



Hi3511/Hi3512 H.264 编解码处理器

用户指南

文档版本 06

发布日期 2009-03-23

部件编码 N/A

秘密

版权所有 © 深圳市海思半导体有限公司

深圳市海思半导体有限公司为客户提供全方位的技术支持，用户可与就近的海思办事处联系，也可直接与公司总部联系。

深圳市海思半导体有限公司

地址: 深圳市龙岗区坂田华为基地华为电气生产中心 邮编: 518129

网址: <http://www.hisilicon.com>

客户服务电话: +86-755-28788858

客户服务传真: +86-755-28357515

客户服务邮箱: support@hisilicon.com

版权所有 © 深圳市海思半导体有限公司 2008~2009。保留一切权利。

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

 **HISILICON**、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

秘密

版权所有 © 深圳市海思半导体有限公司



目 录

前 言	1
1 产品概述.....	1-1
1.1 体系架构	1-2
1.1.1 概述.....	1-2
1.1.2 处理器系统.....	1-2
1.1.3 图形处理.....	1-3
1.1.4 视频编解码处理.....	1-3
1.1.5 加解密引擎.....	1-4
1.1.6 存储控制器接口.....	1-4
1.1.7 以太网接口	1-4
1.1.8 视频接口	1-4
1.1.9 音频接口	1-5
1.1.10 MMC/SD/SDIO 控制器.....	1-5
1.1.11 PCI.....	1-5
1.1.12 USB	1-5
1.1.13 其他外设接口	1-5
1.1.14 硬件特性.....	1-6
1.2 应用场景	1-6
2 硬件特性.....	2-1
2.1 管脚描述	2-2
2.1.1 系统管脚	2-2
2.1.2 JTAG 管脚	2-3
2.1.3 VO 管脚	2-4
2.1.4 VI 管脚	2-4
2.1.5 I ² C 管脚	2-7
2.1.6 IR 管脚	2-7
2.1.7 UART 管脚	2-7
2.1.8 SIO 管脚	2-8
2.1.9 SMI 管脚	2-9
2.1.10 ETH 管脚	2-10



2.1.11 DDR2 管脚.....	2-11
2.1.12 USB 1.1 HOST 管脚.....	2-15
2.1.13 USB 2.0 OTG 管脚.....	2-15
2.1.14 RTC 管脚.....	2-16
2.1.15 MMC 管脚.....	2-16
2.1.16 SPI 管脚.....	2-17
2.1.17 PCI 管脚.....	2-17
2.1.18 电源和地管脚.....	2-20
2.2 管脚复用	2-25
2.2.1 VI 管脚复用.....	2-25
2.2.2 ETH 管脚复用	2-28
2.2.3 MMC 管脚复用	2-28
2.2.4 IR 管脚复用.....	2-28
2.2.5 I ² C 管脚复用.....	2-29
2.2.6 UART 管脚复用	2-29
2.2.7 SPI 管脚复用	2-29
2.2.8 SIO 管脚复用	2-30
2.2.9 SMI 管脚复用	2-30
2.2.10 DDR2 管脚复用	2-31
2.2.11 PCI 管脚复用.....	2-31
2.3 上下电顺序推荐	2-32
2.4 外部中断	2-32
2.5 电气特性	2-32
2.5.1 DC/AC 参数.....	2-32
2.5.2 极限参数.....	2-33
2.5.3 推荐工作条件.....	2-34
2.6 PCB 布线建议	2-34
2.7 时序规格	2-35
2.7.1 时序图例.....	2-35
2.7.2 DDR2 接口时序	2-35
2.7.3 ETH 接口时序	2-38
2.7.4 VI 接口时序.....	2-40
2.7.5 VO 接口时序	2-40
2.7.6 PCI 接口时序.....	2-41
2.7.7 I ² C 接口时序.....	2-42
2.7.8 MMC 接口时序	2-43
2.7.9 SSP 接口时序	2-43
2.7.10 UART 接口时序	2-46
2.7.11 SIO 接口时序.....	2-47



2.7.12 SMI 接口时序.....	2-49
2.8 封装和管脚分布	2-49
2.8.1 封装.....	2-49
2.8.2 管脚分布.....	2-52
3 系统控制.....	3-1
3.1 复位.....	3-2
3.1.1 概述.....	3-2
3.1.2 复位控制.....	3-2
3.1.3 复位配置.....	3-3
3.2 时钟.....	3-3
3.2.1 概述.....	3-3
3.2.2 时钟控制框图.....	3-4
3.2.3 时钟配置.....	3-5
3.3 处理器及存储器地址空间映射	3-9
3.3.1 ARM926EJ-S 处理器.....	3-9
3.3.2 存储器地址空间映射	3-9
3.4 中断系统	3-16
3.4.1 概述.....	3-16
3.4.2 中断系统.....	3-16
3.4.3 中断映射	3-18
3.4.4 寄存器概览.....	3-19
3.4.5 寄存器描述	3-20
3.5 直接存储器存取控制器	3-24
3.5.1 概述.....	3-24
3.5.2 特点.....	3-24
3.5.3 功能描述.....	3-25
3.5.4 工作方式.....	3-29
3.5.5 寄存器概览.....	3-31
3.5.6 寄存器描述	3-33
3.6 加解密引擎	3-49
3.6.1 概述.....	3-49
3.6.2 特点.....	3-49
3.6.3 功能描述.....	3-50
3.6.4 工作方式.....	3-60
3.6.5 寄存器概览.....	3-62
3.6.6 寄存器描述	3-63
3.7 Timer	3-77
3.7.1 概述.....	3-77
3.7.2 特点.....	3-77



3.7.3 功能描述.....	3-77
3.7.4 工作方式.....	3-78
3.7.5 寄存器概览.....	3-79
3.7.6 寄存器描述.....	3-80
3.8 看门狗.....	3-88
3.8.1 概述.....	3-88
3.8.2 特点.....	3-88
3.8.3 信号描述.....	3-89
3.8.4 功能描述.....	3-89
3.8.5 工作方式.....	3-90
3.8.6 寄存器概览.....	3-91
3.8.7 寄存器描述.....	3-92
3.9 实时时钟.....	3-95
3.9.1 概述.....	3-95
3.9.2 特点.....	3-95
3.9.3 功能描述.....	3-95
3.9.4 工作方式.....	3-96
3.9.5 寄存器概览.....	3-97
3.9.6 寄存器描述.....	3-97
3.10 系统控制器.....	3-101
3.10.1 概述.....	3-101
3.10.2 特点.....	3-101
3.10.3 功能描述.....	3-102
3.10.4 寄存器概览.....	3-107
3.10.5 寄存器描述.....	3-108
3.11 电源管理与低功耗模式控制.....	3-133
3.11.1 概述.....	3-133
3.11.2 系统工作模式.....	3-134
3.11.3 时钟门控和时钟频率调整.....	3-134
3.11.4 模块级低功耗控制.....	3-135
3.11.5 DDR 低功耗控制.....	3-135
4 存储控制器.....	4-1
4.1 DDR 控制器	4-2
4.1.1 概述.....	4-2
4.1.2 特点.....	4-2
4.1.3 信号描述.....	4-2
4.1.4 功能描述.....	4-4
4.1.5 工作方式.....	4-7
4.1.6 寄存器概览.....	4-9



4.1.7 寄存器描述.....	4-11
4.2 SMI 控制器.....	4-25
4.2.1 概述.....	4-25
4.2.2 特点.....	4-25
4.2.3 信号描述.....	4-25
4.2.4 功能描述.....	4-26
4.2.5 工作方式.....	4-30
4.2.6 寄存器概览.....	4-32
4.2.7 寄存器描述.....	4-33
5 以太网接口.....	5-1
5.1 概述.....	5-2
5.2 特点.....	5-2
5.3 信号描述.....	5-2
5.4 功能描述.....	5-3
5.4.1 MII 接口时序.....	5-3
5.4.2 MDIO 接口时序.....	5-6
5.4.3 收帧过程.....	5-7
5.4.4 发帧过程.....	5-10
5.4.5 中断管理.....	5-13
5.4.6 流量控制.....	5-13
5.4.7 MAC 过滤.....	5-14
5.5 工作方式.....	5-14
5.5.1 管脚复用配置.....	5-14
5.5.2 时钟门控.....	5-14
5.5.3 软复位.....	5-15
5.5.4 初始化.....	5-15
5.5.5 中断收帧流程.....	5-16
5.5.6 查询收帧流程.....	5-17
5.5.7 发帧流程.....	5-17
5.6 寄存器概览.....	5-18
5.7 寄存器描述.....	5-22
5.7.1 ETH MDIO 控制寄存器.....	5-22
5.7.2 ETH MAC 控制寄存器.....	5-26
5.7.3 ETH 统计计数控制寄存器.....	5-31
5.7.4 ETH 全局控制寄存器.....	5-39
5.7.5 ETH 统计结果寄存器.....	5-54
6 视频接口.....	6-1
6.1 视频输入单元.....	6-2
6.1.1 概述.....	6-2



6.1.2 特点.....	6-2
6.1.3 信号描述.....	6-3
6.1.4 功能描述.....	6-5
6.1.5 工作方式.....	6-17
6.1.6 寄存器概览.....	6-22
6.1.7 寄存器描述.....	6-33
6.2 视频输出单元.....	6-66
6.2.1 概述.....	6-66
6.2.2 特点.....	6-66
6.2.3 信号描述.....	6-67
6.2.4 功能描述.....	6-67
6.2.5 工作方式.....	6-75
6.2.6 寄存器概览.....	6-79
6.2.7 寄存器描述.....	6-80
6.2.8 使用指南.....	6-102
7 音频接口.....	7-1
7.1 概述.....	7-2
7.2 特点.....	7-2
7.2.1 PCM 接口特点	7-2
7.2.2 I ² S 接口特点	7-2
7.3 信号描述.....	7-3
7.4 功能描述.....	7-4
7.4.1 典型应用	7-4
7.4.2 功能原理	7-7
7.5 工作方式	7-8
7.5.1 管脚复用配置	7-8
7.5.2 时钟门控	7-8
7.5.3 时钟配置	7-8
7.5.4 软复位	7-9
7.5.5 音频播放和录制	7-9
7.6 寄存器概览	7-11
7.7 寄存器描述	7-12
8 MMC/SD/SDIO 控制器	8-5
8.1 概述	8-5
8.2 特点	8-5
8.3 信号描述	8-5
8.4 功能描述	8-5
8.5 工作方式	8-5
8.5.1 管脚复用配置	8-5



8.5.2 时钟门控.....	8-5
8.5.3 软复位.....	8-5
8.5.4 时钟配置.....	8-5
8.5.5 初始化.....	8-5
8.5.6 非数据传输指令.....	8-5
8.5.7 单块或多块读数据.....	8-5
8.5.8 单块与多块写数据.....	8-5
8.5.9 流数据读写.....	8-5
8.5.10 DMA 方式数据传输.....	8-5
8.5.11 Auto-stop 使用配置	8-5
8.5.12 停止或中止数据传输	8-5
8.5.13 Suspend 和 Resume 操作.....	8-5
8.5.14 Read wait 操作.....	8-5
8.6 寄存器概览.....	8-5
8.7 寄存器描述.....	8-5
9 PCI.....	9-1
9.1 概述.....	9-2
9.2 特点.....	9-2
9.3 信号描述.....	9-3
9.4 功能描述.....	9-4
9.5 工作方式.....	9-7
9.5.1 管脚复用配置.....	9-7
9.5.2 时钟门控.....	9-7
9.5.3 时钟配置.....	9-8
9.5.4 软复位.....	9-8
9.5.5 工作模式配置.....	9-8
9.5.6 通过 window 进行数据传输	9-9
9.5.7 通过 DMA 通道进行数据传输	9-10
9.6 寄存器概览.....	9-11
9.7 寄存器描述.....	9-14
9.7.1 AHB 侧寄存器描述.....	9-14
9.7.2 PCI 配置空间寄存器描述.....	9-28
10 USB	10-1
10.1 USB 1.1 HOST.....	10-2
10.1.1 概述.....	10-2
10.1.2 特点.....	10-2
10.1.3 信号描述	10-2
10.1.4 功能描述	10-3
10.1.5 工作方式	10-4



10.1.6 寄存器概览.....	10-5
10.1.7 寄存器描述.....	10-6
10.2 USB 2.0 OTG.....	10-24
10.2.1 概述.....	10-24
10.2.2 特点.....	10-24
10.2.3 信号描述.....	10-25
10.2.4 功能描述.....	10-26
10.2.5 工作方式.....	10-29
10.2.6 寄存器概览.....	10-35
10.2.7 寄存器描述.....	10-38
11 其他外设接口.....	11-1
11.1 I ² C 控制器.....	11-2
11.1.1 概述.....	11-2
11.1.2 特性.....	11-2
11.1.3 信号描述.....	11-2
11.1.4 功能描述.....	11-2
11.1.5 工作方式.....	11-5
11.1.6 寄存器概览.....	11-9
11.1.7 寄存器描述.....	11-10
11.2 通用异步收发器.....	11-29
11.2.1 概述.....	11-29
11.2.2 特点.....	11-30
11.2.3 信号描述.....	11-30
11.2.4 功能描述.....	11-31
11.2.5 工作方式.....	11-33
11.2.6 寄存器概览.....	11-36
11.2.7 寄存器描述.....	11-37
11.3 同步串口.....	11-51
11.3.1 概述.....	11-51
11.3.2 特点.....	11-51
11.3.3 信号描述.....	11-51
11.3.4 功能描述.....	11-52
11.3.5 工作方式.....	11-60
11.3.6 寄存器概览.....	11-63
11.3.7 寄存器描述.....	11-64
11.4 红外接口.....	11-72
11.4.1 概述.....	11-72
11.4.2 特点.....	11-72
11.4.3 信号描述.....	11-72



11.4.4 功能描述	11-72
11.4.5 工作方式	11-80
11.4.6 寄存器概览	11-82
11.4.7 寄存器描述	11-83
11.5 通用目的输入输出接口	11-96
11.5.1 概述	11-96
11.5.2 特点	11-96
11.5.3 信号描述	11-96
11.5.4 功能描述	11-101
11.5.5 工作方式	11-102
11.5.6 寄存器概览	11-106
11.5.7 寄存器描述	11-107
12 测试接口.....	12-1
12.1 概述	12-2
12.2 工作模式	12-2
12.3 JTAG 调试	12-2
12.3.1 接口信号	12-2
12.3.2 调试模式	12-3
13 视频处理.....	13-1
13.1 视频编解码器	13-2
13.1.1 概述	13-2
13.1.2 特点	13-2
13.2 图形加速	13-3
13.2.1 概述	13-3
13.2.2 特点	13-3
13.3 图形缩放	13-3
13.3.1 概述	13-3
13.3.2 特点	13-3
14 Hi3511 与 Hi3512 差异说明	14-1
14.1 硬件接口差异	14-2
14.2 管脚差异	14-2
A 缩略语	A-1

说明：前言放在正文前提取进总目录；总目录要用工具生成以便带有链接，不能手动刷新。



插图目录

图 1-1 Hi3511/Hi3512 芯片逻辑框图	1-2
图 1-2 Hi3511 8CIF 硬盘录像机 DVR 应用框图	1-7
图 1-3 Hi3511 16CIF 硬盘录像机 DVR 应用框图	1-8
图 1-4 Hi3511 16D1 硬盘录像机 DVR 应用框图	1-9
图 1-5 Hi3511 8CIF 板卡应用框图	1-9
图 2-1 时序图元说明	2-35
图 2-2 dqs_out 相对于 dq_out 的写操作时序图	2-35
图 2-3 dqs_out 相对于 ck 的写操作时序图	2-36
图 2-4 cmd/addr 相对于 ck 的写操作时序图	2-36
图 2-5 DDR2 SDRAM 输出时序图	2-37
图 2-6 rcven 相对于 dqs_in 的读操作时序图	2-37
图 2-7 MII 接口接收时序图	2-38
图 2-8 MII 接口发送时序图	2-39
图 2-9 MDIO 接口时序图	2-39
图 2-10 VI 接口时序图	2-40
图 2-11 VO 接口时序图	2-40
图 2-12 PCI 接口时序(采用芯片内部时钟)	2-41
图 2-13 PCI 接口时序(采用芯片外部时钟)	2-41
图 2-14 I ² C 传输时序	2-42
图 2-15 MMC 接口时序	2-43
图 2-16 SSP_SPICK 时序	2-44
图 2-17 SSP 主模式下接口时序 (sph= 0)	2-44
图 2-18 SSP 主模式下接口时序 (sph= 1)	2-44
图 2-19 UART 接口时序	2-46
图 2-20 SIO0 接口时序	2-47



图 2-21 SIO1 接口时序	2-48
图 2-22 芯片尺寸视图（顶视图）	2-49
图 2-23 芯片尺寸视图（底视图）	2-50
图 2-24 Detail B 的放大图	2-50
图 2-25 芯片尺寸视图（侧视图）	2-51
图 2-26 Detail A 的放大图	2-51
图 2-27 Hi3511 管脚分布图（1~12）	2-53
图 2-28 Hi3511 管脚分布图（13~23）	2-54
图 3-1 复位信号控制图	3-2
图 3-2 时钟管理模块功能框图	3-4
图 3-3 使用 SMI 控制器 BOOT 时的地址空间映射	3-10
图 3-4 使用 SMI 控制器 BOOT 时清除重映射后的地址分布	3-11
图 3-5 使用 DDR BOOT 时的地址空间映射	3-12
图 3-6 INT 功能框图	3-17
图 3-7 DMAC 功能框图	3-25
图 3-8 LLI 更新通道寄存器示意图	3-27
图 3-9 DMAC 与其他模块连接关系图	3-28
图 3-10 DMAC 链表结构示例	3-43
图 3-11 3 个密钥和 2 个密钥的 3DES 加密操作	3-50
图 3-12 3 个密钥和 2 个密钥的 3DES 解密操作	3-51
图 3-13 AES/DES 的电子密码本（ECB）模式	3-51
图 3-14 3DES 的电子密码本（ECB）模式	3-52
图 3-15 AES/DES 的密码分组链接（CBC）模式	3-53
图 3-16 3DES 的密码分组链接（CBC）模式	3-54
图 3-17 AES/DES 的 s 位密码反馈（CFB）模式	3-55
图 3-18 3DES 的 s 位密码反馈（CFB）模式	3-56
图 3-19 AES 的输出反馈模式	3-57
图 3-20 DES 的 s 位输出反馈模式	3-58
图 3-21 3DES 的 s 位输出反馈模式	3-59
图 3-22 AES 的 CTR 模式	3-60
图 3-23 WatchDog 应用框图	3-89
图 3-24 系统模式切换图	3-104



图 3-25 芯片 ID 寄存器位分配图.....	3-107
图 4-1 DDRC 与 2 片 16bit DDR2 SDRAM 的连接示意图	4-4
图 4-2 DDRC 与单片 16bit DDR2 SDRAM 的连接示意图	4-5
图 4-3 SMI 控制器与异步静态存储器连接示意图	4-26
图 4-4 SMI 控制器与带异步静态存储器接口的控制器件连接示意图	4-27
图 4-5 SMI 控制器时序参数模式时序图（读写）	4-28
图 4-6 SMI 控制器时序参数模式时序图（page 读）	4-29
图 4-7 SMI 控制器异步等待模式时序图（wait 读写）	4-30
图 5-1 MII 接口 100Mbit/s 接收时序	5-4
图 5-2 MII 接口 100Mbit/s 发送时序	5-4
图 5-3 MII 接口 10Mbit/s 接收时序	5-4
图 5-4 MII 接口 10Mbit/s 发送时序	5-5
图 5-5 MII 接口接收时序参数.....	5-5
图 5-6 MII 接口发送时序参数.....	5-5
图 5-7 MDIO 接口读时序	5-6
图 5-8 MDIO 接口写时序	5-7
图 5-9 MDIO 接口时序参数	5-7
图 5-10 中断方式收帧流程图	5-9
图 5-11 查询方式收帧流程图	5-10
图 5-12 中断方式发帧流程图	5-12
图 5-13 查询方式发帧流程图	5-13
图 6-1 VIU 功能框图.....	6-2
图 6-2 VIU 典型应用图.....	6-6
图 6-3 像素输入时序.....	6-6
图 6-4 模拟全电视信号对应的数字行采样时序.....	6-7
图 6-5 ITU-R BT. 601 行时序图	6-7
图 6-6 BT.656 YCbCr 4:2:2 行数据格式	6-8
图 6-7 NTSC 制式垂直同步时序图.....	6-9
图 6-8 PAL 制式垂直同步时序图.....	6-10
图 6-9 两路 BT.656 时分复用时序	6-11
图 6-10 四路 BT.656 时分复用时序	6-11
图 6-11 高清接口输入时序.....	6-12



图 6-12 数字摄像头支持的水平、垂直时序图	6-12
图 6-13 有效图像区域与水平垂直消隐关系图	6-13
图 6-14 YCbCr 4:2:2 co-sited 采样格式	6-13
图 6-15 YCbCr 4:2:2 interspersed 采样格式	6-14
图 6-16 YCbCr4:2:2 的存储模式	6-15
图 6-17 big endian 和 little endian 图像存储模式	6-15
图 6-18 Y/Cb/Cr 图像存储模式	6-16
图 6-19 图象存储 package 模式	6-17
图 6-20 raw data 8bit 存储模式	6-17
图 6-21 VIU 的时钟选择框图	6-18
图 6-22 VIU 的硬件工作流程	6-20
图 6-23 软件操作流程	6-21
图 6-24 亮度统计值软件读取流程	6-22
图 6-25 图像获取参数示意图	6-37
图 6-26 VOU 功能框图	6-66
图 6-27 VOU 输出图像结构示意图	6-68
图 6-28 关键色处理流程图	6-71
图 6-29 图像分层示意图	6-72
图 6-30 图形层的处理流程示意图	6-73
图 6-31 VOU 的硬件工作流程	6-102
图 6-32 VOU 的软件工作流程	6-103
图 7-1 I ² S 接口主模式连接示意图一	7-5
图 7-2 I ² S 接口主模式连接示意图二	7-5
图 7-3 I ² S 接口从模式连接示意图一	7-5
图 7-4 I ² S 接口从模式连接示意图二	7-6
图 7-5 PCM 接口连接示意图 (由 SIO 提供时钟和同步信号)	7-6
图 7-6 PCM 接口连接示意图 (由 CODEC 提供时钟和同步信号)	7-7
图 7-7 I ² S 接口时序	7-7
图 7-8 PCM 接口标准模式时序	7-7
图 7-9 PCM 接口自定义模式时序	7-8
图 8-1 MMC 控制器典型应用电路图	8-5
图 8-2 MMC 控制器功能框图	8-5



图 8-3 MMC 控制器非数据指令操作	8-5
图 8-4 MMC 控制器指令格式	8-5
图 8-5 卡设备响应格式	8-5
图 8-6 单块与多块读操作	8-5
图 8-7 单块与多块写操作	8-5
图 8-8 1bit 数据线传输模式下的块数据格式	8-5
图 8-9 4bit 数据线传输模式下的块数据格式	8-5
图 9-1 PCI 总线典型应用	9-5
图 9-2 Hi3511/Hi3512 PCI 模块架构示意图	9-5
图 9-3 PCI 总线 memory (I/O) 读操作时序图	9-6
图 9-4 PCI 总线 memory (I/O) 写操作时序图	9-6
图 9-5 PCI 总线配置访问时序图	9-7
图 10-1 USB 1.1 HOST 参考设计图	10-3
图 10-2 USB 1.1 HOST 应用框图	10-3
图 10-3 USB 2.0 OTG 参考设计图	10-25
图 10-4 USB 2.0 OTG 应用框图	10-26
图 10-5 USB 2.0 OTG 中断处理流程图	10-28
图 10-6 USB 2.0 OTG 寄存器地址分配图	10-35
图 11-1 I ² C 典型应用电路原理图	11-3
图 11-2 I ² C 数据传输时序图	11-3
图 11-3 I ² C 数据传输帧格式图	11-4
图 11-4 UART 的典型应用框图一	11-31
图 11-5 UART 的典型应用框图二	11-32
图 11-6 UART 帧格式	11-33
图 11-7 当 SSP 接单 slave 时的应用	11-52
图 11-8 当 SSP 接双 slave 时的应用	11-53
图 11-9 当 SSP 作 slave 时的应用	11-53
图 11-10 Motorola SPI 单帧帧格式 (spo=0、sph=0)	11-54
图 11-11 Motorola SPI 连续帧帧格式 (spo=0、sph=0)	11-54
图 11-12 Motorola SPI 单帧帧格式 (spo=0、sph=1)	11-55
图 11-13 Motorola SPI 连续帧帧格式 (spo=0、sph=1)	11-55
图 11-14 Motorola SPI 单帧帧格式 (spo=1、sph=0)	11-56



图 11-15 Motorola SPI 连续帧帧格式 (spo=1、sph=0)	11-56
图 11-16 Motorola SPI 单帧帧格式 (spo=1、sph=1)	11-57
图 11-17 Motorola SPI 连续帧帧格式 (spo=1、sph=1)	11-57
图 11-18 TI 同步串行单帧帧格式	11-58
图 11-19 TI 同步串行连续帧帧格式	11-58
图 11-20 National Semiconductor Microwire 单帧帧格式	11-59
图 11-21 National Semiconductor Microwire 连续帧帧格式	11-59
图 11-22 IR 模块功能框图	11-73
图 11-23 发送单个 NEC with simple repeat code 码的帧格式	11-75
图 11-24 持续按键连续发送 NEC with simple repeat code 码的帧格式	11-75
图 11-25 NEC simple repeat code 码 bit0 和 bit1 的位定义	11-75
图 11-26 NEC simple repeat code 码单发代码格式	11-76
图 11-27 NEC simple repeat code 码连发代码格式	11-76
图 11-28 发送单个 NEC with full repeat code 码的帧格式	11-76
图 11-29 持续按键连续发送 NEC with full repeat code 码的帧格式	11-76
图 11-30 NEC full repeat code 码 bit0 和 bit1 的位定义	11-77
图 11-31 NEC full repeat code 码单发代码格式	11-77
图 11-32 发送单个 TC9012 码的帧格式	11-78
图 11-33 持续按键连续发送 TC9012 码的帧格式	11-78
图 11-34 TC9012 码 bit0 和 bit1 的位定义	11-78
图 11-35 TC9012 码单发代码格式	11-78
图 11-36 TC9012 码连发代码格式 (C0=1)	11-79
图 11-37 TC9012 码连发代码格式 (C0=0)	11-79
图 11-38 发送单个 SONY 帧格式	11-79
图 11-39 持续按键连续发送 SONY 码帧格式图	11-79
图 11-40 bit0 和 bit1 的位定义	11-80
图 11-41 IR 模块初始化操作流程	11-81
图 11-42 读取解码数据的操作流程	11-82
图 12-1 对 ARM 进行调试的系统示例图	12-3



表格目录

表 2-1 管脚 I/O 类型说明	2-2
表 2-2 系统管脚.....	2-3
表 2-3 JTAG 管脚	2-3
表 2-4 VO 管脚.....	2-4
表 2-5 VI 管脚	2-5
表 2-6 I ² C 管脚	2-7
表 2-7 IR 管脚.....	2-7
表 2-8 UART 管脚	2-8
表 2-9 SIO 管脚	2-8
表 2-10 SMI 管脚.....	2-9
表 2-11 ETH 管脚	2-10
表 2-12 DDR2 管脚	2-11
表 2-13 USB 1.1 HOST 管脚.....	2-15
表 2-14 USB 2.0 OTG 管脚.....	2-15
表 2-15 RTC 管脚	2-16
表 2-16 MMC 管脚.....	2-16
表 2-17 SPI 管脚.....	2-17
表 2-18 PCI 管脚	2-17
表 2-19 电源和地管脚.....	2-20
表 2-20 VI0 同步信号管脚复用.....	2-26
表 2-21 VI0 数据信号管脚复用.....	2-26
表 2-22 VI1 数据信号管脚复用.....	2-26
表 2-23 VI2 同步信号管脚复用.....	2-27
表 2-24 VI2 数据信号管脚复用.....	2-27
表 2-25 VI3 数据信号管脚复用.....	2-27



表 2-26 ETH 管脚复用	2-28
表 2-27 MMC 管脚复用	2-28
表 2-28 IR 管脚复用	2-28
表 2-29 I ² C 管脚复用	2-29
表 2-30 UART1 管脚复用	2-29
表 2-31 SPI 管脚复用	2-29
表 2-32 SPI 片选选择	2-30
表 2-33 SIO0 管脚复用	2-30
表 2-34 SIO1 管脚复用	2-30
表 2-35 SMI 管脚复用	2-31
表 2-36 DDR2 管脚复用	2-31
表 2-37 PCI 管脚复用	2-31
表 2-38 DC 参数表 (DVDD33=3.3V)	2-32
表 2-39 DC 参数表 (DVDD18=1.8V)	2-33
表 2-40 AC 参数 (DVDD18=1.8V)	2-33
表 2-41 极限参数	2-33
表 2-42 推荐工作条件	2-34
表 2-43 Clock Parameters	2-37
表 2-44 Memory Device Parameters	2-37
表 2-45 Package/Board Parameters	2-38
表 2-46 25MHz 时 MII 接口的时序参数表	2-39
表 2-47 MDIO 接口时序参数	2-40
表 2-48 VI 接口时序参数	2-40
表 2-49 VO 接口时序参数表	2-41
表 2-50 PCI 接口时序参数表	2-41
表 2-51 I ² C 接口时序参数表	2-42
表 2-52 MMC 接口时序参数	2-43
表 2-53 SSP 接口时序参数	2-44
表 2-54 UART 接收数据信号时序参数	2-46
表 2-55 UART 发送数据信号时序参数	2-47
表 2-56 SIO0 接口时序参数	2-48
表 2-57 SIO0 接口时序参数	2-48



表 2-58 封装参数说明表.....	2-51
表 2-59 Hi3511 管脚数目统计表.....	2-52
表 2-60 管脚排列表.....	2-55
表 3-1 复位信号分类表.....	3-2
表 3-2 ARM/SCLK/HCLK 频率配置.....	3-5
表 3-3 系统控制器状态和时钟切换对应关系.....	3-5
表 3-4 MMC 时钟频率配置.....	3-6
表 3-5 USB 2.0 OTG PHY/USB 1.1 HOST 时钟频率配置.....	3-6
表 3-6 PCI 时钟频率配置.....	3-7
表 3-7 SMI 时钟频率配置.....	3-7
表 3-8 VO 时钟频率配置.....	3-7
表 3-9 VI 时钟频率配置.....	3-7
表 3-10 SIO0/SIO1 时钟频率配置.....	3-8
表 3-11 Hi3511/Hi3512 地址空间映射表.....	3-13
表 3-12 PCI 地址空间分配.....	3-15
表 3-13 中断映射表.....	3-18
表 3-14 INT 寄存器概览（基址是 0x1014_0000）.....	3-19
表 3-15 DMAC 硬件请求和相应设备的对应关系.....	3-28
表 3-16 DMAC 寄存器概览（基址是 0x1013_0000）.....	3-31
表 3-17 DBSize 及 SBSIZE 的值与其对应的 burst 长度.....	3-45
表 3-18 DWidth 和 SWidth 的值与其对应传输位宽.....	3-46
表 3-19 DMAC_CX_CONTROL 寄存器 Prot 段属性及定义.....	3-47
表 3-20 流控制器及传输类型位定义.....	3-49
表 3-21 CIPHER 寄存器概览（基址是 0x9004_0000）.....	3-62
表 3-22 Timer 寄存器概览（基址是 0x101E_2000、0x101E_3000）.....	3-79
表 3-23 WatchDog 接口信号描述.....	3-89
表 3-24 WatchDog 寄存器概览（基址是 0x101E_1000）.....	3-91
表 3-25 RTC 寄存器概览（基址是 0x101E_8000）.....	3-97
表 3-26 系统控制器状态和时钟切换对应关系表.....	3-103
表 3-27 系统控制器寄存器概览（基址是 0x101E_0000）.....	3-108
表 3-28 VEDU 时钟低功耗模式配置.....	3-135
表 4-1 DDRC 接口信号描述.....	4-2



表 4-2 主流 DRAM 产商的 DDR2 SDRAM 器件的寄存器配置值 (135MHz)	4-6
表 4-3 DDRC 命令真值表.....	4-6
表 4-4 DDRC 寄存器概览 (基址是 0x1015_0000)	4-10
表 4-5 DDRC 地址映射表 (16bit 模式)	4-16
表 4-6 DDRC 地址映射表 (32bit 模式)	4-16
表 4-7 SMI 控制器接口信号一览表.....	4-25
表 4-8 SMI 控制器时序参数范围 (总线时钟 $f_{BUSCLK}=135MHz$)	4-27
表 4-9 SMI 控制器读写时序参数表.....	4-29
表 4-10 SMI 控制器 page 读时序参数表	4-29
表 4-11 SMI 控制器支持的存储器地址空间	4-31
表 4-12 SMI 控制器寄存器概览 (基址是 0x1010_0000)	4-32
表 5-1 ETH 接口信号描述	5-3
表 5-2 MII 接口时序参数说明.....	5-6
表 5-3 MDIO 接口时序参数	5-7
表 5-4 CPU 接收帧描述子数据结构	5-8
表 5-5 CPU 发送帧描述子数据结构	5-11
表 5-6 ETH MDIO 控制寄存器 (基址是 0x9003_0000)	5-18
表 5-7 ETH MAC 控制寄存器 (基址是 0x9003_0000)	5-18
表 5-8 ETH 统计计数控制寄存器 (基址是 0x9003_0000)	5-19
表 5-9 ETH 全局控制寄存器 (基址是 0x9003_0000)	5-19
表 5-10 ETH 统计结果寄存器 (基址是 0x9003_0000)	5-20
表 6-1 视频输入接口信号.....	6-3
表 6-2 SAV/EAV Format	6-8
表 6-3 有效 SAV/EAV 值	6-8
表 6-4 PAL 和 NTSC 制式 TV 图像帧对比	6-9
表 6-5 BT.601 一帧有效行数据	6-10
表 6-6 BT.656 帧时序	6-10
表 6-7 VIU 寄存器概览 (基址是 0x9006_0000)	6-22
表 6-8 VOU 视频接口信号	6-67
表 6-9 VOU 数据格式在 SDRAM 中的存放方式 (高 16 位)	6-68
表 6-10 VOU 数据格式在 SDRAM 中的存放方式 (低 16 位)	6-69
表 6-11 VOU 存储图像参数	6-70



表 6-12 VOU 显示图像参数	6-70
表 6-13 32×32×2bpp 双色和透明模式列表	6-74
表 6-14 32×32×2bpp 四色模式列表	6-74
表 6-15 2×32×2bpp 三色和透明模式列表	6-74
表 6-16 hclk 和 clk_vo 的比较	6-75
表 6-17 VOU 复位描述	6-76
表 6-18 视频层存储参数表	6-77
表 6-19 视频层显示参数	6-77
表 6-20 图形层数据存储参数表	6-78
表 6-21 图形层显示参数	6-78
表 6-22 硬件鼠标层的存储参数	6-78
表 6-23 硬件鼠标层的显示参数	6-79
表 6-24 VOU 寄存器概览（基址是 0x9005_0000）	6-79
表 7-1 SIO0 接口信号描述	7-3
表 7-2 SIO1 接口信号描述	7-4
表 7-3 管脚复用关系	7-8
表 7-4 SIO 寄存器概览	7-11
表 8-1 MMC 接口信号描述	8-5
表 8-2 信号线负载参数	8-5
表 8-3 非数据指令 MMC_CMD 参考配置	8-5
表 8-4 单块或多块读数据 MMC_CMD 参考配置	8-5
表 8-5 单块或多块写数据 MMC_CMD 参考配置	8-5
表 8-6 Resume 操作 MMC_CMDARG 参考配置	8-5
表 8-7 MMC 寄存器概览（基址是 0x9002_0000）	8-5
表 9-1 PCI 接口信号描述	9-3
表 9-2 PCI 接口信号描述	9-10
表 9-3 AHB 侧寄存器概览（基址是 0xB000_0000）	9-11
表 9-4 PCI 配置空间寄存器概览	9-13
表 10-1 USB 1.1 HOST 接口信号描述	10-2
表 10-2 USB 1.1 HOST 寄存器概览（基址是 0xA000_0000）	10-5
表 10-3 USB 2.0 OTG 接口信号描述	10-25
表 10-4 USB 2.0 OTG 时钟配置相关寄存器列表	10-29



表 10-5 USB 2.0 OTG 通用寄存器概览（基址是 0x8009_0000）	10-36
表 10-6 USB 2.0 OTG 端点映射控制状态寄存器（设备模式）	10-36
表 10-7 USB 2.0 OTG 端点映射控制状态寄存器（主机模式）	10-37
表 10-8 USB 2.0 OTG 控制与配置寄存器	10-37
表 10-9 USB 2.0 OTG 内置 DMA 寄存器	10-38
表 10-10 USB 2.0 OTG 接收包计数寄存器	10-38
表 10-11 查询间隔定义	10-68
表 11-1 I ² C 接口信号描述	11-2
表 11-2 I2C_SS_SCL_HCNT 典型配置值	11-5
表 11-3 I2C_SS_SCL_LCNT 典型配置值	11-5
表 11-4 I2C_FS_SCL_HCNT 典型配置值	11-5
表 11-5 I2C_FS_SCL_LCNT 典型配置值	11-6
表 11-6 I ² C 寄存器概览（基址是 0x101F_6000）	11-9
表 11-7 UART0 接口信号描述	11-30
表 11-8 UART1 接口信号描述	11-30
表 11-9 UART2 接口信号描述	11-31
表 11-10 RTS 流控信号输出模式框图	11-32
表 11-11 UART 寄存器概览	11-36
表 11-12 SSP 接口信号描述	11-51
表 11-13 时钟分频的典型配置	11-61
表 11-14 SSP 寄存器概览（基址是 0x101F_4000）	11-63
表 11-15 IR 接口信号表	11-72
表 11-16 红外接收数据码型的统计表-NEC 简单重复码	11-73
表 11-17 红外接收数据码型的统计表-NEC 完整重复码	11-74
表 11-18 红外接收数据码型的统计表-TC9012 和 SONY 码	11-74
表 11-19 IR 寄存器概览（基址是 0x101E_9000）	11-83
表 11-20 GPIO 接口信号描述	11-96
表 11-21 8 组 GPIO 寄存器基地址	11-106
表 11-22 GPIO 寄存器概览	11-107
表 12-1 Hi3511/Hi3512 工作模式说明表	12-2
表 12-2 Hi3511/Hi3512 的 JTAG 接口信号表	12-2
表 14-1 Hi3511 与 Hi3512 硬件接口差异	14-2



表 14-2 Hi3511 与 Hi3512 VI 管脚差异 14-2



前 言

概述

本文档介绍了 Hi3511/Hi3512 芯片的特性、逻辑结构，详细描述各个模块的功能、工作方式、相关寄存器定义，用图表的方式给出了接口时序关系和相关参数，并详细描述了芯片的管脚定义和用途以及芯片的性能参数和封装尺寸。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3511 H.264 编解码处理器	V100
Hi3512 H.264 编解码处理器	V100

读者对象

本文档主要适用于以下工程师：

- 电子产品设计维护人员
- 电子产品元器件市场销售人员

内容简介

本文档包含 14 章，内容如下。

章节	内容
1 产品概述	简要介绍了 Hi3511/Hi3512 芯片的应用场景和体系架构。

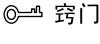


章节	内容
2 硬件特性	详细介绍了 Hi3511/Hi3512 芯片的封装、管脚、电气特性、PCB 布线建议等。
3 系统	详细介绍 Hi3511/Hi3512 芯片系统级的复位、时钟、处理器及存储器地址空间映射、中断系统以及直接存储器存取控制器、Timer、看门狗等模块。
4 存储器控制器	详细描述 Hi3511/Hi3512 芯片的存储器控制器的特点、功能、工作方式和寄存器。
5 以太网接口	详细描述 Hi3511/Hi3512 芯片的以太网接口的特点、功能、工作方式和寄存器。
6 视频接口	详细描述 Hi3511/Hi3512 芯片的视频接口的特点、功能、工作方式和寄存器。
7 音频接口	详细描述了 Hi3511/Hi3512 芯片的音频接口的特点、功能、工作方式和寄存器。
8 MMC	详细描述了 Hi3511/Hi3512 芯片的 MMC/SD/SDIO 控制器的特点、功能、工作方式和寄存器。
9 PCI	详细描述了 Hi3511/Hi3512 芯片的 PCI 的特点、功能、工作方式和寄存器。
10 USB	详细描述了 Hi3511/Hi3512 芯片的 USB 1.1 Host、USB 2.0 OTG 的特点、功能、工作方式和寄存器。
11 其他外设接口	详细描述了 Hi3511/Hi3512 芯片的其他外设接口的特点、功能、工作方式和寄存器。
12 测试接口	详细描述了 Hi3511/Hi3512 芯片的 JTAG 接口的工作模式和调试模式。
13 视频处理	详细描述了 Hi3511/Hi3512 芯片的视频编解码器、图形加速和图形缩放。
14 Hi3511 与 Hi3512 差异说明	描述了 Hi3511 与 Hi3512 差异说明。
A 缩略语	描述本文档出现的缩略语。

约定符号约定

在本文中可能出现下列标志，它们所代表的含义如下。



符号	说明
 危险	以本标志开始的文本表示有高度潜在危险, 如果不能避免, 会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险, 如果不能避免, 可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险, 如果忽视这些文本, 可能导致设备或器件损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
 说明	以本标志开始的文本是正文的附加信息, 是对正文的强调和补充。

通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体, 并且在内容前后增加线条与正文隔离。
“Terminal Display” 格式	“Terminal Display” 格式表示屏幕输出信息。此外, 屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。

表格内容约定

内容	说明
-	表格中的无内容单元。
*	表格中的内容用户可根据需要进行配置。



寄存器访问类型约定

类型	说明	类型	说明
RO	只读, 不可写。	W0C	可读, 写 0 清零, 写 1 保持不变。
WO	只写。	W1S	可读, 写 1 置 1, 写 0 保持不变。
RW	可读可写。	W0S	可读, 写 0 置 1, 写 1 保持不变。
RC	读清零。	OSW	可读, 写 1 后片内自清零, 即产生一个脉冲。
W1C	可读, 写 1 清零, 写 0 保持不变。		

数值单位约定

数据容量、频率、数据速率等的表达方式说明如下。

类别	符号	对应的数值
数据容量 (如 RAM 容量)	1K	1024
	1M	1,048,576
	1G	1,073,741,824
频率、数据速率等	1k	1000
	1M	1,000,000
	1G	1,000,000,000

地址、数据的表达方式说明如下。

符号	举例	说明
0x	0xFE04、0x18	用 16 进制表示的数据值、地址值。
0b	0b000、0b00 00000000	表示 2 进制的数据值以及 2 进制序列 (寄存器描述中除外)。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。



修订日期	版本	修订说明
2009-03-23	06	<p>第 1 章 产品概述</p> <p>修改了 1.1.6 存储控制器接口中支持的 DDR 的空间容量。</p> <p>第 3 章 系统控制</p> <p>修改了 3.10.5 系统控制器中 SC_PERCTRL4 的[4:1]的描述。</p> <p>第 7 章 音频接口</p> <p>修改表 7-1 中 SIO0_XFS、SIO0_XCK、SIO0_RFS、SIO0_RCK 的信号描述。</p> <p>修改表 7-2 中 SIO1_RFS、SIO1_RCK 的信号描述。</p> <p>修改 7.3 信号描述中表 7-1 和表 7-2 之间的描述。</p> <p>修改图 7-4、图 7-5 之间的描述。</p> <p>删除 7.4.2 功能原理中涉及标准模式的描述。</p> <p>修改图 7-5、图 7-6 和图 7-7。</p> <p>修改 7.5.3 时钟配置中 3 种方式的描述。</p> <p>修改寄存器 SIO_MODE、SIO_I2S_RIGHT_RD、I2S_CT_SET、PCM_CT_SET、I2S_LENGTH_CF 的描述及修改个别比特的相关描述。</p> <p>第 11 章 其他外设接口</p> <p>修改了图 11-3。</p> <p>修改 I2C_RAW_INTR_STAT、I2C_INTR_STAT、I2C_INTR_MASK 比特 7 和比特 5 的描述，原来是保留。</p> <p>删除 11.5.2 特性中 GPIO 支持 OD 输出的特性。</p> <p>修改了 GPIO 管脚复用时的部分系统控制器的名称。</p>



修订日期	版本	修订说明
2008-11-18	05	<p>第 2 章 硬件特性</p> <p>修改表 2-4 输出数据的驱动电流。</p> <p>修改表 2-18。</p> <p>修改表 2-58。</p> <p>修改表 2-14 中 OTGREXT 的输出电压, 由 3.3V 改为 1.2V。</p> <p>第 6 章 视频接口</p> <p>6.1.4 增加 BT.656 多路时分复用时序和高清接口。</p> <p>修改 VI_CH_CFG 中 bit29 的描述。</p> <p>修改 VI_CFG 中 bit31~bit23 的描述。</p> <p>增加 VI_LUM_STRH 和 VI_LUM_DIFF_ADDER 两个寄存器。</p> <p>第 9 章 PCI</p> <p>修改 9.5.6 通过 window 进行数据传输中关于 Hi3511/Hi3512 作为 PCI Master 或 Target 时支持的命令。</p> <p>第 10 章 USB</p> <p>修改 10.1.2 特点中关于 roothub 支持的下行端口数。</p> <p>修改 10.1.4 功能描述中中断传输的描述。</p> <p>修改 10.2.2 特点中关于 CPU 传输方式的描述。</p> <p>修改 10.2.4 功能描述中中断传输的描述。</p> <p>修改 OTG_TXMAXP 中[15:11]的描述。</p> <p>修改 OTG_CSR0 中[7]、[6]的描述。</p> <p>修改 OTG_RXMAXP 中[15:11]的描述。</p> <p>修改 OTG_TXTYPE 中[3:0]的描述。</p> <p>修改 OTG_RXTYPE 中[3:0]的描述。</p>



修订日期	版本	修订说明
2009-01-16	04	<p>增加 Hi3512 芯片信息。</p> <p>第 2 章 硬件特性</p> <p>修改了 2.7.10 URAT 接口时序中的关于接收和发送的描述。</p> <p>修改表 2-2 中 T21 管脚的描述。</p> <p>第 4 章 存储器接口</p> <p>修改了寄存器 DDRC_CHn_QOS 中各通道 QOS 的功能描述。</p> <p>第 12 章 测试接口</p> <p>修改了表 12-2 中 TDI、TMS 的描述。</p> <p>第 14 章 Hi3511 与 Hi3512 差异说明</p> <p>增加本章。</p> <p>A 缩略语</p> <p>修改 IR 的全称。</p>
2008-08-08	03	<p>第 10 章 USB</p> <p>修改了 10.2.5 工作方式中表 10-4 涉及 USB OTG PHY 24MHz 时钟源选择的描述。</p> <p>第 3 章 系统</p> <p>修改了表 3-12。</p>



修订日期	版本	修订说明
2008-07-30	02	<p>第 1 章 产品概述</p> <p>修改了 1.1.4 视频编解码处理的特点。</p> <p>修改了 1.1.13 其它外设接口中 GPIO 的描述。。</p> <p>第 2 章 硬件特性</p> <p>修改了 2.1.8 SIO 管脚中 ACKOUT 管脚的描述。</p> <p>修改了 2.1.14 中 RTCBATT 的电压取值范围。</p> <p>修改了 2.5.3 中 DVDD12 的最小值。</p> <p>第 6 章 视频接口</p> <p>修改了 6.1.4.2 数字摄像头接口中的描述。</p> <p>修改 6.1.7 中 VI_CFG[16:15]和 VI_CFG[14:13]中的说明。</p> <p>附录 A 缩略语</p> <p>新增。</p>
2008-06-10	01	第一次发布。



