read_data;
    //set target speed
    read_data = REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 0, 0xa0);
    REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 0, 0xa0) =
(read_data&~0xf|tgt_spd);
    if(do_eq23 == 0x0){//disable EQ23
      read_data = REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 8, 0x90);
      REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 8, 0x90) = (read_data |
0x200);
    }
    read_data = (start_preset << 16)|start_preset;
    for(i = 0; i < lane_cnt/2; i = i+1){//set start PRESET
      REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 1, (0x4c + i*4)) = read_data;
    }
    //direct speed change and config PHY Tx swing to full swing
    read_data = REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 8, 0xc);
    REG_HEADER(TYPE0_CFGBASE, PBUS_NUM, dev_num, 0, 8, 0xc) = (read_data|0x20000);
```

```
    //start link
    read_data = *((volatile unsigned int *)(HT_BASE + bar0base));
    *((volatile unsigned int *)(HT_BASE + bar0base)) = read_data|0x8;
    wait_speedchange(dev_num, tgt_spd);
    printf("pcie link speed is change to target speed %d\n", tgt_spd);
    wait_linkup(bar0base);
    printf("pcie link is up\n");
}

void pcie_hot_reset(unsigned int bar0base)
{
    unsigned long pcie_reg_base = HT_BASE + bar0base;
    tmp_var = *(volatile unsigned int *)( pcie_reg_base);
    //enable soft reset
    *(volatile unsigned int *)( pcie_reg_base) = tmp_var |0x8000;
    //triger hot reset
    *(volatile unsigned int *)( pcie_reg_base +0x4) = 0x4;
}

void local_header_init(unsigned int io_base,
                    unsigned int mem_base,
                    unsigned long long pref_mem_base,
                    unsigned int dev_num,
                    unsigned int func_num,
                    unsigned int sub_busnum,
                    unsigned int second_busnum){
    //enable io/mem space, bus master, parity error response and SErr
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x4 ) = 0x147;
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x1c) = 0xf1000000 |
(((io_base >> 12) & 0xf) << 12) | (((io_base >> 12) & 0xf) << 4);//io space
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x30) = ((io_base >> 16) <<
16) | (io_base >> 16);//io limit/base upper 16bit
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x20) = ((mem_base >> 16) <<
16) | (mem_base >> 16);//mem limit/base
    //mem limit/base
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x24) = ((pref_mem_base >> 16)
<< 16) | (pref_mem_base >> 16);
    /pref mem base upper 32bit
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x28) = 0x00000000;/
    //pref mem limit upper 32bit
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x2c) = 0x00000000;
    //bridge control
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x3c) = 0x20000;
    //pme_int_en,sys_err_f_err_en,sys_err_nf_err_en,sys_err_cor_err_en