1 产品概述

关于本章

本章描述内容如下表所示。

标题	内容
1.1 体系架构	介绍 Hi3511/Hi3512 芯片的体系架构。
1.2 应用场景	介绍 Hi3511/Hi3512 芯片的应用场景。

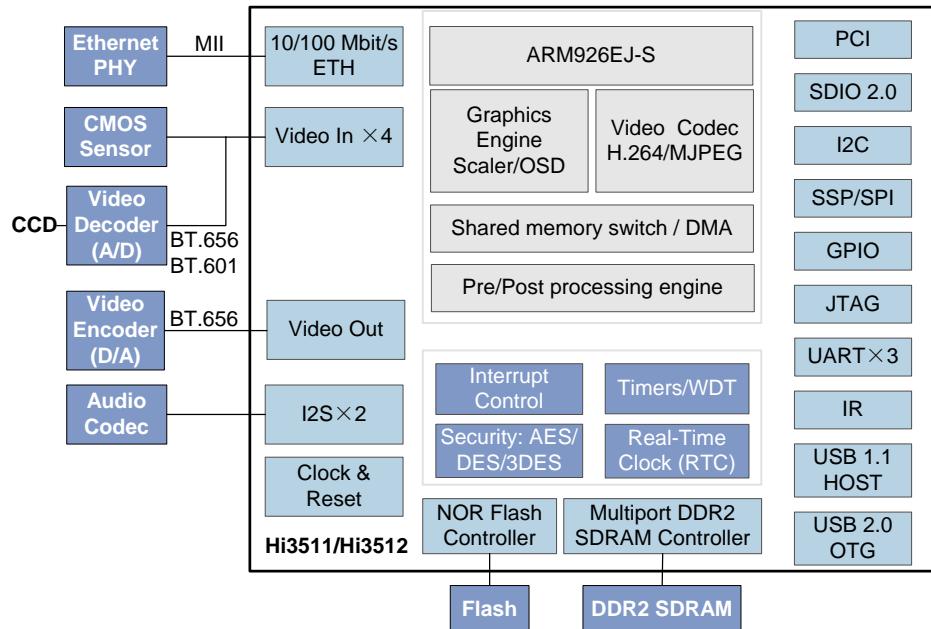


1.1 体系架构

1.1.1 概述

Hi3511/Hi3512 芯片逻辑框图如图 1-1 所示。

图1-1 Hi3511/Hi3512 芯片逻辑框图



1.1.2 处理器系统

Hi3511/Hi3512 处理器系统是基于高性能的 ARM926EJ-S 处理器平台搭建，处理器系统主要包含以下几个部分：

- ARM926 处理器：Hi3511/Hi3512 集成高性能的 ARM926EJ-S 处理器，作为整个系统的主控 CPU (Central Processing Unit)，协同硬件加速器一起完成音视频编解码功能以及系统调度操作。处理器内嵌 16KB 指令 cache 和 16KB 数据 cache，内嵌 2KB 指令紧耦合存储器 ITCM (Instruction Tightly-Coupled Memory)，工作频率可以达到 270MHz。
- 直接存储器访问控制器：DMAC (Direct Memory Access Controller) 直接在存储器和外设之间、外设和外设之间、存储器和存储器之间进行数据传输。
- 中断系统模块：为系统提供中断管理功能。支持 32 个中断源。
- 时钟模块：为整个系统提供时钟管理。包括芯片主时钟管理、各个模块的门控时钟管理。
- 复位模块：复位模块对整个系统的复位、各功能模块的复位进行统一的管理，包括：上电复位的管理和控制，系统软复位、功能模块单独软复位控制。



- 时钟模块：Timer 单元提供 2 组 Dual-Timer，每组 Dual-Timer 提供 2 个独立的计时器。
- 看门狗模块：看门狗用于系统异常情况下，复位整个系统。
- 实时时钟：用于实现时间显示和定时报警功能。RTC（Real Time Clock）可独立供电。
- 系统控制器：系统控制器模块提供了控制系统运行的手段，它控制系统运行的模式，监控系统运行状态，管理系统中的重要模块（如时钟、复位、管脚复用等），完成对外设的某些功能的配置。

1.1.3 图形处理

Hi3511/Hi3512 图形处理模块对视频输入图像或者视频输出图像进行加工处理，使其达到更好的图形显示效果。图形处理主要功能如下：

- 支持图像色彩和对比度增强功能。
- 支持图像去噪功能。
- 支持图形缩放功能，不超过 8 倍的任意大小的缩放功能。
- 支持前后 OSD（On Screen Display）图像叠加功能，支持 4 个区域的视频遮挡功能。
- 支持图像抗闪烁功能。
- 支持 2D 硬件加速。

1.1.4 视频编解码处理

Hi3511/Hi3512 集成高性能的 H.264/JPEG 硬件编码器和解码器，主要特点如下：

- H.264 Main Profile @ Level 3 编解码
- H.264 BaseLine @ Level 3 编解码
- JPEG/MJPEG Baseline 编解码
- 支持大小码流编码
 - 大码流最大 60fps@D1 或 240fps@CIF
 - 小码流最大支持 60fps@CIF
 - 大小码流可分别选择 H.264 和 MJPEG 协议
- H.264 单独编码、单独解码最大支持 75fps@D1 或 300fps@CIF，同编同解最大支持 30fps@D1 或 120fps@CIF
- MJPEG/JPEG 最大支持 300 万像素的编码，帧率 5fps
- H.264 支持 CBR/VBR 码率控制，16kbit/s~20Mbit/s
- 支持编码前处理：Deinterlace、时域滤波、前端 OSD、运动检测



1.1.5 加解密引擎

加解密引擎集成 AES (Advanced Encryption Standard) /DES (Data Encryption Standard) /3DES 多种加解密算法以及数字水印加解密技术，主要特点如下：

- DES/3DES 和 AES 算法符合 FIPS46-3/FIPS 197 标准，DES/3DES 和 AES 的工作模式均符合 FIPS -81/NIST special800-38a 标准。
- 数字水印技术。

1.1.6 存储控制器接口

系统提供 DDR2 SDRAM (Synchronous Dynamic Random Access Memory) 控制器、扩展总线、SMI (Static Memory Interface) 控制器接口，主要特点如下：

- DDRC (Double Data-Rate Controller)
 - 提供 1 个片选的 DDR2 接口，兼容数据位宽为 16bit 和 32bit 的 DDR2 SDRAM。
 - 16bit DDR2 SDRAM 最大支持 256MB 存储器空间
 - 32bit DDR2 SDRAM 最大支持 512MB 存储器空间
- SMI 控制器
 - 8 bit 数据位宽
 - 2 个片选
 - 每个片选最大支持 32MByte

1.1.7 以太网接口

以太网接口 ETH (Ethernet MAC) 模块实现网络接口数据的接收和发送，主要特点如下：

- 工作在 10Mbit/s 或者 100Mbit/s。
- 支持全双工或者半双工工作模式，提供 MII (Media Independent Interface) 接口。
- 提供可配置的 2 个 DMA 地址过滤表。
- 提供使用本机 MAC (Media Access Control) 地址进行帧过滤。
- 对网口的帧进行选择过滤接收。
- 提供流量限制功能。

1.1.8 视频接口

视频接口（以 Hi3511 为例）包括 4 个视频输入接口和 1 个视频输出接口：

- 视频输入接口：
 - 支持 4 个 BT.656 YCrCb 4:2:2 接口（其中 2 个支持 BT.601 接口），8bit 接口，每个接口可以支持 2 路 BT.656 复用视频输入。
 - 支持 1 路 SMPTE296M 720P 高清信号输入，Y/C 4:2:2，16bit。
 - 支持 2 路数字 camera 接口。
- 视频输出接口：1 路 BT.656 接口。



1.1.9 音频接口

音频输入输出接口 SIO (Sonic Input/Output) 包括 2 个 I²S (Inter-IC Sound) 音频接口，支持 8/16/32 位采样位宽，采样率可配置 (4KHz~48KHz)，每个 I²S 音频接口可以满足 8 路音频数据信号的输入。

1.1.10 MMC/SD/SDIO 控制器

MMC (Multi-media Card) /SD (Secure Digital) /SDIO (Secure Digital Input/Output) 控制器用来处理对 SD/MMC 存储卡的读写等操作，并可以通过 SDIO 协议实现对扩展外设（如 Blue Tooth、WiFi 等）的支持。MMC/SD/SDIO 控制器可以控制符合以下协议的设备：

- SD mem-version 2.00
- SDIO-version 1.10
- MMC-version 4.2

1.1.11 PCI

Hi3511/Hi3512 PCI (Peripheral Component Interconnect) 接口支持主桥 (Host Bridge) 模式和从桥 (Simple Bridge) 模式，既可以作为主控制器对外扩展 PCI 设备，也可以作为 PCI 接口的从设备。PCI 支持 33MHz 或者 66MHz，最大支持 5 个 PCI Device。

1.1.12 USB

Hi3511/Hi3512 集成了 1 个 USB (Universal Serial Bus) 2.0 OTG (On The Go) 接口和 1 个 USB 1.1 HOST 接口。

- USB 2.0 OTG: 支持 USB 2.0 协议，同时支持 USB 2.0 OTG 补充协议。
 - USB 2.0 OTG 作为设备支持高速 (480Mbit/s) 和全速(12Mbit/s)。
 - USB 2.0 OTG 作为主机支持高速 (480Mbit/s)、全速(12Mbit/s)和低速 (1.5Mbit/s)。
- USB 1.1 HOST: 完全兼容 USB Spec.1.1。
 - 可以支持 USB high speed/full speed/low speed 三种设备。
 - 完全符合 OHCI (Open Host Controller Interface) Rev1.0 规范。

1.1.13 其他外设接口

- I²C

I²C (The Inter-Integrated Circuit) 控制器实现标准 I²C 主从设备功能，兼容 Philips I²C 总线协议，可完成对 I²C 总线上的从设备的数据发送和接收。

- UART

UART (Universal Asynchronous Receiver Transmitter) 是一个异步串行的通信接口，可以和外部芯片的 UART 对接，实现芯片间的通信。Hi3511/Hi3512 支持 3 个 UART 接口。



- SSP

SSP (Synchronous Serial Protocol) 控制器可以作为一个 master 或 slave 与外部的设备来进行同步串行通信。支持以下通信协议：

- A Motorola SPI (Synchronous Peripheral Interface) -compatible interface
- A Texas Instruments synchronous serial interface
- A National Semiconductor Microwire interface

- IR

IR (Infrared Remoter) 通过红外接口接收红外数据，可以支持 NEC with simple repeat code、NEC with full repeat code、SONY 和 TC9012 四种数据格式解码，并且具备接收数据错误检测和红外遥控唤醒等功能。

- GPIO

Hi3511/Hi3512 最大支持 8 组 GPIO (General Purpose Input/Output)，每组 GPIO 提供 8 个可编程的输入输出管脚。GPIO 与其它业务管脚复用。每个管脚可以配置为输入或者输出，用于生成特定应用的输出信号或采集特定应用的输入信号。

1.1.14 硬件特性

- 800mW 典型功耗 (DVR 应用场景)
- 支持多级省电模式
- 0.13μm 工艺，1.2/1.8/3.3 V 芯片供电电压
- 441 pin TFBGA 封装，0.8mm 管脚间距，19mm×19mm
- 工作环境温度：-25°C～+85°C

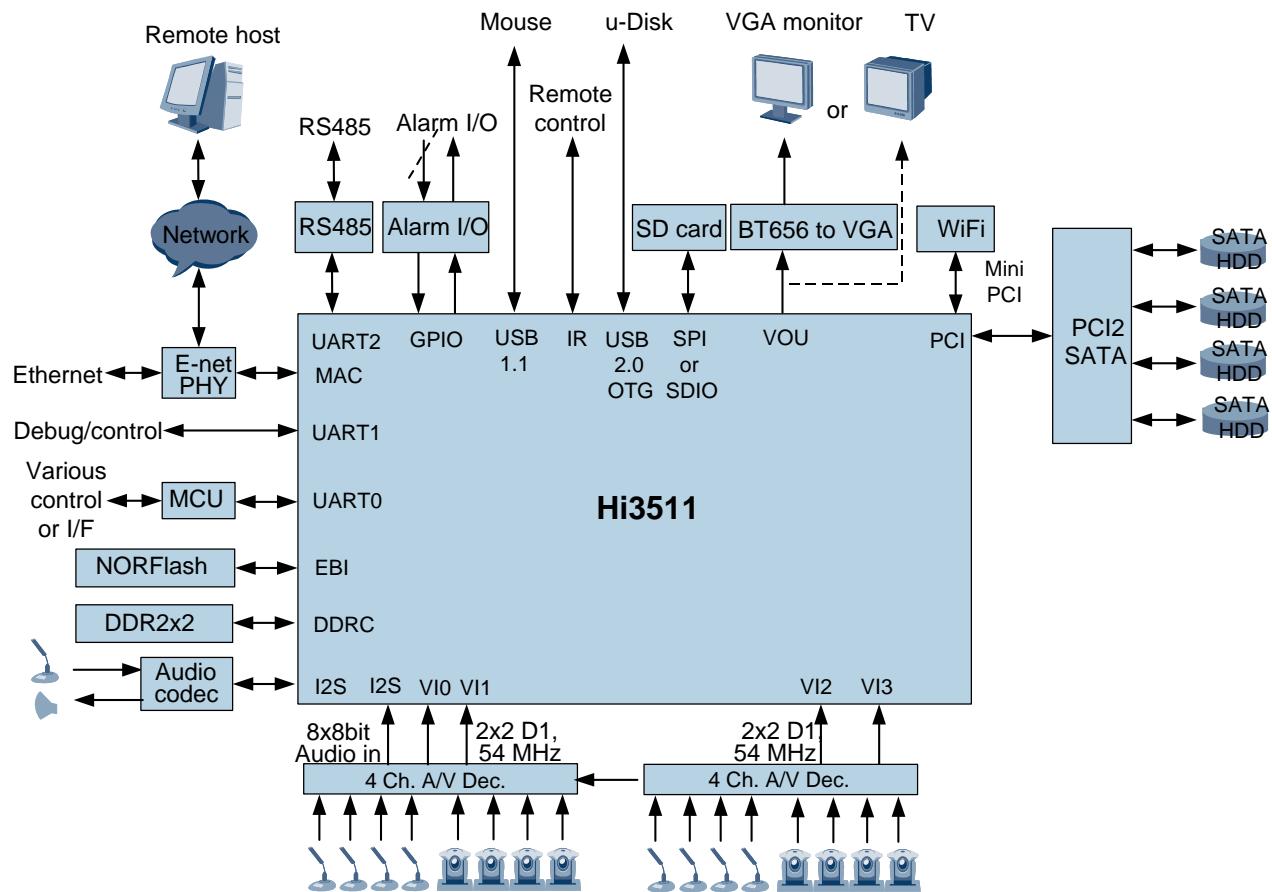
1.2 应用场景

Hi3511/Hi3512 (以下框图以 Hi3511 为例) 是一款基于 ARM9 处理器内核以及视频硬件加速引擎的高性能通信媒体处理器，具有高集成、可编程、支持 MPEG-4 AVC/H.264 和 MJPEG 等多协议的优点，可广泛应用于实时视频通信、数字图像监控等领域。

Hi3511 应用于 8CIF 硬盘录像机 DVR (Digital Video Recorder) 的应用框图如图 1-2 所示。

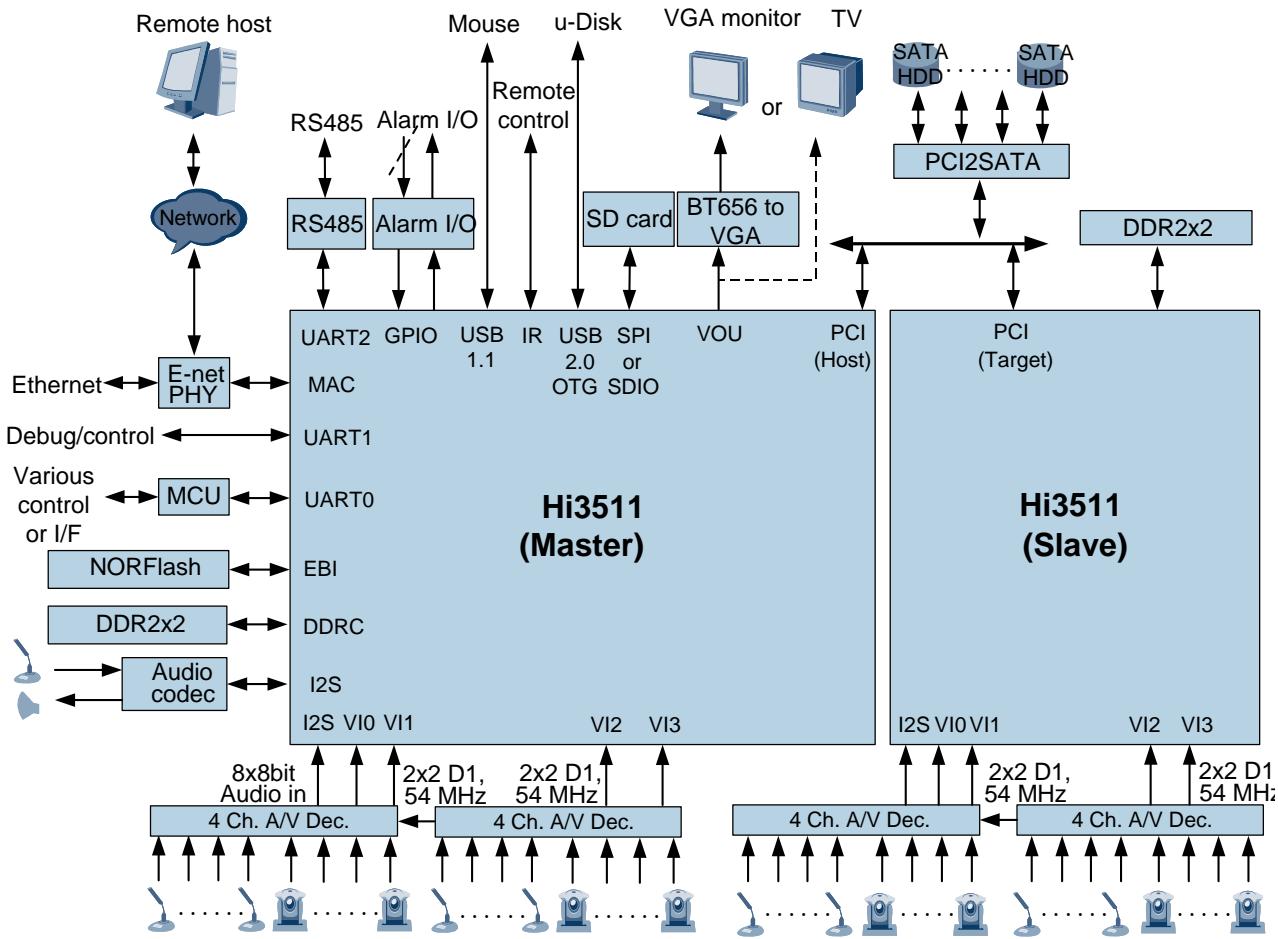


图1-2 Hi3511 8CIF 硬盘录像机 DVR 应用框图



Hi3511 应用于 16CIF 硬盘录像机 DVR 的应用框图如图 1-3 所示。

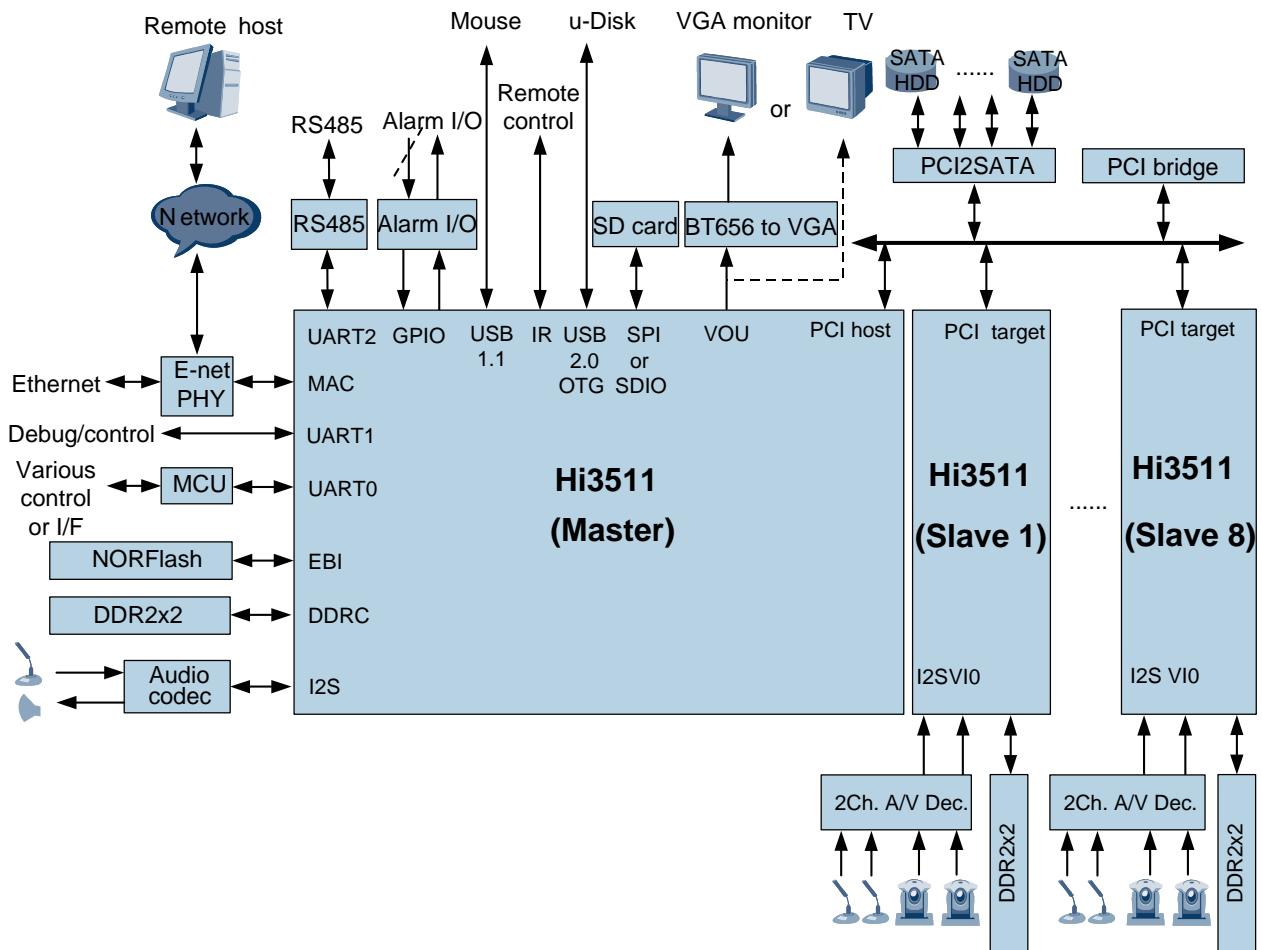
图1-3 Hi3511 16CIF 硬盘录像机 DVR 应用框图



Hi3511 应用于 16D1 硬盘录像机 DVR 的应用框图如图 1-4 所示。

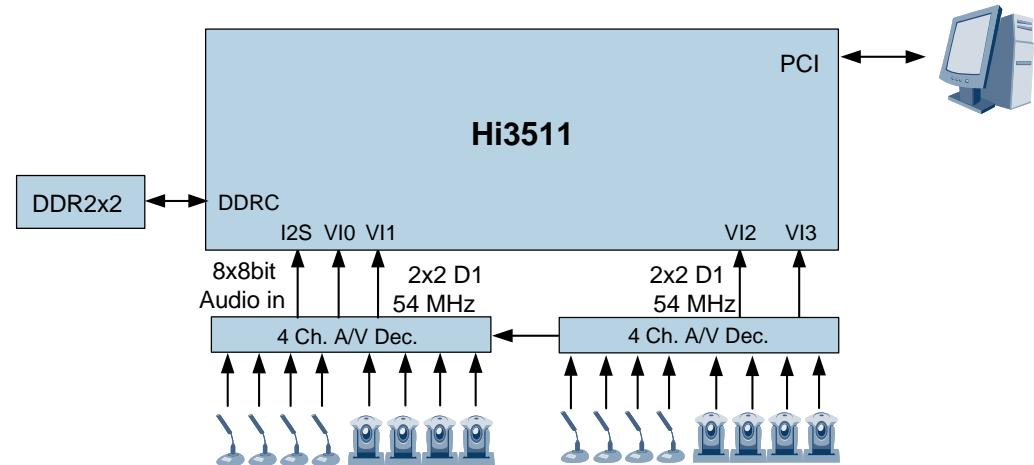


图1-4 Hi3511 16D1 硬盘录像机 DVR 应用框图



Hi3511 应用于 8CIF 板卡的应用框图如图 1-5 所示。

图1-5 Hi3511 8CIF 板卡应用框图





2 硬件特性

关于本章

本章描述内容如下表所示。

标题	内容
2.1 管脚描述	介绍管脚 I/O 类型。
2.2 管脚复用	介绍管脚复用信号。
2.3 上下电顺序推荐	介绍推荐的上下电顺序。
2.4 外部中断	介绍外部中断。
2.5 电气特性	介绍电气性能参数。
2.6 PCB 布线建议	给出 PCB 布线建议。
2.7 时序规格	介绍时序规格。
2.8 封装和管脚分布	介绍封装和管脚分布。



2.1 管脚描述

管脚 I/O 类型说明如表 2-1 所示。

表2-1 管脚 I/O 类型说明

I/O	说明
I	输入信号。
I _{PD}	输入信号, 内部下拉。
I _{PU}	输入信号, 内部上拉。
I _S	输入信号, 带施密特触发器。
I _{SPD}	输入信号, 带施密特触发器, 内部下拉。
I _{SPU}	输入信号, 带施密特触发器, 内部上拉。
O	输出信号。
O _{OD}	输出, 漏极开路。
I/O	双向输入/输出信号。
I _{PD} /O	双向, 输入下拉。
I _{PU} /O	双向, 输入上拉。
I _{SPU} /O	双向, 输入上拉, 带施密特触发器。
I _{PD} /O _{OD}	双向, 输入下拉, 输出漏极开路。
I _{PU} /O _{OD}	双向, 输入上拉, 输出漏极开路。
I _S /O	双向, 输入带施密特触发器。
I _S /O _{OD}	双向, 输入带施密特触发器, 输出漏极开路。
CIN	Crystal Oscillator, 晶振输入。
COUT	Crystal Oscillator, 晶振输出。
P	电源。
G	地。

2.1.1 系统管脚

系统管脚如表 2-2 所示。



表2-2 系统管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
U23	XIN	CIN	-	3.3	27MHz 晶振时钟输入或钟振时钟输入。
U22	XOUT	COUT	-	3.3	27MHz 晶振时钟输出。如果接钟振，该管脚悬空。
J2	XIN24	CIN	-	3.3	24MHz 晶振时钟输入或钟振时钟输入。为 OTG 提供时钟。
J1	XOUT24	COUT	-	3.3	24MHz 晶振时钟输出。如果接钟振，该管脚悬空。
R20	RSTN	I _{SPU}	-	3.3	系统上电复位信号输入，低电平有效。
T23	FUNSEL0	I _{SPD}	-	3.3	加载模式选择。 0: 自加载，通过 Flash 接口 BOOT。 1: 被加载，通过 DDR 接口 BOOT。
T21	TESTMO DE0	I _{SPD}	-	3.3	功能模式和测试模式选择。 0: 正常工作模式，ARM 可进入 debug 模式。 1: 测试模式，ARM 不可进入 debug 模式。
N21	WDGRST	O _{OD}	8	3.3	看门狗复位输出，低电平有效，OD 输出。

2.1.2 JTAG 管脚

JTAG (Joint Test Action Group) 管脚如表 2-3 所示。

表2-3 JTAG 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
R21	TDI	I _{SPU}	-	3.3	JTAG 数据输入。
R23	TCK	I _{SPD}	-	3.3	JTAG 时钟输入。
U21	TMS	I _{SPU}	-	3.3	JTAG 模式选择输入。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
P20	TRSTN	I _{SPD}	-	3.3	JTAG 复位输入, 低电平有效。
P23	TDO	O	8	3.3	JTAG 数据输出。

2.1.3 VO 管脚

VO (Video Output) 管脚如表 2-4 所示。

表2-4 VO 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
E2	VOCK	O	8	3.3	图像输出的时钟。
G1	VODAT0	O	4	3.3	图像输出的数据。
F1	VODAT1	O	4	3.3	
F2	VODAT2	O	4	3.3	
F3	VODAT3	O	4	3.3	
F4	VODAT4	O	4	3.3	
F5	VODAT5	O	4	3.3	
E1	VODAT6	O	4	3.3	
E3	VODAT7	O	4	3.3	

2.1.4 VI 管脚



说明

此处以 Hi3511 芯片为例进行介绍, Hi3512 芯片的相关内容请参见“14 Hi3511 与 Hi3512 差异说明”。

VI (Video Input) 管脚与 GPIO、中断输入、SPI 片选、VI0 和 VI2 的水平同步和垂直同步信号、高清模式下对应 VI 的低两比特数据线存在复用关系, 复用信息请参见“[2.2.1 VI 管脚复用](#)”。

VI 管脚如表 2-5 所示。



表2-5 VI 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
VI0 接口 (如整个接口不使用, 所有管脚可悬空)					
T1	VI0CK	I _{PU} /O	-	3.3	VI0 接口的时钟。
R5	VI0HS	I _{SPU} /O	4	3.3	VI0 接口的水平同步信号。(复用管脚)。如不使用, 该管脚可悬空。
T3	VI0VS	I _{PU} /O	4	3.3	VI0 接口的垂直同步信号。(复用管脚)。如不使用, 该管脚可悬空。
R3	VI0DAT2	I _{PU} /O	4	3.3	VI0 接口的数据 DAT2。对应 656 接口的 data0。
R2	VI0DAT3	I _{PU} /O	4	3.3	VI0 接口的数据 DAT3。对应 656 接口的 data1。
R4	VI0DAT4	I _{PU} /O	4	3.3	VI0 接口的数据 DAT4。对应 656 接口的 data2。
T2	VI0DAT5	I _{PU} /O	4	3.3	VI0 接口的数据 DAT5。对应 656 接口的 data3。
U3	VI0DAT6	I _{PU} /O	4	3.3	VI0 接口的数据 DAT6。对应 656 接口的 data4。
T4	VI0DAT7	I _{PU} /O	4	3.3	VI0 接口的数据 DAT7。对应 656 接口的 data5。
U2	VI0DAT8	I _{PU} /O	4	3.3	VI0 接口的数据 DAT8。对应 656 接口的 data6。
U1	VI0DAT9	I _{PU} /O	4	3.3	VI0 接口的数据 DAT9。对应 656 接口的 data7。
VI1 接口 (如整个接口不使用, 所有管脚可悬空)					
R1	VI1CK	I _{PU}	-	3.3	VI1 接口的时钟。
N5	VI1DAT0	I _{PU} /O	4	3.3	VI1 接口的数据 DAT0。
N3	VI1DAT1	I _{PU} /O	4	3.3	VI1 接口的数据 DAT1。
N4	VI1DAT2	I _{PU} /O	4	3.3	VI1 接口的数据 DAT2。
P2	VI1DAT3	I _{PU} /O	4	3.3	VI1 接口的数据 DAT3。
P3	VI1DAT4	I _{PU} /O	4	3.3	VI1 接口的数据 DAT4。
P5	VI1DAT5	I _{PU} /O	4	3.3	VI1 接口的数据 DAT5。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
P1	VI1DAT6	I _{PU} /O	4	3.3	VI1 接口的数据 DAT6。
P4	VI1DAT7	I _{PU} /O	4	3.3	VI1 接口的数据 DAT7。
VI2 接口 (如整个接口不使用, 所有管脚可悬空)					
Y2	VI2CK	I _{PU}	-	3.3	VI2 接口的时钟。
W2	VI2HS	I _{PU} /O	4	3.3	VI2 接口的水平同步信号。(复用管脚)。如不使用, 该管脚可悬空。
V4	VI2VS	I _{PU} /O	4	3.3	VI2 接口的垂直同步信号。(复用管脚)。如不使用, 该管脚可悬空。
V5	VI2DAT2	I _{PU} /O	4	3.3	VI2 接口的数据 DAT2。对应 656 接口的 data0。
V3	VI2DAT3	I _{PU}	4	3.3	VI2 接口的数据 DAT3。对应 656 接口的 data1。
V1	VI2DAT4	I _{PU} /O	4	3.3	VI2 接口的数据 DAT4。对应 656 接口的 data2。
V2	VI2DAT5	I _{PU}	4	3.3	VI2 接口的数据 DAT5。对应 656 接口的 data3。
U4	VI2DAT6	I _{PU} /O	4	3.3	VI2 接口的数据 DAT6。对应 656 接口的 data4。
Y1	VI2DAT7	I _{PU}	4	3.3	VI2 接口的数据 DAT7。对应 656 接口的 data5。
W1	VI2DAT8	I _{PU}	4	3.3	VI2 接口的数据 DAT8。对应 656 接口的 data6。
AA1	VI2DAT9	I _{PU}	4	3.3	VI2 接口的数据 DAT9。对应 656 接口的 data7。
VI3 接口 (如整个接口不使用, 所有管脚可悬空)					
AB1	VI3CK	I _{PU}	-	3.3	VI3 接口的时钟。
W4	VI3DAT0	I _{PU} /O	4	3.3	VI3 接口的数据 DAT0。
W3	VI3DAT1	I _{PU} /O	4	3.3	VI3 接口的数据 DAT1。
Y3	VI3DAT2	I _{PU} /O	4	3.3	VI3 接口的数据 DAT2。
AA2	VI3DAT3	I _{PU} /O	4	3.3	VI3 接口的数据 DAT3。
AB2	VI3DAT4	I _{PU} /O	4	3.3	VI3 接口的数据 DAT4。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AC2	VI3DAT5	I _{PU} /O	4	3.3	VI3 接口的数据 DAT5。
AC3	VI3DAT6	I _{PU} /O	4	3.3	VI3 接口的数据 DAT6。
AB3	VI3DAT7	I _{PU} /O	4	3.3	VI3 接口的数据 DAT7。

2.1.5 I²C 管脚

I²C 管脚与 GPIO 管脚存在复用关系，复用信息请参见“[2.2.5 I²C 管脚复用](#)”。

I²C 管脚如表 2-6 所示。

表2-6 I²C 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
R22	SDA	I _{SPU} / O _{OD}	8	3.3	I ² C 总线数据/地址，OD 输出。 PCB (Printed Circuit Board) 上需要外接上拉电阻。
P21	SCL	I _{SPU} / O _{OD}	8	3.3	I ² C 总线时钟，OD 输出。PCB 上需要外接上拉电阻。

2.1.6 IR 管脚

IR 管脚与 GPIO 管脚存在复用关系，复用信息请参见“[2.2.4 IR 管脚复用](#)”。

IR 管脚如表 2-6 所示。

表2-7 IR 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
P22	IRRCV	I _{PU} /O	4	3.3	红外遥控接收。

2.1.7 UART 管脚

UART1 管脚与 GPIO 管脚存在复用关系，复用信息请参见“[2.2.6 UART 管脚复用](#)”；
UART2 和 VI2 的同步信号存在复用关系，复用信息请参见“[2.2.1 VI 管脚复用](#)”。

UART0/UART1 管脚如表 2-8 所示。



表2-8 UART 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
C2	URXD0	I _{SPU}	-	3.3	UART0 数据接收。如不使用，该管脚可悬空。
B2	UTXD0	O	4	3.3	UART0 数据发送。
C1	URXD1	I _{SPU/O}	-	3.3	UART1 数据接收。如不使用，该管脚可悬空。
B1	UTXD1	I _{PD/O}	4	3.3	UART1 数据发送。
D2	URTSN1	I _{PD/O}	4	3.3	UART1 的请求发送信号。
D1	UCTSN1	I _{PD/O}	4	3.3	UART1 的清除发送信号。如不使用，该管脚可悬空。

2.1.8 SIO 管脚

SIO0、SIO1 管脚与 GPIO 管脚存在复用关系，复用信息请参见 “[2.2.8 SIO 管脚复用](#)”。

SIO0 和 SIO1 管脚如表 2-9 所示。

表2-9 SIO 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
J5	SIO0DI	I _{PU}	-	3.3	SIO 接口 0 数据输入。如不使用，该管脚可悬空。
G4	SIO0DO	O	4	3.3	SIO 接口 0 数据输出。
H4	SIO0XFS	I _{PU/O}	4	3.3	SIO 接口 0 发送侧帧同步信号。
K5	SIO0RFS	I _{PU/O}	4	3.3	SIO 接口 0 接收侧帧同步信号。
G3	SIO0XCK	I _{PU/O}	4	3.3	SIO 接口 0 发送侧位流时钟。
G2	SIO0RCK	I _{PU/O}	4	3.3	SIO 接口 0 接收侧位流时钟。
H3	ACKOUT	I _{PU/O}	8	3.3	SIO 接口的可编程输出时钟，用于支持低端 Audio Codec 器件。
J3	SIO1DI	I _{PU/O}	4	3.3	SIO 接口 1 数据输入。如不使用，该管脚可悬空。
J4	SIO1RFS	I _{PU/O}	4	3.3	SIO 接口 1 接收侧帧同步信号。
H1	SIO1RCK	I _{PU/O}	4	3.3	SIO 接口 1 接收侧位流时钟。



2.1.9 SMI 管脚

SMI 管脚与 GPIO 管脚存在复用关系，复用信息请参见“[2.2.9 SMI 管脚复用](#)”。

SMI 管脚如表 2-10 所示。

表2-10 SMI 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
J23	EBIWEN	O	8	3.3	SMI 接口的写使能信号，低电平有效。
L20	EBIOEN	O	8	3.3	SMI 接口的读使能信号，低电平有效。
M19	EBICS0N	O	8	3.3	SMI 片选信号 0，可配置成低电平有效或高电平有效，默认为低电平有效。系统支持从该片选启动。
K20	EBICS1N	I/O	8	3.3	SMI 片选信号 1，可配置成低电平有效或高电平有效，默认为低电平有效。
B23	EBIRDYN	I _{SPU} /O	4	3.3	SMI 接口的输入 Ready 指示信号，低电平有效。如不使用，该管脚可悬空。
J21	EBIADR0	O	8	3.3	SMI 地址总线。
J22	EBIADR1	O	8	3.3	
C22	EBIADR2	O	8	3.3	
C23	EBIADR3	O	8	3.3	
D21	EBIADR4	O	8	3.3	
D22	EBIADR5	O	8	3.3	
D23	EBIADR6	O	8	3.3	
E20	EBIADR7	O	8	3.3	
F20	EBIADR8	O	8	3.3	
H19	EBIADR9	O	8	3.3	
H20	EBIADR10	O	8	3.3	
F22	EBIADR11	O	8	3.3	
F23	EBIADR12	O	8	3.3	
G23	EBIADR13	O	8	3.3	
H23	EBIADR14	O	8	3.3	



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
G22	EBIADR15	O	8	3.3	SMI 数据总线。
G21	EBIADR16	O	8	3.3	
H22	EBIADR17	O	8	3.3	
E21	EBIADR18	O	8	3.3	
G20	EBIADR19	O	8	3.3	
E23	EBIADR20	O	8	3.3	
E22	EBIADR21	O	8	3.3	
F21	EBIADR22	O	8	3.3	
J20	EBIADR23	O	8	3.3	
J19	EBIADR24	I/O	8	3.3	
M22	EBIDQ0	I/O	8	3.3	SMI 数据总线。
M23	EBIDQ1	I/O	8	3.3	
L22	EBIDQ2	I/O	8	3.3	
L21	EBIDQ3	I/O	8	3.3	
K22	EBIDQ4	I/O	8	3.3	
L23	EBIDQ5	I/O	8	3.3	
K21	EBIDQ6	I/O	8	3.3	
K23	EBIDQ7	I/O	8	3.3	

2.1.10 ETH 管脚

ETH 管脚与 GPIO 管脚存在复用关系，复用信息请参见 “[2.2.2 ETH 管脚复用](#)”。

ETH 管脚如[表 2-11](#) 所示。

表2-11 ETH 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AC7	MDCK	O	4	3.3	MDIO (Management Data Input/Output) 接口时钟输出。
AB7	MDIO	I/O	4	3.3	MDIO 接口的输入/输出信号。
Y8	ETXD0	O	4	3.3	MII 接口发送数据。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AA8	ETXD1	O	4	3.3	
AB8	ETXD2	O	4	3.3	
AC8	ETXD3	O	4	3.3	
AA7	ETXEN	O	4	3.3	MII 接口发送使能信号。
AB6	ETXCK	I _{PU}	-	3.3	MII 接口发送时钟。
Y5	ERXD0	I _{PU}	-	3.3	MII 接口接收数据。
AA5	ERXD1	I _{PU}	-	3.3	
AB5	ERXD2	I _{PU}	-	3.3	
W7	ERXD3	I _{PU}	-	3.3	
Y6	ERXDV	I _{PU}	-	3.3	MII 接口接收数据有效。
Y7	ERXERR	I _{PU/O}	4	3.3	MII 接口接收数据错误。
AC5	ERXCK	I _{PU}	-	3.3	MII 接口接收时钟。
AA6	ECRS	I _{PU/O}	4	3.3	MII 载波侦听信号。
AC6	ECOL	I _{PU/O}	4	3.3	MII 碰撞指示信号。

2.1.11 DDR2 管脚

DDR2 管脚如表 2-12 所示，复用信息请参见 “[2.2.10 DDR2 管脚复用](#)”。

表2-12 DDR2 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
E4	DDRMRCV I	I/O	4	3.3	DDR2 控制器接收使能信号。 PCB 板反馈输入信号，与 DDRMRCVO 相连。如果不采用绕线方案，可使用复用功能。
D3	DDRMRCV O	I/O	12	3.3	DDR2 控制器接收使能输出信号。 PCB 板反馈输出信号，与 DDRMRCVI 相连，走线长度等于差分时钟平均长度与 DQS (Data Strobe) 平均长度之和。如果不采用绕线方案，可使用复用功能。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
B14	DDRCKP0	O	13.4	1.8	0 组正向差分时钟, 接 DDR2 SDRAM 0。
A14	DDRCKN0	O	13.4	1.8	0 组反向差分时钟, 接 DDR2 SDRAM 0。
B12	DDRCKP1	O	13.4	1.8	1 组正向差分时钟, 接 DDR2 SDRAM 1。
A12	DDRCKN1	O	13.4	1.8	1 组反向差分时钟, 接 DDR2 SDRAM 1。
C13	DDRCKE0	O	13.4	1.8	DDR2 SDRAM 的时钟使能信号, 高电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。
A15	DDRRASN	O	13.4	1.8	输出到 DDR2 SDRAM 的行地址选通信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。
B15	DDRCASN	O	13.4	1.8	输出到 DDR2 SDRAM 的列地址选通信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。
E15	DDRWEN	O	13.4	1.8	输出到 DDR2 SDRAM 的写使能信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。
D15	DDRCSN	O	13.4	1.8	输出到 DDR2 SDRAM 的片选信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。
D11	DDRADDR0	O	13.4	1.8	DDR2 SDRAM 地址信号。
A9	DDRADDR1	O	13.4	1.8	
C12	DDRADDR2	O	13.4	1.8	
C10	DDRADDR3	O	13.4	1.8	
B10	DDRADDR4	O	13.4	1.8	
D10	DDRADDR5	O	13.4	1.8	
E11	DDRADDR6	O	13.4	1.8	



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
B9	DDRADR7	O	13.4	1.8	DDR2 SDRAM bank 0 选择信号。
D9	DDRADR8	O	13.4	1.8	
E10	DDRADR9	O	13.4	1.8	
A10	DDRADR10	O	13.4	1.8	
B11	DDRADR11	O	13.4	1.8	
A11	DDRADR12	O	13.4	1.8	
C8	DDRADR13	O	13.4	1.8	
D14	DDRBA0	O	13.4	1.8	
D13	DDRBA1	O	13.4	1.8	
C14	DDRBA2	O	13.4	1.8	
D16	DDRDM0	O	13.4	1.8	输出到 DDR2 SDRAM 0 的字节屏蔽信号，对应数据总线 DDRDQ[7:0]。
D19	DDRDM1	O	13.4	1.8	输出到 DDR2 SDRAM 0 的字节屏蔽信号，对应数据总线 DDRDQ[15:8]。
D6	DDRDM2	O	13.4	1.8	输出到 DDR2 SDRAM1 的字节 Mask 信号，对应数据总线 DDRDQ[23:16]。
D4	DDRDM3	O	13.4	1.8	输出到 DDR2 SDRAM1 的字节 Mask 信号，对应数据总线 DDRDQ[31:24]。
E16	DDRODT	O	13.4	1.8	输出的 ODT 信号，连接到两片 DDR2 SDRAM
B16	DDRDQ0	I/O	13.4	1.8	DDR2 SDRAM 的数据总线。
A18	DDRDQ1	I/O	13.4	1.8	
C16	DDRDQ2	I/O	13.4	1.8	
C18	DDRDQ3	I/O	13.4	1.8	
B18	DDRDQ4	I/O	13.4	1.8	
A16	DDRDQ5	I/O	13.4	1.8	



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
D18	DDRDQ6	I/O	13.4	1.8	
A17	DDRDQ7	I/O	13.4	1.8	
C21	DDRDQ8	I/O	13.4	1.8	
A21	DDRDQ9	I/O	13.4	1.8	
A20	DDRDQ10	I/O	13.4	1.8	
A19	DDRDQ11	I/O	13.4	1.8	
A22	DDRDQ12	I/O	13.4	1.8	
C20	DDRDQ13	I/O	13.4	1.8	
B21	DDRDQ14	I/O	13.4	1.8	
B19	DDRDQ15	I/O	13.4	1.8	
C6	DDRDQ16	I/O	13.4	1.8	
B8	DDRDQ17	I/O	13.4	1.8	
A6	DDRDQ18	I/O	13.4	1.8	
D7	DDRDQ19	I/O	13.4	1.8	
A8	DDRDQ20	I/O	13.4	1.8	
B6	DDRDQ21	I/O	13.4	1.8	
A7	DDRDQ22	I/O	13.4	1.8	
A5	DDRDQ23	I/O	13.4	1.8	
B3	DDRDQ24	I/O	13.4	1.8	
C4	DDRDQ25	I/O	13.4	1.8	
A4	DDRDQ26	I/O	13.4	1.8	
B5	DDRDQ27	I/O	13.4	1.8	
E7	DDRDQ28	I/O	13.4	1.8	
A2	DDRDQ29	I/O	13.4	1.8	
D5	DDRDQ30	I/O	13.4	1.8	
A3	DDRDQ31	I/O	13.4	1.8	
B17	DDRDQS0	I/O	13.4	1.8	DDR2 SDRAM 0 的数据选通信号， 对应数据总线 DDRDQ[7:0]。
B20	DDRDQS1	I/O	13.4	1.8	DDR2 SDRAM 0 的数据选通信号， 对应数据总线 DDRDQ[15:8]。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
B7	DDR DQS2	I/O	13.4	1.8	DDR2 SDRAM 1 的数据选通信号， 对应数据总线 DDRDQ[23:16]。
B4	DDR DQS3	I/O	13.4	1.8	DDR2 SDRAM 1 的数据选通信号， 对应数据总线 DDRDQ[31:24]。

2.1.12 USB 1.1 HOST 管脚

USB 1.1 HOST 的管脚都是模拟的信号管脚。

USB 1.1 HOST 管脚如表 2-13 所示。

表2-13 USB 1.1 HOST 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AC4	USB D-DM	I/O	-	3.3	USB 1.1 HOST 端口 D-差分数据总线，为模拟信号。
AB4	USB D+DP	I/O	-	3.3	USB 1.1 HOST 端口 D+差分数据总线，为模拟信号。

2.1.13 USB 2.0 OTG 管脚

USB 2.0 OTG 的管脚都是模拟的信号管脚。

USB 2.0 OTG 管脚如表 2-14 所示。

表2-14 USB 2.0 OTG 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
N1	OTG ID	I	-	3.3	USB 2.0 OTG 类型识别管脚，为模拟信号。
L1	OTG D-DM	I/O	-	3.3	USB 2.0 OTG 端口 D-差分数据总线，为模拟信号。
L2	OTG D+DP	I/O	-	3.3	USB 2.0 OTG 端口 D+差分数据总线，为模拟信号。
K3	OTG REEXT	I/O	-	1.2	USB 2.0 OTG 外置电阻连接端。
N2	OTG VBUS	I/O	-	5	USB 2.0 OTG VBUS 电源。



2.1.14 RTC 管脚

RTC 的管脚是单独供电的。RTC 的管脚如表 2-14 所示。

表2-15 RTC 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
V21	XINRTC	CIN	-	1.2	RTC 晶振时钟输入，时钟频率为 32.768kHz。 1.2V 的供电由 PMU 内部产生。
V20	XOUTRTC	COUT	-	1.2	RTC 晶振时钟输出。 1.2V 的供电由 PMU 内部产生。
U19	RTCAVDD12	P	-	1.2	RTC 1.2V 电容滤波电源。 该 1.2V 电源由 PMU 内部产生。
W20	RTCAGND	G	-	0	RTC 地。
U20	RTCBATT	P	-	2.0~3.6	RTC 电池供电电源。

2.1.15 MMC 管脚

MMC 管脚与 GPIO 管脚存在复用关系，复用信息请参见“[2.2.3 MMC 管脚复用](#)”。

MMC 的管脚如表 2-16 所示。

表2-16 MMC 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
Y23	SDIOCMD	I _{SPU} /O	8	3.3	MMC 命令。
W22	SDIODAT0	I _{SPU} /O	8	3.3	MMC 数据 0。
Y21	SDIODAT1	I _{SPU} /O	8	3.3	MMC 数据 1。
W23	SDIODAT2	I _{SPU} /O	8	3.3	MMC 数据 2。
W21	SDIODAT3	I _{SPU} /O	8	3.3	MMC 数据 3。
Y22	SDIOCK	I _{SPU} /O	12	3.3	MMC 输出时钟。



2.1.16 SPI 管脚

SPI (SSP) 管脚与 GPIO 管脚存在复用关系，复用信息请参见 “[2.2.7 SPI 管脚复用](#)”。

SPI 的管脚如表 2-17 所示。

表2-17 SPI 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
M20	SPIDI	I _{SPU} /O	4	3.3	SPI 输入数据。如不使用，该管脚可悬空。
N23	SPIDO	I _{PU} /O	4	3.3	SPI 的输出数据。
N20	SPICSN0	I _{SPU} /O	4	3.3	SPI 的片选信号 0。
N22	SPICK	I _{SPU} /O	8	3.3	SPI 的时钟信号。

2.1.17 PCI 管脚

PCI 的部分管脚与 GPIO 管脚存在复用关系，复用信息请参见 “[2.2.11 PCI 管脚复用](#)”。

PCI 的管脚如表 2-18 所示。

表2-18 PCI 管脚

Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AB10	PCICLK	I/O	-	3.3	PCI 工作时钟。
W10	PCIRSTN	I _S /O	-	3.3	PCI 总线复位。
Y20	PCIAD0	I/O	-	3.3	PCI 地址数据复用总线 0。
AA20	PCIAD1	I/O	-	3.3	PCI 地址数据复用总线 1。
AA23	PCIAD2	I/O	-	3.3	PCI 地址数据复用总线 2。
AA22	PCIAD3	I/O	-	3.3	PCI 地址数据复用总线 3。
AB23	PCIAD4	I/O	-	3.3	PCI 地址数据复用总线 4。
AB22	PCIAD5	I/O	-	3.3	PCI 地址数据复用总线 5。
AC22	PCIAD6	I/O	-	3.3	PCI 地址数据复用总线 6。
AB21	PCIAD7	I/O	-	3.3	PCI 地址数据复用总线 7。
AB20	PCIAD8	I/O	-	3.3	PCI 地址数据复用总线 8。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AC20	PCIAD9	I/O	-	3.3	PCI 地址数据复用总线 9。
Y19	PCIAD10	I/O	-	3.3	PCI 地址数据复用总线 10。
AA19	PCIAD11	I/O	-	3.3	PCI 地址数据复用总线 11。
AB19	PCIAD12	I/O	-	3.3	PCI 地址数据复用总线 12。
AC19	PCIAD13	I/O	-	3.3	PCI 地址数据复用总线 13。
Y18	PCIAD14	I/O	-	3.3	PCI 地址数据复用总线 14。
AA18	PCIAD15	I/O	-	3.3	PCI 地址数据复用总线 15。
AB16	PCIAD16	I/O	-	3.3	PCI 地址数据复用总线 16。
AC16	PCIAD17	I/O	-	3.3	PCI 地址数据复用总线 17。
Y15	PCIAD18	I/O	-	3.3	PCI 地址数据复用总线 18。
AA15	PCIAD19	I/O	-	3.3	PCI 地址数据复用总线 19。
AB15	PCIAD20	I/O	-	3.3	PCI 地址数据复用总线 20。
AC15	PCIAD21	I/O	-	3.3	PCI 地址数据复用总线 21。
Y14	PCIAD22	I/O	-	3.3	PCI 地址数据复用总线 22。
AA14	PCIAD23	I/O	-	3.3	PCI 地址数据复用总线 23。
W13	PCIAD24	I/O	-	3.3	PCI 地址数据复用总线 24。
Y13	PCIAD25	I/O	-	3.3	PCI 地址数据复用总线 25。
AA13	PCIAD26	I/O	-	3.3	PCI 地址数据复用总线 26。
AB13	PCIAD27	I/O	-	3.3	PCI 地址数据复用总线 27。
AC13	PCIAD28	I/O	-	3.3	PCI 地址数据复用总线 28。
AB12	PCIAD29	I/O	-	3.3	PCI 地址数据复用总线 29。
AC12	PCIAD30	I/O	-	3.3	PCI 地址数据复用总线 30。
Y12	PCIAD31	I/O	-	3.3	PCI 地址数据复用总线 31。
AC10	PCIFRAME N	I/O	-	3.3	PCI 总线 FRAME 信号线。
AC21	PCICBE0	I/O	-	3.3	PCI 字节使能和命令复用总线 0。
AB18	PCICBE1	I/O	-	3.3	PCI 字节使能和命令复用总线 1。
AA16	PCICBE2	I/O	-	3.3	PCI 字节使能和命令复用总线 2。



Pin	管脚名称	类型	驱动 (mA)	电压 (V)	描述
AC14	PCICBEE3	I/O	-	3.3	PCI 字节使能和命令复用总线 3。
Y16	PCIIRDYN	I/O	-	3.3	PCI 总线 IRDY 信号, 低电平有效。
AC17	PCITRDYN	I/O	-	3.3	PCI 总线 TRDY 信号, 低电平有效。
AB17	PCISTOPN	I/O	-	3.3	PCI 总线 STOP 信号, 低电平有效。
AA17	PCIDEVSEL N	I/O	-	3.3	PCI 总线 DEVSEL 信号, 低电平有效。
AC18	PCIPAR	I/O	-	3.3	PCI 总线 PAR 信号。
AC11	PCIPERRN	I/O	-	3.3	PCI 总线 PERR 信号, 低电平有效。 如不使用, 该管脚需外接上拉电阻, 电阻阻值需大于 $4.7\text{k}\Omega$ 。
Y17	PCISERRN	I/O _{OD}	-	3.3	PCI 总线 SERR 信号, 低电平有效。 该管脚需外接上拉电阻, 电阻阻值需大于 $4.7\text{k}\Omega$ 。
W16	PCIIDSEL	I _S /O	-	3.3	PCI 总线 IDSEL 信号。
AA9	PCIINTAN	I/O _{OD}	-	3.3	PCI 中断 INTA 信号, 低电平有效。 使用时单板上拉。
Y9	PCIGRANT0 N	I/O	-	3.3	PCI 总线仲裁信号 0, 低电平有效。
AB9	PCIGRANT1 N	I/O	-	3.3	PCI 总线仲裁信号 1, 低电平有效。
AC9	PCIGRANT2 N	I/O	-	3.3	PCI 总线仲裁信号 2, 低电平有效。
Y10	PCIGRANT3 N	I/O	-	3.3	PCI 总线仲裁信号 3, 低电平有效。
AB14	PCIGRANT4 N	I/O	-	3.3	PCI 总线仲裁信号 4, 低电平有效。
AB11	PCIREQ0N	I/O	-	3.3	PCI 总线申请信号 0, 低电平有效。
Y11	PCIREQ1N	I _S /O	-	3.3	PCI 总线申请信号 1, 低电平有效。
AA11	PCIREQ2N	I _S /O	-	3.3	PCI 总线申请信号 2, 低电平有效。
W11	PCIREQ3N	I _S /O	-	3.3	PCI 总线申请信号 3, 低电平有效。
W17	PCIREQ4N	I _S /O	-	3.3	PCI 总线申请信号 4, 低电平有效。



2.1.18 电源和地管脚

电源和地管脚如表 2-19 所示。

表2-19 电源和地管脚

Pin	管脚名称	类型	电压 (V)	描述
USB 2.0 OTG				
M5	OTGVDD12	P	1.2	USB 2.0 OTG 1.2V 数字电源。
L4	OTGVDDA33C	P	3.3	USB 2.0 OTG 3.3V 模拟电源。
K4	OTGVSSA33C	G	0	USB 2.0 OTG 3.3V 模拟电源对应地。
M3	OTGVSS	G	0	USB 2.0 OTG 的数字地。
M2	OTGVDDA33T	P	3.3	USB 2.0 OTG 3.3V 模拟电源。
M1	OTGVDDA33T	P	3.3	USB 2.0 OTG 3.3V 模拟电源。
L3	OTGVSSA33T	G	0	USB 2.0 OTG 3.3V 模拟电源对应地。
K2	OTGVSSA33T	G	0	USB 2.0 OTG 3.3V 模拟电源对应地。
K1	OTGVSSA33T	G	0	USB 2.0 OTG 3.3V 模拟电源对应地。
PLL				
V22	AVDD33_PLL	P	3.3	PLL 3.3V 模拟电源。
V23	AVSS33_PLL	G	0	PLL 3.3V 模拟电源对应地。
T19	DVDD12_0	P	1.2	PLL 1.2V 数字电源。
T20	VSS_0	G	0	PLL 1.2V 数字地。
USB 1.1 HOST				
AA4	SAVDD33	P	3.3	USB 1.1 HOST 3.3V 模拟电源。
Y4	SAVSS	G	0	USB 1.1 HOST 3.3V 模拟电源对应地。
DDR2				
E19	DVDD18	P	1.8	DDR2 1.8V 数字电源。
E18	DVDD18	P	1.8	
E14	DVDD18	P	1.8	
E13	DVDD18	P	1.8	
E9	DVDD18	P	1.8	
E6	DVDD18	P	1.8	



Pin	管脚名称	类型	电压 (V)	描述
H16	DVDD18	P	1.8	
H15	DVDD18	P	1.8	
H13	DVDD18	P	1.8	
H11	DVDD18	P	1.8	
H9	DVDD18	P	1.8	
H8	DVDD18	P	1.8	
D8	VREF3	P	0.9	DDR2 0.9V 参考电源。
E8	VSSREF3	G	0	DDR2 0.9V 参考电源对应地。
D12	VREF2	P	0.9	DDR2 0.9V 参考电源。
E12	VSSREF2	G	0	DDR2 0.9V 参考电源对应地。
D17	VREF1	P	0.9	DDR2 0.9V 参考电源。
E17	VSSREF1	G	0	DDR2 0.9V 参考电源对应地。
其它数字电源地				
W19	DVDD33	P	3.3	3.3V 数字电源。
W18	DVDD33	P	3.3	
W15	DVDD33	P	3.3	
W12	DVDD33	P	3.3	
W9	DVDD33	P	3.3	
W6	DVDD33	P	3.3	
W5	DVDD33	P	3.3	
T5	DVDD33	P	3.3	
P19	DVDD33	P	3.3	
L5	DVDD33	P	3.3	
K19	DVDD33	P	3.3	
G19	DVDD33	P	3.3	
G5	DVDD33	P	3.3	
F19	DVDD33	P	3.3	
T16	DVDD33	P	3.3	
T15	DVDD33	P	3.3	



Pin	管脚名称	类型	电压 (V)	描述
T13	DVDD33	P	3.3	1.2V 核电压电源。
T11	DVDD33	P	3.3	
T9	DVDD33	P	3.3	
T8	DVDD33	P	3.3	
R16	DVDD33	P	3.3	
R8	DVDD33	P	3.3	
N16	DVDD33	P	3.3	
N8	DVDD33	P	3.3	
L16	DVDD33	P	3.3	
L8	DVDD33	P	3.3	
J16	DVDD33	P	3.3	
J8	DVDD33	P	3.3	
T14	DVDD12	P	1.2	数字地。
T12	DVDD12	P	1.2	
T10	DVDD12	P	1.2	
P16	DVDD12	P	1.2	
P8	DVDD12	P	1.2	
M16	DVDD12	P	1.2	
M8	DVDD12	P	1.2	
K16	DVDD12	P	1.2	
K8	DVDD12	P	1.2	
H14	DVDD12	P	1.2	
H12	DVDD12	P	1.2	
H10	DVDD12	P	1.2	
AC23	VSS	G	0	
AC1	VSS	G	0	
AA21	VSS	G	0	
AA12	VSS	G	0	
AA10	VSS	G	0	
AA3	VSS	G	0	



Pin	管脚名称	类型	电压 (V)	描述
W14	VSS	G	0	
W8	VSS	G	0	
V19	VSS	G	0	
U5	VSS	G	0	
T22	VSS	G	0	
R19	VSS	G	0	
N19	VSS	G	0	
M21	VSS	G	0	
M4	VSS	G	0	
L19	VSS	G	0	
H21	VSS	G	0	
H5	VSS	G	0	
H2	VSS	G	0	
E5	VSS	G	0	
D20	VSS	G	0	
C19	VSS	G	0	
C17	VSS	G	0	
C15	VSS	G	0	
C11	VSS	G	0	
C9	VSS	G	0	
C7	VSS	G	0	
C5	VSS	G	0	
C3	VSS	G	0	
B22	VSS	G	0	
B13	VSS	G	0	
A23	VSS	G	0	
A13	VSS	G	0	
A1	VSS	G	0	
R15	VSS	G	0	
R14	VSS	G	0	



Pin	管脚名称	类型	电压 (V)	描述
R13	VSS	G	0	
R12	VSS	G	0	
R11	VSS	G	0	
R10	VSS	G	0	
R9	VSS	G	0	
P15	VSS	G	0	
P14	VSS	G	0	
P13	VSS	G	0	
P12	VSS	G	0	
P11	VSS	G	0	
P10	VSS	G	0	
P9	VSS	G	0	
N15	VSS	G	0	
N14	VSS	G	0	
N13	VSS	G	0	
N12	VSS	G	0	
N11	VSS	G	0	
N10	VSS	G	0	
N9	VSS	G	0	
M15	VSS	G	0	
M14	VSS	G	0	
M13	VSS	G	0	
M12	VSS	G	0	
M11	VSS	G	0	
M10	VSS	G	0	
M9	VSS	G	0	
L15	VSS	G	0	
L14	VSS	G	0	
L13	VSS	G	0	
L12	VSS	G	0	



Pin	管脚名称	类型	电压 (V)	描述
L11	VSS	G	0	
L10	VSS	G	0	
L9	VSS	G	0	
K15	VSS	G	0	
K14	VSS	G	0	
K13	VSS	G	0	
K12	VSS	G	0	
K11	VSS	G	0	
K10	VSS	G	0	
K9	VSS	G	0	
J15	VSS	G	0	
J14	VSS	G	0	
J13	VSS	G	0	
J12	VSS	G	0	
J11	VSS	G	0	
J10	VSS	G	0	
J9	VSS	G	0	

2.2 管脚复用

2.2.1 VI 管脚复用



说明

此处以 Hi3511 芯片为例进行介绍, Hi3512 芯片的相关内容请参见“14 Hi3511 与 Hi3512 差异说明”。

Hi3511 有 4 个 VI 接口, 其中 VI0 和 VI2 支持 BT.601 接口。VI0 和 VI2 的同步信号的管脚复用较为复杂, 其他数据信号与 GPIO 管脚存在复用关系。具体的复用情况如表 2-20、表 2-21、表 2-22、表 2-23、表 2-24 和表 2-25 所示。



表2-20 VI0 同步信号管脚复用

主功能信号 SC_PERCTRL1[1:0]=0b01	复用信号 1 SC_PERCTRL1[1:0]=0b00		复用信号 2 SC_PERCTRL1[1:0]=0b11		复用信号 3 SC_PERCTRL1[1:0]=0b10	
VI0HS	GPIO0_0	通用 GPIO	VI0DAT0	高清 10bit 模式, VI0 数据 0 输入。	INTRN	外部中断输入, 低电平有效。
VI0VS	GPIO0_1	通用 GPIO	VI0DAT1	高清 10bit 模式, VI0 数据 1 输入。	SPICSN1	SPI 的片选 1, 低电平有效。

表2-21 VI0 数据信号管脚复用

主功能信号 SC_PERCTRL1[2]=0b0	复用信号 1 SC_PERCTRL1[2]=0b1	
VI0DAT2	GPIO4_0	通用 GPIO。
VI0DAT3	GPIO4_1	通用 GPIO。
VI0DAT4	GPIO4_2	通用 GPIO。
VI0DAT5	GPIO4_3	通用 GPIO。
VI0DAT6	GPIO4_4	通用 GPIO。
VI0DAT7	GPIO4_5	通用 GPIO。
VI0DAT8	GPIO4_6	通用 GPIO。
VI0DAT9	GPIO4_7	通用 GPIO。

表2-22 VI1 数据信号管脚复用

主功能信号 SC_PERCTRL1[23]=0b0	复用信号 1 SC_PERCTRL1[23]=0b1	
VI1DAT0	GPIO6_4	通用 GPIO。
VI1DAT1	GPIO6_5	通用 GPIO。
VI1DAT2	GPIO6_6	通用 GPIO。
VI1DAT3	GPIO6_7	通用 GPIO。
VI1DAT4	GPIO7_0	通用 GPIO。
VI1DAT5	GPIO7_1	通用 GPIO。
VI1DAT6	GPIO7_2	通用 GPIO。



VI1DAT7	GPIO7_3	通用 GPIO。
---------	---------	----------

表2-23 VI2 同步信号管脚复用

主功能信号 SC_PERCTRL1 [8:7]=0b01	复用信号 1 SC_PERCTRL1[8:7]=0b00	复用信号 2 SC_PERCTRL1[8:7]=0b11	复用信号 3 SC_PERCTRL1[8:7] =0b10			
VI2HS	GPIO0_5	通用 GPIO	VI2DAT0	高清 10bit 模式， VI2 数据 0 输入。	URXD2	UART2 数 据接收。
VI2VS	GPIO0_6	通用 GPIO	VI2DAT1	高清 10bit 模式， VI2 数据 1 输入。	UTXD2	UART2 数 据发送。

表2-24 VI2 数据信号管脚复用

主功能信号 SC_PERCTRL1[3]=0b0	复用信号 1 SC_PERCTRL1[3]=0b1	
VI2DAT2	GPIO3_6	通用 GPIO。
VI2DAT4	GPIO7_7	通用 GPIO。
VI2DAT6	GPIO3_7	通用 GPIO。

表2-25 VI3 数据信号管脚复用

主功能信号 SC_PERCTRL1[6]=0b0	复用信号 1 SC_PERCTRL1[6]=0b1	
VI3DAT0	GPIO5_0	通用 GPIO。
VI3DAT1	GPIO5_1	通用 GPIO。
VI3DAT2	GPIO5_2	通用 GPIO。
VI3DAT3	GPIO5_3	通用 GPIO。
VI3DAT4	GPIO5_4	通用 GPIO。
VI3DAT5	GPIO5_5	通用 GPIO。
VI3DAT6	GPIO5_6	通用 GPIO。
VI3DAT7	GPIO5_7	通用 GPIO。



2.2.2 ETH 管脚复用

ETH 部分功能管脚与 GPIO 管脚存在复用关系，当 ETH 采用全双工模式工作时，可以将这部分管脚配置成 GPIO。复用信息如表 2-26 所示。

表2-26 ETH 管脚复用

主功能信号	复用信号 1	
SC_PERCTRL1[24]=0b0	SC_PERCTRL1[24]=0b1	
ERXERR	GPIO3_0	通用 GPIO。
ECRS	GPIO3_1	通用 GPIO。
ECOL	GPIO3_2	通用 GPIO。

2.2.3 MMC 管脚复用

MMC 管脚与 GPIO 管脚存在复用关系，缺省为 GPIO。复用信息如表 2-27 所示。

表2-27 MMC 管脚复用

主功能信号	复用信号 1	
SC_PERCTRL1[9]=0b1	SC_PERCTRL1[9]=0b0	
SDIOCMD	GPIO1_3	通用 GPIO。
SDIODAT0	GPIO1_4	通用 GPIO。
SDIODAT1	GPIO1_5	通用 GPIO。
SDIODAT2	GPIO1_6	通用 GPIO。
SDIODAT3	GPIO2_1	通用 GPIO。
SDIOCK	GPIO2_2	通用 GPIO。

2.2.4 IR 管脚复用

IR 管脚与 GPIO 管脚存在复用关系，复用信息如表 2-28 所示。

表2-28 IR 管脚复用

主功能信号	复用信号 1	
SC_PERCTRL1[17]=0b0	SC_PERCTRL1[17]=0b1	
IRRCV	GPIO2_3	通用 GPIO。



2.2.5 I²C 管脚复用

I²C 管脚与 GPIO 管脚存在复用关系，复用信息如表 2-29 所示。

表2-29 I²C 管脚复用

主功能信号 SC_PERCTRL1[14]=0b0	复用信号 1 SC_PERCTRL1[14]=0b1	
SDA	GPIO3_3	通用 GPIO。
SCL	GPIO3_4	通用 GPIO。

2.2.6 UART 管脚复用

UART1 的管脚与 GPIO 管脚存在复用关系，复用信息如表 2-30 所示。

表2-30 UART1 管脚复用

主功能信号 SC_PERCTRL1[16]=0b1	复用信号 1 SC_PERCTRL1[16]=0b0	
URXD1	GPIO2_4	通用 GPIO。
UTXD1	GPIO2_5	通用 GPIO。
URTSN1	GPIO2_6	通用 GPIO。
UCTSN1	GPIO2_7	通用 GPIO。

2.2.7 SPI 管脚复用

SPI 管脚与 GPIO 管脚存在复用关系，复用信息如表 2-31 所示。

表2-31 SPI 管脚复用

主功能信号 SC_PERCTRL1[10]=0b1	复用信号 1 SC_PERCTRL1[10]=0b0	
SPIDI	GPIO6_0	通用 GPIO。
SPIDO	GPIO7_6	通用 GPIO。
SPICK	GPIO7_5	通用 GPIO。
SPICSN0	GPIO6_1	通用 GPIO。



SPI 的两个片选管脚 SPICSN0 和 SPICSN1（请参见表 2-20）的选择如表 2-32 所示。

表2-32 SPI 片选选择

主功能信号 SC_PERCTRL1[12]=0b0	复用信号 1 及描述 SC_PERCTRL1[12]=0b1	
SPICSN0	SPICSN1	SPI 的片选 1。

2.2.8 SIO 管脚复用

SIO0、SIO1 管脚与 GPIO 管脚存在复用关系，复用信息分别如表 2-33 和表 2-34 所示。

表2-33 SIO0 管脚复用

主功能信号 SC_PERCTRL1[18]=0b1	复用信号 1 SC_PERCTRL1[18]=0b0	
SIO0XFS	GPIO6_2	通用 GPIO。
主功能信号 SC_PERCTRL1[11]=0b1	复用信号 1 SC_PERCTRL1[11]=0b0	
SIO0XCK	GPIO6_3	通用 GPIO。
主功能信号 SC_PERCTRL1[19]=0b1	复用信号 1 SC_PERCTRL1[19]=0b0	
ACKOUT	GPIO3_5	通用 GPIO。

表2-34 SIO1 管脚复用

主功能信号 SC_PERCTRL1[5]=0b1	复用信号 1 SC_PERCTRL1[5]=0b0	
SIO1DI	GPIO0_4	通用 GPIO。
SIO1RFS	GPIO0_2	通用 GPIO。
SIO1RCK	GPIO0_3	通用 GPIO。

2.2.9 SMI 管脚复用

SMI 管脚与 GPIO 管脚存在复用关系，复用信息如表 2-35 所示。



表2-35 SMI 管脚复用

主功能信号 SC_PERCTRL1[20]=0b1	复用信号 1 SC_PERCTRL1[20]=0b0	
EBICS1N	GPIO1_1	通用 GPIO。
主功能信号 SC_PERCTRL1[21]=0b0	复用信号 1 SC_PERCTRL1[21]=0b1	
EBIADR24	GPIO1_2	通用 GPIO。
主功能信号 SC_PERCTRL1[4]=0b0	复用信号 1 SC_PERCTRL1[4]=0b1	
EBIRDYN	GPIO7_4	通用 GPIO。

2.2.10 DDR2 管脚复用

DDR2 管脚与 GPIO 管脚存在复用关系，复用信息如表 2-36 所示。

表2-36 DDR2 管脚复用

主功能信号 SC_PERCTRL1[15]=0b1	复用信号 1 SC_PERCTRL1[15]=0b0	
DDRMRCVO	GPIO0_7	通用 GPIO。
DDRMRCVI	GPIO1_0	通用 GPIO。

2.2.11 PCI 管脚复用

PCI 管脚与 GPIO 管脚存在复用关系，复用信息如表 2-37 所示。

表2-37 PCI 管脚复用

主功能信号 SC_PERCTRL4[17]=0b0, 即 PCI 为从模式	复用信号 1 SC_PERCTRL4[17]=0b1, 即 PCI 为主模式	
PCIIDSEL	PCIINTBN	PCI 中断 B 输入信号, 低电平有效。
主功能信号 SC_PERCTRL4[17]=0b1, 即 PCI 为主模式	复用信号 1 SC_PERCTRL4[17]=0b0, 即 PCI 为从模式	
PCIREQ0N	PCIGRANT_SL AVEN	从模式下, 输入的 PCI 总线仲裁信号, 低电平有效。



PCIGRANT0N	PCIREQ_SLAVE	从模式下, 输入的 PCI 总线请求信号, 低电平有效。
主功能信号 SC_PERCTRL1[13]=0b1	复用信号 1 SC_PERCTRL1[13]=0b0	
PCIREQ4N	GPIO1_7	通用 GPIO。
PCIGRANT4N	GPIO2_0	通用 GPIO。

2.3 上下电顺序推荐

推荐先上高电压, 后上低电压; 先关低电压, 后关高电压。

2.4 外部中断

请参见表 3-13。

2.5 电气特性

2.5.1 DC/AC 参数

DC/AC 参数如表 2-38~表 2-40 所示。

表2-38 DC 参数表 (DVDD33=3.3V)

符号	参数	最小值	典型值	最大值	单位	说明
V _{IH}	非 PCI 高电平输入电压	2.0	-	5.5	V	-
	PCI 高电平输入电压	0.5×DVD D33	-	-	V	-
V _{IL}	非 PCI 低电平输入电压	-0.3	-	0.8	V	-
	PCI 低电平输入电压	-	-	0.3×DVD D33	V	-
I _L	输入漏电流	-	-	±1	μA	-
I _{OZ}	三态输出漏电流	-	-	±1	μA	-
V _{OH}	非 PCI 高电平输出电压	2.4	-	-	V	-
	PCI 高电平输出电压	0.9×DVD D33	-	-	V	-



符号	参数	最小值	典型值	最大值	单位	说明
V _{OL}	非 PCI 低电平输出电压	-	-	0.4	V	-
	PCI 低电平输出电压	-	-	0.1×DV _{D33}	V	
R _{PU}	上拉电阻	62	77	112	kΩ	-
R _{PD}	下拉电阻	48	85	174	kΩ	-

表2-39 DC 参数表 (DVDD18=1.8V)

符号	参数	最小值	典型值	最大值	单位	说明
V _{ref}	参考电压	0.49×DV _{DD18}	0.50×DV _{DD18}	0.51×DV _{DD18}	V	-
V _{IH(dc)}	高电平输入电压	V _{ref} +0.125	-	DVDD18+0.3	V	-
V _{IL(dc)}	低电平输入电压	-0.3	-	V _{ref} -0.125	V	-
I _{OH(dc)}	高电平输出电流	−13.4	-	-	mA	V _{oh(dc)} =1.42V
I _{OL(dc)}	低电平输出电流	13.4	-	-	mA	V _{ol(dc)} =0.28V

表2-40 AC 参数 (DVDD18=1.8V)

符号	参数	最小值	典型值	最大值	单位	说明
V _{IH(ac)}	高电平 AC 输入电压	V _{ref} +0.25	-	-	V	-
V _{IL(ac)}	低电平 AC 输入电压	-	-	V _{ref} -0.25	V	-

2.5.2 极限参数

极限参数如表 2-41 所示。

表2-41 极限参数

符号	参数	使用范围	单位
T _{STG}	存储温度	−65~+150	℃
DVDD33	3.3V 供电电压	−0.5~+4.6	V



符号	参数	使用范围	单位
DVDD18	1.8V 供电电压	-0.5~+2.5	V
DVDD12	Core 供电电压	-0.5~+1.8	V
V _{I_3.3V}	任何一个 3.3V 输入管脚上的电压	-0.5~+6	V
V _{O_3.3V}	任何一个 3.3V 输出管脚上的电压	-0.5~+4.6	V
V _{I_1.8V}	任何一个 1.8V 输入管脚上的电压	-0.5~+2.5	V
V _{O_1.8V}	任何一个 1.8V 输出管脚上的电压	-0.5~+2.5	V

2.5.3 推荐工作条件

推荐工作条件如表 2-42 所示。

表2-42 推荐工作条件

符号	参数	最小值	典型值	最大值	单位
T _{OPT}	操作环境温度	-20	-	85	°C
DVDD12	内部 Core 电源	1.16	1.2	1.32	V
DVDD33	I/O 电源	2.97	3.3	3.63	V
DVDD18	I/O 电源	1.7	1.8	1.9	V
VREF1, VREF2, VREF3	DDR 参考电源	0.49×DVD D18	0.5×DVD D18	0.51×DVD D18	V
RTCBATT	RTC 电池供电	2.5	-	3.3	V
SAVDD33	USB 1.1 模拟电源	3.0	3.3	3.6	V
AVDD33_PLL	PLL 模拟电源	3.0	3.3	3.6	V
OTGVDDA33C OTGVDDA33T	OTG 模拟电源	3.135	3.3	3.465	V
OTGVDD12	OTG 数字电源	1.14	1.2	1.26	V

2.6 PCB 布线建议

请参见《Hi3511/Hi3512 硬件设计 用户指南》。

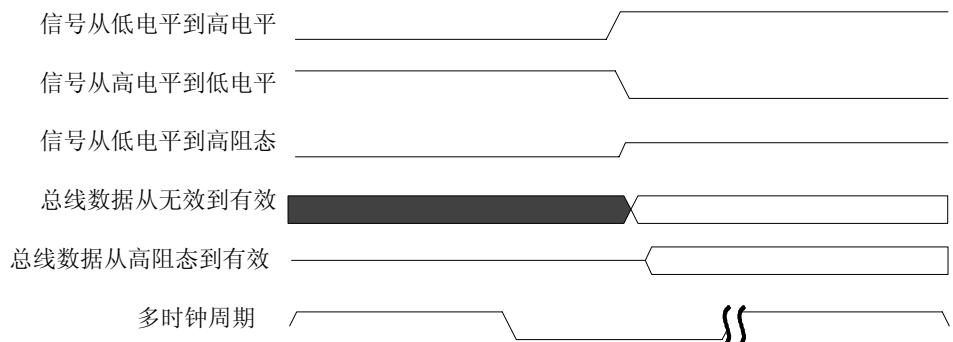


2.7 时序规格

2.7.1 时序图例

图 2-1 介绍了本手册的时序图中的图元。

图2-1 时序图元说明



2.7.2 DDR2 接口时序

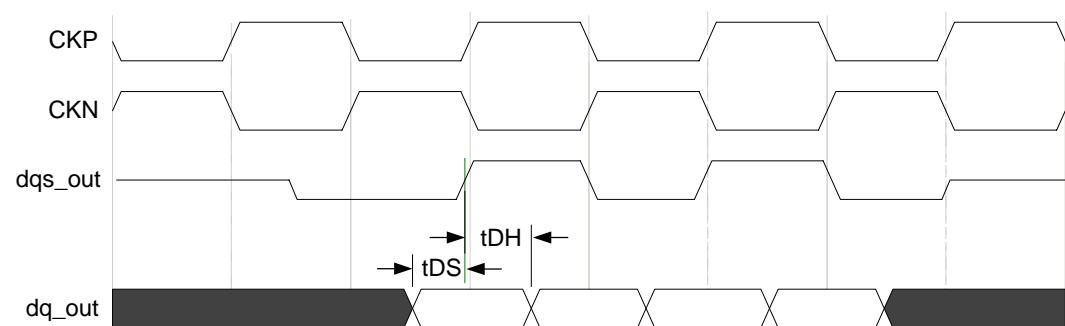
2.7.2.1 写操作时序

dqs_out 相对于 dq_out 的写操作时序

dqs_out 相对于 dq_out 的写操作时序中需要检查的时序参数是 tDS 和 tDH。在 DDR533 中, tDS=0.150; tDH=0.225。

dqs_out 相对 DDR 时钟 CKP/CKN 以及 dq_out 的时序如图 2-2 所示。

图2-2 dqs_out 相对于 dq_out 的写操作时序图

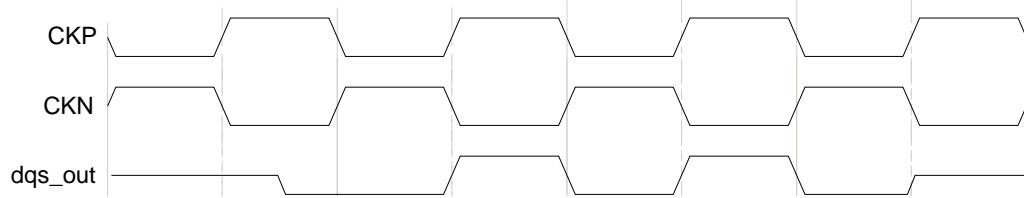




dqs_out 相对于 ck 的写操作时序

dqs_out 相对于 ck 的写操作时序中需要检查的时序参数是 tDSS 和 tDSH，这两个参数的值为 tCK×0.2，时序关系如图 2-3 所示。

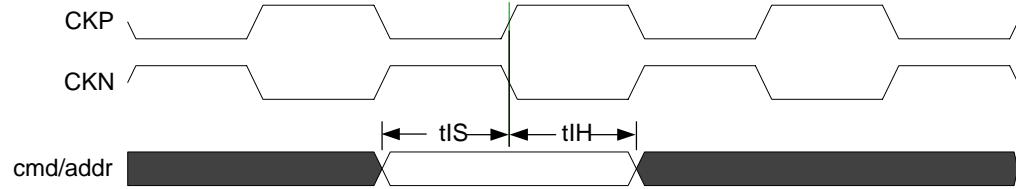
图2-3 dqs_out 相对于 ck 的写操作时序图



cmd/addr 相对于 ck 的写操作时序

cmd/addr 相对于 ck 的写操作时序如图 2-4 所示。

图2-4 cmd/addr 相对于 ck 的写操作时序图



2.7.2.2 读操作时序

cmd/addr 相对于 ck 的读操作时序

请参见“[2.7.2.1 写操作时序](#)”中“[cmd/addr 相对于 ck 的写操作时序](#)”的描述。

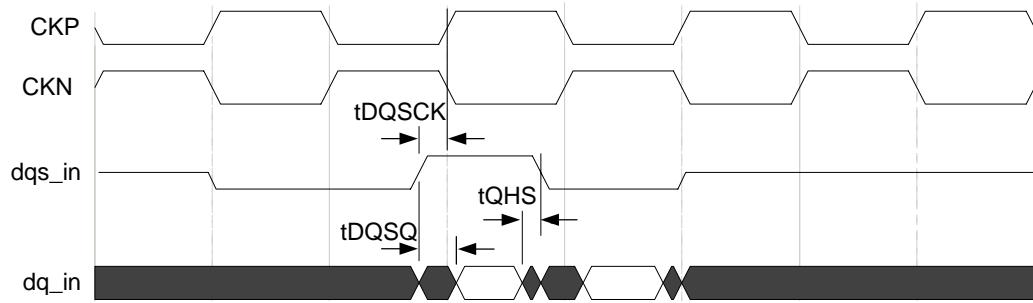
dqs_in 相对于 dq_in 的读操作时序

对于 DDR SDRAM 输出时序，理想情况下，DQS 和 CK 是同相位，但是有 tDQSCK 的偏斜。该参数的值为 0.45ns。tDQSQ 是 dq 和 dqs 之间的抖动，是最晚有效的 dq 相对于 dqs 的抖动，该值为 0.3ns，tQHS 是最早有效的 dq 相对于 dqs 的抖动，其值为 0.4ns。

DDR2 SDRAM 输出时序如图 2-5 所示。



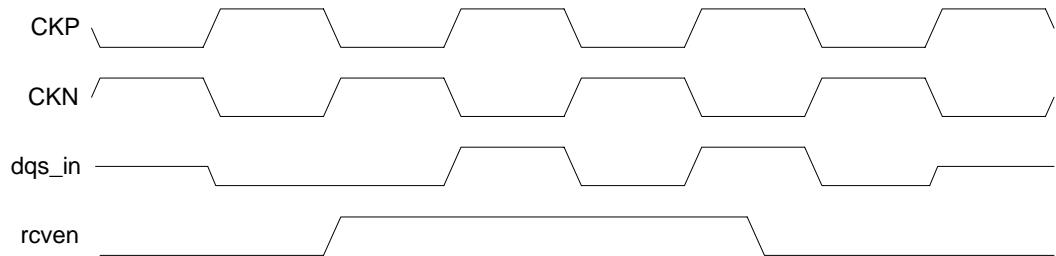
图2-5 DDR2 SDRAM 输出时序图



rcven 相对于 dqs_in 的读操作时序

DDRCVENO 到 DDRCVENI 的走线 rcven，其走线长度等于差分时钟平均长度与 DQS 平均长度之和；rcven 相对于 dqs_in 的读操作时序如图 2-6 所示。

图2-6 rcven 相对于 dqs_in 的读操作时序图



2.7.2.3 时序参数

本文中描述的时序都是 DDR PHY 侧的时序。对于 Hi3511/Hi3512，以 DDR533 的时序参数为依据，具体参数信息如表 2-43～表 2-45 所示。

表2-43 Clock Parameters

参数	典型值	单位
Memory clock frequency	135.00	MHz
PLL clock jitter	0.200	ns
PLL duty cycle	45.000	%
CLK Skew	0.200	ns

表2-44 Memory Device Parameters

参数	典型值	单位
Memory Type	DDR533	-



参数	典型值	单位
tDSS: DQS falling edge-CK setup time	1.480	ns
tDSH: DQS falling edge-CK hold time	1.480	ns
tDS: DQ/DM-DQS setup time	0.150	ns
tDH: DQ/DM-DQS hold time	0.225	ns
DQS-DQ Skew, tDQSQ	0.300	ns
Data hold Skew Factor, tQHS	0.400	ns
Adr/Cmd Setup time, tIS	0.250	ns
Adr/Cmd Hold time, tIH	0.375	ns
tDQSCK: DQS ouput acces time from CKP/CKN (min)	0.450	ns
tDQSCK: DQS ouput acces time from CKP/CKN (max)	0.450	ns

表2-45 Package/Board Parameters

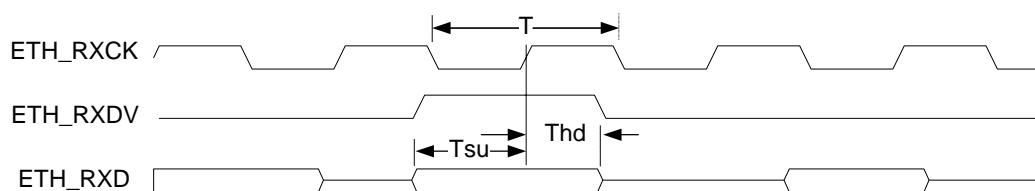
参数	典型值	单位
Package trace skew (max/min)	0.060	ns
Board trace length mismatch (For Dqs/Dq)	0.050	ns
Board trace length mismatch (For others)	0.100	ns
Board and Package Uncertainty(Output)	0.360	ns
Board and Package Uncertainty(Input)	0.360	ns
ASIC Skew	0.200	ns

2.7.3 ETH 接口时序

MII 接口时序

MII 接口接收时序如图 2-7 所示。

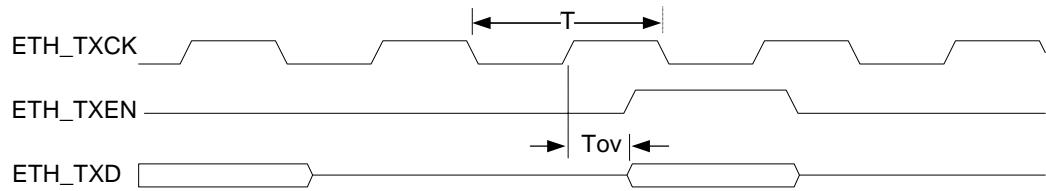
图2-7 MII 接口接收时序图





MII 接口发送时序如图 2-8 所示。

图2-8 MII 接口发送时序图



MII 接口的时序参数表如表 2-46 所示。

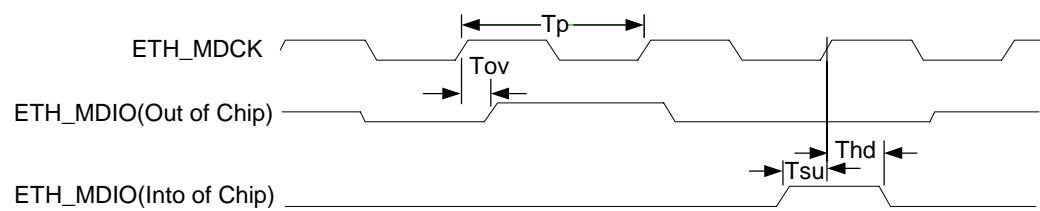
表2-46 25MHz 时 MII 接口的时序参数表

参数	符号	信号	最小值	最大值	单位
MII 时钟周期	T	ETH_RXCK、 ETH_TXCK	400(10Mbit/s)	400	ns
			40(10Mbit/s)	40	ns
MII 信号建立时间	Tsu (RX)	ETH_RXER、 ETH_RXDV、 ETH_RXD[3:0]	10	-	ns
MII 信号保持时间	Thd (RX)	ETH_RXER、 ETH_RXDV、 ETH_RXD[3:0]	10	-	ns
MII 输出信号延时	Tov (MII TX)	ETH_TXD[1:0] 、ETH_TXEN	0	20	ns

MDIO 接口接收时序

MDIO 接口时序图如图 2-9 所示。

图2-9 MDIO 接口时序图



MDIO 接口时序参数如表 2-47 所示。



表2-47 MDIO 接口时序参数

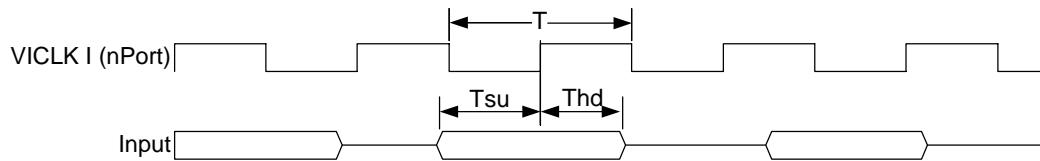
参数	符号	信号	最小值	最大值	单位
MDIO 发送数据延迟时间	T_{ov}	MDIO	37	37	ns
MDIO 时钟周期	T_p	MDC	370 或 740	370 或 740	ns
MDIO 接收数据建立时间	T_{su}	MDIO	10	-	ns
MDIO 接收数据保持时间	T_{hd}	MDIO	10	-	ns

注：MDC 时钟周期 T_p 可通过调整 MDC 频率进行改变，选择 ETH 工作时钟 135MHz 的 100 分频或者 50 分频。

2.7.4 VI 接口时序

VI 接口时序如图 2-10 所示。

图2-10 VI 接口时序图



VI 接口时序参数如表 2-48 所示。

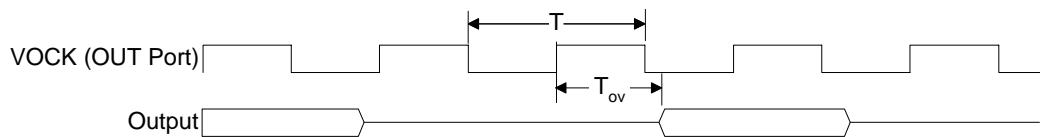
表2-48 VI 接口时序参数

参数	符号	最小值	典型值	最大值	单位
VICLK 时钟周期	T	13.3		-	ns
输入信号建立时间要求	T_{su}	3.5	-	-	ns
输入信号保持时间要求	T_{hd}	4	-	-	ns

2.7.5 VO 接口时序

VO 接口时序如图 2-11 所示。

图2-11 VO 接口时序图





VO 接口时序参数如表 2-49 所示。

表2-49 VO 接口时序参数表

参数	符号	最小值	典型值	最大值	单位
VOCK 时钟周期	T	-	37.03	-	ns
输出信号延时	Tov	3	-	9	ns

2.7.6 PCI 接口时序

PCI 接口时序如图 2-12 和图 2-13 所示。

图2-12 PCI 接口时序(采用芯片内部时钟)

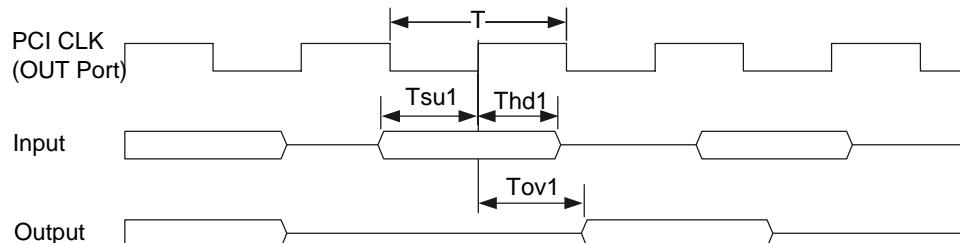
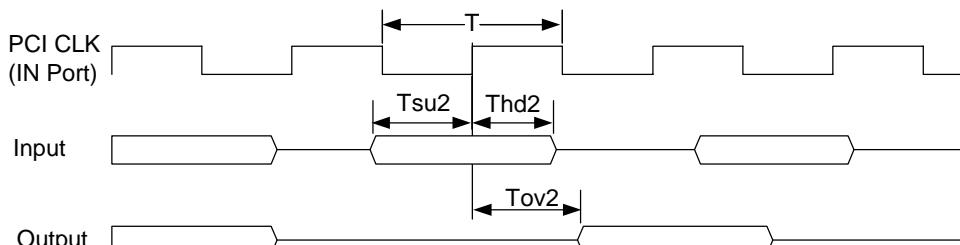


图2-13 PCI 接口时序(采用芯片外部时钟)



PCI 接口时序参数如表 2-49 所示。

表2-50 PCI 接口时序参数表

参数	符号	最小值	典型值	最大值	单位
PCI CLK 时钟周期	T	15.15	-	-	ns
输入信号建立时间	Tos1	5	-	-	ns
	Tos2	4.5	-	-	ns

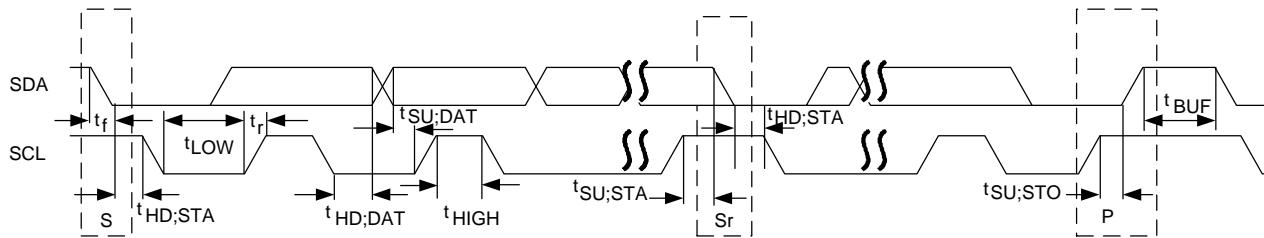


参数	符号	最小值	典型值	最大值	单位
输入信号保持时间	Toh1	0	-	-	ns
	Toh2	0	-	-	ns
输出信号延时	Tov1	4	-	6	ns
	Tov2	2	-	8	ns