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x8c) = 0xf;
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 1, 0x2c) = 0x7;
    //set primary_busnum, 2nd_busnum and sub_busnum
    REG_HEADER(TYPE0_CFGBASE, 0, dev_num, func_num, 0, 0x18) = (sub_busnum << 16) |
                                                    (second_busnum << 8) | 0x0;
}
```

```
void cfg_device_read(
unsigned int bus_num,
unsigned int dev_num, unsigned int func_num,
unsigned int reg_id, unsigned int * read_data
)
{
    unsigned int reg_addrh, reg_addrl, reg_addr;
    reg_addr =  reg_id << 2;
    reg_addrh = (reg_addr&0xf00) >> 8;
    reg_addrl = reg_addr&0xff;
    *(read_data) = REG_HEADER(TYPE1_CFGBASE, bus_num, dev_num, func_num,
                              reg_addrh, reg_addrl);
}
void cfg_device_write(
unsigned int bus_num,
unsigned int dev_num, unsigned int func_num,
unsigned int reg_id, unsigned int write_data
)
{
    unsigned int reg_addrh, reg_addrl, reg_addr;
    reg_addr =  reg_id << 2;
    reg_addrh = (reg_addr&0xf00) >> 8;
    reg_addrl = reg_addr&0xff;
    REG_HEADER(TYPE1_CFGBASE, bus_num, dev_num,
                func_num,reg_addrh, reg_addrl)= write_data;
}
```

31 DMA 控制器（D30:F0）

31.1 DMA 控制器功能概述

2K1500 的 DMA 控制器实现数据访问格式可配置的多通道全地址空间的直接内存访问功能，目前可支持同时四通道数据搬运。控制器内部包括版本寄存器、中断使能、中断状态以及复位控制寄存器，可配置控制器的运行频率。各通道还拥有一组配置寄存器，用于配置描述符的地址信息、通道状态控制、停止模式、发送控制等。

描述符存在内存中，用于 DMA 控制器通道和 CPU 的控制交互，每个通道可配置地址连续的多个描述符，以 ring 的形式首尾相接。描述符分别定义源端和目的端的数据访问格式，包括基地址、访问跨步距离（stride）、数据宽度（size）、访问长度（length）、最大并发访问数量（outstanding）、访问总字节数（count）以及是否地址固定（fixed）、是否缓存（Cached）。

DMA 的传送数据的过程由三个阶段组成：

a） 传送前的预处理：将描述符存在内存的指定位置，由 CPU 配置 DMA 描述符相关的寄存器。

b） 数据传送：将数据从读缓冲区搬运到写缓冲区，在 DMA 控制器的控制下自动完成。

c） 传送结束处理：发送中断请求。

31.2 DMA 配置寄存器

DMA 控制器的 PCI 设备编号为(dev30,func0)，可以通过配置总线设置 DMA 控制器寄存器的基地址。物理地址的低 12 位作为偏移访问配置寄存器。

31.2.1 控制器配置寄存器

（1） 版本寄存器（0x0000）

位域	字段名	访问	复位值	描述
7:0	Version	R	8’h10	配置寄存器版本号
19:16	Channels	R	4’h4	DMA 通道数，最多 16 通道
47:32	Freq	RW	16’b0	控制器运行频率，以 MHz 为单位

（2） 中断使能控制寄存器（0x0010）

位域	字段名	访问	复位值	描述
31:0	Int_enable	RW	32’b0	中断使能寄存器，每两位对应一个通道； 两位中的低位为超时中断，高位为描述符中断

（3） 中断状态寄存器（0x0014）

位域	字段名	访问	复位值	描述
31:0	Int_status	R/C	32’b0	中断状态寄存器，每两位对应一个通道

				写 1 清 0

（4） 复位控制寄存器（0x0018）

位域	字段名	访问	复位值	描述
15:0	Channel_reset	RW	32’b0	复位控制，每 1 位对应一个通道 写 1 将各个通道的 this_ptr 复位为 0

31.2.2 通道配置寄存器

每个通道拥有一组配置寄存器，用于表示描述符信息。基地址从 0x100 开始，每个通道偏移增加 0x20，寄存器描述如下：

（1） 基地址寄存器（偏移：0x0000）

位域	字段名	访问	复位值	描述
63:0	Ring_base	RW	64’h0	描述符基地址，必须 16 字节对齐

（2） 指针寄存器（偏移：0x0008）

位域	字段名	访问	复位值	描述
15:0	Last_ptr	RW	16’b0	描述符最大指针，表示了描述符个数（Last_ptr+1）
31:16	This_ptr	R	16’b0	当前描述符指针

（3） 超时中断设置寄存器（偏移：0x000C）

位域	字段名	访问	复位值	描述
0	Overtimer_en	RW	1’b0	使能超时中断
31:16	Overtimer	RW	16’b0	超时时间设置，以 16 拍为单位。在取回一个无效描述符后开始计数，一旦超时就触发中断

（4） 控制寄存器（偏移：0x0010）

位域	字段名	访问	复位值	描述
0	Poll/status	RW	1’b0	启动/状态 写 1 开始读取描述符； 读到 1 表示本通道正在工作，读到 0 表示已经停止
1	Stop_mode	RW	1’b0	停止模式控制 0：根据描述符状态自动停止 1：根据停止指针停止
2	Dma_uncache	RW	1’b0	使用 uncache 方式进行描述符读写