2.7.7 I²C 接口时序

I²C 传输时序如图 2-14 所示。

图2-14 I²C 传输时序



I²C 接口时序参数如表 2-51 所示。

表2-51 I²C 接口时序参数表

参数	符号	标准模式		快速模式		单位
		最小值	最大值	最小值	最大值	
SCL (Serial Clock) 时钟频率	f _{SCL}	0	100	0	400	KHz
启动保持时间	t _{HD;STA}	4.0	-	0.6	-	μs
SCL 低电平周期	t _{LOW}	4.7	-	1.3	-	μs
SCL 高电平周期	t _{HIGH}	4.0	-	0.6	-	μs
启动建立时间	t _{SU;STA}	4.7	-	0.6	-	μs
数据保持时间	t _{HD;DAT}	0	3.45	0	0.9	μs
数据建立时间	t _{SU;DAT}	250	-	100	-	ns
SDA、SCL 上升时间	t _r	-	1000	20+0.1C _b	300	ns
SDA、SCL 下降时间	t _f	-	300	20+0.1C _b	300	ns

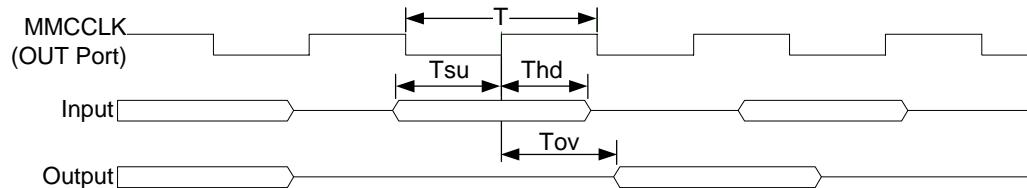


参数	符号	标准模式		快速模式		单位
		最小值	最大值	最小值	最大值	
结束建立时间	$t_{SU,STO}$	4.0	-	0.6	-	μs
开始与结束之间的总线释放时间	t_{BUF}	4.7	-	1.3	-	μs
总线负载	C_b	-	400	-	400	pF
低电平噪声容限	V_{nL}	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
高电平噪声容限	V_{nH}	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

2.7.8 MMC 接口时序

MMC 接口时序如图 2-15 所示。

图2-15 MMC 接口时序



MMC 接口时序参数如表 2-52 所示。

表2-52 MMC 接口时序参数

参数	符号	最小值	典型值	最大值	单位
MMCCLK 时钟周期	T	20.8	-	-	ns
输入信号建立时间	T_{su}	5	-	-	ns
输入信号保持时间	T_{hd}	2.5	-	-	ns
输出信号延时	T_{ov}	6.5	-	14.3	ns

2.7.9 SSP 接口时序



图 2-16 ~ 图 2-18 中，以下缩略语或字母意义不变：

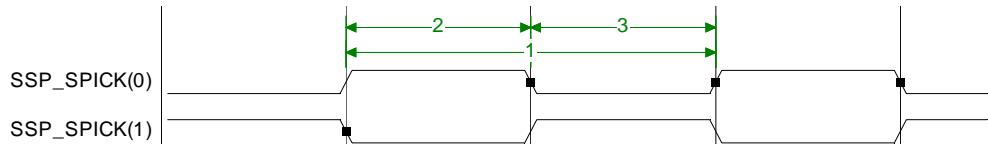
- MSB:Most Significant Bit
- LSB:Least Significant Bit



- SSP_SPICK(0):spo=0
- SSP_SPICK(1):spo=1

SSP 接口时钟时序如图 2-16 所示。

图2-16 SSP_SPICK 时序



SSP 主模式下接口时序分别如图 2-17 和图 2-18 所示。

图2-17 SSP 主模式下接口时序 (sph= 0)

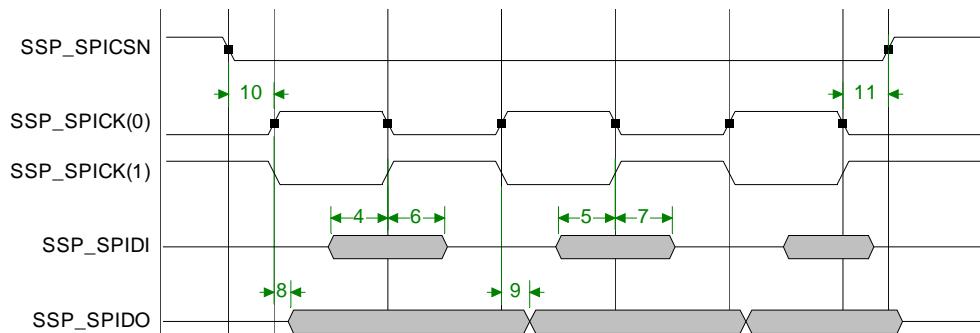
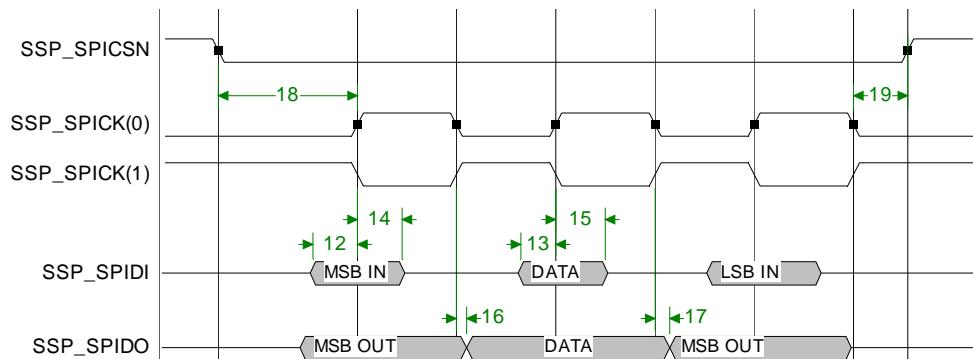


图2-18 SSP 主模式下接口时序 (sph= 1)



SSP 接口时序参数如表 2-53 所示。

表2-53 SSP 接口时序参数

No	参数	符号	最小值	典型值	最大值	单位
1	Cycle time, SSP_SPICK	tc	-	-	-	ns



No	参数	符号	最小值	典型值	最大值	单位
2	Pulse duration, SSP_SPICK high (All Master Modes)	tw1	-	-	-	ns
3	Pulse duration, SSP_SPICK low (All Master Modes)	tw2	-	-	-	ns
4	Setup time, SSP_SPIDI (input) valid before SSP_SPICK (output) falling edge	tsu1	-	-	-	ns
5	Setup time, SSP_SPIDI (input) valid before SSP_SPICK (output) rising edge	tsu2	-	-	-	ns
6	Hold time, SSP_SPIDI (input) valid after SSP_SPICK (output) falling edge	th1	-	-	-	ns
7	Hold time, SSP_SPIDI (input) valid after SSP_SPICK (output) rising edge	th2	-	-	-	ns
8	Delay time, SSP_SPICK (output) rising edge to SSP_SPIDO (output) transition	td1	-	-	-	ns
9	Delay time, SSP_SPICK (output) falling edge to SSP_SPIDO (output) transition	td2	-	-	-	ns
10	Delay time, SSP_SPICSN (output) falling edge to first SSP_SPICK (output) rising or falling edge	td3	-	-	-	ns
11	Delay time, SSP_SPICK (output) rising or falling edge to SSP_SPICSN (output) rising edge	td4	-	-	-	ns
12	Setup time, SSP_SPIDI (input) valid before SSP_SPICK (output) rising edge	tsu3	-	-	-	ns
13	Setup time, SSP_SPIDI (input) valid before SSP_SPICK (output) falling edge	tsu4	-	-	-	ns
14	Hold time, SSP_SPIDI (input) valid after SSP_SPICK (output) rising edge	th3	-	-	-	ns

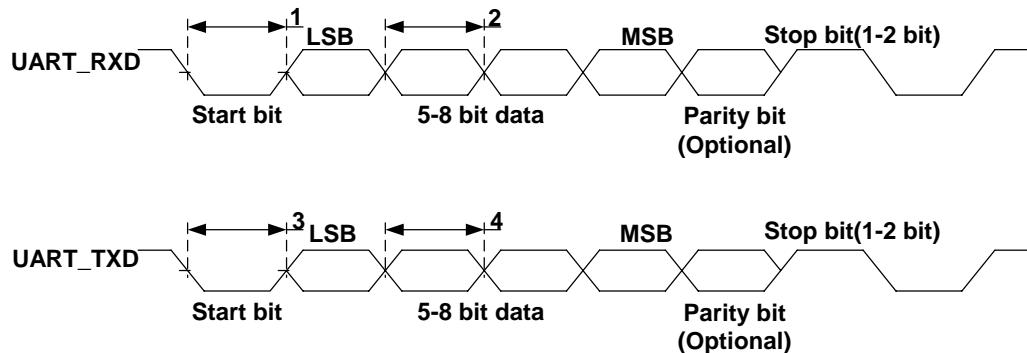


No	参数	符号	最小值	典型值	最大值	单位
15	Hold time, SSP_SPIDI (input) valid after SSP_SPICK (output) falling edge	th4	-	-	-	ns
16	Delay time, SSP_SPICK (output) falling edge to SSP_SPIDO (output) transition	td5	-	-	-	ns
17	Delay time, SSP_SPICK (output) rising edge to SSP_SPIDO (output) transition	td6	-	-	-	ns
18	Delay time, SSP_SPICSN (output) falling edge to first SSP_SPICK (output) rising or falling edge	td7	-	-	-	ns
19	Delay time, SSP_SPICK (output) rising or falling edge to SSP_SPICSN (output) rising edge	td8	-	-	-	ns

2.7.10 UART 接口时序

UART 接口时序如图 2-19 所示。

图2-19 UART 接口时序



UART 接收数据信号 URXD 时序要求如表 2-54 所示。

表2-54 UART 接收数据信号时序参数

No	参数	符号	最小值	典型值	最大值	单位
1	脉冲宽度, 接收 Start Bit	UART_RXD	0.96U	-	1.05U	ns
2	脉冲宽度, 接收 Data Bit	UART_RXD	0.96U	-	1.05U	ns

注: U = UART baud time = 1/baud rate



UART 发送数据信号 UTXD 时序要求如表 2-55 所示。

表2-55 UART 发送数据信号时序参数

No	参数	符号	最小值	典型值	最大值	单位
3	脉冲宽度, 传送 Start Bit	UART_TXD	U-2	-	U+2	ns
4	脉冲宽度, 传送 Data Bit	UART_TXD	U-2	-	U+2	ns

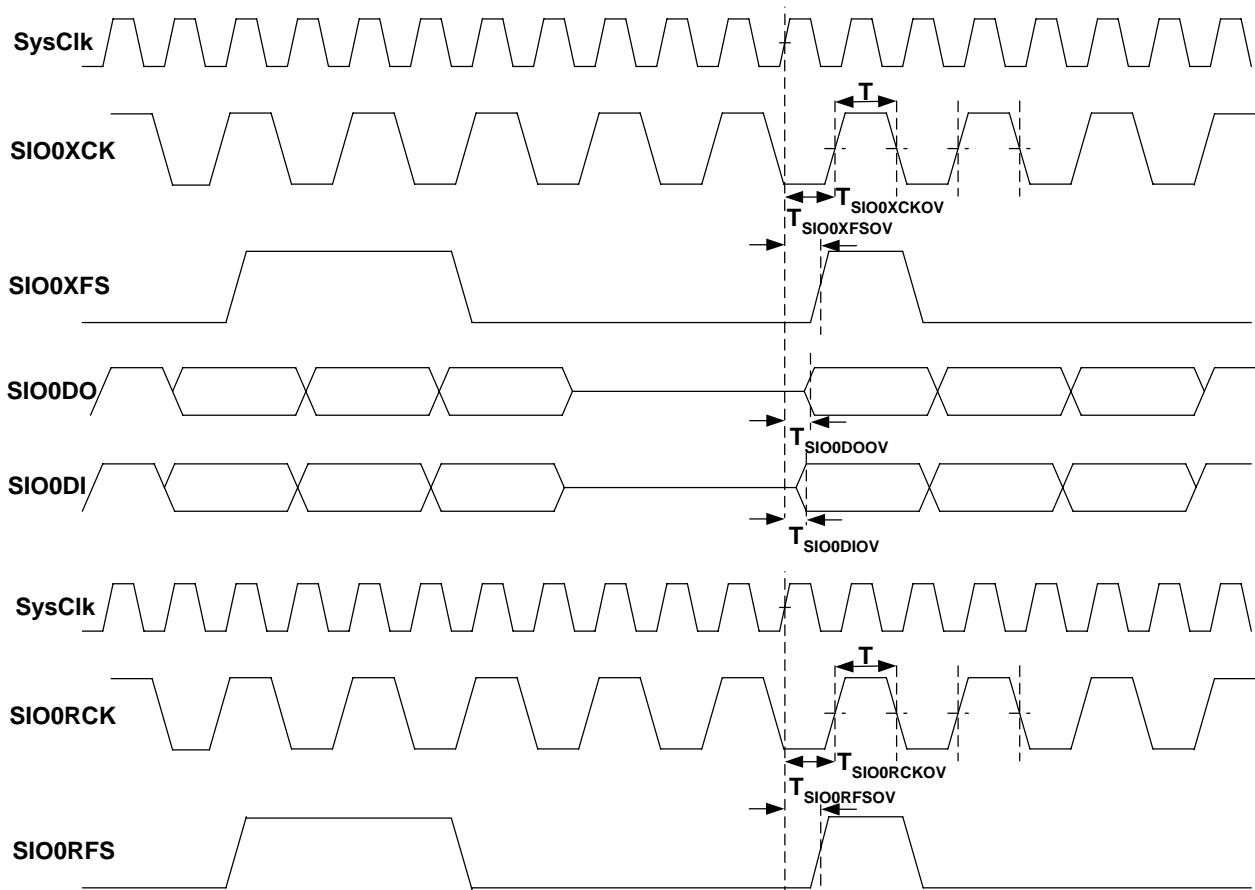
注: U = UART baud time = 1/baud rate

2.7.11 SIO 接口时序

SIO0 接口时序

SIO0 接口时序如图 2-20 所示。

图2-20 SIO0 接口时序





SIO0 接口时序参数如表 2-56 所示。

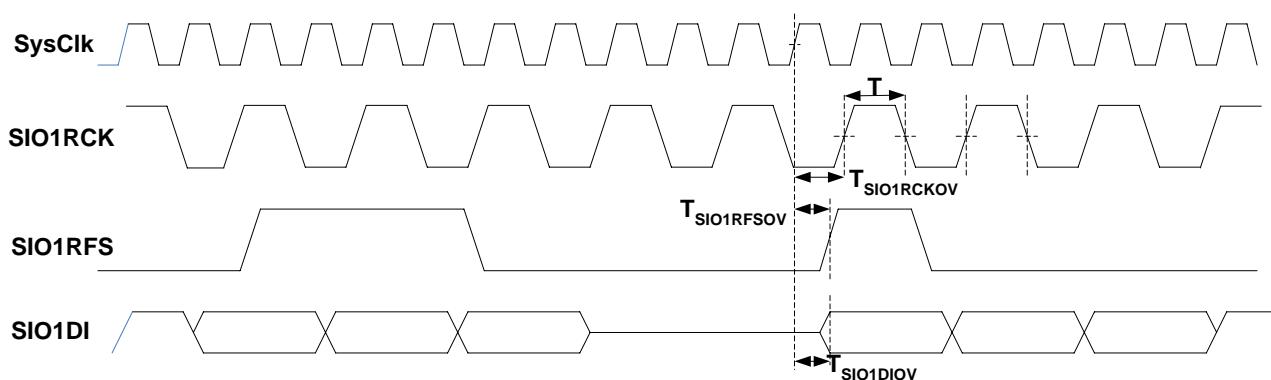
表2-56 SIO0 接口时序参数

参数	符号	最小值	典型值	最大值	单位
SIO0CLK 时钟周期	T	651.04	-	3906.25	ns
SIO0XCK 输出信号延时(int)	$T_{SIO0XCKOV}$	2.19	-	6.04	ns
SIO0XCK 输出信号延时(ext)	$T_{SIO0XCKOV}$	4.42	-	13.09	ns
SIO0XFS 输出信号延时(int)	$T_{SIO0XFSOV}$	2.18	-	5.74	ns
SIO0XFS 输出信号延时(ext)	$T_{SIO0XFSOV}$	4.44	-	12.33	ns
SIO0DO 输出信号延时(int)	$T_{SIO0DOOV}$	2.19	-	5.15	ns
SIO0DO 输出信号延时(ext)	$T_{SIO0DOOV}$	4.44	-	11.22	ns
SIO0RCK 输出信号延时	$T_{SIO0RCKOV}$	1.3	-	1.9	ns
SIO0RFS 输出信号延时	$T_{SIO0RFSOV}$	1.16	-	1.62	ns
SIO0DI 输出信号延时	$T_{SIO0DIOV}$	0.86	-	1.92	ns

SIO1 接口时序

SIO1 接口时序如图 2-21 所示。

图2-21 SIO1 接口时序



SIO0 接口时序参数如表 2-57 所示。

表2-57 SIO0 接口时序参数

参数	符号	最小值	典型值	最大值	单位
SIO1CLK 时钟周期	T	651.04	-	3906.25	ns



参数	符号	最小值	典型值	最大值	单位
SIO1RCK 输出信号延时	$T_{SIO1RCKOV}$	1.69	-	3.87	ns
SIO1RFS 输出信号延时	$T_{SIO1RFSOV}$	1.67	-	3.84	ns
SIO1DI 输出信号延时	$T_{SIO1DIOV}$	1.48	-	3.39	ns

2.7.12 SMI 接口时序

SMI 接口时序请参见“4.2.4 功能描述”中“功能原理”部分的内容。

2.8 封装和管脚分布

2.8.1 封装

Hi3511/Hi3512 芯片采用 $0.13\mu\text{m}$ 工艺、TFBGA441 封装，封装尺寸 $19\text{mm} \times 19\text{mm}$ ，管脚间距为 0.8mm 。具体封装尺寸请参见图 2-22~图 2-26，尺寸参数请参见表 2-58。

图2-22 芯片尺寸视图（顶视图）

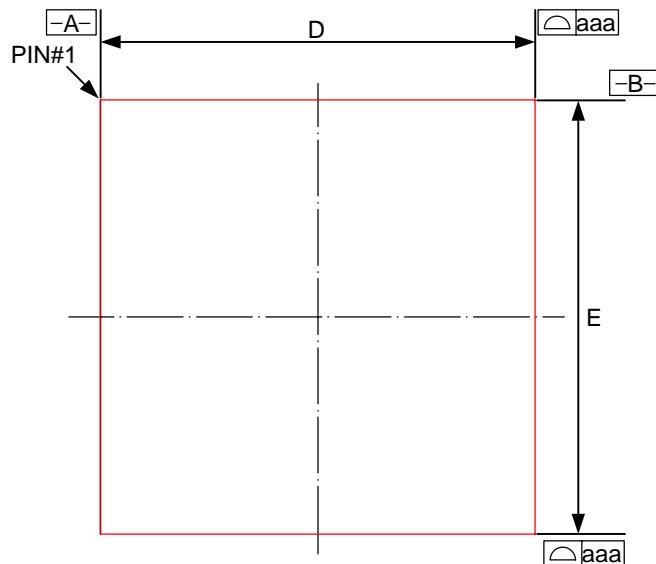




图2-23 芯片尺寸视图（底视图）

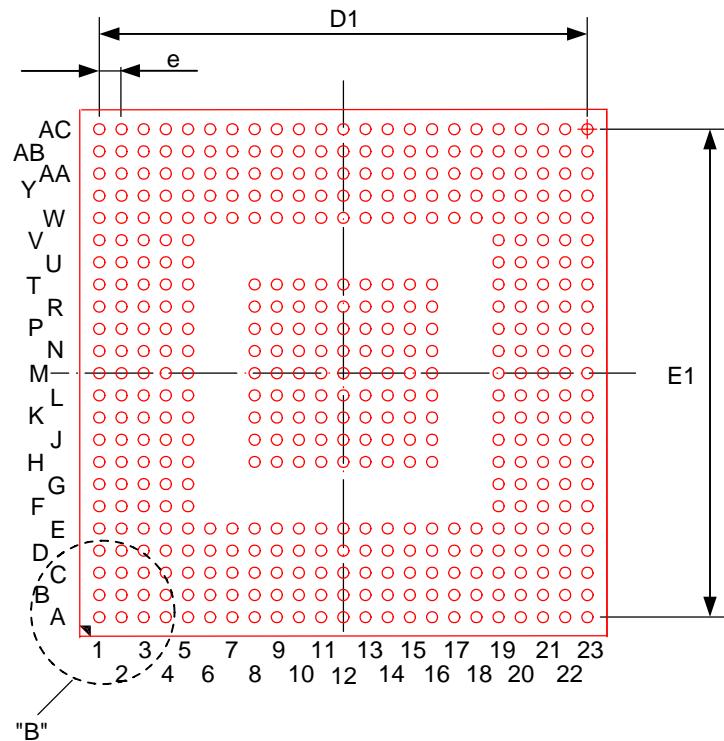
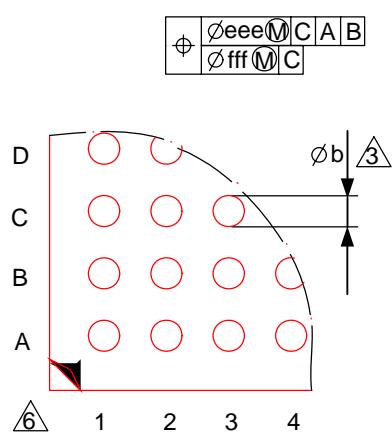


图2-24 Detail B 的放大图



DETAIL:"B"



图2-25 芯片尺寸视图 (侧视图)

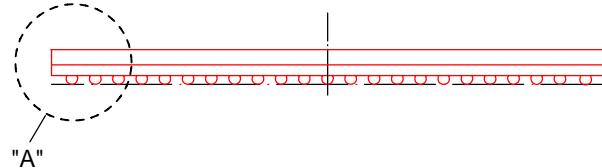


图2-26 Detail A 的放大图

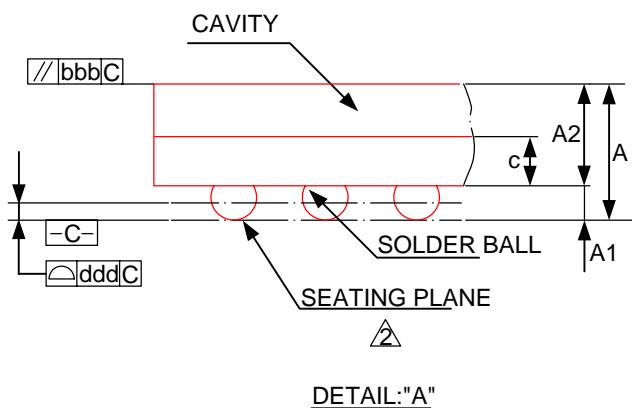


表2-58 封装参数说明表

参数	尺寸 (mm)		
	最小值	典型值	最大值
A	-	-	1.40
A1	0.25	0.30	0.35
A2	0.91	0.96	1.01
b	0.35	0.40	0.45
c	0.22	0.26	0.3
D	18.90	19.00	19.10
E	18.90	19.00	19.10
D1	-	17.60	-
E1	-	17.60	-
e	-	0.80	-
aaa	0.15		
bbb	0.20		
ddd	0.20		



参数	尺寸 (mm)		
	最小值	典型值	最大值
eee	0.15		
fff	0.08		
MD/ME	23/23		

2.8.2 管脚分布



说明

此处以 Hi3511 芯片为例进行介绍, Hi3512 芯片的相关内容请参见“14 Hi3511 与 Hi3512 差异说明”。

Hi3511 的管脚有 441 个, 管脚数目统计表如表 2-59 所示。

表2-59 Hi3511 管脚数目统计表

管脚类别	数量
I/O	282
数字电源	54
数字地	88
其他/模拟电源	7
其他/模拟地	7
DDR2 参考电源	3
总计	441

Hi3511 管脚分布图（顶视图）如图 2-27 和图 2-28 所示。

图2-27 Hi3511 管脚分布图 (1~12)

	1	2	3	4	5	6	7	8	9	10	11	12
A	VSS	DDRDQ29	DDRDQ31	DDRDQ26	DDRDQ23	DDRDQ18	DDRDQ22	DDRDQ20	DDRADR1	DDRADR10	DDRADR12	DDRCKN1
B	UTXD1	UTXD0	DDRDQ24	DDRDQS3	DDRDQ27	DDRDQ21	DDRDQS2	DDRDQ17	DDRADR7	DDRADR4	DDRADR11	DDPCKP1
C	URXD1	URXD0	VSS	DDRDQ25	VSS	DDRDQ16	VSS	DDRADR13	VSS	DDRADR3	VSS	DDRADR2
D	UCTSN1	URTSN1	DDRMRCVO	DDRDIM3	DDRDQ30	DDRDIM2	DDRDQ19	VREF3	DDRADR8	DDRADR5	DDRADR0	VREF2
E	VODAT6	VOCK	VODAT7	DDRMRCVI	VSS	DVDD18	DDRDQ28	VSSREF3	DVDD18	DDRADR9	DDRADR6	VSSREF2
F	VODAT1	VODAT2	VODAT3	VODAT4	VODAT5							
G	VODAT0	SIO0RCK	SIO0XCK	SIO0DO	DVDD33							
H	SIO1RCK	VSS	ACKOUT	SIO0XFS	VSS			DVDD18	DVDD18	DVDD12	DVDD18	DVDD12
J	XOUT24	XIN24	SIO1DI	SIO1RFS	SIO0DI			DVDD33	VSS	VSS	VSS	VSS
K	OTGVSSA33T	OTGVSSA33T	OTGREXT	OTGVSSA33C	SIO0RFS			DVDD12	VSS	VSS	VSS	VSS
L	OTGDM	OTGDP	OTGVSSA33T	OTGVDDA33C	DVDD33			DVDD33	VSS	VSS	VSS	VSS
M	OTGVDDA33T	OTGVDDA33T	OTGVSS	VSS	OTGVDD12			DVDD12	VSS	VSS	VSS	VSS
N	OTGID	OTGVBUS	VI1DAT1	VI1DAT2	VI1DAT0			DVDD33	VSS	VSS	VSS	VSS
P	VI1DAT6	VI1DAT3	VI1DAT4	VI1DAT7	VI1DAT5			DVDD12	VSS	VSS	VSS	VSS
R	VI1CK	VI0DAT3	VI0DAT2	VI0DAT4	VI0HS			DVDD33	VSS	VSS	VSS	VSS
T	VI0CK	VI0DAT5	VI0VS	VI0DAT7	DVDD33			DVDD33	DVDD33	DVDD12	DVDD33	DVDD12
U	VI0DAT9	VI0DAT8	VI0DAT6	VI2DAT6	VSS							
V	VI2DAT4	VI2DAT5	VI2DAT3	VI2VS	VI2DAT2							
W	VI2DAT8	VI2HS	VI3DAT1	VI3DAT0	DVDD33	DVDD33	ERXD3	VSS	DVDD33	PCIRSTN	PCIREQ3N	DVDD33
Y	VI2DAT7	VI2CK	VI3DAT2	SAVSS	ERXD0	ERXDV	ERXERR	ETXD0	PCIGRANT0N	PCIGRANT3N	PCIREQ1N	PCIAD31
AA	VI2DAT9	VI3DAT3	VSS	SAVDD33	ERXD1	ECRS	ETXEN	ETXD1	PCIINTAN	VSS	PCIREQ2N	VSS
AB	VI3CK	VI3DAT4	VI3DAT7	USBDP	ERXD2	ETXCK	MDIO	ETXD2	PCIGRANT1N	PCICLK	PCIREQ0N	PCIAD29
AC	VSS	VI3DAT5	VI3DAT6	USBDM	ERXCK	ECOL	MDCK	ETXD3	PCIGRANT2N	PCIFRAMEN	PCIPERRN	PCIAD30
	1	2	3	4	5	6	7	8	9	10	11	12

图2-28 Hi3511 管脚分布图 (13~23)

13	14	15	16	17	18	19	20	21	22	23	
VSS	DDRCKN0	DDRRASN	DDRDQ5	DDRDQ7	DDRDQ1	DDRDQ11	DDRDQ10	DDRDQ9	DDRDQ12	VSS	A
VSS	DDRCKP0	DDRCASN	DDRDQ0	DDRDQS0	DDRDQ4	DDRDQ15	DDRDQS1	DDRDQ14	VSS	EBIRDYN	B
DDRCKE0	DDRBA2	VSS	DDRDQ2	VSS	DDRDQ3	VSS	DDRDQ13	DDRDQ8	EADR2	EADR3	C
PDDRBA1	DDRBA0	DDRCMN	DDRDMD0	VREF1	DDRDQ6	DDRDMD1	VSS	EADR4	EADR5	EADR6	D
DVDD18	DVDD18	DDRWEN	DDRODT	VSSREF1	DVDD18	DVDD18	EADR7	EADR18	EADR21	EADR20	E
						DVDD33	EADR8	EADR22	EADR11	EADR12	F
						DVDD33	EADR19	EADR16	EADR15	EADR13	G
DVDD18	DVDD12	DVDD18	DVDD18			EADR9	EADR10	VSS	EADR17	EADR14	H
VSS	VSS	VSS	DVDD33			EADR24	EADR23	EADR0	EADR1	EWIEN	J
VSS	VSS	VSS	DVDD12			DVDD33	EBICS1N	EIDQ6	EIDQ4	EIDQ7	K
VSS	VSS	VSS	DVDD33			VSS	EBIOEN	EIDQ3	EIDQ2	EIDQ5	L
VSS	VSS	VSS	DVDD12			EBICS0N	SPIDI	VSS	EIDQ0	EIDQ1	M
VSS	VSS	VSS	DVDD33			VSS	SPICSN0	WDGRST	SPICK	SPIDO	N
VSS	VSS	VSS	DVDD12			DVDD33	TRSTN	SCL	IRRCV	TDO	P
VSS	VSS	VSS	DVDD33			VSS	RSTN	TDI	SDA	TCK	R
DVDD33	DVDD12	DVDD33	DVDD33			DVDD12_0	VSS_0	TESTMODE0	VSS	FUNSEL0	T
						RTCAVDD12	RTCBATT	TMS	XOUT	XIN	U
						VSS	XOUTRTC	XINRTC	AVDD33_PLL	AVSS_PLL	V
PCIAD24	VSS	DVDD33	PCIIDSEL	PCIREQ4N	DVDD33	DVDD33	RTCAGN	SDIODAT3	SDIODAT0	SDIODAT2	W
PCIAD25	PCIAD22	PCIAD18	PCIIRDYN	PCISERRN	PCIAD14	PCIAD10	PCIAD0	SDIODAT1	SDIOCK	SDIOMCMD	Y
PCIAD26	PCIAD23	PCIAD19	PCICBE2	PCIDEVSELN	PCIAD15	PCIAD11	PCIAD1	VSS	PCIAD3	PCIAD2	AA
PCIAD27	PCIGRANT4N	PCIAD20	PCIAD16	PCISTOPN	PCICBE1	PCIAD12	PCIAD8	PCIAD7	PCIAD5	PCIAD4	AB
PCIAD28	PCICBE3	PCIAD21	PCIAD17	PCITRDYN	PCIPAR	PCIAD13	PCIAD9	PCICBE0	PCIAD6	VSS	AC
13	14	15	16	17	18	19	20	21	22	23	



Hi3511 的管脚按管脚位置排序的排列表如表 2-60 所示。

表2-60 管脚排列表

位置	管脚名称	位置	管脚名称
A1	VSS	M13	VSS
A2	DDRDQ29	M14	VSS
A3	DDRDQ31	M15	VSS
A4	DDRDQ26	M16	DVDD12
A5	DDRDQ23	M19	EBICS0N
A6	DDRDQ18	M20	SPIDI
A7	DDRDQ22	M21	VSS
A8	DDRDQ20	M22	EBIDQ0
A9	DDRADR1	M23	EBIDQ1
A10	DDRADR10	N1	OTGID
A11	DDRADR12	N2	OTGVBUS
A12	DDRCKN1	N3	VI1DAT1
A13	VSS	N4	VI1DAT2
A14	DDRCKN0	N5	VI1DAT0
A15	DDRRASN	N8	DVDD33
A16	DDRDQ5	N9	VSS
A17	DDRDQ7	N10	VSS
A18	DDRDQ1	N11	VSS
A19	DDRDQ11	N12	VSS
A20	DDRDQ10	N13	VSS
A21	DDRDQ9	N14	VSS
A22	DDRDQ12	N15	VSS
A23	VSS	N16	DVDD33
B1	UTXD1	N19	VSS
B2	UTXD0	N20	SPICSN0
B3	DDRDQ24	N21	WDGRST
B4	DDRDQS3	N22	SPICK
B5	DDRDQ27	N23	SPIDO

位置	管脚名称	位置	管脚名称
B6	DDRDQ21	P1	VI1DAT6
B7	DDRDQS2	P2	VI1DAT3
B8	DDRDQ17	P3	VI1DAT4
B9	DDRADR7	P4	VI1DAT7
B10	DDRADR4	P5	VI1DAT5
B11	DDRADR11	P8	DVDD12
B12	DDRCKP1	P9	VSS
B13	VSS	P10	VSS
B14	DDRCKP0	P11	VSS
B15	DDRCASN	P12	VSS
B16	DDRDQ0	P13	VSS
B17	DDRDQS0	P14	VSS
B18	DDRDQ4	P15	VSS
B19	DDRDQ15	P16	DVDD12
B20	DDRDQS1	P19	DVDD33
B21	DDRDQ14	P20	TRSTN
B22	VSS	P21	SCL
B23	EBIRDYN	P22	IRRCV
C1	URXD1	P23	TDO
C2	URXD0	R1	VI1CK
C3	VSS	R2	VI0DAT3
C4	DDRDQ25	R3	VI0DAT2
C5	VSS	R4	VI0DAT4
C6	DDRDQ16	R5	VI0HS
C7	VSS	R8	DVDD33
C8	DDRADR13	R9	VSS
C9	VSS	R10	VSS
C10	DDRADR3	R11	VSS
C11	VSS	R12	VSS
C12	DDRADR2	R13	VSS
C13	DDRCKE0	R14	VSS



位置	管脚名称	位置	管脚名称
C14	DDRBA2	R15	VSS
C15	VSS	R16	DVDD33
C16	DDRDQ2	R19	VSS
C17	VSS	R20	RSTN
C18	DDRDQ3	R21	TDI
C19	VSS	R22	SDA
C20	DDRDQ13	R23	TCK
C21	DDRDQ8	T1	VI0CK
C22	EBIADR2	T2	VI0DAT5
C23	EBIADR3	T3	VI0VS
D1	UCTSN1	T4	VI0DAT7
D2	URTSN1	T5	DVDD33
D3	DDRMRCVO	T8	DVDD33
D4	DDRDM3	T9	DVDD33
D5	DDRDQ30	T10	DVDD12
D6	DDRDM2	T11	DVDD33
D7	DDRDQ19	T12	DVDD12
D8	VREF3	T13	DVDD33
D9	DDRADR8	T14	DVDD12
D10	DDRADR5	T15	DVDD33
D11	DDRADR0	T16	DVDD33
D12	VREF2	T19	DVDD12_0
D13	PDDRBA1	T20	VSS_0
D14	DDRBA0	T21	TESTMODE0
D15	DDRCSEN	T22	VSS
D16	DDRDM0	T23	FUNSEL0
D17	VREF1	U1	VI0DAT9
D18	DDRDQ6	U2	VI0DAT8
D19	DDRDM1	U3	VI0DAT6
D20	VSS	U4	VI2DAT6
D21	EBIADR4	U5	VSS



位置	管脚名称	位置	管脚名称
D22	EBIADR5	U19	RTCAVDD12
D23	EBIADR6	U20	RTCBATT
E1	VODAT6	U21	TMS
E2	VOCK	U22	XOUT
E3	VODAT7	U23	XIN
E4	DDRMRCVI	V1	VI2DAT4
E5	VSS	V2	VI2DAT5
E6	DVDD18	V3	VI2DAT3
E7	DDRDQ28	V4	VI2VS
E8	VSSREF3	V5	VI2DAT2
E9	DVDD18	V19	VSS
E10	DDRADR9	V20	XOUTRTC
E11	DDRADR6	V21	XINRTC
E12	VSSREF2	V22	AVDD33_PLL
E13	DVDD18	V23	AVSS_PLL
E14	DVDD18	W1	VI2DAT8
E15	DDRWEN	W2	VI2HS
E16	DDRODT	W3	VI3DAT1
E17	VSSREF1	W4	VI3DAT0
E18	DVDD18	W5	DVDD33
E19	DVDD18	W6	DVDD33
E20	EBIADR7	W7	ERXD3
E21	EBIADR18	W8	VSS
E22	EBIADR21	W9	DVDD33
E23	EBIADR20	W10	PCIRSTN
F1	VODAT1	W11	PCIREQ3N
F2	VODAT2	W12	DVDD33
F3	VODAT3	W13	PCIAD24
F4	VODAT4	W14	VSS
F5	VODAT5	W15	DVDD33
F19	DVDD33	W16	PCIIDSEL



位置	管脚名称	位置	管脚名称
F20	EBIADR8	W17	PCIREQ4N
F21	EBIADR22	W18	DVDD33
F22	EBIADR11	W19	DVDD33
F23	EBIADR12	W20	RTCAGND
G1	VODAT0	W21	SDIODAT3
G2	SIO0RCK	W22	SDIODAT0
G3	SIO0XCK	W23	SDIODAT2
G4	SIO0DO	Y1	VI2DAT7
G5	DVDD33	Y2	VI2CK
G19	DVDD33	Y3	VI3DAT2
G20	EBIADR19	Y4	SAVSS
G21	EBIADR16	Y5	ERXD0
G22	EBIADR15	Y6	ERXDV
G23	EBIADR13	Y7	ERXERR
H1	SIO1RCK	Y8	ETXD0
H2	VSS	Y9	PCIGRANT0N
H3	ACKOUT	Y10	PCIGRANT3N
H4	SIO0XFS	Y11	PCIREQ1N
H5	VSS	Y12	PCIAD31
H8	DVDD18	Y13	PCIAD25
H9	DVDD18	Y14	PCIAD22
H10	DVDD12	Y15	PCIAD18
H11	DVDD18	Y16	PCIIRDYN
H12	DVDD12	Y17	PCISERRN
H13	DVDD18	Y18	PCIAD14
H14	DVDD12	Y19	PCIAD10
H15	DVDD18	Y20	PCIAD0
H16	DVDD18	Y21	SDIODAT1
H19	EBIADR9	Y22	SDIOCK
H20	EBIADR10	Y23	SDIOCMD
H21	VSS	AA1	VI2DAT9



位置	管脚名称	位置	管脚名称
H22	EBIADR17	AA2	VI3DAT3
H23	EBIADR14	AA3	VSS
J1	XOUT24	AA4	SAVDD33
J2	XIN24	AA5	ERXD1
J3	SIO1DI	AA6	ECRS
J4	SIO1RFS	AA7	ETXEN
J5	SIO0DI	AA8	ETXD1
J8	DVDD33	AA9	PCIINTAN
J9	VSS	AA10	VSS
J10	VSS	AA11	PCIREQ2N
J11	VSS	AA12	VSS
J12	VSS	AA13	PCIAD26
J13	VSS	AA14	PCIAD23
J14	VSS	AA15	PCIAD19
J15	VSS	AA16	PCICBE2
J16	DVDD33	AA17	PCIDEVSELN
J19	EBIADR24	AA18	PCIAD15
J20	EBIADR23	AA19	PCIAD11
J21	EBIADR0	AA20	PCIAD1
J22	EBIADR1	AA21	VSS
J23	EBIWEN	AA22	PCIAD3
K1	OTGVSSA33T	AA23	PCIAD2
K2	OTGVSSA33T	AB1	VI3CK
K3	OTGREXT	AB2	VI3DAT4
K4	OTGVSSA33C	AB3	VI3DAT7
K5	SIO0RFS	AB4	USBDP
K8	DVDD12	AB5	ERXD2
K9	VSS	AB6	ETXCK
K10	VSS	AB7	MDIO
K11	VSS	AB8	ETXD2
K12	VSS	AB9	PCIGRANT1N



位置	管脚名称	位置	管脚名称
K13	VSS	AB10	PCICLK
K14	VSS	AB11	PCIREQ0N
K15	VSS	AB12	PCIAD29
K16	DVDD12	AB13	PCIAD27
K19	DVDD33	AB14	PCIGRANT4N
K20	EBICS1N	AB15	PCIAD20
K21	EBIDQ6	AB16	PCIAD16
K22	EBIDQ4	AB17	PCISTOPN
K23	EBIDQ7	AB18	PCICBE1
L1	OTGDM	AB19	PCIAD12
L2	OTGDP	AB20	PCIAD8
L3	OTGVSSA33T	AB21	PCIAD7
L4	OTGVDDA33C	AB22	PCIAD5
L5	DVDD33	AB23	PCIAD4
L8	DVDD33	AC1	VSS
L9	VSS	AC2	VI3DAT5
L10	VSS	AC3	VI3DAT6
L11	VSS	AC4	USBDM
L12	VSS	AC5	ERXCK
L13	VSS	AC6	ECOL
L14	VSS	AC7	MDCK
L15	VSS	AC8	ETXD3
L16	DVDD33	AC9	PCIGRANT2N
L19	VSS	AC10	PCIFRAMEN
L20	EBIOEN	AC11	PCIPERRN
L21	EBIDQ3	AC12	PCIAD30
L22	EBIDQ2	AC13	PCIAD28
L23	EBIDQ5	AC14	PCICBE3
M1	OTGVDDA33T	AC15	PCIAD21
M2	OTGVDDA33T	AC16	PCIAD17
M3	OTGVSS	AC17	PCITRDYN



位置	管脚名称	位置	管脚名称
M4	VSS	AC18	PCIPAR
M5	OTGVDD12	AC19	PCIAD13
M8	DVDD12	AC20	PCIAD9
M9	VSS	AC21	PCICBE0
M10	VSS	AC22	PCIAD6
M11	VSS	AC23	VSS
M12	VSS	-	-



3 系统控制

关于本章

本章描述内容如下表所示。

标题	内容
3.1 复位	介绍复位模块的功能、复位控制和复位配置。
3.2 时钟	介绍时钟模块的功能、时钟控制框图和时钟配置。
3.3 处理器及存储器地址空间映射	介绍 ARM926EJ-S 处理器的特点和存储器地址空间映射。
3.4 中断系统	介绍中断系统的功能、特点、中断映射、寄存器概览和寄存器描述。
3.5 直接存储器存取控制器	介绍直接存储器存取控制器的功能、特点、工作方式、寄存器概览和寄存器描述。
3.6 加解密引擎	介绍加解密引擎的功能、特点、工作方式、寄存器概览和寄存器描述。
3.7 Timer	介绍 Timer 模块的功能、特点、工作方式、寄存器概览和寄存器描述。
3.8 看门狗	介绍看门狗的功能、特点、工作方式、寄存器概览和寄存器描述。
3.9 实时时钟	介绍实时时钟的功能、特点、工作方式、寄存器概览和寄存器描述。
3.10 系统控制器	介绍系统控制器的功能、特点、工作方式、寄存器概览和寄存器描述。
3.11 电源管理与低功耗模式控制	介绍低功耗模式、系统工作模式、时钟门控和时钟频率调整、模块级低功耗控制和 DDR 低功耗控制。



3.1 复位

3.1.1 概述

复位管理模块对整个芯片的复位、各功能模块的复位进行统一的管理，包括：

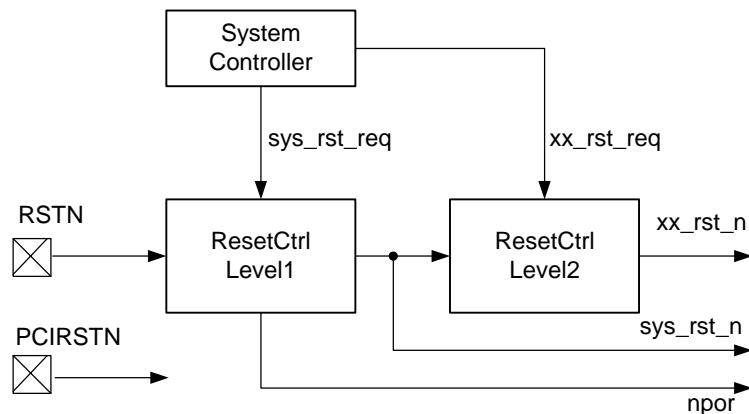
- 上电复位的管理和控制
- 系统软复位、功能模块单独软复位控制
- 复位信号同步到各模块对应时钟域

复位管理模块生成芯片内部各功能模块的复位信号。

3.1.2 复位控制

复位信号控制如图 3-1 所示。

图3-1 复位信号控制图



RSTN: 上电复位信号，源自芯片管脚 RSTN 输入。

PCIRSTN: PCI 总线复位信号，源自芯片管脚 PCIRSTN 输入。

sys_rst_req: 全局软复位请求信号，源自系统控制器。

xx_rst_req: 子模块单独软复位请求信号，源自系统控制器。

xx_rst_n、sys_rst_n、npor: 复位信号。

表3-1 复位信号分类表

复位信号类型	产生方式	用途
全局硬复位 npor	来自复位管脚 RSTN	对整芯片进行全局复位。
全局软复位 sys_rst_n	软件配置系统控制器的全局软复位寄存器	对整芯片中除了时钟复位电路和测试电路的所有模块进行全局复位。
子模块复位 xx_rst_n	软件配置系统控制寄存器的子模块复位控制寄存器	芯片各子模块的单独复位。



复位信号类型	产生方式	用途
PCI 总线复位 PCIRSTN	来自 PCI 的管脚 PCIRSTN	当 Hi3511/Hi3512 处于从模式时，此管脚复位能够复位 PCI 子模块。

3.1.3 复位配置

上电复位

RSTN 是 Hi3511/Hi3512 芯片的功能复位输入 IO，完成上电复位过程必须同时满足以下条件：

- 上电复位 IO 输入一个低电平脉冲。
- 晶振时钟输入管脚 XIN 输入的时钟稳定。
- 输入的上电复位信号低电平维持时间大于 12 个 XIN 晶振时钟周期。

系统复位

实现系统复位有两种途径：

- 上电复位。
- 全局软复位，通过系统控制器 [SC_SYSSTAT](#) 控制。

软复位

软复位控制通过配置相应的系统控制器来实现，具体配置请参见 [SC_PERCTRL0](#) 的描述。



注意

- 系统软复位请求发出后，电路必须等待至少 360 个系统时钟周期才完成复位撤消。
- 各模块单独软复位不会自动撤消，比如某模块的复位是配置 1 时，模块处于复位状态，那么必须再配置为 0，该模块复位才会撤消。
- USB 2.0 OTG PHY 和控制器、USB 1.1 PHY 及控制器、MMC、ETH、VI、VO 和 RTC 这些模块上电时默认均处于复位状态，正常工作前必须先撤销复位。

3.2 时钟

3.2.1 概述

时钟管理模块对芯片时钟输入、时钟生成和控制进行统一的管理，包括：

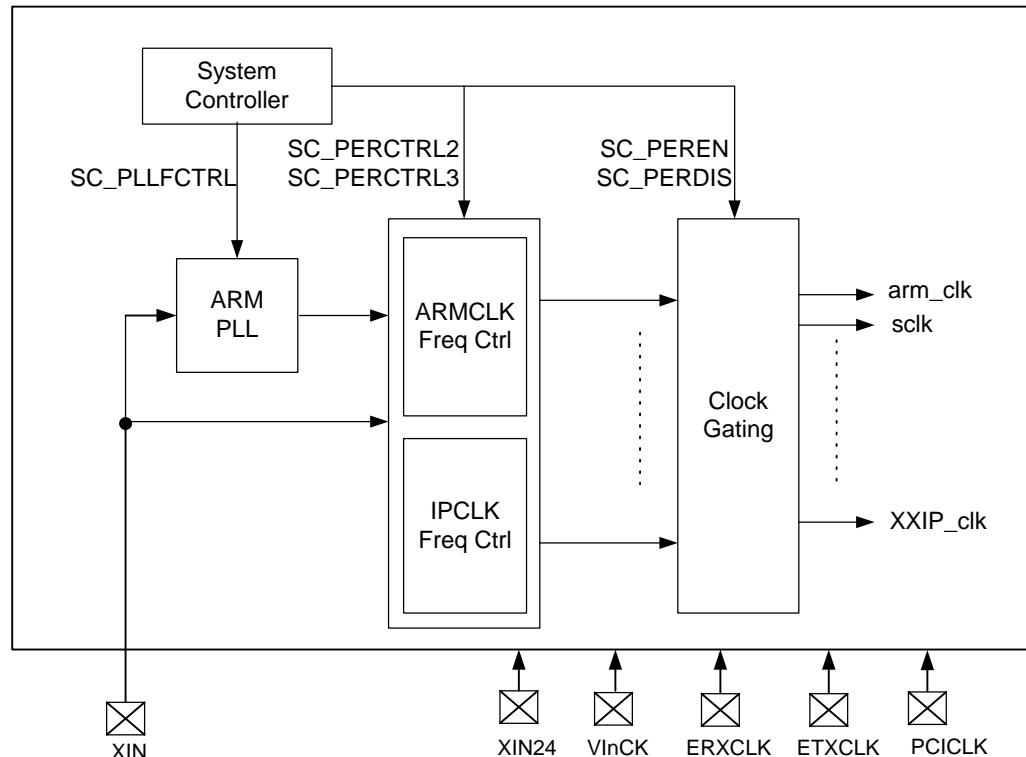


- 时钟输入的管理和控制
- 时钟分频和控制
- 生成各模块的工作时钟

3.2.2 时钟控制框图

时钟管理模块功能框图如图 3-2 所示。

图3-2 时钟管理模块功能框图



注：图中的 VInCK 中的 n 取值范围为 0~3。

时钟管理模块有以下两部分输入：

- 源自芯片管脚的时钟输入：XIN、XIN24、VI0CK、VI1CK、VI2CK、VI3CK、ERXCLK、ETXCLK 和 PCICLK。
 - XIN 为 PLL 输入时钟
 - XIN24 为 USB 2.0 OTG PHY 输入时钟
 - VI0CK、VI1CK、VI2CK、VI3CK 为视频输入时钟
 - ERXCLK、ETXCLK 为 ETH 模块接口时钟
 - PCICLK 为 PCI 模块接口时钟
- 源自系统控制器的时钟控制寄存器。
 - PLL 频率配置 [SC_PLLFCTRL](#)
 - IP 时钟频率配置 [SC_PERCTRL2](#)



- 时钟门控配置 [SC_PEREN](#) 和 [SC_PERDIS](#)

时钟管理模块功能主体主要包括三部分：

- ARMPLL，用于产生 ARM 和总线时钟，也可以经过分频生成其它外设所需时钟。
- ARM 频率控制单元 ARMCLK Freq Ctrl 和模块时钟频率控制单元 IPCLK Freq Ctrl。
- 时钟门控管理单元 Clock Gating。

3.2.3 时钟配置

PLL 配置寄存器

PLL 配置寄存器主要涉及到以下内容：

- PLL 输出频率配置寄存器，用于配置 PLL 输出频率。ARMPLL 采用管脚 XIN 输入的晶振时钟作为输入时钟，输出频率配置方法请参见 [SC_PLLFCTRL](#)。
- ARMPLL_EN 配置方法。ARMPLL 的开与关，有两种控制方式：
 - 由软件直接控制开关。
 - 通过改变系统状态，间接导致 ARMPLL 的开关状态发生改变。详细内容请参见 [SC_PLLCTRL\[0\]](#) 和 [SC_CTRL\[2:0\]](#)。

ARM/SCLK/HCLK 频率配置

ARM/SCLK/HCLK 频率配置方法如表 3-2 所示。

表3-2 ARM/SCLK/HCLK 频率配置

信号名	描述
sysmode[3:0]	ARM 频率切换。 默认为 0xB，为晶振模式。可通过配置 SC_CTRL[2:0] 控制该信号。
sleep_mode	除 SCLK/IR 模块/CRG (Clock Reset Generation) 模块以外所有时钟关闭，系统进入睡眠模式。可通过配置 SC_CTRL[2:0] 控制该信号。
arm_hclk_sel	ARM 频率模式。可通过配置 SC_PERCTRL2[7] 控制该信号。

系统控制器的状态和时钟切换的对应关系如表 3-3 所示。

表3-3 系统控制器状态和时钟切换对应关系

系统控制器状态	45kHz 时钟使能状态	27MHz 晶振使能状态	ARMPLL 使能状态	系统时钟状态
NORMAL	使能	使能	使能	ARM 子系统的工作时钟都来自 PLL 输出。



系统控制器状态	45kHz 时钟使能状态	27MHz 晶振使能状态	ARMPLL 使能状态	系统时钟状态
SLOW	使能	使能	不使能	ARM 子系统的工作时钟都来自 27MHz 晶振输入。
DOZE	使能	使能	不使能	ARM 子系统的工作时钟都来自 27MHz 晶振时钟分频得到的 45KHz 时钟。
SLEEP	使能	使能	不使能	除系统控制器和红外模块工作在 45KHz 外，其它模块的时钟都处于关闭状态。

模块时钟频率配置

MMC 时钟频率配置方法如表 3-4 所示。

表3-4 MMC 时钟频率配置

信号名	描述
mmcclk_sel	MMC 模块工作时钟频率控制，可通过配置 SC_PERCTRL2[5:4] 控制该信号。
mmcsap_sel	MMC 模块采样卡数据的时钟正反相控制，可通过配置 SC_PERCTRL2[6] 控制该信号。

USB 2.0 OTG PHY 24MHz 时钟，USB 1.1 HOST 48MHz 时钟频率配置方法如表 3-5 所示。

表3-5 USB 2.0 OTG PHY/USB 1.1 HOST 时钟频率配置

信号名	描述
otg24_sel	USB 2.0 OTG 24MHz 时钟源选择，可通过配置控制 SC_PERCTRL2[25] 配置该信号。 注意： USB OTG PHY 24MHz 时钟若选择 PLL 内部时钟，一定要先正确配置 usb48clk_sel 。
usb48clk_sel	USB 1.1 HOST 48MHz 时钟频率选择控制，可通过配置 SC_PERCTRL2[9:8] 配置该信号。



PCI 钟频率配置方式如表 3-6 所示。

表3-6 PCI 时钟频率配置

信号名	描述
pciclk_sel[3:0]	PCI 模块的工作时钟频率控制, 可通过配置 SC_PERCTRL2[23:20] 控制该信号。

SMI 时钟频率配置方式如表 3-7 所示。

表3-7 SMI 时钟频率配置

信号名	描述
ssmcclk_sel	SMI 模块工作时钟频率选择, 提供 AHB 总线的 1 或 2 分频。可通过配置 SC_PERCTRL2[24] 控制该信号。

VO 输出时钟配置方式如表 3-8 所示。

表3-8 VO 时钟频率配置

信号名	描述
voout_sel	VO 模块输出时钟相位选择, 提供 VO 工作时钟的正反相输出。可通过配置 SC_PERCTRL2[0] 控制该信号。

VI 输出时钟配置方式如表 3-9 所示。

表3-9 VI 时钟频率配置

信号名	描述
vi3div_sel[1:0]	VI3 路接口分频时钟选择, 可提供 VI3 路时钟的 1/2/4 分频选择。可通过配置 SC_PERCTRL2[19:18] 控制该信号。
vi2div_sel[1:0]	VI2 路接口分频时钟选择, 可提供 VI2 路时钟的 1/2/4 分频选择。可通过配置 SC_PERCTRL2[17:16] 控制该信号。
vi1div_sel[1:0]	VI1 路接口分频时钟选择, 可提供 VI1 路时钟的 1/2/4 分频选择。可通过配置 SC_PERCTRL2[15:14] 控制该信号。
vi0div_sel[1:0]	VI0 路接口分频时钟选择, 可提供 VI0 路时钟的 1/2/4 分频选择。可通过配置 SC_PERCTRL2[13:12] 控制该信号。
vi3_vi2_sel	VI3 路接口时钟选择, 可通过配置 SC_PERCTRL2[3] 控制该信号。

信号名	描述
vi2_vi0_sel	VI2 路接口时钟选择, 可通过配置 SC_PERCTRL2[2] 控制该信号。
vi1_vi0_sel	VI1 路接口时钟选择, 可通过配置 SC_PERCTRL2[1] 控制该信号。

SIO0/SIO1 时钟频率配置方法如[表 3-10](#) 所示。

表3-10 SIO0/SIO1 时钟频率配置

信号名	描述
sioclk_sel[23:0]	用于 SIO0/SIO1 模块的输出接口主时钟 SIO_MCLK 的分频因子控制, 可通过配置 SC_PERCTRL3[23:0] 控制该信号。
siobclk_sel[2:0]	配置 SIO0/SIO1 模块在主模式下的位流时钟 SIO_XCK (BCLK) 的频率。可通过配置 SC_PERCTRL3[26:24] 控制该信号。
siolrclk_sel[2:0]	配置 SIO0/SIO1 模块在主模式下的输出采样率时钟 SIO_RFS/SIO_XFS (FSCLK) 的频率, 可通过配置 SC_PERCTRL3[29:27] 控制该信号。

一般的应用场景会给定采样率时钟频率 FSCLK, 而比特时钟 BCLK 和主时钟 MCLK 相对于 FSCLK 分别具有可变的倍数关系, 时钟频率配置方法示例如下:

假设时钟源 ARMPLL 已经配置输出频率为 540MHz, 此时要求配置出 SIO 工作时钟频率为 FSCLK=48kHz、MCLK=256FSCLK=12.288MHz、BCLK=16FSCLK=384KHz。配置方法如下:

- ARMPLL 到 MCLK 分频比计算为: $N=12.288/540$, 则 sioclk_sel[23:0]= $N \times 2^{27}$, 根据四舍五入取整原则计算为 3054199, 因此配置 sioclk_sel=0x002E_9A77, 即可得到 MCLK 的正确频率。
- BCLK 由 MCLK 分频获得, 分频比为 BCLK/MCLK=16/256=1/16, 因此根据配置表中的对应关系, 配置 siobclk_sel[2:0]=0b100 (对应 16 分频) 便可得到 BCLK 的正确频率。
- FSCLK 由 BCLK 分频得到, 分频比为 1/16, 因此根据配置表中的对应关系, 配置 siolrclk_sel=0b011 (对应 16 分频) 便可得到 FSCLK 的正确频率。

时钟门控配置

时钟门控配置请参见系统控制器 [SC_PEREN](#)。

注意事项

时钟配置需要注意以下事项:



- ARM 工作时钟上电默认为晶振模式，即选择 XIN 输入的晶振时钟。
- PLL 在变更频率配置时，需要等待 0.5ms 才能输出稳定的时钟。更改 PLL 频率配置只能在系统处于 SLOW 模式下进行。
- 在 PLL 输出时钟未稳定的情况下，不能执行系统切换到 PLL 模式。

3.3 处理器及存储器地址空间映射

3.3.1 ARM926EJ-S 处理器

ARM926EJ-S 处理器的特点如下：

- 采用 32bit ARM v5TEJ，5 级流水，兼容 32bit ARM、16bit Thumb 指令集。
- 内嵌增强型 DSP 指令。
- 支持 Java。
- 提供独立的 16KB 指令 Cache、16kB 数据 Cache、4 路组相连 Cache，Cache Line 大小为 32 byte；数据 Cache 支持 write-back 和 write-through 操作可配置。
- Cache 支持伪随机或者 round-robin 替换算法，并可进行配置。
- 独立 32 bit 指令和数据总线接口，总线工作频率与 ARM926EJ-S 系统时钟频率支持 1:1 和 1:2 的配置。
- 包含 MMU，支持 VxWorks、Linux、WindowCE、PalmOS 等开放操作系统。
- 提供独立 2KB ITCM (Instruction TCM)。
- 采用小端字序模式 (little endian)。
- 支持快速中断请求 FIQ (Fast Interrupt Request) 和一般中断请求 IRQ (Interrupt Request)。
- 支持 JTAG 调试接口。
- 支持动态功耗管理和静态功耗管理。

3.3.2 存储器地址空间映射

Hi3511/Hi3512 内部使用的地址总线和数据总线均为 32bit 位宽，可寻址的地址空间为 4GB。Hi3511/Hi3512 支持以下两种 BOOT 方式：

- 使用 SMI 接口 BOOT。
- 使用 DDR 接口 BOOT。

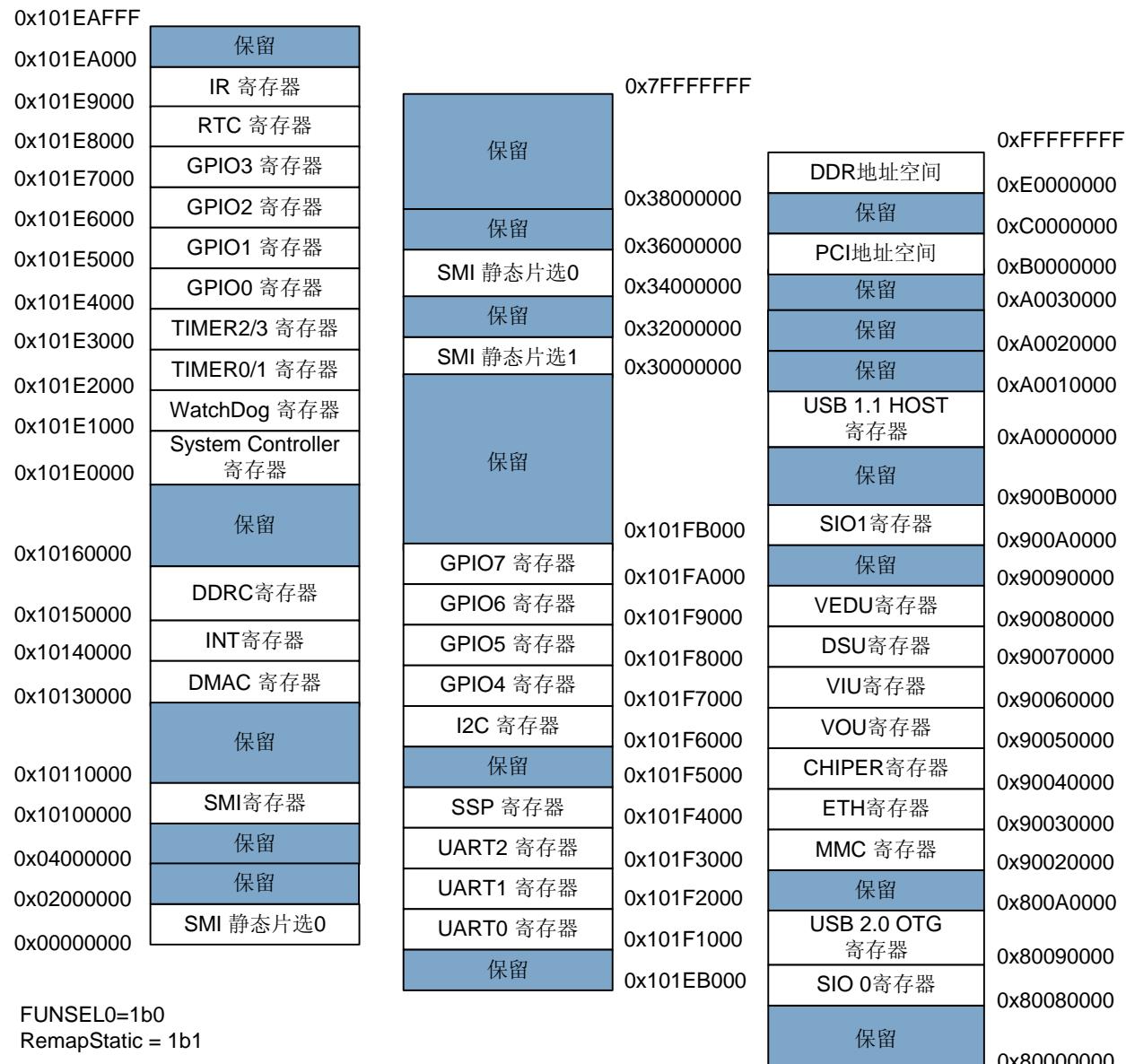
使用 SMI 控制器 BOOT

当使用 SMI 控制器 BOOT 时，外部连接的存储器一般为异步 Nor Flash，此时需要设置芯片外部管脚 FUNSEL0 电平为 0，用于选择 SMI 接口：

- 通过设置 FUNSEL0 管脚为 0，芯片支持从 SMI 接口挂接的异步 Nor Flash 启动。
- Hi3511/Hi3512 只支持 8bit 的 Nor Flash 存储器。

BOOT 时芯片内部的存储空间映射如图 3-3 所示。

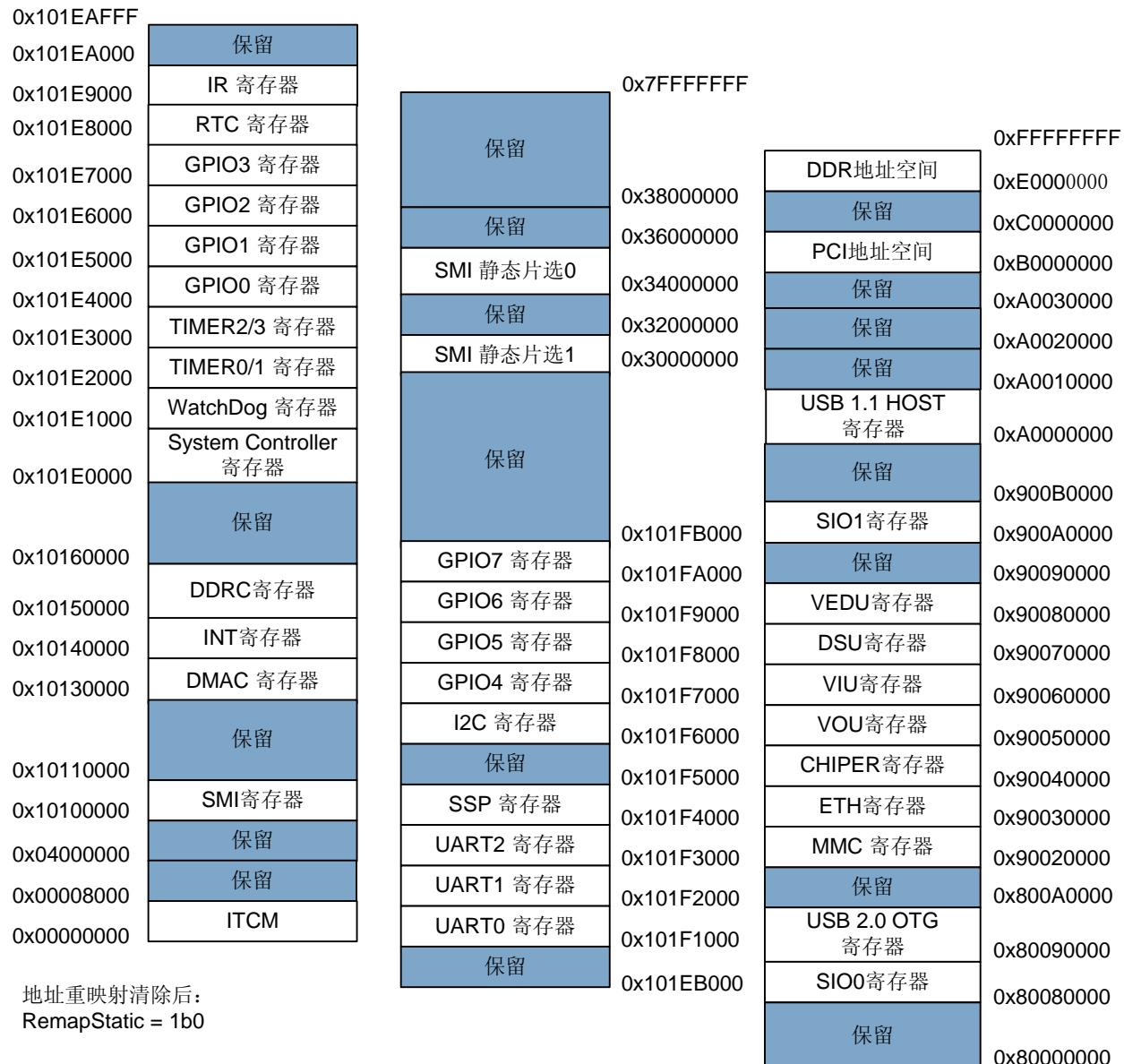
图3-3 使用 SMI 控制器 BOOT 时的地址空间映射



使用 SMI 控制器 BOOT 时清除重映射后的地址分布如图 3-4 所示。



图3-4 使用 SMI 控制器 BOOT 时清除重映射后的地址分布



注意

- ITCM 的大小除了可以设置成 0KB 以外，最小只能选择 4KB。系统实际上只提供了 2KB 的 ITCM 地址空间，因此，软件需要保证在 ITCM 的程序和数据必须在 2KB 的范围内。
- 如果需要使用 ITCM，则必须通过 ARM 的系统控制协处理器 CP15 提供的寄存器设置 ITCM 使能，并且配置 ITCM 的大小等信息。使用 MCR 指令配置 C9 寄存器为 0xD。

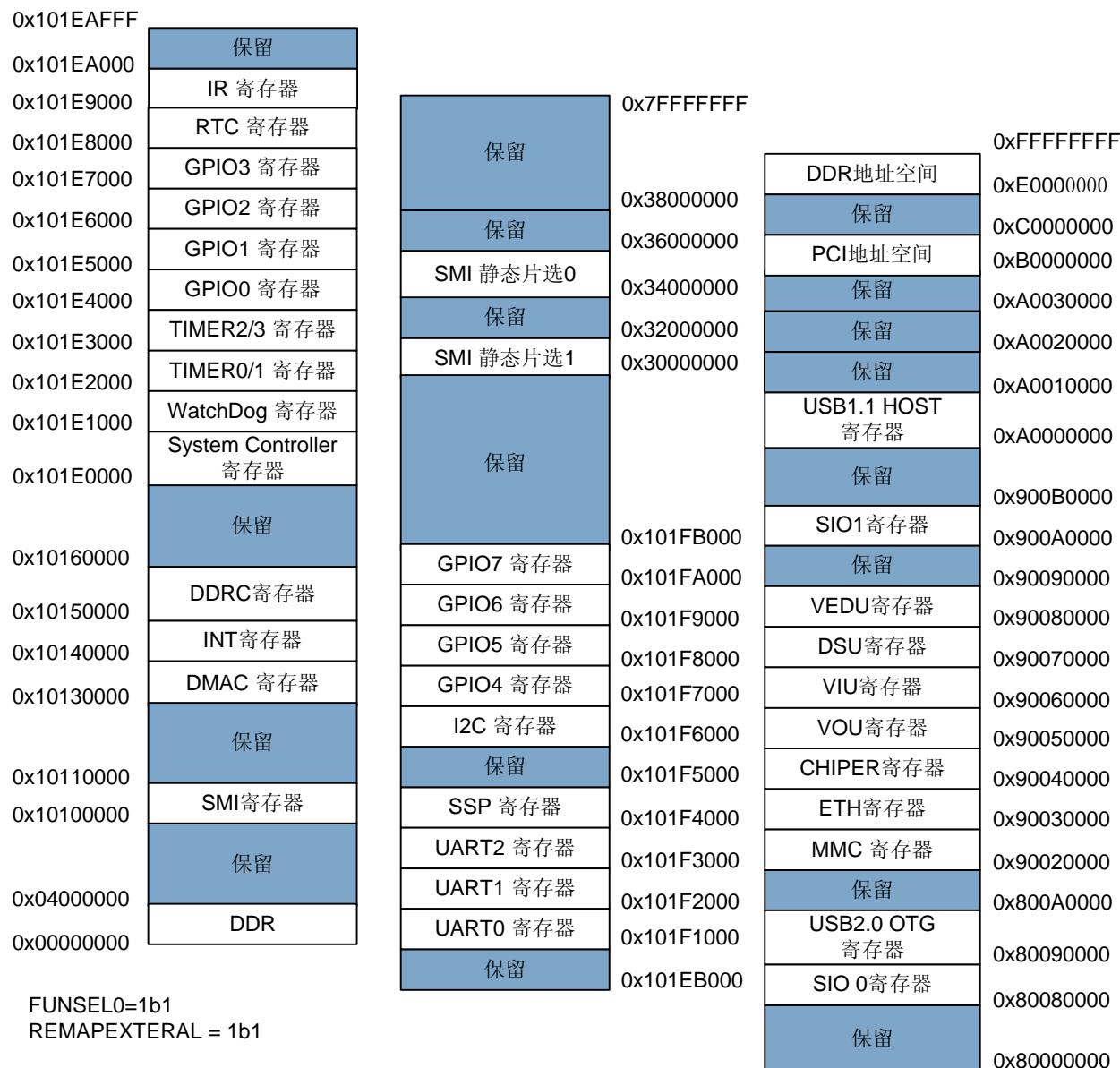
使用 DDR BOOT

当使用 DDR BOOT 时, SMI 接口外部可以不用连接存储器, 此时需要设置芯片外部管脚 FUNSEL0 为 1, 用来选择从 DDR 接口 BOOT。此时, 芯片需要一片主芯片配合:

- FUNSEL0 = 1, 芯片支持从 DDR 接口挂接的 DDR 器件启动。
- 上电复位撤消, 芯片的 ARM 处于复位状态。
- 主芯片 PCI 接口配置芯片, 将启动程序写入 0x0000_0000(此时指向 DDR 器件)。
- 主芯片 PCI 接口配置芯片, 将 ARM 的复位撤消。
- 芯片从 DDR 器件加载程序, 进行 BOOT。

使用 DDR BOOT 时芯片内部的存储空间映射如图 3-5 所示。

图3-5 使用 DDR BOOT 时的地址空间映射





使用 DDR BOOT 时清除重映射后的地址分布请参见图 3-4。

地址空间列表

Hi3511/Hi3512 内部的地址空间映射如表 3-11 所示。

表3-11 Hi3511/Hi3512 地址空间映射表

起始地址	结束地址	功能	大小	说明
0xE000_0000	0xFFFF_FFFF	DDR	512MB	动态存储器寻址空间
0xC000_0000	0xDFFF_FFFF	保留	512MB	-
0xB000_0000	0xBFFF_FFFF	PCI 地址空间	256MB	此段地址空间分配请参见表 3-12。
0xA003_0000	0xAFFF_FFFF	保留	255.8MB	-
0xA002_0000	0xA002_FFFF	保留	64KB	-
0xA001_0000	0xA001_FFFF	保留	64KB	-
0xA000_0000	0xA000_FFFF	USB 1.1 HOST 寄存器	64KB	-
0x900B_0000	0x9FFF_FFFF	保留	255.3MB	-
0x900A_0000	0x900A_FFFF	SIO1 寄存器	64KB	-
0x9009_0000	0x9009_FFFF	保留	64KB	-
0x9008_0000	0x9008_FFFF	VEDU 寄存器	64KB	-
0x9007_0000	0x9007_FFFF	DSU 寄存器	64KB	-
0x9006_0000	0x9006_FFFF	VIU 寄存器	64KB	-
0x9005_0000	0x9005_FFFF	VOU 寄存器	64KB	-
0x9004_0000	0x9004_FFFF	CIPHER 寄存器	64KB	-
0x9003_0000	0x9003_FFFF	ETH 寄存器	64KB	-
0x9002_0000	0x9002_FFFF	MMC 寄存器	64KB	-
0x9001_0000	0x9001_FFFF	保留	64KB	-
0x9000_0000	0x9000_FFFF	保留	64KB	-
0x800A_0000	0x8FFF_FFFF	保留	255MB	-
0x8009_0000	0x8009_FFFF	USB 2.0 OTG 寄存器	64KB	-
0x8008_0000	0x8008_FFFF	SIO0 寄存器	64KB	-
0x8000_0000	0x8007_FFFF	保留	512KB	-

起始地址	结束地址	功能	大小	说明
0x3600_0000	0x7FFF_FFFF	保留	1183MB	-
0x3400_0000	0x35FF_FFFF	SMI CS0	32MB	-
0x3200_0000	0x33FF_FFFF	保留	32MB	-
0x3000_0000	0x31FF_FFFF	SMI CS1	32MB	-
0x2C00_0000	0x2FFF_FFFF	保留	64MB	-
0x2800_0000	0x2BFF_FFFF	保留	64MB	-
0x2400_0000	0x27FF_FFFF	保留	64MB	-
0x101F_E000	0x23FF_FFFF	保留	318MB	-
0x101F_D000	0x101F_DFFF	保留	4KB	-
0x101F_B000	0x101F_CFFF	保留	16KB	-
0x101F_A000	0x101F_AFFF	GPIO7 寄存器	4KB	-
0x101F_9000	0x101F_9FFF	GPIO6 寄存器	4KB	-
0x101F_8000	0x101F_8FFF	GPIO5 寄存器	4KB	-
0x101F_7000	0x101F_7FFF	GPIO4 寄存器	4KB	-
0x101F_6000	0x101F_6FFF	I ² C 寄存器	4KB	-
0x101F_5000	0x101F_5FFF	保留	4KB	-
0x101F_4000	0x101F_4FFF	SSP 寄存器	4KB	-
0x101F_3000	0x101F_3FFF	UART2 寄存器	4KB	-
0x101F_2000	0x101F_2FFF	UART1 寄存器	4KB	-
0x101F_1000	0x101F_1FFF	UART0 寄存器	4KB	-
0x101F_0000	0x101F_0FFF	保留	4KB	-
0x101E_F000	0x101E_FFFF	保留	4KB	-
0x101E_B000	0x101E_EFFF	保留	16KB	-
0x101E_A000	0x101E_AFFF	保留	4KB	-
0x101E_9000	0x101E_9FFF	IR 寄存器	4KB	-
0x101E_8000	0x101E_8FFF	RTC 寄存器	4KB	-
0x101E_7000	0x101E_7FFF	GPIO3 寄存器	4KB	-
0x101E_6000	0x101E_6FFF	GPIO2 寄存器	4KB	-
0x101E_5000	0x101E_5FFF	GPIO1 寄存器	4KB	-



起始地址	结束地址	功能	大小	说明
0x101E_4000	0x101E_4FFF	GPIO0 寄存器	4KB	-
0x101E_3000	0x101E_3FFF	Timer2/3 寄存器	4KB	-
0x101E_2000	0x101E_2FFF	Timer0/1 寄存器	4KB	-
0x101E_1000	0x101E_1FFF	WatchDog 寄存器	4KB	-
0x101E_0000	0x101E_0FFF	系统控制寄存器	4KB	-
0x1016_0000	0x101D_FFFF	保留	512KB	-
0x1015_0000	0x1015_FFFF	DDRC 寄存器	64KB	-
0x1014_0000	0x1014_FFFF	INT 寄存器	64KB	-
0x1013_0000	0x1013_FFFF	DMA 寄存器	64KB	-
0x1012_0000	0x1012_FFFF	保留	64KB	-
0x1011_0000	0x1011_FFFF	保留	64KB	-
0x1010_0000	0x1010_FFFF	SMI 寄存器	64KB	-
0x0100_0000	0x100F_FFFF	保留	257MB	-
0x0000_0000	0x00FF_FFFF	-	16MB	<p>复位时根据管脚 FUNCSEL0 的电平决定当前存储空间所对应的器件：</p> <ul style="list-style-type: none">• FUNCSEL0=0，芯片支持从 SMI 接口挂接的 Nor Flash 启动；• FUNCSEL0=1，芯片支持从 DDR 接口挂接的 DDR 启动。 <p>通常情况下，清除地址重映射之后，会把 ITCM 配置到该空间，ITCM 的大小为 2KB，对应地址空间范围为 0x0000_0000~0x0000_0800。</p>

PCI 地址空间分配情况如表 3-12 所示。

表3-12 PCI 地址空间分配

起始地址	结束地址	功能	大小	说明
0xB000_0000	0xB000_03FF	PCI 控制寄存器。	1KB	-
0xB000_0400	0xB000_FFFF	保留。	63KB	-



起始地址	结束地址	功能	大小	说明
0xB001_0000	0xB001_FFFF	IO 地址空间。	64KB	当 PCI 接口处于主模式时，该段地址空间有效。
0xB002_0000	0xB7FF_FFFF	NP 地址空间。	127MB	-
0xB800_0000	0xBFFF_FFFF	FP 地址空间。	128MB	-

3.4 中断系统

3.4.1 概述

中断系统 INT (Interrupt System) 为系统提供中断管理功能。

INT 具有以下特性：

- 支持 32 个中断源，高电平触发。
- 中断类型可配置，各中断源可根据需要配置为 IRQ (Interrupt Request) 或 FIQ (Fast Interrupt Request)。
- 支持中断源屏蔽。
- 支持原始中断源状态查询和屏蔽后中断源状态查询。
- 支持访问保护功能。当设置为启用访问保护状态时，只有处于特权模式下的 CPU 才能访问 INT。
- 支持在 INT 的工作时钟关断的情况下产生中断请求。

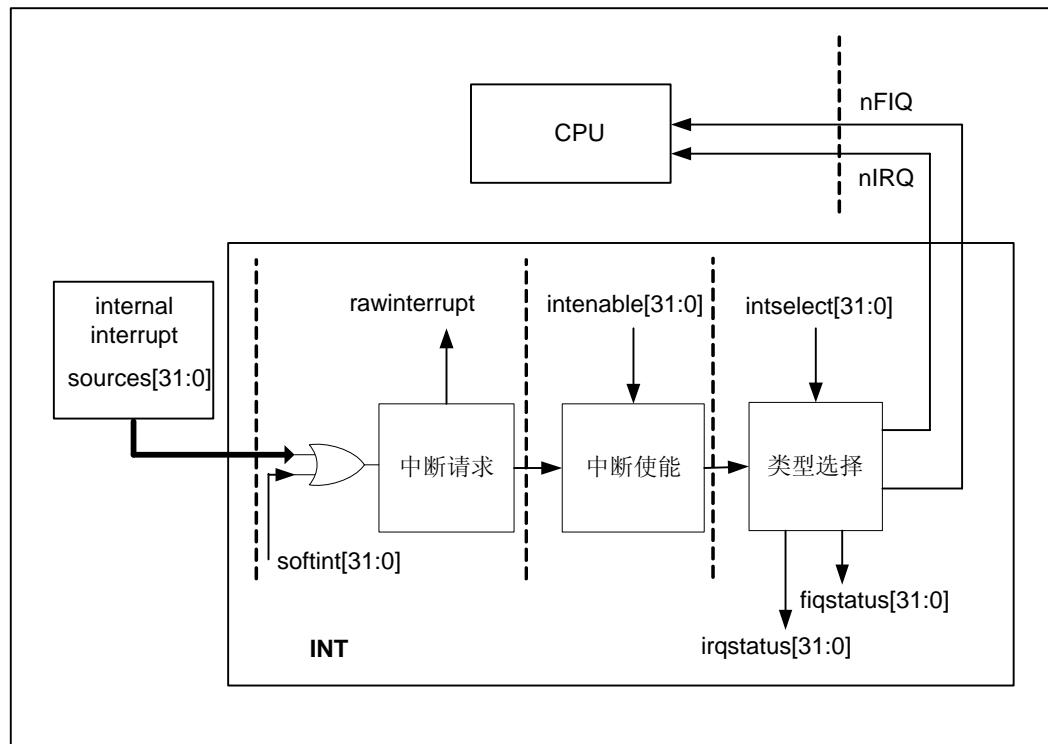
3.4.2 中断系统

功能框图

INT 的功能框图如图 3-6 所示。



图3-6 INT 功能框图



CPU 可通过内部总线访问 INT 的功能寄存器。例如：

- 通过配置 `INT_INTENABLE` 可实现对各中断源中断请求的屏蔽功能。
- 通过配置 `INT_INTSELECT` 可实现对各中断源中断请求类型的选择（通常最多只有一个中断请求会被配置为 FIQ）。
- 通过配置 `INT_SOFTINT` 和 `INT_SOFTINTCLEAR` 分别产生和清除中断请求。
- 通过 `INT_IRQSTATUS`、`INT_FIQSTATUS` 来查询当 INT 产生中断请求时的中断源状态。
- 通过 `INT_RAWINTR` 查询中断源及软件中断所产生的原始中断请求。

中断处理流程

当中断发生时，中断处理的步骤如下：

- 步骤 1 ARM 处理器切换到 IRQ 或 FIQ 中断模式。
- 步骤 2 跳转到初级中断处理程序并压栈。
- 步骤 3 进入 System 模式。
- 步骤 4 查询寄存器 `INT_IRQSTATUS` 以确定中断来源，若存在多个等待服务的中断源则比较各中断的优先级。



步骤 5 跳转到中断源所对应的中断服务程序 ISR(Interrupt Service Routine)，通过配置 [INT_INTENABLE](#) 寄存器屏蔽当前处理的中断号，如果有需要可通过配置 [INT_INTENABLE](#) 寄存器开启相应中断使能。

步骤 6 执行 ISR。

步骤 7 清除当前中断源，若是软中断则配置 [INT_SOFTINTCLEAR](#) 寄存器的相应比特位。

步骤 8 打开 ARM CPSR 寄存器中 IRQ 和 FIQ 屏蔽位。

步骤 9 出栈，并从中断中返回。

----结束

3.4.3 中断映射

中断映射如[表 3-13](#) 所示。

表3-13 中断映射表

中断号	对应的中断源
0	WatchDog 中断
1	软件中断
2	COMMRX 中断
3	COMMTX 中断
4	Dual-Timer0 中断
5	Dual-Timer1 中断
6	GPIO0 中断
7	GPIO1 中断
8	GPIO2、GPIO3、GPIO4、GPIO5、GPIO6、GPIO7 组合中断
9	IR 中断
10	RTC 中断
11	SSP 中断
12	UART0 中断
13	UART1 中断
14	UART2 中断
15	ETH 中断
16	VOU 中断



中断号	对应的中断源
17	DMAC 中断
18	SIO1 中断
19	I ² C 中断
20	USB 1.1 HOST 中断
21	CIPHER 中断
22	外部管脚 INTRN (该管脚为复用管脚) 中断
23	USB 2.0 OTG 中断
24	MMC 中断
25	VIU 中断
26	DSU 中断
27	SIO0 中断
28	VEDU 中断
29	保留
30	PCI 中断
31	TDE 中断

注：COMMRX、COMMTX 中断为 CPU 调试使用。

3.4.4 寄存器概览

INT 寄存器概览如表 3-14 所示。

表3-14 INT 寄存器概览 (基址是 0x1014_0000)

偏移地址	名称	描述	页码
0x000	INT_IRQSTATUS	IRQ 中断状态寄存器	3-20
0x004	INT_FIQSTATUS	FIQ 中断状态寄存器	3-20
0x008	INT_RAWINTR	原始中断状态寄存器	3-21
0x00C	INT_INTSELECT	中断选择寄存器	3-21
0x010	INT_INTENABLE	中断使能寄存器	3-21
0x014	INT_INTENCLEAR	中断使能清除寄存器	3-22
0x018	INT_SOFTINT	软中断寄存器	3-22



偏移地址	名称	描述	页码
0x01C	INT_SOFTINTCLEAR	软中断清除寄存器	3-23
0x020	INT_PROTECTION	保护使能寄存器	3-23

3.4.5 寄存器描述

INT_IRQSTATUS

INT_IRQSTATUS 为 IRQ 中断状态寄存器，用来提供 32 个 IRQ 中断状态。

Offset Address		Register Name		Total Reset Value			
Bit	0x0	INT_IRQSTATUS		0x0000_0000			
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0						
Name	irqstatus						
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0						
Bits	Access	Name	Description				
[31:0]	RO	irqstatus	各比特位对应中断源 IRQ 类型中断状态。 0: 无效。 1: 有效，并向处理器发出 IRQ 中断。				

INT_FIQSTATUS

INT_FIQSTATUS 为 FIQ 中断状态寄存器，用来提供 32 个 FIQ 中断状态。

Offset Address		Register Name		Total Reset Value			
Bit	0x4	INT_FIQSTATUS		0x0000_0000			
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0						
Name	fiqstatus						
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0						
Bits	Access	Name	Description				
[31:0]	RO	fiqstatus	各比特位对应中断源 FIQ 类型中断状态。 0: 无效。 1: 有效，并向处理器发出 FIQ 中断。				



INT_RAWINTR

INT_RAWINTR 为原始中断状态寄存器，用来提供屏蔽前的中断请求状态，也包括软件操作 INT_SOFTINT 产生的中断。

Offset Address																Register Name								Total Reset Value								
0x8																INT_RAWINTR								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rawinterrupt																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:0]	RO	rawinterrupt		各位对应中断源屏蔽前中断请求状态。 0: 无效。 1: 有效。																												

INT_INTSELECT

INT_INTSELECT 为中断选择寄存器，该寄存器的每位选择对应的中断源是生成 IRQ 中断还是 FIQ 中断。

Offset Address																Register Name								Total Reset Value								
0xC																INT_INTSELECT								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	intselect																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:0]	RW	intselect		各位所对应中断源的中断请求类型。 0: IRQ 中断。 1: FIQ 中断。																												

INT_INTENABLE

INT_INTENABLE 为中断使能寄存器，用于使能中断请求线。复位时，由于 INT_INTENABLE 的值变为 0x0000_0000，所有中断源都被屏蔽。

Offset Address																Register Name	Total Reset Value															
0x10																INT_INENABLE	0x0000_0000															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	intenable																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:0]	RW	intenable		<p>读该寄存器时，返回的是各中断源的屏蔽状态。 0: 被屏蔽。 1: 未被屏蔽。</p> <p>写该寄存器时，其作用是按位使能中断源。 0: 对应位的当前值不受影响。 1: 对应位被置 1，使能对应的中断请求。</p>																												

INT_INENCLEAR

INT_INENCLEAR 为中断使能清除寄存器，该寄存器清除中断使能寄存器的相应比特位。该寄存器为只写寄存器，无复位值。

Offset Address																Register Name	Total Reset Value															
0x14																INT_INENCLEAR	-															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	intenableclear																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		
Bits	Access	Name		Description																												
[31:0]	WO	intenableclear		<p>写该寄存器时，其作用是按位屏蔽中断源。 0: INT_INENABLE 寄存器对应位的当前值不受影响。 1: INT_INENABLE 寄存器对应位被清零，对应中断源被屏蔽。</p>																												

INT_SOFTINT

INT_SOFTINT 为软中断寄存器。通过软件可控制中断源输入线产生软件中断。软件中断的清除可通过写软件中断清除寄存器 INT_SOFTINTCLEAR 实现，清除通常是在中断服务程序结束时进行。



Offset Address																Register Name	Total Reset Value															
0x18																INT_SOFTINT	0x0000_0000															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	softint																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:0]	RW	softint		在指定的中断源上产生一个屏蔽前软件中断。 0: 对应位不受影响。 1: 对应位置位, 产生一个软件中断。																												

INT_SOFTINTCLEAR

INT_SOFTINTCLEAR 为软件中断清除寄存器, 用于清除相应的软中断寄存器比特位。该寄存器为只写寄存器, 无复位值。

Offset Address																Register Name	Total Reset Value															
0x1C																INT_SOFTINTCLEAR	-															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	softintclear																															
Reset	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?		
Bits	Access	Name		Description																												
[31:0]	WO	softintclear		将 INT_SOFTINT 寄存器的特定位清零。 0: 寄存器 INT_SOFTINT 的对应位不受影响。 1: 寄存器 INT_SOFTINT 的对应位清零。																												

INT_PROTECTION

INT_PROTECTION 为保护使能寄存器, 用于关断或使能被保护的寄存器访问权限。复位时该寄存器被清零。用户模式或特权模式都可访问 INT 的寄存器。当 CPU 无法产生正确的保护信息 (H PROT) 时, 将该寄存器复位后即可允许在用户模式下访问 INT 的寄存器。



Offset Address												Register Name												Total Reset Value											
0x20												INT_PROTECTION												0x0000_0000											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved																												protection						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
Bits	Access	Name	Description																																
[31:1]	-	reserved	保留。																																
[0]	RW	protection	使能寄存器的访问保护。 0: 不使能寄存器的访问保护, CPU 采用特权模式 (privileged mode) 和用户模式 (user mode) 都可以访问 INT 的寄存器。 1: 使能寄存器访问保护, CPU 只能采用特权模式访问 INT 的寄存器。																																

3.5 直接存储器存取控制器

3.5.1 概述

直接存储器访问 (DMA) 方式, 是一种完全由硬件执行 I/O 交换的工作方式。在这种方式中, 直接存储器访问控制器 (DMAC) 直接在存储器和外设、外设和外设、存储器和存储器之间进行数据传输, 避免处理器干涉并减少了处理器中断处理开销。DMA (Directory Memory Access) 方式一般用于高速传输成组的数据。DMAC (Directory Memory Access Controller) 在收到 DMA 传输请求后根据 CPU 对通道的配置启动总线主控制器, 向存储器和外设发出地址和控制信号, 对传输数据的个数计数, 并且以中断方式向 CPU 报告传输操作的结束或错误。

3.5.2 特点

DMAC 有如下特点:

- 支持 8bit、16bit、32bit 数据位宽方式传输。
- 提供 8 个 DMA 通道, 每个通道可配置用于一种单向传输。
- DMA 通道优先级固定, 优先级从高到低对应的通道号依次为 0~7。当来自 2 个外设的 DMA 请求同时有效时, 优先级高的通道先开始传输。
- DMAC 通道 0~通道 5 中各包含 1 个 4×32bit 的 FIFO, DMAC 通道 6~通道 7 中各包含 1 个 16×32bit 的 FIFO (First In First Out)。
- 提供 2 个总线宽度为 32bit 的 Master 总线接口用于数据传输。
- 外设可使用单次传输 (single) 和连续传输 (burst) 2 种 DMA 请求。



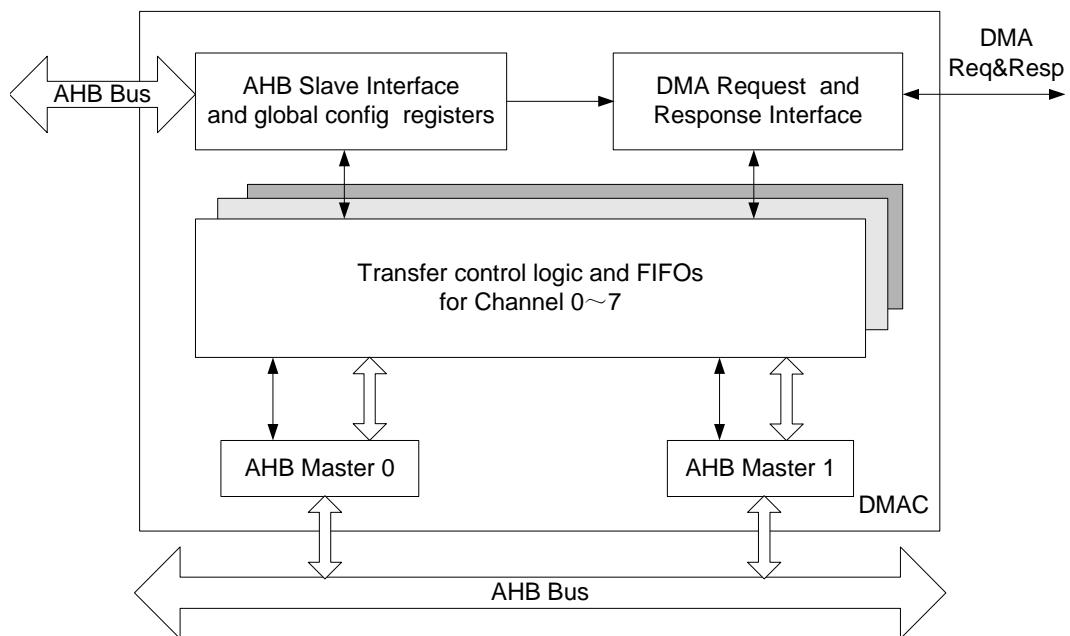
- 提供 16 组 DMA 请求输入，可通过配置，作为通道的源端请求或目的端请求。
- 支持软件控制的 DMA 请求。
- 支持通过编程决定 DMA burst 长度。
- 源地址和目的地址可分别配置为在 DMA 传输过程中自动递增或不递增。
- 支持 4 种数据传输方向：
 - 存储器至外设
 - 存储器至存储器
 - 外设至存储器
 - 外设至外设
- 支持链表 DMA 传输。
- 支持 DMAC 流控。
- 提供 1 个可屏蔽中断输出，支持 DMA 错误和 DMA 传输完成中断屏蔽前后状态查询，及两者的组合中断状态查询。
- 支持 DMAC 使能禁止，用于功耗控制，支持 DMAC 时钟门控。