3	Wait_resp	RW	1’b0	等待响应控制 0：描述符操作完成后直接读取下一个描述符 1:等待描述符 owner 写的响应后再读取下一个描述符（建议当描述符数量较少且成 ring 时配置 1）
4	Tx_ask_req*	RW	1’b0	要求发送端提供 req 信号 0：不需要发送端提供 req 信号 1：需要发送端提供 req 信号

5	Rx_ask_req*	RW	1’b0	要求接收端提供 req 信号 0：不需要接收端提供 req 信号 1：需要接收端提供 req 信号
31:16	Stop_ptr	RW	16’b0	停止指针，当 Stop_mode 为 1 时，能够执行的最后一个指针

*仅 CAN 控制器支持此功能

31.3 DMA 描述符

描述符存在内存中，用于 DMA 控制器通道和 CPU 的控制交互，可以配置为链式，也可配置为 ring 的形式首尾相接。每个通道开始运行时，从描述符基址读取描述符，当描述符运行结束后获取下一个描述符。通道指针寄存器的 last_ptr，或者描述符内的 last_des 标记都规定了描述符链（环）的最后一个描述符，当该描述符执行完毕后，通道重新获取第 0 个描述符，this_ptr 清零。停止模式为 0 时，通道一直运行直到取回一个无效的描述符（owner 位为 0）时停止操作；停止模式为 1 时，通道运行至 stop_ptr 指定的位置停下。

每次 DMA 完成一个描述符对应的数据搬运操作后，会向描述符的 Owner 位写 0 表示该描述符已经完成。当通道获取到一个无效描述符时，重新启动 DMA 需要重新对通道控制寄存器执行 poll 操作，直到获取到一个有效的描述符。

每个描述符占 32 个字节，描述符信息表示如下。

（1） DES0（0x0000）

位域	字段名	访问	复位值	描述
15:0	Wait_time	RW	16’b0	每次从读缓冲区请求数据的间隔时间，以 16 拍为单位
16	Int_trigger	RW	1’b0	描述符完成时触发中断使能
17	Last_des	RW	1’b0	表示 ring 的结尾，下一个描述符直接将 ptr 修改为 0 Last_des/Last_ptr 都可以表示 ring 的结束位置，以小的为准
31	Owner	RW	1’b0	描述符状态 0：CPU 未填充 1：DMA 未完成

（2） DES1（0x0004）

位域	字段名	访问	复位值	描述
15:0	TX.stride	RW	16’b0	每次写访问之间的跨步距离-1 （例：配置为 0xffff 代表跨步距离为 0x10000，下同）
31:16	RX.stride	RW	16’b0	每次读访问之间的跨步距离-1

（3） DES2（0x0008）

位域	字段名	访问	复位值	描述
0	TX.fixed	RW	1’b0	写地址固定，不进行累加
1	TX.cached	RW	1’b0	写地址为 Cached，维护一致性

				当设置为 Cached 时，会自动合并处理
7:4	TX.size	RW	4’b0	每次写时的 size（应该与地址对齐） 0：1 字节；1：2 字节；2：4 字节；3：8 字节； 4：16 字节；其他：保留
11:8	TX.length	RW	4’b0	每次读时的 length，当 size 为 4 时有效，其他情况均为 0 0：1 拍；1：2 拍；2：3 拍；3：4 拍；其他：保留
15:12	TX.outstanding	RW	4’h0	最大并发访问数量 0：1 个；1：2 个；2：3 个；3：4 个；其他：保留
31:16	TX.count	RW	16’b0	以 size 和 length 为单位的访问总字节数-1

（4） DES3（0x000C）

位域	字段名	访问	复位值	描述
0	RX.fixed	RW	1’b0	读地址固定，不进行累加
1	RX.cached	RW	1’b0	读地址为 Cached，维护一致性 当设置为 Cached 时，会自动合并处理
7:4	RX.size	RW	4’b0	每次读时的 size（应该与地址对齐） 0：1 字节；1：2 字节；2：4 字节；3：8 字节； 4：16 字节；其他：保留
11:8	RX.length	RW	4’b0	每次读时的 length，当 size 为 4 时有效，其他情况均为 0 0：1 拍；1：2 拍；2：3 拍；3：4 拍；其他：保留
15:12	RX.outstanding	RW	4’h0	最大并发访问数量 0：1 个；1：2 个；2：3 个；3：4 个；其他：保留
31:16	RX.count	RW	16’b0	以 size 和 length 为单位的访问总字节数-1

（5） DES4（0x0010）

位域	字段名	访问	复位值	描述
63:0	TX.addr	RW	64’b0	写缓冲区基地址

（6） DES6（0x0018）

位域	字段名	访问	复位值	描述
63:0	RX.addr	RW	64’b0	读缓冲区基地址

31.4 描述符配置约束和说明

（1）缓冲区基地址根据 size 向下对齐。如当 size 为 4 时，一次访问 16 个字节，基地址后四位强制为 0；

（2）一次访问请求是指由描述符内 length 和 size 指定规格的数据读或者数据写访问，单次访问请求字节数为 length<<size；

（3）总访问字节数（count+1）根据 length << size 进行对齐，向下取整倍数。即以单次访问请求的字节数作为单位，如 length2，size 为 4，则字节总数强制向下取整为 12

的倍数；

（4）跨步距离（stride+1）同总访问字节数根据 length << size 向下取整倍数对齐；

（5）对于一个描述符所对应的 dma 传输，当发送了 tx.count+1 个字节数后视为 dma 完成。如果发送总访问字节数大于接收总访问字节数，那么以接收的总访问字节数作为发送的总访问字节数。

31.5 DMA 中断

DMA 控制器有两种中断：超时中断和描述符中断，每个通道均可通过使能位对中断进行屏蔽或使能。响应中断后，可通过中断状态寄存器查询到中断源通道，并需要对中断状态寄存器进行写清零操作。

31.5.1 超时中断

当通道取回一个无效的描述符后，通道超时计数器被激活。当计数值达到超时时间预设值的时，触发超时中断，提示软件控制器空闲，需要重新装填描述符。此时可以通过通道指针寄存器读取到当前描述符链行进的位置。在装填完描述符后，需要重新通过通道控制寄存器 poll 位获取新的描述符。在取回一个有效的描述符后，通道重新开始工作。

31.5.2 描述符中断

使能描述符中断不仅需要使能该通道的描述符中断，还需要再描述符的中断触发位配置为 1。当描述符执行完成后，立即产生描述符中断。