3.5.3 功能描述

3.5.3.1 功能框图

DMAC 的功能框图如图 3-7 所示。

图3-7 DMAC 功能框图



DMAC 的每一个通道都内含一组传输控制逻辑和一个 FIFO，传输控制逻辑自动完成以下步骤：



步骤 1 从软件指定的源地址位置读取数据。

步骤 2 缓存到通道内含的 FIFO 中。

步骤 3 从通道 FIFO 中取出数据。

步骤 4 写入到软件指定的目的地址位置。

----结束

3.5.3.2 工作流程

DMAC 基本工作流程如下：

步骤 1 软件选定 DMAC 的一个通道用于 DMA 传输，配置该通道的源地址、目的地址、链表头指针、传输数据个数、源/目的端对应的外设请求线号、源/目的端使用的 Master 并启动该通道。一旦通道被启动，DMAC 硬件即开始检测与该通道相连的源外设和目的设备的 DMA 请求线上的活动。

步骤 2 源设备向 DMAC 发起 DMA 请求（如果源设备为存储器，DMAC 默认其 DMA 请求始终有效）。

步骤 3 DMAC 通道响应源设备 DMA 请求，从源设备读取数据并存入通道内部的 FIFO 中。

步骤 4 目的设备向 DMAC 发起 DMA 请求（如果目的设备为存储器，DMAC 默认其 DMA 请求始终有效）。

步骤 5 DMAC 通道响应目的设备 DMA 请求，从通道内部的 FIFO 中取出数据并写入目的设备。

步骤 6 步骤 2、3 和步骤 4、5 可能是并发执行的，因为源设备和目的设备有可能同时向 DMAC 发起 DMA 请求。

当出现 DMA 通道 FIFO 被写满而目的设备来不及读走、或 DMA 通道 FIFO 被读空而源设备来不及写入时，DMAC 自动阻塞源设备或目的设备的 DMA 请求，直到相应的 FIFO 空满状态解除为止。

在 DMAC 与源设备、目的设备的多次交互过程中，步骤 2~5 反复被执行，直到软件指定的数据传输全部完成时，发出传输完成中断（该中断可被屏蔽）。

如果寄存器 [DMAC_CX_LLI](#) 不为 0，则以该寄存器的值为地址读取链表结点，并将读取值依次载入 [DMAC_CX_SRC_ADDR](#)、[DMAC_CX_DEST_ADDR](#)、[DMAC_CX_LLI](#) 以及 [DMAC_CX_CONTROL](#) 这四个寄存器（如图 3-8 所示），然后回到步骤 2。

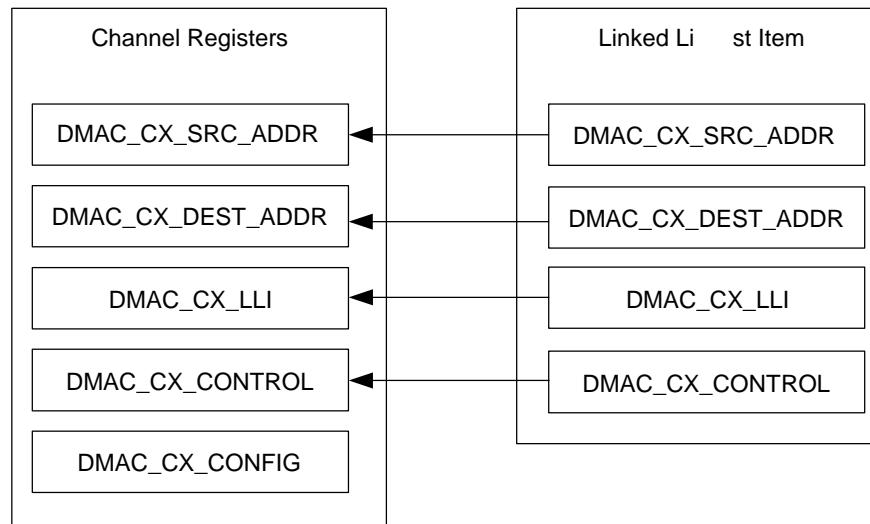
若寄存器 [DMAC_CX_LLI](#) 的值为 0 则停止当前的 DMA 传输，通道自动关闭，传输过程结束。

----结束

LLI 更新通道寄存器示意如图 3-8 所示。



图3-8 LLI 更新通道寄存器示意图



3.5.3.3 DMA 与外设的连接关系

外设利用 DMA 请求信号向 DMAC 请求发起数据传输。

DMA 请求信号

DMAC 为每个外设提供了 2 种 DMA 请求信号，分别为：

- **DMACBREQ**
burst 传输请求信号。该信号引发一次 burst 传输，burst 长度为预先设定值。
- **DMACSREQ**
单次传输请求信号。该信号引发一次单次传输，即 DMAC 从外设读取一个数据或向外设写一个数据。

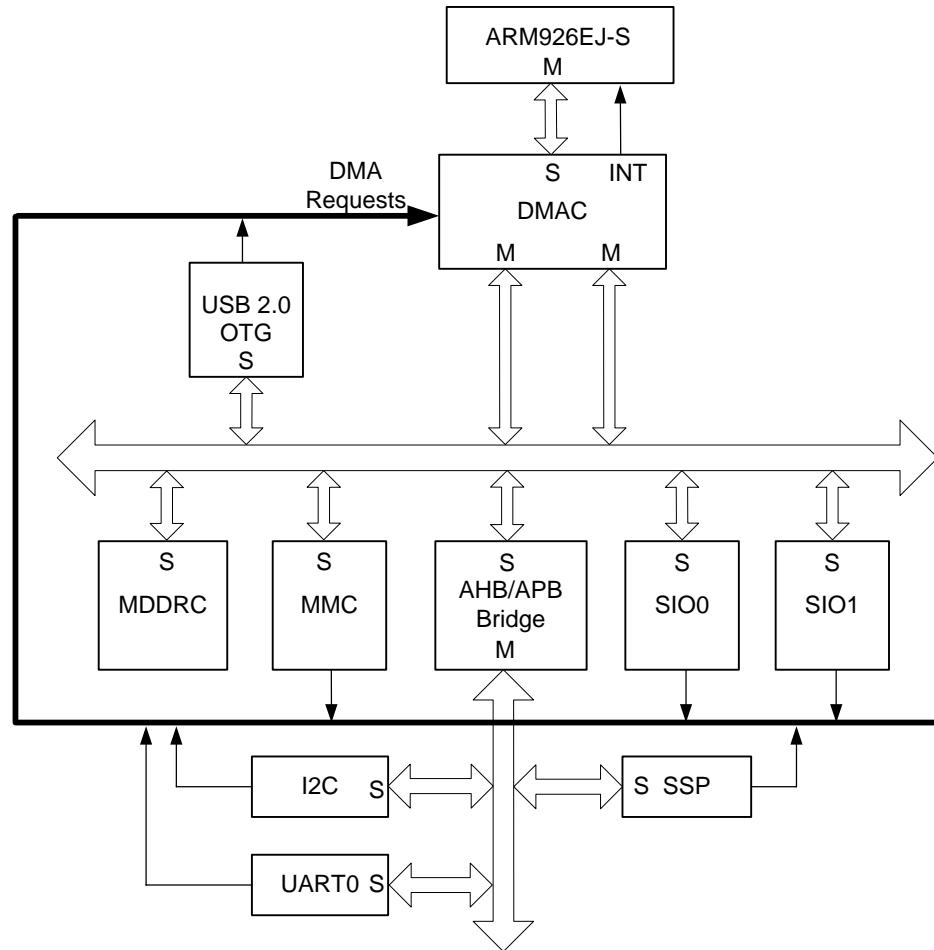
请求清除信号

DMAC 提供一个请求清除信号：

- **DMACLR**
DMAC 向每个外设发出的 DMA 请求清除信号，用于应答外设的 DMA 请求信号。

DMAC 与其他模块的连接关系如图 3-9 所示。

图3-9 DMAC 与其他模块连接关系图



DMAC 请求线

DMAC 的硬件请求和相应设备的对应关系如表 3-15 所示。

表3-15 DMAC 硬件请求和相应设备的对应关系

DMAC 硬件请求编号	对应设备
0	保留。
1	保留。
2	SIO0 接收请求。
3	SIO0 发送请求。
4	SIO1 接收请求。
5	保留。



DMAC 硬件请求编号	对应设备
6	保留。
7	保留。
8	SSP 接收请求。
9	SSP 发送请求。
10	保留。
11	保留。
12	MMC 接收请求。
13	MMC 发送通道。
14	UART0 接收请求。
15	UART0 发送请求。

DMA 通道对应的源端和目的端请求由软件配置。例如，DMA 请求号 2 为 SIO0 的接收通道请求，若希望使用通道 3 传输 SIO0 的接收数据，则应配置 DMA 请求号 2 作为通道 3 的源端请求。

存储器没有 DMA 请求线，当 DMA 传输的一方为存储器时，DMAC 默认其 DMA 请求是始终有效的。由于 DMAC 的通道 6、7 上的传输，每次总线操作之后，都会插入 IDLE 周期，供高优先级通道的 Master 占用总线进行传输；因此，建议将存储器到存储器的传输配置到通道 6、7 进行，以免总线上其它通道较长时间无法占用总线。

3.5.4 工作方式

初始化

初始化 DMAC 步骤如下：

- 步骤 1 配置寄存器 **DMAC_CONFIG**，设置 DMAC 的 Master0、Master1 的 Endianness，向 **DMAC_CONFIG[e]** 写 1，启动 DMAC。
- 步骤 2 配置寄存器 **DMAC_INT_TC_CLR** 和 **DMAC_INT_ERR_CLR** 的所有位为 1，清除所有中断状态。
- 步骤 3 配置寄存器 **DMAC_SYNC** 相应位为 0，设置需要进行同步的 DMA 请求信号组。
- 步骤 4 依次配置并关闭各个通道。向每个通道的 **DMAC_CX_CONFIG[e]** 写 0，关闭通道。

----结束

启动通道

DMAC 初始化完成之后，需要配置并启动 DMA 通道，才可以使用 DMAC 来进行数据传输。通道的配置启动步骤如下：



步骤 1 读寄存器 **DMAC_ENBLD_CHNS**, 找出处于非传输状态的通道, 并从中选择一个通道用于配置。

步骤 2 向寄存器 **DMAC_INT_TC_CLR** 和 **DMAC_INT_ERR_CLR** 的相应位写 1, 清除选定通道的中断状态。

步骤 3 配置并启动选定的通道。配置步骤如下:

1. 配置通道寄存器 **DMAC_CX_SRC_ADDR**, 设置源设备访问首地址。
2. 配置通道寄存器 **DMAC_CX_DEST_ADDR**, 设置目的设备访问首地址。
3. 如果配置通道用于单块数据传输, 则将通道寄存器 **DMAC_CX_LLI** 配置为 0。
4. 如果配置通道用于链表数据传输, 则将通道寄存器 **DMAC_CX_LLI** 配置为链表头指针。
5. 配置通道寄存器 **DMAC_CX_CONTROL**, 设置访问源/目的设备所采用的 Master、源/目的设备的位宽、burst size、地址递增以及 transfer size 等参数。
6. 配置通道寄存器 **DMAC_CX_CONFIG**, 设置本通道的 DMA 请求信号、流控方式及中断屏蔽。**DMAC_CX_CONFIG[e]**此时应写入 0, 即暂不启动该通道。
7. 配置通道寄存器 **DMAC_CX_CONFIG**, 启动该通道。注意此时该寄存器的写入值除 Channel Enable 位改为 1 外, 其他位不变。

----结束

中断处理

DMA 通道配置启动传输完成之后或传输过程中出现错误, 都会上报中断给中断控制器。中断程序的处理步骤如下:

步骤 1 读中断状态寄存器 **DMAC_INT_STAT**, 找出发出中断请求的通道。当多个通道同时发出中断请求时, 先服务优先级最高的中断。

步骤 2 读寄存器 **DMAC_INT_TC_STAT**, 比较选定的位是否为 1, 以确定对应通道发出的中断为传输完成中断。若是, 则转到步骤 4 执行; 否则转到步骤 3 继续执行。

步骤 3 读寄存器 **DMAC_INT_ERR_STAT**, 比较选定的位是否为 1, 以确定对应通道发出的中断为错误中断。若是则转到步骤 5 执行; 否则退出中断处理。

步骤 4 传输完成中断处理。可分为以下 3 个子步骤:

1. 配置寄存器 **DMAC_INT_TC_CLR**, 对选定的位写 1, 清除对应通道的中断状态。
2. 取走或使用掉内存中 buffer 中的数据, 如有需要(例如: 需在内存中新开辟一个 buffer)可重新配置并启动该通道。
3. 退出中断处理。

步骤 5 错误中断处理。可分为以下 3 个子步骤:

1. 配置寄存器 **DMAC_INT_ERR_CLR**, 对选定的位写 1, 清除对应通道的中断状态。
2. 给出错误信息, 有需要可重新配置并启动该通道。
3. 退出中断处理。



----结束

3.5.5 寄存器概览

DMAC 寄存器概览如表 3-16 所示。

表3-16 DMAC 寄存器概览（基址是 0x1013_0000）

偏移地址	寄存器名	功能简述	页码
0x000	DMAC_INT_STAT	DMAC 中断状态寄存器	3-33
0x004	DMAC_INT_TC_STAT	DMAC 传输完成中断状态寄存器	3-33
0x008	DMAC_INT_TC_CLR	DMAC 传输完成中断清除寄存器	3-34
0x00C	DMAC_INT_ERR_STAT	DMAC 错误中断状态寄存器	3-34
0x010	DMAC_INT_ERR_CLR	DMAC 错误中断清除寄存器	3-35
0x014	DMAC_RAW_INT_TC_STAT	DMAC 原始传输完成中断状态寄存器	3-35
0x018	DMAC_RAW_INT_ERR_STAT	DMAC 原始错误中断状态寄存器	3-36
0x01C	DMAC_ENBLD_CHNS	DMAC 通道使能状态寄存器	3-36
0x020	DMAC_SOFT_BREQ	软件 Burst DMA 请求寄存器	3-37
0x024	DMAC_SOFT_SREQ	软件 Single DMA 请求寄存器	3-38
0x028	DMAC_SOFT_LBREQ	软件 Last Burst DMA 请求寄存器	3-38
0x02C	DMAC_SOFT_LSREQ	软件 Last Single DMA 请求寄存器	3-39
0x030	DMAC_CONFIG	DMAC 配置寄存器	3-39
0x034	DMAC_SYNC	DMAC 请求同步寄存器	3-40
0x100	DMAC_C0_SRC_ADDR	通道 0 源地址寄存器	3-41
0x104	DMAC_C0_DEST_ADDR	通道 0 目的地址寄存器	3-42
0x108	DMAC_C0_LLI	通道 0 链表项寄存器	3-42
0x10C	DMAC_C0_CONTROL	通道 0 控制寄存器	3-43
0x110	DMAC_C0_CONFIG	通道 0 配置寄存器	3-47
0x120	DMAC_C1_SRC_ADDR	通道 1 源地址寄存器	3-41



偏移地址	寄存器名	功能简述	页码
0x124	DMAC_C1_DEST_ADDR	通道 1 目的地址寄存器	3-42
0x128	DMAC_C1_LLI	通道 1 链表项寄存器	3-42
0x12C	DMAC_C1_CONTROL	通道 1 控制寄存器	3-43
0x130	DMAC_C1_CONFIG	通道 1 配置寄存器	3-47
0x140	DMAC_C2_SRC_ADDR	通道 2 源地址寄存器	3-41
0x144	DMAC_C2_DEST_ADDR	通道 2 目的地址寄存器	3-42
0x148	DMAC_C2_LLI	通道 2 链表项寄存器	3-42
0x14C	DMAC_C2_CONTROL	通道 2 控制寄存器	3-43
0x150	DMAC_C2_CONFIG	通道 2 配置寄存器	3-47
0x160	DMAC_C3_SRC_ADDR	通道 3 源地址寄存器	3-41
0x164	DMAC_C3_DEST_ADDR	通道 3 目的地址寄存器	3-42
0x168	DMAC_C3_LLI	通道 3 链表项寄存器	3-42
0x16C	DMAC_C3_CONTROL	通道 3 控制寄存器	3-43
0x170	DMAC_C3_CONFIG	通道 3 配置寄存器	3-47
0x180	DMAC_C4_SRC_ADDR	通道 4 源地址寄存器	3-41
0x184	DMAC_C4_DEST_ADDR	通道 4 目的地址寄存器	3-42
0x188	DMAC_C4_LLI	通道 4 链表项寄存器	3-42
0x18C	DMAC_C4_CONTROL	通道 4 控制寄存器	3-43
0x190	DMAC_C4_CONFIG	通道 4 配置寄存器	3-47
0x1A0	DMAC_C5_SRC_ADDR	通道 5 源地址寄存器	3-41
0x1A4	DMAC_C5_DEST_ADDR	通道 5 目的地址寄存器	3-42
0x1A8	DMAC_C5_LLI	通道 5 链表项寄存器	3-42
0x1AC	DMAC_C5_CONTROL	通道 5 控制寄存器	3-43
0x1B0	DMAC_C5_CONFIG	通道 5 配置寄存器	3-47
0x1C0	DMAC_C6_SRC_ADDR	通道 6 源地址寄存器	3-41
0x1C4	DMAC_C6_DEST_ADDR	通道 6 目的地址寄存器	3-42
0x1C8	DMAC_C6_LLI	通道 6 链表项寄存器	3-42
0x1CC	DMAC_C6_CONTROL	通道 6 控制寄存器	3-43



偏移地址	寄存器名	功能简述	页码
0x1D0	DMAC_C6_CONFIG	通道 6 配置寄存器	3-47
0x1E0	DMAC_C7_SRC_ADDR	通道 7 源地址寄存器	3-41
0x1E4	DMAC_C7_DEST_ADDR	通道 7 目的地址寄存器	3-42
0x1E8	DMAC_C7_LLI	通道 7 链表项寄存器	3-42
0x1EC	DMAC_C7_CONTROL	通道 7 控制寄存器	3-43
0x1F0	DMAC_C7_CONFIG	通道 7 配置寄存器	3-47

3.5.6 寄存器描述

DMAC INT STAT

DMAC_INT_STAT 为中断状态寄存器，给出经过屏蔽后的中断状态。若寄存器 DMAC_INT_TC_STAT 和 DMAC_INT_ERR_STAT 的相应位同时被屏蔽，则该寄存器的对应位被屏蔽。该寄存器的每 1 位对应着 DMAC 的 1 个通道。

DMAC INT_TC_STAT

DMAC_INT_TC_STAT 为传输结束状态寄存器，给出经过屏蔽后的传输完成中断状态，对应的屏蔽位为寄存器 DMAC_CX_CONFIG[itc] (其中 X 表示通道号 0~7)。该寄存器必须和寄存器 DMAC_INT_STAT 结合在一起使用。



DMAC INT_TC_CLR

DMAC INT TC CLR 为传输结束状态清除寄存器，用于清除传输完成中断。

DMAC INT ERR STAT

DMAC_INT_ERR_STAT 为错误中断状态寄存器，给出经过屏蔽后的错误中断状态，对应的屏蔽位为寄存器 DMAC_CX_CONFIG[ie]。该寄存器必须和 DMAC_INT_STAT 寄存器结合在一起使用。



DMAC INT ERR CLR

DMAC INT_ERR_CLR 为错误中断清除寄存器，用于清除出错中断。

DMAC RAW INT TC STAT

DMAC_RAW_INT_TC_STAT 为原始传输完成中断状态寄存器，给出各通道屏蔽前的传输完成中断状态。



DMAC RAW INT ERR STAT

DMAC_RAW_INT_ERR_STAT 为原始传输错误中断状态寄存器，给出各通道屏蔽前的错误中断状态。

DMAC ENBLD CHNS

DMAC_ENBLD_CHNS 为使能通道寄存器，用于表明被使能的通道。

如寄存器 `DMAC_ENBLD_CHNS` 的某位为 1，表示对应的通道被使能。使能某个通道，由该通道的通道寄存器 `DMAC_CX_CONFIG` 的使能位决定。当某个通道的 DMA 传输结束时，寄存器 `DMAC_ENBLD_CHNS` 中与该通道对应的位被清零。



Offset Address			Register Name			Total Reset Value																										
0x01C			DMAC_ENBLD_CHNS			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																								enabled_channels							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:8]	-	reserved	保留。																													
[7:0]	RO	enabled_channels	通道使能状态, bit[7:0]分别对应通道 7~通道 0。 0: 禁止。 1: 使能。																													

DMAC_SOFT_BREQ

DMAC_SOFT_BREQ 为软件 Burst 请求寄存器, 用于供软件控制产生 DMA burst 请求。

读该寄存器, 可得知当前正在请求 DMA Burst 传输的设备。外设和该寄存器都可以产生 1 个 DMA 请求。



说明

建议不要同时使用软件 DMA 请求和硬件 DMA 请求。

Offset Address			Register Name			Total Reset Value																										
0x020			DMAC_SOFT_BREQ			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																								soft_breq							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15:0]	RW	soft_breq	用于软件控制产生 DMA burst 传输请求, 每比特对应请求见表 3-15。 写该寄存器时: 0: 无影响。 1: 产生 DMA burst 传输请求, 当传输结束时该寄存器中的相应位被清零。 读该寄存器时: 0: 与请求线 DMACBREQ[15:0]对应的外设未发出 DMA Burst 请求。																													

			1: 与请求线 DMACBREQ[15:0]对应的外设正在请求 DMA Burst 传输。
--	--	--	---

DMAC_SOFT_SREQ

DMAC_SOFT_SREQ 为软件 Single 请求寄存器，用于供软件控制产生 DMA 单次传输请求。

如读该寄存器，可得知当前正在请求 DMA 单次传输的设备。通过 DMAC 的 16 个 DMA 请求输入信号和该寄存器都可以产生 1 个 DMA 请求。

说明

建议不要同时使用软件 DMA 请求和硬件 DMA 请求。

	Offset Address	Register Name	Total Reset Value
	0x024	DMAC_SOFT_SREQ	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	soft_sreq
Reset	0 0		
Bits	Access	Name	Description
[31:16]	-	reserved	保留。
[15:0]	RW	soft_sreq	<p>用于软件控制产生 DMA single 传输请求，每比特对应请求见表 3-15。</p> <p>写该寄存器时：</p> <p>0: 无影响。</p> <p>1: 产生 DMA single 传输请求，当传输结束时该寄存器中的相应位被清零。</p> <p>读该寄存器时：</p> <p>0: 与请求线 DMACBREQ[15:0]对应的外设未发出 DMA signal 请求。</p> <p>1: 与请求线 DMACBREQ[15:0]对应的外设正在请求 DMA signal 传输。</p>

DMAC_SOFT_LBREQ

DMAC_SOFT_LBREQ 为软件最后一个 burst 请求寄存器，用于供软件控制产生 DMA last burst 传输请求。



Offset Address			Register Name			Total Reset Value					
Bit	0x028			DMAC_SOFT_LBREQ			0x0000_0000				
Name	reserved						soft_lbreq				
Reset	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description								
[31:16]	-	reserved	保留。								
[15:0]	WO	soft_lbreq	由软件发起 last burst 请求, 每比特对应请求请参见表 3-15。 0: 无影响。 1: 产生 DMA last burst 传输请求, 当传输结束时该寄存器中的相应位被清零。								

DMAC_SOFT_LSREQ

DMAC_SOFT_LSREQ 为软件最后一个 single 请求寄存器, 用于供软件控制产生 DMA last single 传输请求。

Offset Address			Register Name			Total Reset Value					
Bit	0x02C			DMAC_SOFT_LSREQ			0x0000_0000				
Name	reserved						soft_lsreq				
Reset	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description								
[31:16]	-	reserved	保留。								
[15:0]	WO	soft_lsreq	由软件发起 last single 传输请求, 每比特对应请求请参见表 3-15。 0: 无影响。 1: 产生一个 DMA last single 传输请求, 当传输结束时该寄存器中的相应位被清零。								

DMAC_CONFIG

DMAC_CONFIG 为配置寄存器, 用于配置 DMAC 的操作。通过写该寄存器的 m1 (bit[1]) 和 m2 (bit[2]), 可改变 DMAC 的 2 个 master 接口的 endianness。复位时, DMAC 的 2 个 master 接口被设为 little endian 模式。

 说明

2 个 master 均采用 little endian。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	DMAC_CONFIG	0x0000_0000
Name	reserved		m2 m1 e
Reset	0 0		
Bits	Access	Name	Description
[31:3]	-	reserved	保留。
[2]	RW	m2	Master 2 endianness 配置。 0: little endian 模式。 1: big endian 模式。
[1]	RW	m1	Master 1 endianness 配置。 0: little endian 模式。 1: big endian 模式。
[0]	RW	e	DMAC 使能。 0: 禁止 DMAC。 1: 使能 DMAC。

DMAC_SYNC

DMAC_SYNC 为同步寄存器，用于启用或禁用为 DMA 请求信号提供的同步逻辑。

 说明

建议各请求均不启用同步。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	DMAC_SYNC	0x0000_0000
Name	reserved		dmac_sync
Reset	0 0		
Bits	Access	Name	Description
[31:16]	-	reserved	保留。
[15:0]	RW	dmac_sync	控制是否需要对请求线进行同步，每比特对应请求请参见 表 3-15。 0: 使能对应外设的 DMA 请求信号同步逻辑。



		1: 禁止对应外设的 DMA 请求信号同步逻辑。
--	--	--------------------------

DMAC_CX_SRC_ADDR

DMAC_CX_SRC_ADDR 为源地址寄存器，给出当前待传数据的源地址（字节排序）。

寄存器的偏移地址为 $0x100 + X \times 0x20$ 。其中 X 的取值为 0~7，分别对应 DMA 通道 0~通道 7。

每个寄存器在对应的通道被启动前都要由软件对其直接编程。当通道被启动后，该寄存器在下列情况下更新：

- 当源地址递增时。
- 当传完一个完整的数据块后，从链表结点中载入时。

当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。对该寄存器的读操作一般是用在通道停止传输的时候，此时读取值显示的是 DMAC 读最后一项时的源地址。

源地址和目的地址必须与源设备和目的设备的传输宽度对齐。

	Offset Address	Register Name	Total Reset Value
Bit	0x100+X × 0x20	DMAC_CX_SRC_ADDR	0x0000_0000
Name	src_addr		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	src_addr	DMA 源地址。



说明

DMAC 提供了 8 个通道，每个通道都包括 5 个通道寄存器：

- DMAC_CX_SRC_ADDR 寄存器
- DMAC_CX_DEST_ADDR 寄存器
- DMAC_CX_LLI 寄存器
- DMAC_CX_CONTROL 寄存器
- DMAC_CX_CONFIG 寄存器

当 DMA 从存储器中载入链表结点时，前 4 个寄存器由 DMAC 自动更新。



注意

在 DMA 传输正在进行时，更新通道寄存器会导致 DMAC 产生不可预测的行为。要改变通道的配置，必须先关闭通道然后再配置相关寄存器。



DMAC_CX_DEST_ADDR

DMAC_CX_DEST_ADDR 为目的地址寄存器，偏移地址为：0x104+X×0x20。其中，X 的取值为 0~7，分别对应 DMA 通道 0~通道 7。

通道目的地址寄存器 DMAC_CX_DEST_ADDR 包含了当前待传数据的目的地址（字节排序）。每个寄存器在对应的通道被启动前，都要由软件对其直接编程。当通道被启动后，该寄存器在下列情况下更新：

- 目的地址递增。
 - 传完一个完整的数据块后，从链表结点中载入。

当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。在通道停止传输时，读该寄存器，此时读取值显示的是 DMAC 写最后一项时的目的地址。

DMAC_CX_LLI

DMAC_CX_LLI 为链表寄存器，偏移地址为：0x108+X×0x20。其中，X 的取值为 0~7，分别对应 DMA 通道 0~通道 7。

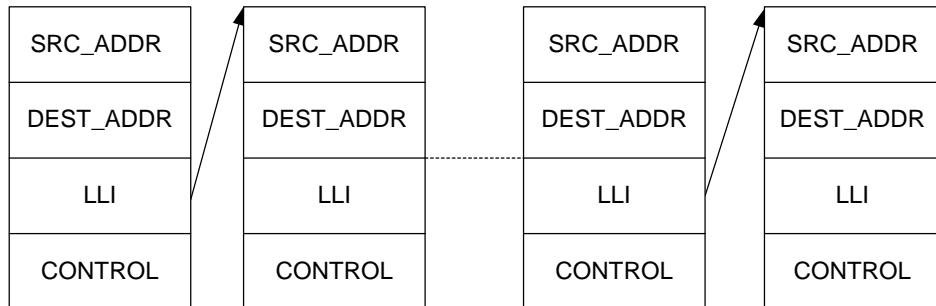
DMAC 的链表结点数据结构为：

- 通道寄存器 **DMAC_CX_SRC_ADDR**, 设置源设备首地址。
 - 通道寄存器 **DMAC_CX_DEST_ADDR**, 设置目的设备首地址。
 - 通道寄存器 **DMAC_CX_LLI**, 设置下一个结点的地址。
 - 通道寄存器 **DMAC_CX_CONTROL**, 设置访问源/目的设备所采用的 Master、源/目的设备的位宽、burst size、地址递增以及 transfer size 等参数。

DMAC 链表结构示例如图 3-10 所示。



图3-10 DMAC 链表结构示例



注意

该寄存器的 LLI 字段不应指定 1 个大于 0xFFFF_FFF0 的数。否则，1 个 4 字的 burst 传输将使地址回卷到 0x0000_0000 处，导致链表结点数据结构不能存储在连续的地址区域中。

如果 LLI 的值为 0，表示当前结点是链表的链尾，则当本结点对应的数据块全部传完后，该通道就会被关闭。

	Offset Address	Register Name	Total Reset Value
Bit	0x108+X × 0x20	DMAC_CXLLI	0x0000_0000
Name	lli		
Reset	0 0		reserved lm
Bits	Access	Name	Description
[31:2]	RW	lli	Linked list item。下一个链表结点地址的[31:2]位，地址位[1:0]为 0。要求链表地址 4 字节对齐。
[1]	RW	reserved	保留，写入时必须写 0，读出时应屏蔽该位。
[0]	RW	lm	用于载入下一个链表结点的 Master。 0: Master 1。 1: Master 2。

DMAC_CX_CONTROL

DMAC_CX_CONTROL 为通道控制寄存器，偏移地址为：0x10C+X × 0x20。其中，X 的取值为 0~7，分别对应 DMA 通道 0~通道 7。



通道控制寄存器包含了 DMA 通道控制信息，如传输长度、burst 长度、传输位宽等。

每个寄存器在对应的通道被启动前，都要由软件对其直接编程。当通道被启动后，该寄存器的值在传完 1 个完整的数据块后，从链表结点载入时更新。

当该通道处于活动状态时，读该寄存器得不到有效信息。因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变。在通道停止传输时，可进行该寄存器的读操作。

Offset Address																Register Name	Total Reset Value															
0x10C + X × 0x20																DMAC_CX_CONTROL	0x0000_0000															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	i	prot	di	si	d	s	dwidth	swidth	dbsize	sbsize																						transfersize
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31]	RW	i	传输完成中断使能位。该位用于决定当前链表结点是否触发传输完成中断。 0：当前链表结点不触发传输完成中断。 1：当前链表结点触发传输完成中断。																													
[30:28]	RW	prot	Master 发出的访问保护 HPROT[2:0]信号，这几位的具体含义请参见表 3-19 所示。																													
[27]	RW	di	目的地址递增。 0：目的地址不递增。 1：目的地址每传一个数就递增一次。 目的设备为外设时目的地址不递增；目的设备为存储器时目的地址递增。																													
[26]	RW	si	源地址递增。 0：源地址不递增。 1：源地址每传一个数就递增一次。 源设备为外设时源地址不递增；源设备为存储器时源地址递增。																													
[25]	RW	d	设置访问目的设备的 Master。 0：SIO0、SIO1、UART0、MMC、SSP、I ² C 使用 Master 1 访问。 1：Nor Flash、DDR 使用 Master 2 访问。																													
[24]	RW	s	设置访问源设备的 Master。 0：SIO0、SIO1、UART0、MMC、SSP、I ² C 使用 Master 1 访问。 1：Nor Flash、DDR 使用 Master 2 访问。																													



[23:21]	RW	dwidth	目的设备传输位宽。 宽于 master 位宽的传输位宽是非法的。 目的设备和源设备的位宽可以不一样，硬件自动对数据进行 pack 和 unpack。 DWidth 的值和具体的位宽对应关系请参见 表 3-18 。
[20:18]	RW	swidth	源设备传输位宽。 宽于 master 位宽的传输位宽是非法的。 目的设备和源设备的位宽可以不一样，硬件自动对数据进行 pack 和 unpack。 SWidth 的值和具体的位宽对应关系请参见 表 3-18 。
[17:15]	RW	dbsize	目的设备 burst 长度。 表示 1 次目的设备 burst 传输所需传输的数据个数，即当 DMACCxBREQ 有效时，传输的数据个数。 该值必须被设为目的设备支持的 burst 大小，或者若目的设备为存储器，被设为到存储地址边界的存储区域大小。 DBSize 的值和具体的传输长度的对应关系请参见 表 3-17 。
[14:12]	RW	sbsize	源设备 burst 长度。 表示 1 次源设备 burst 传输所需传输的数据个数，即当 DMACCxBREQ 有效时，传输的数据个数。 该值必须被设为源设备支持的 burst 大小，或者若源设备为存储器时，被设为到存储地址边界的存储区域大小。 SBSIZE 的值和具体的传输长度的对应关系请参见 表 3-17 。
[11:0]	RW	transfersize	通过写该寄存器可设定 DMA 传输的长度，前提是 DMA 是流控制器。这里 transfer size 表示的源设备待传数据的个数。 读该寄存器可得到在与目的设备相连的总线上已传出的数据个数。 当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。对该寄存器的读操作一般是用在通道被启动后然后又停止传输时。

[DMAC_CX_CONTROL](#) 寄存器的 DBSize 及 SBSIZE 的值与其对应的 burst 长度如 [表 3-17](#) 所示。

表3-17 DBSize 及 SBSIZE 的值与其对应的 burst 长度

DBSize 或 SBSIZE 的值	burst 长度
000	1
001	4

DBSize 或 SBSize 的值	burst 长度
010	8
011	16
100	32
101	64
110	128
111	256

[DMAC_CX_CONTROL](#) 寄存器的 DWidth 和 SWidth 的值与其对应传输位宽如表 3-18 所示。

表3-18 DWidth 和 SWidth 的值与其对应传输位宽

SWidth 或 DWidth 的值	传输位宽
000	Byte (8bit)
001	Halfword (16bit)
010	Word (32bit)
其他	保留

配置寄存器 [DMAC_CX_CONTROL](#) 时需注意：

- 当源设备的传输宽度小于目的设备传输宽度时，源设备的传输宽度与 transfer size 的乘积应为目的设备传输宽度的整数倍，否则 FIFO 中的数据将会滞留并丢失。
- SWidth 和 DWidth 字段不能设置为未定义的位宽。
- transfer size 字段若被写为 0 且 DMAC 又是流控制器，则 DMAC 将不会发生任何传输动作。编程者应负责关闭此 DMA 通道并对此通道重新编程。
- 不应对 DMAC_CX_CONTROL 寄存器进行普通的写入/读出测试。由于 transfer size 字段不是一个普通的可写入并读回相同值的寄存器字段。当写入时，该字段如一个控制寄存器，因为其决定了 DMAC 应传输多少个数据；当读回时，该字段则相当于一个状态寄存器，因为其返回的剩下的待传输数据个数（以源设备位宽为单位）。
- 当 transfer size 字段的设置值大于源设备或目的设备中的 FIFO 的深度（是外设的 FIFO，不是 DMAC 的 FIFO），则 DMAC 的源地址或目的地址必须被设为不递增模式，否则有可能导致外设的 FIFO 溢出。

总线访问信息在传输发生时由 master 接口信号提供给源设备或目的设备。这些访问信息是通过对通道寄存器编程设定的 [DMAC_CX_CONTROL](#)[prot]和 [DMAC_CX_CONFIG](#)[Lock]位。[表 3-19](#) 给出了使用 prot 的 3 个保护位的含义。



表3-19 DMAC_CX_CONTROL 寄存器 Prot 段属性及定义

比特	描述	目的
[2]	Cacheable or nonCacheable	指明访问是可 Cache 还是不可 Cache。 0: 不可 Cache。 1: 可 Cache。 该位控制总线信号 HPROT[3]的输出。
[1]	bufferable or nonbufferable	指明访问是可缓冲还是不可缓冲。 0: 不可缓冲。 1: 可缓冲。 该位控制总线信号 HPROT[2]的输出。
[0]	privileged or user	指明访问是用户模式还是特权模式。 0: 用户模式。 1: 特权模式。 该位控制总线信号 HPROT[1]的输出。

DMAC_CX_CONFIG

DMAC_CX_CONFIG 为通道配置寄存器，偏移地址为：0x110 + X × 0x20。其中，X 的取值为 0~7，分别对应 DMA 通道 0~通道 7。

该寄存器在新的链表结点被载入时不会被更新。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	DMAC_CX_CONFIG	0x0000_0000
Name	reserved	h a l ic ie flow_cntrl reserved dest_peripheral reserved src_peripheral e	
Reset	0 0		
Bits	Access	Name	Description
[31:19]	-	reserved	保留。 写入时必须写入 0，读出时应被屏蔽。
[18]	RW	h	Halt 位。 0: 允许 DMA 请求。 1: 忽略后来的 DMA 请求，通道 FIFO 中的内容都被传完。 该位可以和 Active 位以及 Channel Enable 位一起用于无数据

			丢失地关闭一个 DMA 通道。
[17]	RO	a	Active 位。 0: 通道 FIFO 中没有数据。 1: 通道 FIFO 中有数据。 该位可以和 Halt 位以及 Channel Enable 位一起用于无数据丢失地关闭一个 DMA 通道。
[16]	RW	l	Lock 位。 0: 禁止总线上 lock 传输。 1: 使能总线上 lock 传输。
[15]	RW	itc	传输完成中断屏蔽位。 0: 屏蔽本通道的传输完成中断。 1: 不屏蔽本通道的传输完成中断。
[14]	RW	ie	错误中断屏蔽位。 0: 屏蔽本通道的错误中断。 1: 不屏蔽本通道的错误中断。
[13:11]	RW	flow_cntrl	流控及传输类型字段。 该字段用于指定流控制器和传输类型。流控制器可以是 DMAC、源设备和目的设备。 传输类型可以是存储器到外设、外设到存储器、外设到外设、存储器到存储器。详细描述请参见表 3-20 所示。
[10]	-	reserved	保留。 写入时必须写入 0, 读出时应被屏蔽。
[9:6]	RW	dest_peripheral	目的设备。该字段用于选择一个外设请求信号作为本通道的 DMA 目的设备的请求信号。 如果 DMA 传输的目的设备是存储器则该字段被忽略。
[5]	-	reserved	保留。 写入时必须写入 0, 读出时应被屏蔽。
[4:1]	RW	src_peripheral	源设备。该字段用于选择一个外设请求信号作为本通道的 DMA 源设备的请求信号。 如果 DMA 传输的源设备是存储器则该字段被忽略。
[0]	RW	e	通道使能位。读该寄存器可获得知本通道的状态（也可通过读寄存器 DMACEnbldChns 获得）。 0: 关闭通道。 1: 启动通道。 通过清零可关闭通道。将该位被清零时, 当前的总线传输会继续执行直到完成。然后通道关闭, FIFO 中剩余的数据全部丢失; 当最后一个 LLI 完成或传输中出现错误时, 通道也会



		<p>被关闭，同时该位被清零；如果要关闭通道，而又不使通道 FIFO 中的数据丢失，则 Halt 位也必须同时被置位，使通道忽略后来的 DMA 请求。然后必须轮询 active 位，直到其值变为 0，表明通道 FIFO 中不再留有数据。此时才能够清除 enable 位。</p> <p>通过置位启动通道必须先重新初始化通道，然后才能再次启动通道；若通过简单的置位启动通道，会引发不可预测性的后果。</p>
--	--	--

注：当刚通过写 Channel Enable 位关闭一个通道时，必须要等到轮询到寄存器 **DMAC_ENBLD_CHNS** 中的相应 bit 为 0 之后，才能将 Channel Enable 位重新置位。这是因为通道实际的关闭并没有在将 Channel Enable 位清零后立即生效。总线 burst 的运行时延时也必须要考虑到。

表 3-20 描述了 **DMAC_CX_CONFIG** 寄存器的 flow_cntrl 字段对应的流控和传输类型。

表3-20 流控制器及传输类型位定义

比特值	传输类型	控制器
000	存储器至存储器	DMAC
001	存储器至外设	DMAC
010	外设至存储器	DMAC
011	源设备至目的设备	DMAC
其他	保留	保留

3.6 加解密引擎

3.6.1 概述

CIPHER 是一个实现 DES/3DES 和 AES 加解密处理的模块，DES/3DES 和 AES 算法的实现符合 FIPS46-3/FIPS 197 标准。DES/3DES 和 AES 的工作模式符合 FIPS -81/NIST special800-38a 标准。

CIPHER 模块适用于进行大量数据的高效加解密处理，可以提供一次实现多个分组的加解密操作和一次实现单个分组的加解密操作。

3.6.2 特点

加解密引擎有如下特点：

- AES 密钥长度支持 128、192、256 位。
- 支持 DES 的密钥长度为 64 位。
- 3DES 支持 3 个密钥的方式，也支持 2 个密钥的方式。



- AES 支持 ECB、CBC、1/8/128-CFB、OFB 和 CTR 几种工作模式，工作模式符合 NIST special800-38a 标准。
- DES/3DES 支持 ECB、CBC、1/8/64-CFB、1/8/64-OFB 几种工作模式，工作模式符合 FIPS-81 标准。
- ECB、CBC、CFB、OFB 工作模式下，支持一次实现多个分组的加解密运算，也支持一次实现单个分组的加解密运算。
- AES 的 CTR 工作模式下，只支持一次实现单个分组的加解密运算。
- 提供中断状态查询、中断屏蔽和中断清除功能。
- 提供强行中止模块运行功能。
- 提供对输入数据（包括分组输入、向量输入、密钥）和结果输出数据（包括分组输出、向量输出）的字节序调整功能。

3.6.3 功能描述

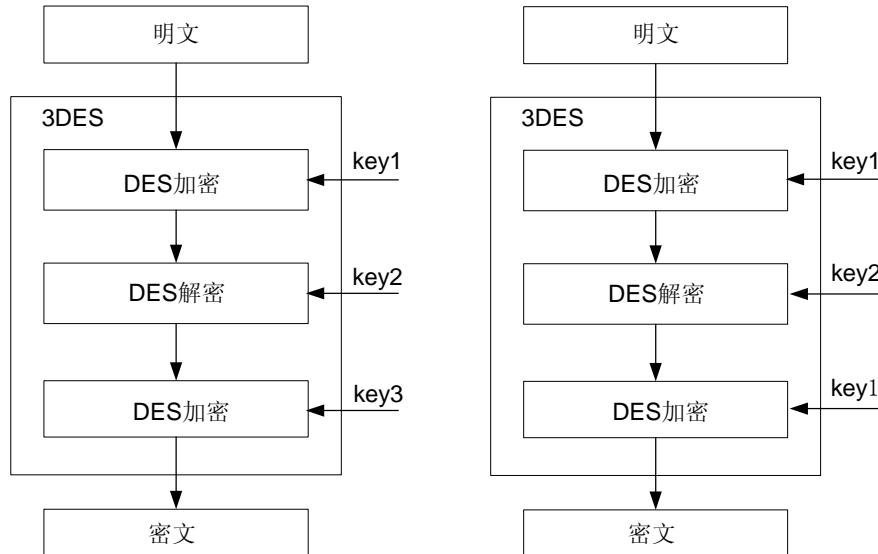
DES/3DES、AES 算法支持的几种工作模式分别符合 FIPS-81 标准和 NIST special800-38a 标准，对于 DES/3DES 和 AES 算法，ECB、CBC 和 CFB 工作模式是相同，OFB 和 CTR（只有 AES 算法中包括）工作模式略有区别。

3.6.3.1 3DES 算法

3DES 支持 3 个密钥和 2 个密钥的运算，2 个密钥的运算可以看作 3 个密钥的一种简化情况，在 2 个密钥的操作中的第三个密钥（key3）都使用第一个密钥（key1）代替。

3 个密钥和 2 个密钥的 3DES 加密运算过程如图 3-11 所示。

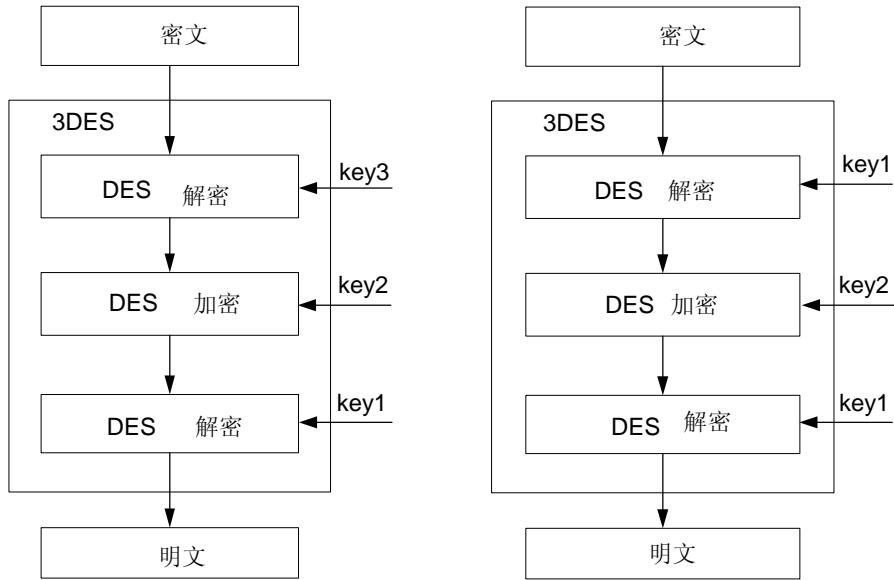
图3-11 3 个密钥和 2 个密钥的 3DES 加密操作



3 个密钥和 2 个密钥的 3DES 解密运算过程如图 3-12 所示。



图3-12 3个密钥和2个密钥的3DES解密操作



3.6.3.2 ECB 模式

ECB (Electronic CodeBook) 模式中, 加、解密算法是直接应用到各个分组数据, 而且各个分组的运算均独立。这个特点使得明文的加密操作和密文的解密操作可以并行进行。AES/DES 和 3DES 的电子密码本 (ECB) 模式分别如图 3-13 和图 3-14 所示。

图3-13 AES/DES 的电子密码本 (ECB) 模式

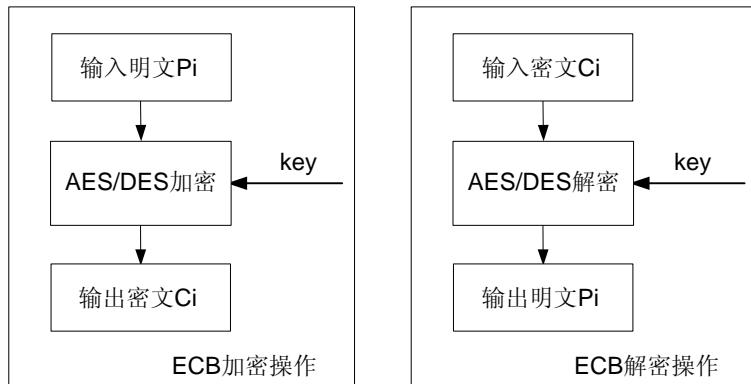
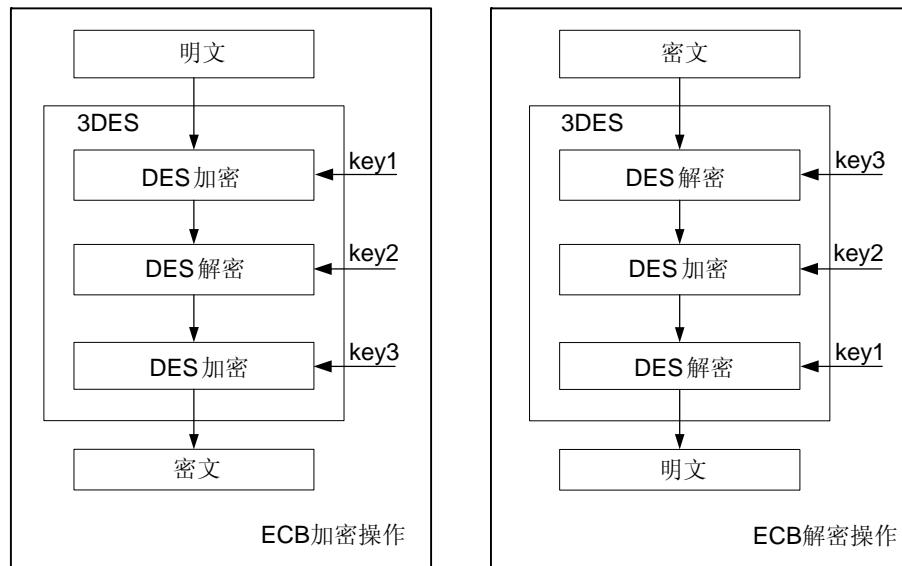




图3-14 3DES 的电子密码本 (ECB) 模式



3.6.3.3 CBC 模式

CBC (Cipher Block Chaining) 模式下, 加密的输入明文分组需要先与输入向量 IV (Initialization Vector) 进行异或操作, 才进入加密操作, 而每个明文分组的加密处理都与上一个明文分组处理的结果 (即密文) 相关, 因此 CBC 模式下的加密操作是不能进行并行处理的。但是解密操作不依赖于上一个分组的明文输出, 是可以进行并行处理的。



图3-15 AES/DES 的密码分组链接 (CBC) 模式

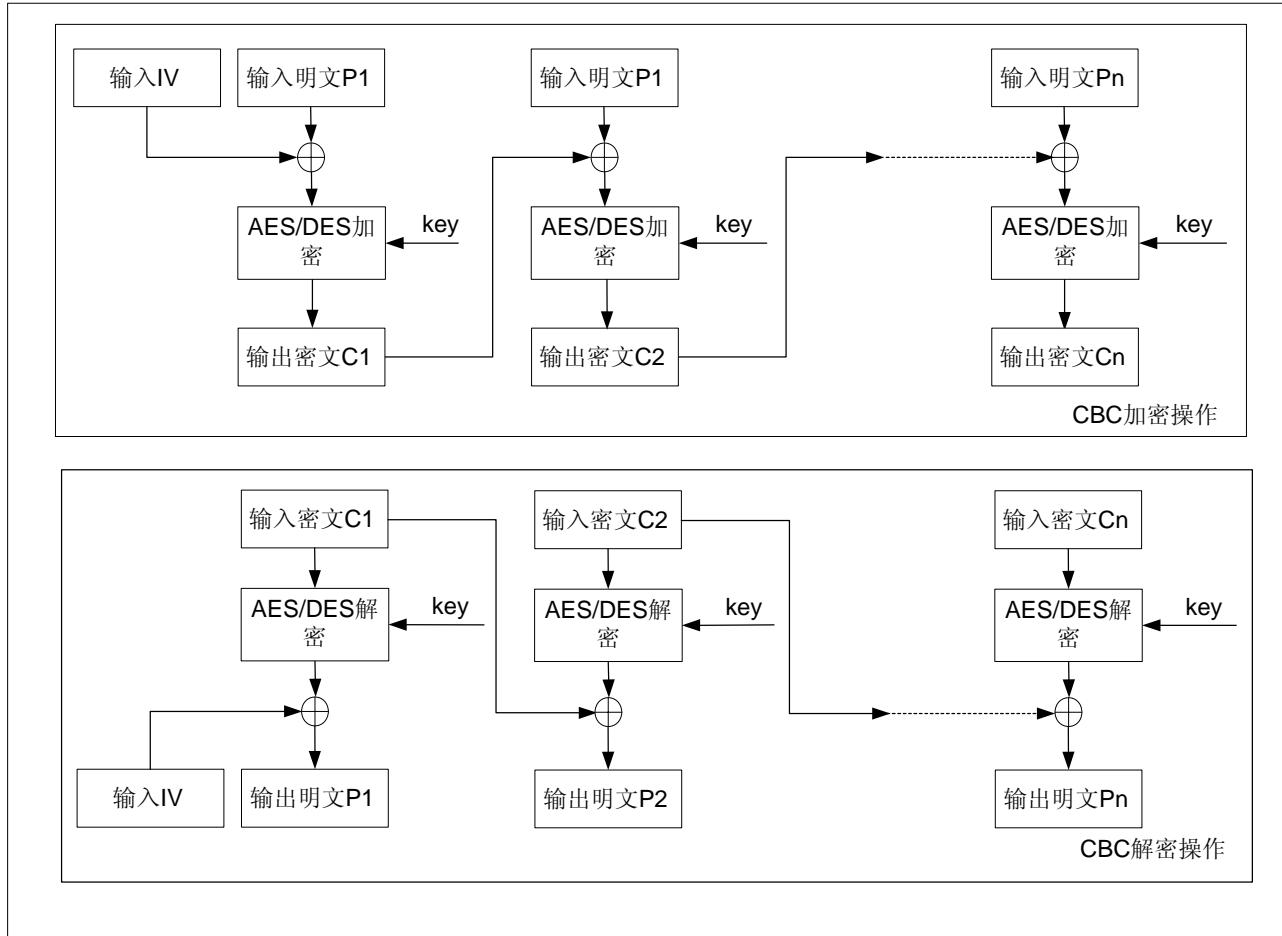
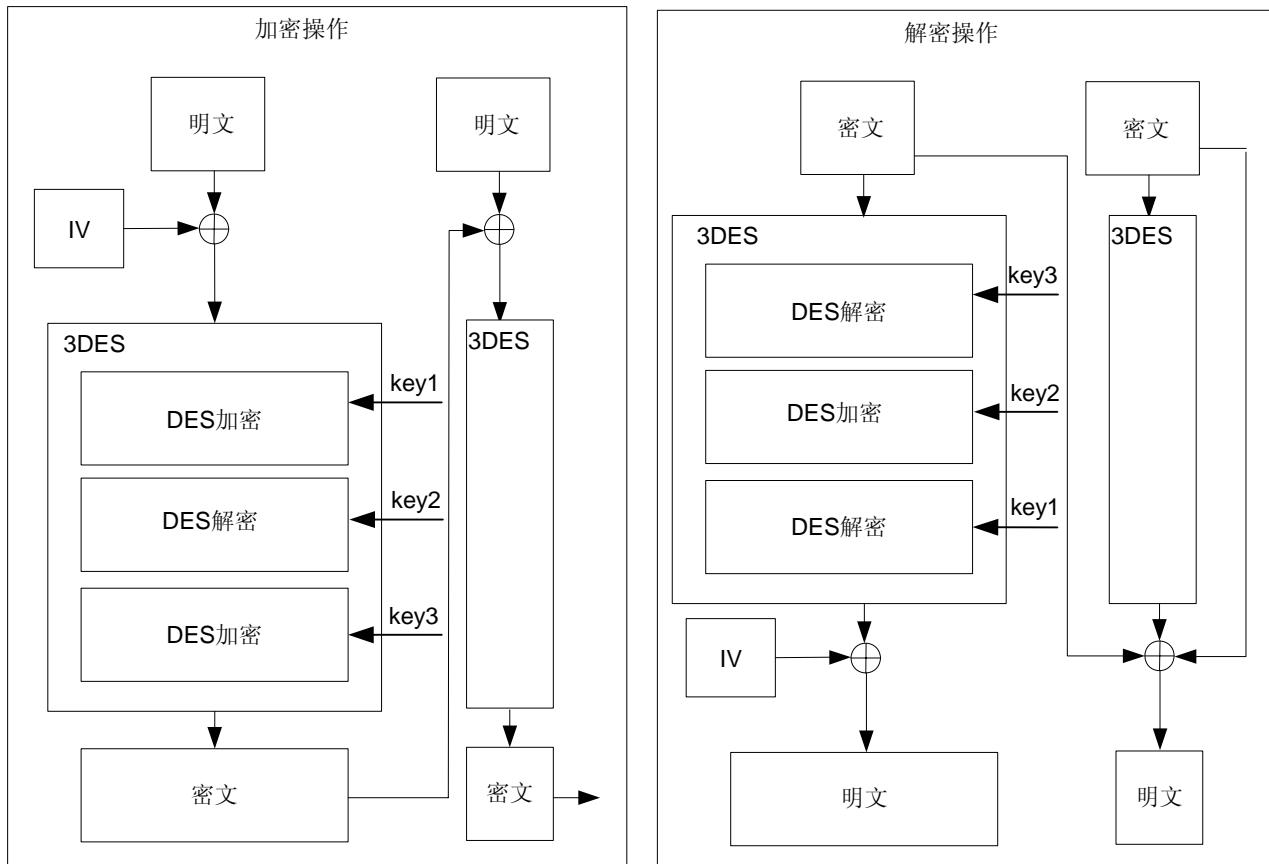


图3-16 3DES 的密码分组链接 (CBC) 模式



3.6.3.4 CFB 模式

CFB (Cipher FeedBack) 模式是将分组密码转换成流密码的一种工作模式，可以通过选择 CFB 的操作位数来实现。移位操作的位数用 s 位表示，关于 s 位存在以下 2 种情况：

- 对于 DES/3DES，s 位可以是 1 位、8 位或 64 位。
- 对于 AES，s 位可以是 1 位、8 位或 128 位。

AES/DES 的 s 位密码反馈 (CFB) 模式和 3DES 的 s 位密码反馈 (CFB) 模式分别如图 3-17 和图 3-18 所示。



图3-17 AES/DES 的 s 位密码反馈 (CFB) 模式

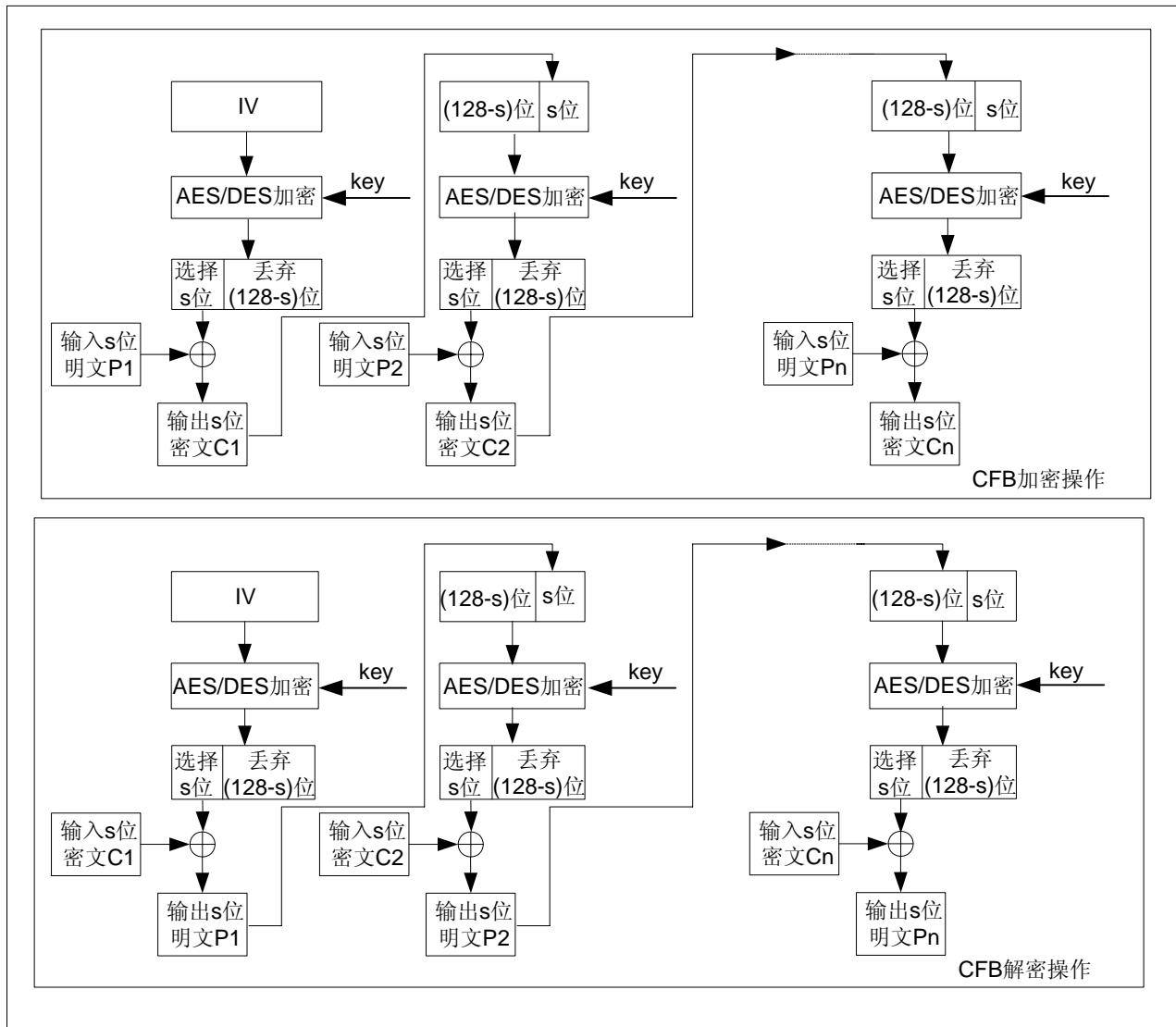
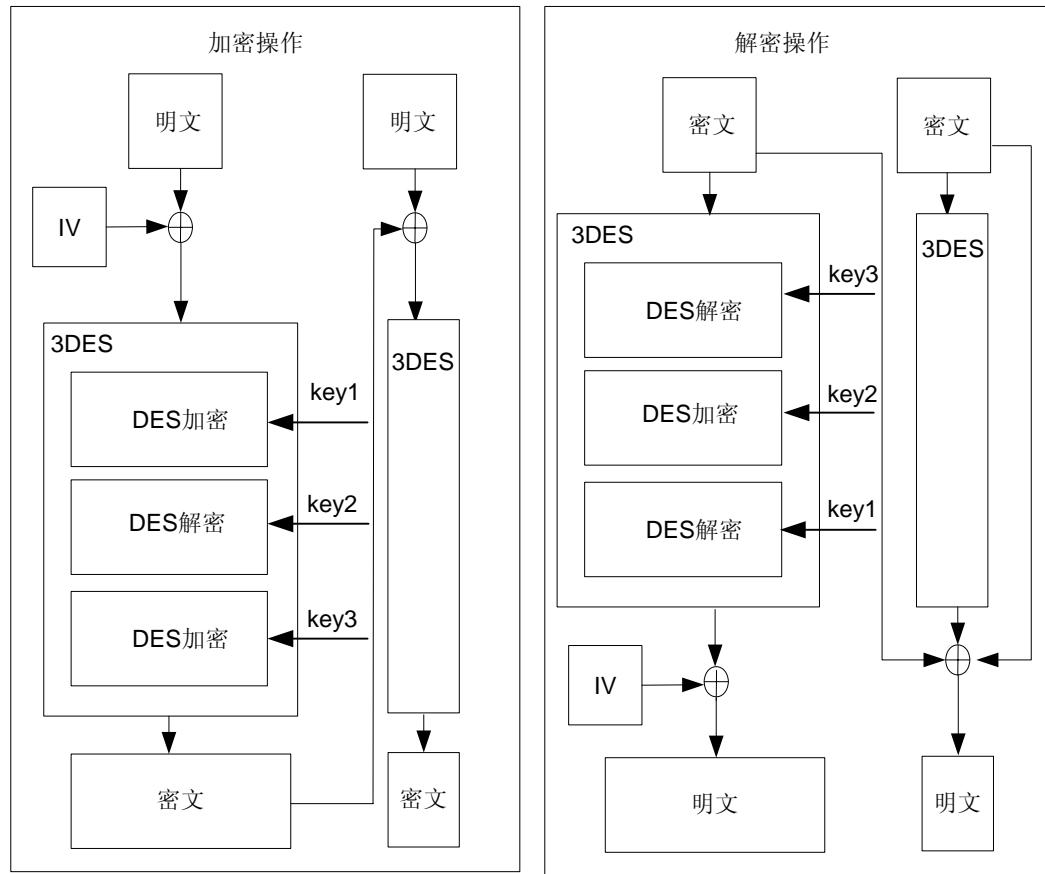


图3-18 3DES 的 s 位密码反馈 (CFB) 模式



3.6.3.5 OFB 模式

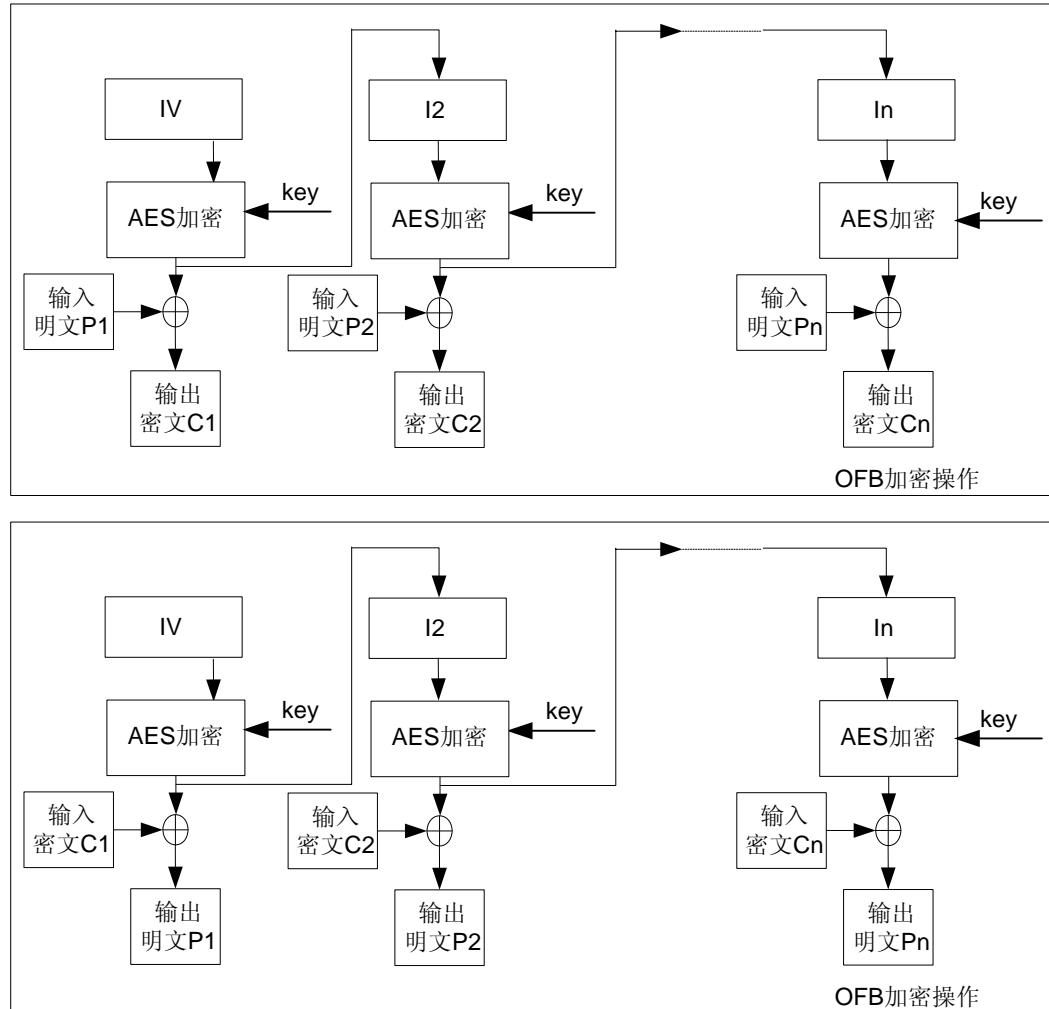
OFB (Output FeedBack) 模式下, 将 IV 直接作为加密操作的输入, 因此对同一个密钥的操作情况下, 应该使用不相同的 IV, 避免降低操作的安全性。关于 s 位, 存在以下 2 种情况:

- 对于 DES/3DES, s 位可以是 1 位、8 位或 64 位。
- 对于 AES, s 位只能是 128 位。



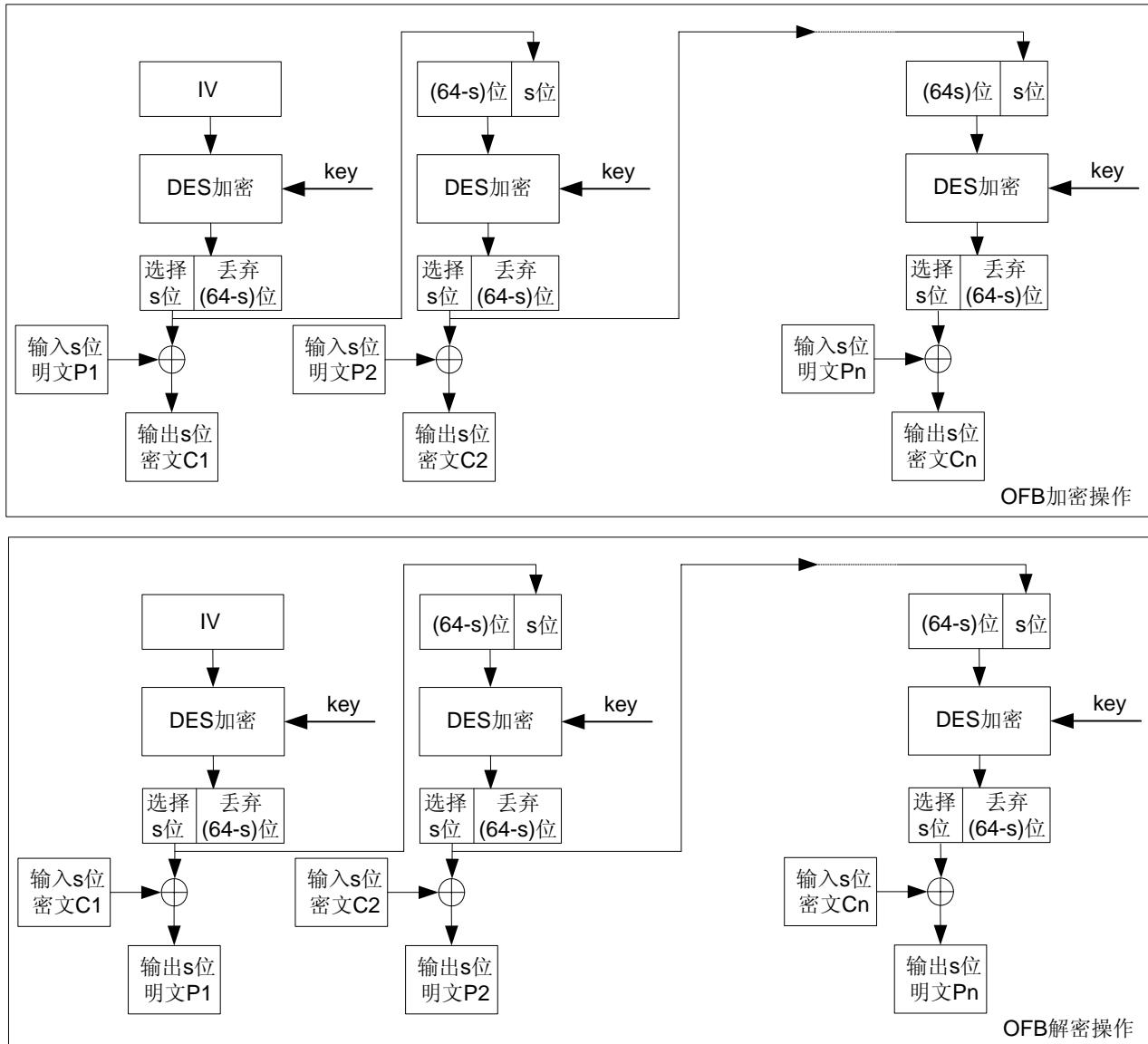
AES 的输出反馈 (OFB) 模式如图 3-19 所示。

图3-19 AES 的输出反馈模式



DES 的 s 位输出反馈模式如图 3-20 所示。

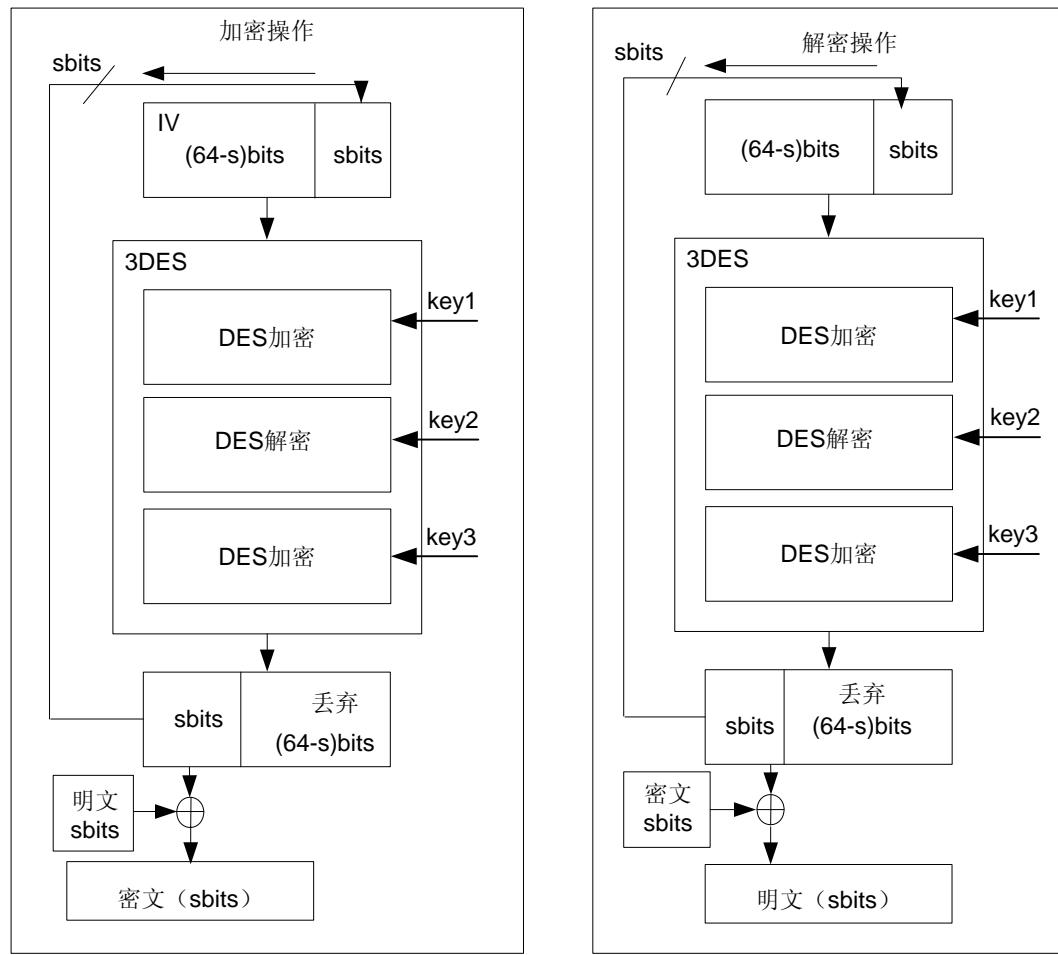
图3-20 DES 的 s 位输出反馈模式



3DES 的 s 位输出反馈模式如图 3-21 所示。



图3-21 3DES 的 s 位输出反馈模式



3.6.3.6 CTR 模式

CTR (Counter) 模式下, 向 AES 加密或解密处理模块输入不同的数据来保证数据处理的安全性, 这种数据可以是计数的值。因此, 计数值 CTR_n 的选取也决定了这种方式应用的安全性。

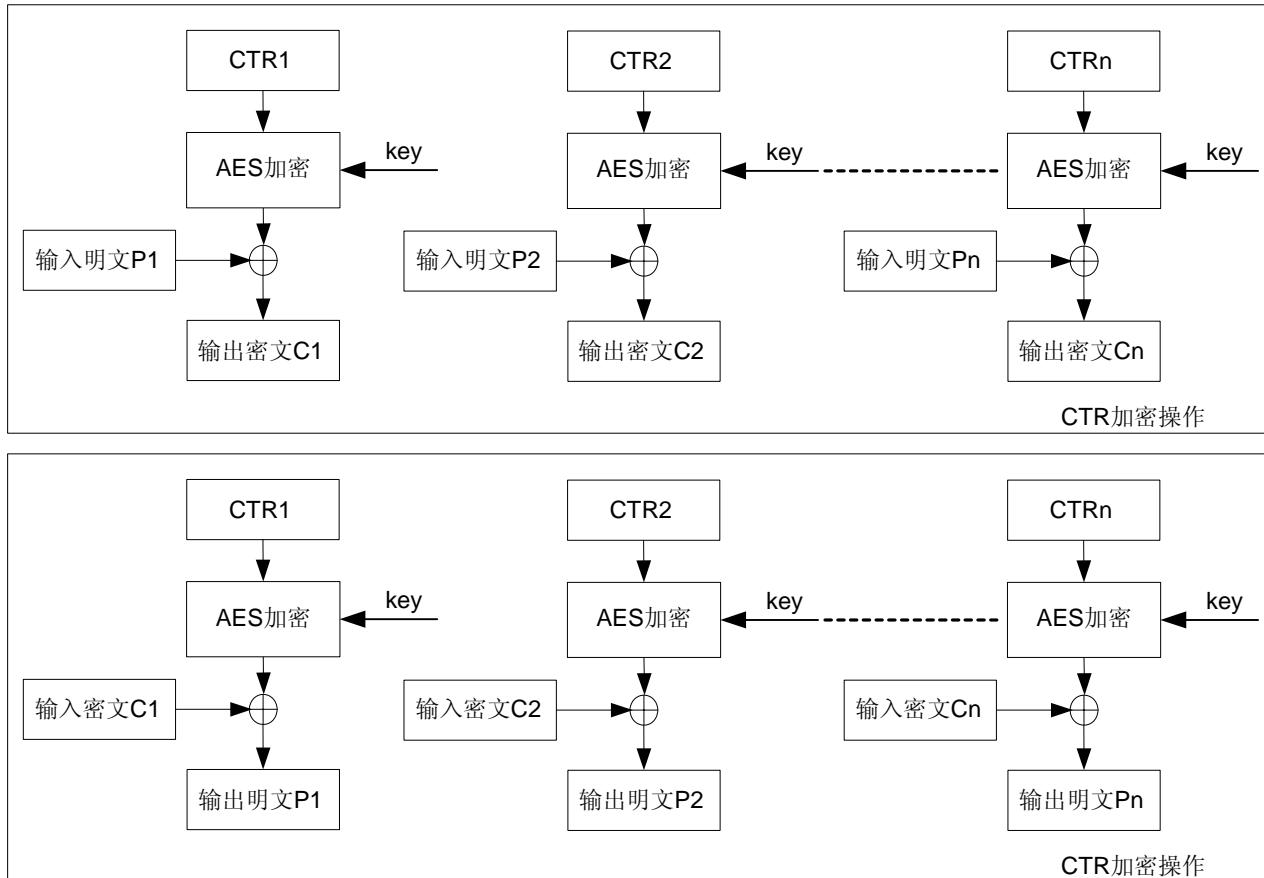


说明

CTR_n 一般采用累加计数的方式获取。

AES 的 CTR 模式如图 3-22 所示。

图3-22 AES 的 CTR 模式



3.6.4 工作方式

3.6.4.1 CIPHER 的单分组操作流程

CIPHER 的单分组操作流程如下：

- 步骤 1 读取 **CIPHER_BUSY** 状态寄存器，如果状态寄存器是 0，则执行步骤 2，否则继续读取该寄存器。
- 步骤 2 配置控制寄存器 **CIPHER_CTRL**。
- 步骤 3 配置寄存器 **CIPHER_DIN**、**CIPHER_IVIN** 和 **CIPHER_KEY**。
- 步骤 4 配置启动寄存器 **CIPHER_ST** 为 1。
- 步骤 5 两种方式等待操作结束：
 1. 中断方式：如果有中断，则执行步骤 6，否则继续等待中断。
 2. 查询方式：读取 **CIPHER_BUSY** 状态寄存器，如果状态寄存器是 0，则执行步骤 6，否则继续读取该寄存器。
- 步骤 6 读取结果寄存器 **CIPHER_DOUT**、**CIPHER_IVOUT**。



----结束

所有的工作模式都可以选择进行单分组操作流程。

如果需要使用单分组操作来实现一段数据的加、解密，则需要将上述中的流程执行多次。为了提高执行效率，需要注意以下几个方面：

- 对于同一段数据，如果操作的密钥 CIPHER_KEY 相同，则只有第一个分组需要对 CIPHER_KEY 寄存器进行配置。
- 对于 CBC、CFB 和 OFB 操作模式，执行完一个分组操作之后，默认操作会自动将本次操作中的 CIPHER_IVOUT 的值作为下次的 CIPHER_IVIN，因此，启动下次操作有 2 种方法：
 - 配置 CIPHER_CTRL[ivin_sel] 为 0，再配置 CIPHER_DIN 寄存器。
 - 读出本次的 CIPHER_IVOUT 中的值，配置到 CIPHER_IVIN 寄存器中，并配置 CIPHER_CTRL[ivin_sel] 为 1，再配置 CIPHER_CTRL 寄存器。
- 对于 CBC、CFB 和 OFB 操作模式，如果本次操作的 CIPHER_IVIN 寄存器中的值与上个分组的 CIPHER_IVOUT 寄存器中的值没有关系，则必须重新配置 CIPHER_IVIN 寄存器，并将 CIPHER_CTRL[ivin_sel] 置为 1。

3.6.4.2 CIPHER 的多分组操作流程

CIPHER 的多分组操作流程如下：

步骤 1 读取 CIPHER_BUSY 状态寄存器，如果状态寄存器是 0，则执行步骤 2，否则继续读取该寄存器。

步骤 2 配置控制寄存器 CIPHER_CTRL。

步骤 3 配置数据存放的地址、数据量寄存器：SRC_START_ADDR、DEST_START_ADDR 和 MEM_LENGTH。

步骤 4 配置寄存器 CIPHER_IVIN 和 CIPHER_KEY。

步骤 5 配置启动寄存器 CIPHER_ST 为 1。

步骤 6 两种方式等待操作结束：

1. 中断方式，如果有中断，则执行步骤 7，否则继续等待中断。
2. 查询方式，读取 CIPHER_BUSY 状态寄存器，如果状态寄存器是 0，则执行步骤 7，否则继续读取该寄存器。

步骤 7 读取结果寄存器 CIPHER_IVOUT。

----结束

除了 AES CTR 工作模式外，其它的工作模式都可以使用多分组操作流程。

多分组操作方式适用于进行大数据量的加解密运算的应用环境，这种操作方式减少了 CPU 在每次分组运算启动和结束时对寄存器的读写操作。CPU 配置完必需的控制和数据寄存器，再配置 CIPHER_ST 寄存器之后，硬件自动即可实现所有的运算和操作，直到数据运算完成。这种方式可释放 CPU 资源。



多分组操作方式一次只能对一个连续地址空间的数据（输入和结果数据存放的地址空间可以是不相同的）进行加解密操作，如果需要将多个不同地址空间的数据使用同一个密钥进行加密或解密操作，则需要注意以下的操作要求：

- 不需要配置 **CIPHER_KEY** 寄存器。
- 需要重新配置源和目的地址，数据长度等信息。
- 对于 CBC、CFB 和 OFB 操作模式，执行完一个分组操作之后，默认操作会自动将本次操作中 **cipher_ivout** 的值作为下次的 **cipher_ivin**，因此，启动下次操作有 2 种方法：
 - 配置 **CIPHER_CTRL[10]** 为 0，再配置 **CIPHER_DIN** 寄存器。
 - 读出本次的 **CIPHER_IVOUT** 中的值，配置到 **CIPHER_IVIN** 寄存器中，并配置 **CIPHER_CTRL[10]** 为 1，再配置 **CIPHER_DIN** 寄存器。

3.6.4.3 时钟门控

当不需要进行加密操作，且状态寄存器 **CIPHER_BUSY[0]**=0 时，可以关断 CIPHER 模块时钟以降低功耗。

- 可通过向 **SC_PERDIS[2]** 写 1，关闭 CIPHER 模块时钟。
- 当需要启动 CIPHER 模块，向 **SC_PEREN[2]** 写 1，使能 CIPHER 模块时钟。

3.6.4.4 软复位

当判断出某单元出现问题时，可以通过软复位恢复单元的错误。

- 向 **SC_PERCTRL0[2]** 写 1，软复位 CIPHER 模块。
- 向 **SC_PERCTRL0[2]** 写 0，撤消对 CIPHER 模块的软复位，进行正常操作。

3.6.5 寄存器概览

表3-21 CIPHER 寄存器概览（基址是 0x9004_0000）

偏移地址	名称	描述	页码
0x000、0x004、 0x008、0x00C	CIPHER_DIN	CIPHER 模块的 128 位分组输入寄存器	3-63
0x010、0x014、 0x018、0x01C	CIPHER_IVIN	CIPHER 模块的向量分组的输入寄存器（ECB 工作模式下无需配置）	3-64
0x020、0x024、 0x028、0x02C 0x030、0x034、 0x038、0x03C	CIPHER_KEY	CIPHER 模块的密钥输入寄存器	3-65
0x040、0x044、 0x048、0x04C	CIPHER_DOUT	CIPHER 模块 128 位分组输出寄存器	3-66



偏移地址	名称	描述	页码
0x050、0x054、0x058、0x05C	CIPHER_IVOUT	CIPHER 模块操作完成之后的向量输出寄存器 (ECB、CTR 工作模式下无需关注)	3-67
0x060	CIPHER_CTRL	CIPHER 模块控制寄存器	3-68
0x064	INT_CIPHER	CIPHER 模块屏蔽后中断寄存器	3-69
0x068	CIPHER_BUSY	CIPHER 模块运算状态指示寄存器	3-69
0x06C	CIPHER_ST	CIPHER 运算启动/停止控制信号寄存器	3-69
0x070	SRC_START_A DDR	待处理分组数据存储在片外 memory 的起始地址寄存器	3-69
0x074	MEM_LENGTH	待处理分组数据的长度 (以 32 位数据宽度衡量的长度) 寄存器	3-69
0x078	DEST_START_ ADDR	运算结果分组存储在片外 memory 的起始地址寄存器	3-69
0x07C	INT_MASK	CIPHER 模块中断屏蔽寄存器	3-69
0x080	INT_CIPHER_S TATUS	CIPHER 模块中断状态寄存器	3-69

3.6.6 寄存器描述

CIPHER_DIN

CIPHER_DIN 为 CIPHER 模块的 128 位分组输入寄存器。

如果选择进行单分组的处理, 即 [CIPHER_CTRL\[11\]=0](#) 时, 需要配置该寄存器:

- 如果选择进行 AES 运算 ([CIPHER_CTRL\[9:8\]=10](#))
 - 如果选择 CFB 操作模式 ([CIPHER_CTRL\[3:1\]=010](#))
如果选择 1-CFB 操作 ([CIPHER_CTRL\[5:4\]=10](#)), 低 1 位有效, 即 [CIPHER_DOUT\[0\]](#) 为有效数据。
如果选择 8-CFB 操作 ([CIPHER_CTRL\[5:4\]=01](#)), 低 8 位有效, 即 [CIPHER_DOUT\[7:0\]](#) 为有效数据。
如果选择 128-CFB 操作 ([CIPHER_CTRL\[5:4\]=00、11](#)), 128 位数据均有效。
- 如果选择其他操作模式, 128 位数据均有效。
- 如果选择进行 DES 或 3DES 运算 ([CIPHER_CTRL\[9:8\]=00、01 或 11](#))
 - 如果选择 CFB 或 OFB 操作模式 ([CIPHER_CTRL\[3:1\]=010、011](#))
如果选择 1-CFB/1-OFB 操作 ([CIPHER_CTRL\[5:4\]=10](#)), 低 1 位有效, 即 [CIPHER_DOUT\[0\]](#) 为有效数据。



如果选择 8-CFB/8-OFB 操作 (**CIPHER_CTRL[5:4]**=01)，低 8 位有效，即 **CIPHER_DOUT[7:0]** 为有效数据。

如果选择 64-CFB/64-OFB 操作 (**CIPHER_CTRL**[5:4]=00、11)，低 64 位数有效，即 **CIPHER_DOUT**[63:0] 为有效数据。

- 如果选择其他操作模式，低 64 位数有效，即 `CIPHER_DOUT[63:0]` 为有效数据。

如果选择进行多分组的处理，即 `CIPHER_CTRL[11]=1`，不需要配置该寄存器。

Offset Address		Register Name	Total Reset Value
0x000~0x00C		CIPHER_DIN	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	cipher_din		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	cipher_din	<p>CIPHER 模块的 128 位分组输入，每个地址对应一个 32 位宽的数据。</p> <p>CIPHER_DIN[31:0]: 0x00 地址。</p> <p>CIPHER_DIN[63:32]: 0x04 地址。</p> <p>CIPHER_DIN[95:64]: 0x08 地址。</p> <p>CIPHER_DIN[127:96]: 0x0C 地址。</p>

CIPHER_IVIN

CIPHER IVIN 为 CIPHER 模块的向量分组的输入寄存器。

配置该寄存器时需要注意：

- 如果选择进行单分组的处理 (**CIPHER_CTRL[11]=0**), 且执行的不是 ECB 模式 (**CIPHER_CTRL[3:1]=001、010、011 或 100**) 时:
 - 如果选择不需要进行输入向量配置 (**CIPHER_CTRL[10]=0**), 则不需要进行该寄存器的配置。
 - 如果选择需要进行输入向量配置 (**CIPHER_CTRL[10]=1**), 则需要进行该寄存器的配置。如果此时选择进行 AES 运算 (**CIPHER_CTRL[9:8]=10**), 128 位数据均有效; 如果选择进行 DES 或 3DES 运算 (**CIPHER_CTRL[9:8]=00、01 或 11**), 低 64 位数据有效, 即 **CIPHER_IVIN[63:0]** 为有效数据。
 - 如果选择进行多分组的处理 (**CIPHER_CTRL[11]=1**), 且执行的是 CBC 或 CFB 或 OFB 模式 (**CIPHER_CTRL[3:1]=001、010 或 011**) 时:
 - 如果选择不需要进行输入向量配置, 即 **CIPHER_CTRL[10]=0**, 则不需要进行该寄存器的配置, 第一个分组运算的输入向量与该寄存器中的值无关。
 - 如果选择需要进行输入向量配置, 即 **CIPHER_CTRL[10]=1**, 则需要进行该寄存器的配置, 第一个分组运算的输入向量从该寄存器获取。



CIPHER_KEY

CIPHER_KEY 为 CIPHER 模块的密钥输入寄存器。

配置本寄存器时需要注意：

- 选择 DES 运算 (**CIPHER_CTRL**[9:8]=00 或 11) 时, 低 64 位数据有效, 即 **CIPHER_KEY**[63:0] 为有效数据。
 - 选择 3DES 运算 (**CIPHER_CTRL**[9:8]=01), 且选择 3 个密钥运算 (**CIPHER_CTRL**[7:6]=00、01 或 10) 时, 低 192 位数据有效, 此时:
 - **CIPHER_KEY**[63:0] 表示第一个密钥。
 - **CIPHER_KEY**[127:64] 表示第二个密钥。
 - **CIPHER_KEY**[191:128] 表示第三个密钥。
 - 选择 3DES 运算 (**CIPHER_CTRL**[9:8]=01), 且选择 2 个密钥运算 (即 **CIPHER_CTRL**[7:6]=11) 时, 低 128 位数据有效, 此时:
 - **CIPHER_KEY**[63:0] 表示第一个密钥。
 - **CIPHER_KEY**[127:64] 表示第二个密钥。
 - 选择 AES 运算 (**CIPHER_CTRL**[9:8]=10) 时:
 - 如果选择 128 位密钥操作 (**CIPHER_CTRL**[7:6]=00 或 11), 低 128 位数据有效, 即 **CIPHER_KEY**[127:0] 为有效数据。
 - 如果选择 192 位密钥操作 (**CIPHER_CTRL**[7:6]=01), 低 192 位数据有效, 即 **CIPHER_KEY**[191:0] 为有效数据。
 - 如果选择 256 位密钥操作 (**CIPHER_CTRL**[7:6]=10), 256 位数据均有效。



CIPHER DOUT

CIPHER DOUT 为 CIPHER 模块 128 位分组输出寄存器。

读取本寄存器时需要注意：

- 如果选择进行单分组的处理（**CIPHER_CTRL[11]**=0），从该寄存器中读取的数据是单分组的运算的结果数据。
 - 如果选择进行 AES 运算（**CIPHER_CTRL[9:8]**=10）

如果选择 8-CFB 操作，即 **CIPHER_CTRL[5:1]**=0b01010，低 8 位有效，即 **CIPHER_DOUT[7:0]** 为有效数据；如果选择 128-CFB 操作，即 **CIPHER_CTRL[5:1]**=0b00010 或 0b11010，128 位数据均有效；其它情况下 128 位数据均有效。
 - 如果选择进行 DES 或 3DES 运算（**CIPHER_CTRL[9:8]**=00 或 01 或 11）

如果选择 1-CFB/1-OFB 操作，即 **CIPHER_CTRL[5:1]**=0b10010 或 0b10011，低 1 位有效，即 **CIPHER_DOUT[0]** 为有效数据；如果选择 8-CFB/8-OFB 操作，即 **CIPHER_CTRL[5:1]**=0b01010 或 0b01011，低 8 位有效，即 **CIPHER_DOUT[7:0]** 为有效数据；如果选择 64-CFB/64-OFB 操作，即 **CIPHER_CTRL[5:1]**=0b00010、0b00011、0b11010 或 0b11011，低 64 位数有效，即 **CIPHER_DOUT[63:0]** 为有效数据；其它情况下低 64 位数有效，即 **CIPHER_DOUT[63:0]** 为有效数据。
 - 如果选择进行多分组的处理，即 **CIPHER_CTRL[10]**=1，从该寄存器中读取的数据是最后一个分组运算的结果数据。



CIPHER_IVOUT

CIPHER_IVOUT 为 CIPHER 操作完成之后的向量输出寄存器。

读取本寄存器时需要注意：

- 如果执行的是 ECB 或 CTR 工作模式 (**CIPHER_CTRL**[3:1]=000、100、101、110 或 111)，不需要关注此寄存器。
 - 如果选择进行单分组的处理 (**CIPHER_CTRL**[11]=0)，该寄存器中的数据是该分组的向量结果输出，可以作为同一数据包的下一个分组运算的向量输入。
 - 如果选择进行 AES 运算 (**CIPHER_CTRL**[9:8]=10)，128 位数据均有效。
 - 如果选择进行 DES 或 3DES 运算 (**CIPHER_CTRL**[9:8]=00、01 或 11)，低 64 位数据有效，即 **CIPHER_IVOUT**[63:0] 为有效数据，
 - 如果选择进行多分组的处理，即 **CIPHER_CTRL**[10]=1，该寄存器中读取的数据是最后一个分组运算的向量结果输出。
 - 如果选择进行 AES 运算 (**CIPHER_CTRL**[9:8]=10)，128 位数据均有效。
 - 如果选择进行 DES 或 3DES 运算 (**CIPHER_CTRL**[9:8]=00、01 或 11)，低 64 位数据有效，即 **CIPHER_IVOUT**[63:0] 为有效数据。



CIPHER_CTRL

CIPHER_CTRL 为控制 CIPHER 操作的寄存器。

配置本寄存器时需要注意：

- 在进行模块的其它寄存器配置之前，必须先配置该寄存器。
 - AES 的 CTR 模式不支持多分组工作方式，即在选择 AES 的 CTR 工作模式时，不允许配置 CIPHER_CTRL[cipher_mode] 为 1。
 - AES 下除了 CFB 模式之外，其它模式不允许将 width 位配置为 01 或 10。
 - DES/3DES 下除了 CFB 和 OFB 模式之外，其它模式不允许将 width 位配置为 01 或 10。
 - 对于 CIPHER_CTRL[byte_seq_reg] 和 CIPHER_CTRL[byte_seq_ram] 位，分别是针对寄存器配置/结果寄存器读取操作和对结果寄存器中的数据进行字节序调整。如果送给该模块的数据（以 DES 操作的数据为例）是字符流的形式，则需要启动字节调整。

例如文本“7654321”对应0x3736_3534_3332_3120，以字符的方式接收数据之后，存放在存储器中的顺序如下：

- 0x0 地址存放的数据是: 0x3435_3637
 - 0x4 地址存放的数据是: 0x2031_3233

假设数据是从 0x0 的偏移地址开始存放的，这种情况下，则需要把 byte_seq_ram 和 byte_seq_reg 位均配置为 1。



Offset Address		Register Name	Total Reset Value
0x060		CIPHER_CTRL	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved		dest_addr_set cipher_mode ivin_sel alg_sel key_length width mode decrypt
Reset	0 0		
Bits	Access	Name	Description
[31:15]	-	reserved	保留。
[14]	RW	byte_seq_ram	用来控制对 memory 地址空间的数据字节序调整。 0: 不进行字节序调整。 1: 进行字节序调整。
[13]	RW	byte_seq_reg	用来控制对输入数据配置的字节序调整和输出数据寄存器读取的字节序调整。 输入数据寄存器包括: CIPHER_KEY 、 CIPHER_IVIN 、 CIPHER_DIN 。 输出数据寄存器包括: CIPHER_IVOUT 、 CIPHER_DOUT 。 0: 不进行字节序调整。 1: 进行字节序调整。
[12]	RW	dest_addr_set	待处理分组数据和运算结果分组数据存储在片外 memory 的起始地址关系控制。 0: 待处理分组数据和运算结果分组数据存放的起始地址相同, 可以不配置 SRC_START_ADDR 和 DEST_START_ADDR 。 1: 待处理分组数据和运算结果分组数据存放的起始地址不相同, 需要分别配置 SRC_START_ADDR 和 DEST_START_ADDR 。
[11]	RW	cipher_mode	CIPHER 模块工作方式选择控制。 0: 选择进行单分组的操作。 1: 选择进行多分组的操作。
[10]	RW	ivin_sel	CIPHER_IVIN 的输入选择控制。 0: CIPHER_IVIN 不需要进行配置。 1: CIPHER_IVIN 需要配置。
[9:8]	RW	alg_sel	算法类型选择控制。 00: DES 运算。 01: 3DES 运算。

			10: AES 运算。 11: DES 运算。
[7:6]	RW	key_length	密钥长度控制。 AES 算法下: 00: 128 位密钥长度。 01: 192 位密钥长度。 10: 256 位密钥长度。 11: 128 位密钥长度。 DES 算法下: 00: 3 个密钥。 01: 3 个密钥。 10: 3 个密钥。 11: 2 个密钥。
[5:4]	RW	width	位宽控制。 DES/3DES 算法下: 00: 64 位模式。 01: 8 位模式。 10: 1 位模式。 11: 64 位模式。 AES 算法下: 00: 128 位模式。 01: 8 位模式。 10: 1 位模式。 11: 64 位模式。
[3:1]	RW	mode	工作模式控制。 在 AES 算法下: 000: ECB 模式。 001: CBC 模式。 010: CFB 模式。 011: OFB 模式。 100: CTR 模式。 其它: ECB 模式。 在 DES 算法下: 000: ECB 模式。 001: CBC 模式。 010: CFB 模式。 011: OFB 模式。



			其它: ECB 模式。
[0]	RW	decrypt	加解密控制。 0: 加密。 1: 解密。

INT_CIPHER

INT_CIPHER 为屏蔽后中断寄存器，即中断状态寄存器 (INT_CIPHER_STATUS) 经过中断屏蔽寄存器 (INT_MASK) 屏蔽之后的中断寄存器。

	Offset Address		Register Name	Total Reset Value																												
	0x064		INT_CIPHER	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:2]	-	reserved	保留。																													
[1]	RO	int_error	错误中断状态，CIPHER 模块访问总线出错，或使用非 word 方式访问。 0: 无错误。 1: 有错误。																													
[0]	RO	int_done	CIPHER 运算完成中断。 0: 未完成操作。 1: 完成操作。																													

CIPHER_BUSY

CIPHER_BUSY 为运算状态指示寄存器。



CIPHER_ST

CIPHER_ST 为 CIPHER 运算启动/停止控制信号寄存器。

配置本寄存器时需要注意：

- 如果向 **CIPHER_ST** 寄存器地址写 0x0000_0001 之后, CIPHER 模块将启动运算。
 - 如果向 **CIPHER_ST** 寄存器地址写 0x0000_0000 之后, CIPHER 模块将中止运行。
 - 如果选择进行多分组的处理, 即 **CIPHER_CTRL[11]**=1, 如果正在进行总线读写, 则将完成总线读写之后, 中止模块运行; 否则, 硬件在读到该值之后, 将立刻中止模块运行。



SRC_START_ADDR

SRC_START_ADDR 为待处理分组数据存储在片外 memory 的起始地址寄存器。

如果选择进行多个分组的处理, 即 **CIPHER_CTRL[11]=1**, 则需要在启动 CIPHER 模块之前, 配置该寄存器。

	Offset Address	Register Name	Total Reset Value
Bit	0x070	SRC_START_ADDR	0x0000_0000
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name		src_start_addr	
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	src_start_addr	待处理分组数据存储在片外 memory 的起始地址。

MEM_LENGTH

MEM_LENGTH 为待处理分组数据的长度 (以 32 位数据宽度衡量的长度) 寄存器。

配置本寄存器时需要注意:

- 如果选择进行多分组的处理 (**CIPHER_CTRL[11]=1**), 则需要在启动 CIPHER 模块之前, 配置该寄存器。
- mem_length** 表示以字 (32 位) 长度划分的长度。
- 在选择 AES 运算时:
 - 如果选择 ECB、CBC、OFB 或 CTR 模式 (即 **CIPHER_CTRL[3:1]=000**、**001**、**011**、**100**、**101**、**110** 或 **111**), 要求 **mem_length** 的值可以整除 4。
 - 如果选择 128-CFB 模式 (**CIPHER_CTRL[3:1]=010**, 且 **CIPHER_CTRL[5:4]=00** 或 **11**), 要求 **mem_length** 的值可以整除 4。
 - 如果选择执行 8-CFB 模式 (**CIPHER_CTRL[3:1]=010** 且 **CIPHER_CTRL[5:4]=01**) 时, 对 **mem_length** 的值没有要求。对数据填充, 要求将数据填充在高位, 有效数据从低字节开始有效。例如, **mem_length=0x0000_0001**, 表示仅有一个地址数据有效, 该地址的数据用 **data[31:0]** 表示, 如果实际数据中, 只有一个字节有效, 即 **data[7:0]** 是有效的数据, 则其它数据是填充数据, 对应的结果, 也是只有低字节有效; 如果只有两个字节有效, 即 **data[15:0]** 是有效数据, 其它数据是填充数据, 对应的结果, 也是只有低 16 位数据有效; 依此类推。
 - 选择执行 1-CFB 模式, 即 **CIPHER_CTRL[3:1]=010**, 且 **CIPHER_CTRL[5:4]=11** 时, 对 **mem_length** 的值没有要求。对数据填充, 要求将数据填充在高位, 有效数据从低位开始有效。例如, **mem_length=0x0000_0001**, 表示仅有一个地址数据有效, 该地址的数据用 **data[31:0]** 表示, 如果实际数据中, 只有一位有效, 即 **data[0]** 是有效的数据, 其它数据是填充数据, 对应的结果, 也是只有最低位有效; 如果只有两位有效, 即 **data[1:0]** 是有效数据, 其它数据是填充数据, 对应的结果, 也是只有低 2 位数据有效; 依此类推。



- 在选择 DES/3DES 运算时：
 - 选择执行 ECB 或 CBC 模式 (**CIPHER_CTRL**[3:1]=000、001、100、101、110 或 111)，要求 **mem_length** 的值是偶数。
 - 选择执行 64-CFB 或 64-OFB 模式 (**CIPHER_CTRL**[3:1]=010 或 011，且 **CIPHER_CTRL**[5:4]=00 或 11)，要求 **mem_length** 的值是偶数。
 - 选择执行 8-CFB 模式或 8-OFB (**CIPHER_CTRL**[3:1]=010 或 011，且 **CIPHER_CTRL**[5:4]=01)，对 **mem_length** 的值没有要求，但是，对数据填充，要求将数据填充在高位，有效数据从低字节开始有效。例如，**mem_length**=0x0000_0001，表示仅有一个地址数据有效，该地址的数据用 **data**[31:0]表示，如果实际数据中，只有一个字节有效，即 **data**[7:0]是有效的数据，其它数据是填充数据，对应的结果，也是只有低字节有效；如果只有两个字节有效，即 **data**[15:0]是有效数据，其它数据是填充数据，对应的结果，也是只有低 16 位数据有效；依此类推。
 - 选择执行 1-CFB 或 8-OFB 模式，即 **CIPHER_CTRL**[3:1]=010 或 011，且 **CIPHER_CTRL**[5:4]=11，对 **mem_length** 的值没有要求，但是，对数据填充，要求将数据填充在高位，有效数据从低位开始有效。例如，**mem_length**=0x0000_0001，表示仅有一个地址数据有效，该地址的数据用 **data**[31:0]表示，如果实际数据中，只有一位有效，即 **data**[0]是有效的数据，其它数据是填充数据，对应的结果，也是只有最低位有效；如果只有两位有效，即 **data**[1:0]是有效数据，其它数据是填充数据，对应的结果，也是只有低 2 位数据有效；依此类推。

DEST_START_ADDR

DEST_START_ADDR 为运算结果分组存储在片外 memory 的起始地址寄存器。



注意

- 如果待处理分组数据和运算结果分组数据存放的起始地址相同，即 [CIPHER_CTRL\[12\]=0](#) 时，不需要配置该寄存器。
- 如果待处理分组数据和运算结果分组数据存放的起始地址不相同，即 [CIPHER_CTRL\[12\]=1](#) 时，需要配置该寄存器。

	Offset Address	Register Name	Total Reset Value
	0x078	DEST_START_ADDR	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		dest_start_addr	
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	dest_start_addr	运算结果分组存储在片外 memory 的起始地址。

INT_MASK

INT_MASK 为中断屏蔽寄存器。该寄存器用来对中断状态寄存器进行屏蔽控制，决定是否产生中断。

向 INT_MASK 寄存器地址写 0x0000_0003 之后，CIPHER 模块将屏蔽 [INT_CIPHER_STATUS](#) 寄存器的状态，即不产生中断，即 INT_CIPHER[1:0] 为 00。

向 INT_MASK 寄存器地址写 0x0000_0001 之后，CIPHER 模块将屏蔽 [INT_CIPHER_STATUS](#) 寄存器中的 int_done_status 状态，该状态不触发中断，即 INT_CIPHER[0] 为 0。

向 [INT_CIPHER_STATUS](#) 寄存器地址写 0x0000_0002 之后，CIPHER 模块将屏蔽 [INT_CIPHER_STATUS](#) 寄存器中的 int_error_status 状态，该状态不触发中断，即 INT_CIPHER[1] 为 0。



Offset Address		Register Name	Total Reset Value																						
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	INT_MASK	0x0000_0000																						
Name	reserved																								int_error_mask
Reset	0 0																								
Bits	Access	Name	Description																						
[31:2]	-	reserved	保留。																						
[1]	RW	int_error_mask	是否产生 int_error 的屏蔽控制。 0: 产生中断。 1: 不产生中断。																						
[0]	RW	int_done_mask	是否产生 int_done 的屏蔽控制。 0: 产生中断。 1: 不产生中断。																						

INT CIPHER STATUS

INT_CIPHER_STATUS 为中断状态寄存器。该寄存器是写清寄存器，向 INT_CIPHER_STATUS 寄存器地址写 0x0000_0003 之后，CIPHER 模块将 INT_CIPHER_STATUS 寄存器清 0。向 INT_CIPHER_STATUS 寄存器地址写 0x0000_0001 之后，CIPHER 模块将清除 INT_DONE_STATUS 中断信号；向 INT_CIPHER_STATUS 寄存器地址写 0x0000_0002 之后，CIPHER 模块将清除 int_error_status 中断信号。



[1]	WC	int_error_status	错误中断状态, CIPHER 模块的 Master 接口访问 AHB 出错, 以及不使用 word 方式访问 slave 接口出错。 0: 无错误。 1: 有错误。
[0]	WC	int_done_status	CIPHER 运算完成中断状态。 0: 未完成操作。 1: 完成操作。

3.7 Timer

3.7.1 概述

Timer 单元提供 2 组 Dual-Timer: Dual-Timer0 和 Dual-Timer1。2 组 Dual-Timer 模块自身并无差异, 除基地址不同以外, 各自在系统中的应用也有所不同。其中:

- Dual-Timer0 包括 Timer0、Timer1, 它们共用同一个基地址和同一中断线。
- Dual-Timer1 包括 Timer2、Timer3, 它们共用同一个基地址和同一中断线。

每组 Dual-Timer 包含两个功能完全相同的 Timer。

3.7.2 特点

Dual-Timer 模块具有以下特点:

- 2 个带可编程 8 位预分频器的 32bit/16bit 减法定时器/计数器。
- 计数时钟可配置, 可选为系统总线时钟或 3MHz 时钟。
- 支持 3 种计数模式: 自由运行模式、周期模式和单次计数模式。
- 有 2 种载入计数初值的方法, 分别通过 **TIMERx_LOAD** 和 **TIMERx_BGLOAD** 寄存器实现。
- 当前的计数值可随时读取。
- 当计数值减到 0 时会产生一个中断。

3.7.3 功能描述

典型应用

Hi3511/Hi3512 中的 Timer 主要是供软件使用。基于这种应用, Hi3511/Hi3512 的 2 组 Dual-Timer 提供不同的计时时钟, Dual-Timer 的典型应用如下:

Dual-Timer0 中的 Timer0 和 Timer1、Dual-Timer1 中的 Timer2 和 Timer3 均供软件使用。



- 系统处于 Normal 模式或 Slow 模式时，计数时钟可选择为总线时钟或 3MHz 时钟。
- 系统处于 Doze 模式时，不可配置使用 3MHz 时钟。
- 系统处于 Sleep 模式时，此组 Dual-Timer 无时钟，停止工作。

功能原理

Timer 基于一个 32bit/16bit（可配置）减法计数器。计数器的值在每个计数时钟的上升沿减 1。当计数值递减到零，Timer 将产生一个中断。

Timer 有以下 3 种计数模式：

- **自由运行模式**
定时器持续计数，当计数值减到 0 时又自动回转到其最大值，并继续计数。当计数长度为 32bit 时，最大值为 0xFFFF_FFFF。当计数长度为 16bit 时，最大值为 0xFFFF。在自由模式下，也可以载入计数值，并立即从载入值减计数，但计到 0 时回转到其最大值。
- **周期模式**
定时器持续计数，当计数值减到 0 时从 **TIMERx_BGLOAD** 寄存器中再次载入初值并继续计数。
- **单次计数模式**
向定时器中载入计数初值。当定时器的计数值减到 0 时就停止计数，直到重新被载入新值且定时器处于使能状态，才再次开始计数。

每个 Timer 具有一个预分频计数器，可将其工作时钟在 Timer 内部再次进行 1 分频、16 分频或 256 分频。进一步提高计数时钟频率的选择灵活性。

对定时器载入计数初值的方法如下：

- 通过配置 **TIMERx_LOAD** 寄存器可对定时器载入计数初值。当定时器处于工作状态时，如果向 **TIMERx_LOAD** 寄存器写入值，会导致定时器立刻从新值开始重新计数。适用于所有计数模式。
- 通过配置 **TIMERx_BGLOAD** 寄存器可以设定周期计数模式的计数周期。写该寄存器不会立刻影响定时器的当前计数，定时器会继续计数直到计数值减到 0。然后载入 **TIMERx_BGLOAD** 寄存器中的新值开始计数。

3.7.4 工作方式

初始化

系统初始化时应对 Timer 进行初始化。初始化 TimerX（X 为 0/1 或 2/3）时应按以下步骤进行配置：

- 步骤 1 配置 **TIMERx_LOAD** 寄存器，为 Timer 载入计数初值。
- 步骤 2 当需要 Timer 工作在周期计数模式下，配置 **TIMERx_BGLOAD** 寄存器，设置 Timer 的计数周期。
- 步骤 3 配置系统控制器的 **SC_CTRL** 寄存器，设置 Timer 的时钟使能信号的参考时钟。



步骤 4 配置寄存器 **TIMERx_CONTROL**，设置 Timer 的计数模式、计数器长度、预分频因子及中断屏蔽，同时启动 Timer 计数。

----结束

中断处理

Timer 主要用于定时产生中断，因此 Timer 的中断处理主要是激活等待定时中断的进程。典型步骤如下：

步骤 1 配置 **TIMERx_INTCLR** 寄存器，清除 Timer 中断。

步骤 2 继续执行中断服务程序。

步骤 3 恢复中断现场，继续执行当前被中断的程序。

----结束

3.7.5 寄存器概览

Timer 模块中的 4 个定时器各自有一组寄存器，这 4 组寄存器除基址和偏移地址不相同外，其他特性都相同。其中：

- Timer0 和 Timer1 共用一个基址：0x101E_2000。
- Timer2 和 Timer3 共用一个基址：0x101E_3000。



说明

TIMERx 中的 “x” 取值为 0、1、2、3。

表3-22 Timer 寄存器概览（基址是 0x101E_2000、0x101E_3000）

Timer0/2 的偏移地址	Timer1/3 的偏移地址	名称	描述	页码
0x000	0x020	TIMERx_LOAD	计数初值寄存器	3-69
0x004	0x024	TIMERx_VALUE	当前计数值寄存器	3-69
0x008	0x028	TIMERx_CONTROL	Timer 控制寄存器	3-69
0x00C	0x02C	TIMERx_INTCLR	中断清除寄存器	3-69
0x010	0x030	TIMERx_RIS	原始中断寄存器	3-69
0x014	0x034	TIMERx_MIS	屏蔽后中断寄存器	3-69
0x018	0x038	TIMERx_BGLOAD	循环模式计数初值寄存器	3-69

3.7.6 寄存器描述

说明

- TIMER0_XXXX 和 TIMER2_XXXX 的偏移地址相同，关于这两个相似的寄存器描述均以 TIMER0_XXXX 为例进行介绍。
- TIMER1_XXXX 和 TIMER3_XXXX 的偏移地址相同，关于这两个相似的寄存器描述均以 TIMER1_XXXX 为例进行介绍。

TIMERx_LOAD

TIMERx_LOAD 为计数初值寄存器，用来配置定时器的计数初值。Timer0~3 各有 1 个计数初值寄存器。

1. TIMER0_LOAD

	Offset Address	Register Name	Total Reset Value
	0x000	TIMER0_LOAD	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	timer0_load		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	timer0_load	Timer0 的计数初值。

2. TIMER1_LOAD

	Offset Address	Register Name	Total Reset Value
	0x020	TIMER1_LOAD	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	timer1_load		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	timer1_load	Timer1 的计数初值。

当定时器处于周期模式且计数值递减到 0 时，将 [TIMERx_LOAD](#) 的值重新载入计数器。当直接写 [TIMERx_LOAD](#) 寄存器时，定时器当前的计数器将在被 TIMCLKENx 使能的下一个 TIMCLK 的上升沿更新为写入值。

说明

- 向 [TIMERx_LOAD](#) 寄存器写入的最小有效值为 1。
- 当向 [TIMERx_LOAD](#) 写 0 时，Dual-Timer 将会立刻产生 1 个中断。

当向 `TIMERx_BGLOAD` 寄存器写入值时, `TIMERx_LOAD` 的值也会被覆盖, 但定时器计数的当前值不会受到影响。

如果在被 `TIMCLKENx` 使能的 `TIMCLK` 的上升沿到来之前，向 `TIMERx_BGLOAD` 寄存器和 `TIMERx_LOAD` 寄存器都写入了值，则在被 `TIMCLKENx` 使能的 `TIMCLK` 的下一个上升沿定时计数器的值首先更新为 `TIMERx_LOAD` 的写入值。此后，每当计数器递减到 0 时，重新载入 `TIMERx_BGLOAD` 与 `TIMERx_LOAD` 中最晚被写入的寄存器的写入值。

在分别对 `TIMERx_BGLOAD` 寄存器和 `TIMERx_LOAD` 寄存器进行了 2 次写入之后，读 `TIMERx_LOAD` 返回的值为 `TIMERx_BGLOAD` 的写入值。

TIMERx_VALUE

TIMERx_VALUE 为当前计数值寄存器，用于给出正在递减的计数器的当前值。Timer0~3 各有 1 个当前计数值寄存器。

当向 `TIMERx_LOAD` 寄存器的写操作发生后, `TIMERx_VALUE` 在 `PCLK` 时钟域立刻反映出计数器的新载入值, 不用等到下一个被 `TIMCLKENx` 使能的 `TIMCLK` 时钟沿到来。



说明

当定时器处于 16bit 模式时，32 位的 **TIMERx_VALUE** 寄存器的高 16 位并未被自动设为 0。若该定时器以前处于 32bit 模式，并且自从进入 16bit 模式后 **TIMERx_LOAD** 从未被写过，则 **TIMERx_VALUE** 寄存器的高 16 位可能具有非零值。

1. TIMER0_VALUE

2. TIMER1 VALUE



TIMERx_CONTROL

TIMERx_CONTROL TIMER 控制寄存器。Timer0~3 各有 1 个控制寄存器。

说明

当选择用周期模式进行计数时，需要将 TIMERx_CONTROL[timermode] 置 1、
TIMERx_CONTROL[oneshot] 置 0。

1. TIMER0_CONTROL



Offset Address		Register Name		Total Reset Value											
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8	7	6	5	4	3	2	1	0	oneshot					
Name	reserved										timeren	intenable	reserved	timerpre	timersize
Reset	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0				
Bits	Access	Name	Description												
[31:8]	-	reserved	保留。												
[7]	RW	timeren	定时器使能。 0: Timer 禁止。 1: Timer 使能。												
[6]	RW	timermode	定时器的计数模式。 0: 自由运行模式。 1: 周期模式。												
[5]	RW	intenable	TIMERx_RIS 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。												
[4]	-	reserved	保留。												
[3:2]	RW	timerpre	该字段用于设置 Timer 的预分频因子。 00: 不经过预分频, 时钟频率除以 1。 01: 4 级预分频, 将 Timer 时钟频率除以 16。 10: 8 级预分频, 将 Timer 时钟频率除以 256。 11: 未定义, 若设为该值, 相当于 8 级预分频, 将 Timer 时钟频率除以 256。												
[1]	RW	timersize	选择 16bit/32bit 计数器操作模式。 0: 16bit 计数器。 1: 32bit 计数器。												
[0]	RW	oneshot	选择计数模式为单次计数模式还是周期计数模式。 0: 周期计数模式或自由运行模式。 1: 单次计数模式。												



2. TIMER1_CONTROL

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																													oneshot		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		
Bits	Access	Name	Description																													
[31:8]	-	reserved	保留。																													
[7]	RW	timeren	定时器使能。 0: Timer 禁止。 1: Timer 使能。																													
[6]	RW	timermode	定时器的计数模式。 0: 自由运行模式。 1: 周期模式。																													
[5]	RW	intenable	TIMERx_RIS 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																													
[4]	-	reserved	保留。																													
[3:2]	RW	timerpre	该字段用于设置 Timer 的预分频因子。 00: 不经过预分频, 时钟频率除以 1。 01: 4 级预分频, 将 Timer 时钟频率除以 16。 10: 8 级预分频, 将 Timer 时钟频率除以 256。 11: 未定义, 若设为该值, 相当于预分频因子等于 10。																													
[1]	RW	timersize	选择 16bit/32bit 计数器操作模式。 0: 16bit 计数器。 1: 32bit 计数器。																													
[0]	RW	oneshot	选择计数模式为单次计数模式还是周期计数模式。 0: 周期计数模式。 1: 单次计数模式。																													



TIMERx_INTCLR

TIMERx_INTCLR 为中断清除寄存器，对该寄存器的任何写操作都会清除相应计数器的中断状态。Timer0~3 各有 1 个中断清除寄存器。

1. TIMER0_INTCLR

注：本寄存器是只写寄存器，写进去任意值，都会引起Timer清中断，内部并不记忆写入的值，无复位值。

2. TIMER1 INTCLR

注：本寄存器是只写寄存器，写进去任意值，都会引起 Timer 清中断，内部并不记忆写入的值，无复位值。

TIMERx RIS

TIMERx_RIS 为原始中断（屏蔽前中断）寄存器。Timer0~3 各有 1 个原始中断寄存器。



Offset Address		Register Name	Total Reset Value
0x010		TIMER0_RIS	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	timer0ris
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。写入无效，读时返回 0。
[0]	RO	timer0ris	Timer0 的原始中断状态。 0: 未产生中断。 1: 已产生中断。

2. TIMER1 RIS

TIMERx_MIS

TIMERx_MIS 为屏蔽后中断寄存器。Timer0~3 各有 1 个屏蔽后中断寄存器。

1. TIMER0_MIS



2. TIMER1_MIS

Offset Address		Register Name	Total Reset Value
0x034		TIMER1_MIS	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved		timer1mis
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。
[0]	RO	timer1mis	屏蔽后的 Timer1 的中断状态。 0: 中断无效。 1: 中断有效。

TIMERx_BGLOAD

TIMERx_BGLOAD 为周期模式计数初值寄存器。Timer0~3 各有 1 个周期模式计数初值寄存器。

TIMERx_BGLOAD 寄存器中包含了定时器的计数初值。该寄存器用于在周期模式下，当定时器的计数值递减到 0 时重新载入计数初值。

该寄存器提供了访问 **TIMERx_LOAD** 寄存器的另一种方法。不同之处在于写入值到 **TIMERx_BGLOAD** 寄存器中不会导致定时器立即从新写入值开始计数。

1. TIMER0_BGLOAD

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	TIMER0_BGLOAD	0x0000_0000
Name	timer0bgload		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	timer0bgload	Timer0 的计数初值。 (注意和 TIMERx_LOAD 寄存器有区别, 具体请参见 TIMERx_LOAD 寄存器的描述。)

2. TIMER1_BGLOAD

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	TIMER1_BGLOAD	0x0000_0000
Name	timer1bgload		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	timer1bgload	Timer1 的计数初值。 (注意和 TIMERx_LOAD 寄存器有区别, 具体请参见 TIMERx_LOAD 寄存器的描述。)

3.8 看门狗

3.8.1 概述

看门狗 WatchDog 用于系统异常情况下, 一定时间内发出复位信号, 以复位整个系统。

3.8.2 特点

WatchDog 具备以下特点:

- 内部具有一个 32bit 减法计数器, 计数时钟可配置。
- 支持超时时间间隔 (即计数初值) 可配置。
- 支持寄存器锁定, 防止寄存器被误改。
- 支持超时中断产生。
- 支持复位信号产生。
- 支持调试模式。



3.8.3 信号描述

表3-23 WatchDog 接口信号描述

信号名称	方向	描述	对应管脚
WDG_RST	O	WatchDog 输出的复位信号，低电平有效。	WDGRST



注意

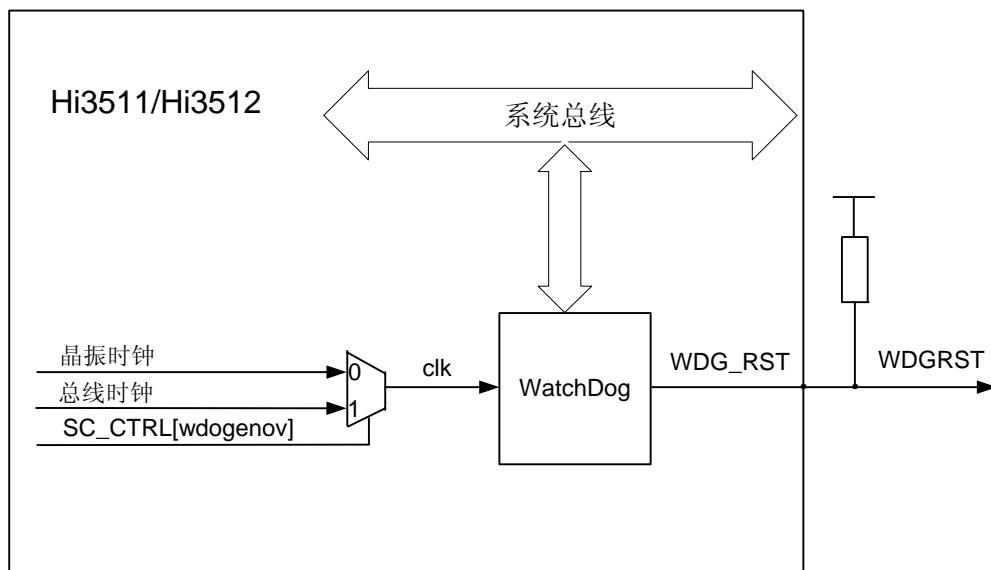
WDGRST 管脚为 OD (Open-Drain) 门输出，需要外接上拉电阻。

3.8.4 功能描述

应用框图

WatchDog 应用框图如图 3-23 所示。

图3-23 WatchDog 应用框图



功能原理

WatchDog 的运行基于 1 个 32bit 减法计数器，计数初值由寄存器 [WDG_LOAD](#) 载入。在 WatchDog 时钟使能情况下，计数器的值在每个计数时钟的上升沿减 1。当计数值递减到 0，WatchDog 将产生一个中断。然后在下一个计数时钟上升沿，计数器又从寄存器 [WDG_LOAD](#) 中重新载入计数初值，开始递减计数。



如果计数器第二次计数递减到 0 时, CPU 还没有清除 WatchDog 中断, 则 WatchDog 将发出复位信号 [WDG_RST](#), 计数器停止计数。

根据实际应用需要, 可通过配置 [WDG_CONTROL](#) 使能或者禁止 WatchDog 产生中断和复位信号。此时, 存在以下两种情况:

- 当禁止产生中断时, 计数器将停止计数。
- 当重新开启中断时, WatchDog 将从 [WDG_LOAD](#) 的设定值开始计数, 而不是从计数器上次停止时的计数值开始计数。在中断到来之前, 可以重新载入初值。

WatchDog 的计数时钟可以选择晶振时钟或者总线时钟, 便于选择不同的计数时间范围。

通过配置 [WDG_LOCK](#) 寄存器, 可以禁止对 WatchDog 内部寄存器进行写操作:

- 向 [WDG_LOCK](#) 写入 0x1ACC_E551, 可以打开所有 WatchDog 寄存器的写权限。
- 向 [WDG_LOCK](#) 寄存器写入其他任何值, 可以关闭所有 WatchDog 寄存器 ([WDG_LOCK](#) 寄存器除外) 的写权限。

该保护 WatchDog 的寄存器的特性不会被软件错误地修改, 从而使得在异常情况下, WatchDog 不致被软件错误地中止操作。

在调试模式下, WatchDog 自动关闭, 以防止干扰正常的调试操作。



注意

在系统进入 SLEEP 模式之前, 必须关闭 WatchDog。具体操作请参见 “[3.8.5 工作方式](#)” 中的 “[关闭 WatchDog](#)”。

3.8.5 工作方式

计数时钟频率配置

系统支持 2 种 WatchDog 计数时钟: 晶振时钟和总线时钟, 通过系统控制器中 [SC_CTRL\[wdogenov\]](#) 进行配置。

WatchDog 计数时间为:

$$T_{WDG} = Value_{WDG_LOAD} \times (1/f_{clk})$$



说明

T_{WDG} 表示 WatchDog 计数时间, $Value_{WDG_LOAD}$ 表示 WatchDog 计数初值, f_{clk} 表示 WatchDog 计数时钟频率。

WatchDog 在不同时钟下的计数时间范围值如下:

- 当选择晶振时钟 (3MHz) 时, 计数时间范围为 0~1431s。
- 当选择总线时钟 (135MHz) 时, 计数时间范围为 0~31s。



系统初始化配置

系统上电复位后 WatchDog 计数器处于停止计数状态，在系统初始化过程中需要将 WatchDog 初始化并启动其运行。WatchDog 的初始化过程如下：

- 步骤 1 配置 [WDG_LOAD](#)，设定计数初值。
 - 步骤 2 配置 [WDG_CONTROL](#)，打开中断屏蔽并启动 WatchDog 计数。
 - 步骤 3 配置 [WDG_LOCK](#)，给 WatchDog 上锁，防止软件错误修改 WatchDog 的配置。
- 结束

中断处理过程

收到 WatchDog 发出的中断后，应及时清除其中断状态，并使其载入计数初值重新开始计数。WatchDog 中断处理的过程如下所示：

- 步骤 1 向 [WDG_LOCK](#) 写 0x1ACC_E551，为 WatchDog 开锁。
 - 步骤 2 配置 [WDG_INTCLR](#)，清除 WatchDog 的中断状态，同时也使 WatchDog 自动载入计数初值重新开始计数。
 - 步骤 3 配置 [WDG_LOCK](#)，写入 0x1ACC_E551 以外的任何值，给 WatchDog 上锁。
- 结束

关闭 WatchDog

在系统进入 SLEEP 模式之前，必须关闭 WatchDog。向寄存器 [WDG_CONTROL\[inten\]](#) 控制位写入 0，即关闭 WatchDog。向该位写入 1，即打开 WatchDog。

3.8.6 寄存器概览

WatchDog 寄存器概览如表 3-24 所示。

表3-24 WatchDog 寄存器概览（基址是 0x101E_1000）

偏移地址	名称	描述	页码
0x000	WDG_LOAD	计数初值寄存器	3-69
0x004	WDG_VALUE	计数器当前值寄存器	3-69
0x008	WDG_CONTROL	控制寄存器	3-69
0x00C	WDG_INTCLR	中断清除寄存器	3-69
0x010	WDG_RIS	原始中断寄存器	3-69
0x014	WDG_MIS	屏蔽后中断寄存器	3-69
0x018 ~ 0xBFC	RESERVED	保留	-

偏移地址	名称	描述	页码
0xC00	WDG_LOCK	LOCK 寄存器	3-69
0xC04 ~ 0xFFC	RESERVED	保留	-

3.8.7 寄存器描述

WDG_LOAD

WDG_LOAD 为计数初值寄存器，用来配置 WatchDog 内部计数器的计数初值。

	Offset Address	Register Name	Total Reset Value
	0x000	WDG_LOAD	0xFFFF_FFFF
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	wdg_load		
Reset	1 1		
Bits	Access	Name	Description
[31:0]	RW	wdg_load	计数初值。

WDG_VALUE

WDG_VALUE 为计数器当前值寄存器，用来读出 WatchDog 内部计数器的当前计数值。

	Offset Address	Register Name	Total Reset Value
	0x004	WDG_VALUE	0xFFFF_FFFF
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	wdogvalue		
Reset	1 1		
Bits	Access	Name	Description
[31:0]	RO	wdogvalue	WatchDog 计数器当前值。

WDG_CONTROL

WDG_CONTROL 为控制寄存器，用来控制 WatchDog 的打开/关闭、中断和复位功能。



Offset Address			Register Name			Total Reset Value																										
Bit	0x008																															
Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												resen	inten		
Bits	Access	Name	Description																													
[31:2]	-	reserved	保留。																													
[1]	RW	resen	WatchDog 复位信号输出使能。 0: 禁止。 1: 使能。																													
[0]	RW	inten	WatchDog 中断信号输出使能。 0: 计数器停止计数, 计数值保持当前值不变, WatchDog 被关闭。 1: 既启动计数器又使能中断, WatchDog 被启动。																													



说明

若 [WDG_CONTROL\[inten\]](#) 位表示的中断先前被禁止, 则当中断再次被使能时, 计数器从 [WDG_LOAD](#) 中载入初值, 并重新开始计数。

WDG_INTCLR

WDG_INTCLR 为中断清除寄存器, 用来清除 WatchDog 中断, 使 WatchDog 重新载入初值进行计数。本寄存器是只写寄存器, 写进去任意值, 都会引起 WatchDog 清中断, 内部并不记忆写入的值, 无复位值。

Offset Address			Register Name			Total Reset Value																										
Bit	0x00C																															
Reset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	wdg_intclr																															
Bits	Access	Name	Description																													
[31:0]	WO	wdg_intclr	对该寄存器写入任意值可清除 WatchDog 的中断, 并使 WatchDog 从寄存器 WDG_LOAD 中重新载入初值重新计数。																													

WDG_RIS

WDG_RIS 为原始中断寄存器，用来反映 WatchDog 原始中断状态。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	WDG_RIS	0x0000_0000
Name	reserved		wdogris
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。
[0]	RO	wdogris	WatchDog 原始中断状态，当计数器的计数值递减到 0 时，该位置 1。 0: 未产生中断。 1: 已产生中断。

WDG_MIS

WDG_MIS 为屏蔽后中断寄存器，用来反映屏蔽后的 WatchDog 中断状态。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	WDG_MIS	0x0000_0000
Name	reserved		wdogmis
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。
[0]	RO	wdogmis	WatchDog 的屏蔽后中断状态。 0: 未产生中断或者中断被屏蔽。 1: 已产生中断。

WDG_LOCK

WDG_LOCK 为 LOCK 寄存器，用来控制 WatchDog 寄存器的读写权限。



3.9 实时时钟

3.9.1 概述

实时时钟 RTC (Real Time Clock) 用于实现时间显示和定时报警功能。

3.9.2 特点

RTC 具备以下特点：

- 内部具有 1 个 32bit 加法计数器
 - 计数时钟 1Hz
 - 计数初值可配置
 - 计数比较值可配置
 - 支持超时中断产生
 - 支持软复位

3.9.3 功能描述

RTC 的运行基于 1 个 32bit 加法计数器，计数初值由寄存器 **RTC_LR** 载入。计数器的值在每个计数时钟的上升沿加 1。当计数值递加到 **RTC_LR** 寄存器与 **RTC_MR** 寄存器值相等时，RTC 将产生一个中断，然后在下一个计数时钟上升沿，计数器继续递加计数。

RTC 在系统中的中断号为“10”。

根据实际应用需要，可通过配置 `RTC_IMSC` 使能或者禁止 RTC 产生中断信号。此时，存在以下两种情况：



- 当禁止产生中断时，RTC 计数器继续递加计数，将不会对外产生中断，在 [RTC_MIS](#) 中显示屏蔽后中断的状态，在 [RTC_RIS](#) 中显示原始中断状态。
- 当重新开启中断时，RTC 计数器仍然继续递加计数，当计数值递加到 [RTC_LR](#) 寄存器与 [RTC_MR](#) 寄存器值相等时，RTC 将产生一个中断。

RTC 的计数时钟采用的是 1Hz 时钟，便于通过计数值转换为具体的年、月、日、时、分、秒。

RTC 采用独立供电方式，通过芯片管脚 RTCBATT 提供 3.3V 电源或者纽扣电池电源，以保证在系统断电情况下，RTC 仍然维持电力进行计数。

3.9.4 工作方式

3.9.4.1 计数时钟频率

RTC 采用 1Hz 时钟进行计数，计数最大时间为：

$$T_{RTC} = (2^{32} - 1) \times (1/f_{rtcclk}) = 4294967295(\text{秒}) \approx 49710(\text{天})$$



说明

T_{RTC} 表示 RTC 计数时间， $2^{32} - 1$ 表示 RTC 计数最大值， f_{rtcclk} 表示 RTC 计数时钟频率：1Hz。

3.9.4.2 软复位

通过配置系统控制器 [SC_PERCTRL0\[rtc_srst\]](#)，可以实现对 RTC 的单独软复位。软复位后各个 RTC 配置寄存器的值均恢复为默认值，因此软复位后需要重新对这些寄存器进行初始化配置。

软复位步骤如下：

- 步骤 1 向 [SC_PERCTRL0\[rtc_srst\]](#) 写 0，对 RTC 软复位。
- 步骤 2 向 [SC_PERCTRL0\[rtc_srst\]](#) 写 1，撤消对 RTC 的软复位。

----结束

3.9.4.3 系统初始化

系统上电复位后，RTC 计数器处于停止计数状态，在系统初始化过程中需要将 RTC 初始化并启动其运行。RTC 的初始化过程如下：

- 步骤 1 配置 [RTC_CR\[rtc_start\]](#)=0b1，启动 RTC 计数器开始计数。
- 步骤 2 配置 [RTC_IMSC\[rtc_imsc\]](#)=0b0，设置 RTC 中断屏蔽位。
- 步骤 3 配置 [RTC_MR](#)，设置 RTC 比较值。
- 步骤 4 配置 [RTC_LR](#)，设置 RTC 计数初始值。
- 步骤 5 RTC 按照 1Hz 的计数时钟频率，从 [RTC_LR](#) 中的值开始计数，当计数到 [RTC_MR](#) 中的值时，将根据 [RTC_IMSC](#) 的设置，决定是否产生中断。

----结束



3.9.4.4 中断处理

系统收到 RTC 发出的中断后，表示定时时间到，随后转入“定时开机”、“定时关机”等相应操作，RTC 计数器仍然保持递加计数。RTC 中断处理的过程如下：

步骤 1 配置 **RTC_ICR[rtc_icr]=0b1**，清除 RTC 的中断状态。

步骤 2 如果需要继续设置定时时间，则向寄存器 **RTC_MR** 写入新的比较值。

----结束

3.9.4.5 关闭 RTC

一旦配置 **RTC_CR**，启动 RTC 计数后，RTC 将一直处于计数状态。只有对 RTC 复位后，才能关闭 RTC。对 RTC 的软复位操作请参见“[3.9.4.2 软复位](#)”操作。

3.9.5 寄存器概览

表3-25 RTC 寄存器概览（基址是 0x101E_8000）

偏移地址	名称	描述	页码
0x000	RTC_DR	计数器当前值寄存器	3-69
0x004	RTC_MR	RTC 比较寄存器	3-69
0x008	RTC_LR	RTC 加载寄存器	3-69
0x00C	RTC_CR	RTC 使能寄存器	3-69
0x010	RTC_IMSC	中断屏蔽寄存器	3-69
0x014	RTC_RIS	原始中断寄存器	3-69
0x018	RTC_MIS	屏蔽后中断寄存器	3-69
0x01C	RTC_ICR	中断清除寄存器	3-69
0x020~0xFFC	RESERVED	保留	-

3.9.6 寄存器描述

RTC_DR

RTC_DR 为计数器当前值寄存器，用来读取 RTC 内部计数器的当前值。

RTC_MR

RTC_MR 为 RTC 比较寄存器，用来设置 RTC 的比较值。

Offset Address										Register Name										Total Reset Value												
0x004										RTC_MR										0x0000_0000												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rtc_match																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits		Access		Name				Description																								
[31:0]		RW		rtc_match				设置的 RTC 比较值。																								

RTC LR

RTC LR 为 RTC 加载寄存器，用来设置 RTC 计数初始值。



RTC_CR

RTC_CR 为控制寄存器，用来使能 RTC。一旦使能，只有系统复位才能清除该寄存器。对该寄存器的任何写操作不起作用。读则返回当前值。

Offset Address		Register Name	Total Reset Value
0x00C		RTC_CR	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	rtc_start
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。
[0]	RW	rtc_start	RTC 使能。 0: 禁止 RTC。 1: 使能 RTC。

RTC_IMSC

RTC_IMSC 为中断屏蔽设置/清除寄存器，用来反映 RTC 中断屏蔽状态。

RTC_RIS

RTC RIS 为原始中断状态寄存器，用来反映 RTC 原始中断状态。

Offset Address			Register Name	Total Reset Value																												
0x014			RTC_RIS	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												rtc_ris			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:1]	-	reserved	保留。																													
[0]	RO	rtc_ris	RTC 原始中断状态。 0: 未产生中断。 1: 已产生中断。																													

RTC_MIS

RTC_MIS 为 RTC 屏蔽后中断寄存器，用来反映屏蔽后的 RTC 中断状态。

Offset Address			Register Name	Total Reset Value																												
0x018			RTC_MIS	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												rtc_mis			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:1]	-	reserved	保留。																													
[0]	RO	rtc_mis	RTC 屏蔽后的中断状态。 0: 未产生中断或者中断被屏蔽。 1: 已产生中断。																													

RTC_ICR

RTC_ICR 为 RTC 清除中断寄存器，用来清除 RTC 中断。



Offset Address		Register Name	Total Reset Value
0x01C		RTC_ICR	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	rtc_icr
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。
[0]	WO	rtc_icr	清除 RTC 中断。 0: 无影响。 1: 清除中断。

3.10 系统控制器

3.10.1 概述

系统控制器模块提供了控制系统运行的手段，它控制系统运行的模式，监控系统运行状态，管理系统中的重要模块（如时钟、复位、管脚复用等），完成对外设的某些功能的配置。

3.10.2 特点

系统控制器具有以下特点：

- 控制并监控系统的运行模式
 - 提供系统时钟控制和状态查询
 - 提供对外设时钟门控的控制、复位控制和查询
 - 系统的中断模式的控制
 - 提供对系统地址重映射的控制和状态监控
 - 提供通用外设控制寄存器对外设进行各种控制
 - 提供对管脚复用的控制
 - 提供对关键寄存器的写保护功能
 - 提供芯片的标识寄存器



3.10.3 功能描述

系统运行模式控制

系统工作在以下 4 种工作模式：

- **NORMAL 模式**

系统正常工作在 NORMAL 模式下。在此模式下，系统由片内 ARMPLL 的输出时钟驱动，所有的模块均能正常工作于此时钟源。

- **SLOW 模式**

SLOW 模式是慢速模式。在此模式下，系统由外接晶振时钟驱动，只有部分片内模块可以工作，如系统控制器、Timer、SMI 等。所有对高速时钟有要求的模块在此时钟下无法工作，如 DDRC、USB 1.1 HOST、由 CRG 提供时钟的 AMBA-PCI 桥、SIO0、SIO1、MMC 等。

- **DOZE 模式**

DOZE 模式是低速模式。在此模式下，系统由外接晶振分频的 45kHz 低频时钟驱动。大部分片内外设和存储器接口无法工作，CPU 和少量片内模块（如系统控制器、Timer 等）可以工作于该模式。

- **SLEEP 模式**

SLEEP 模式是休眠模式。在该模式下，CPU 和大部分模块因时钟关断而停止工作。只有系统控制器、IR 模块可以在晶振分频的 45kHz 低频时钟下工作，另外 AMBA-PCI 桥的 AHB 侧也保留低频时钟。

系统控制器提供了一个系统模式切换机制，用于控制系统时钟源的切换。

模式切换由模式控制寄存器 **SC_CTRL[modectrl]** 配置，这 3 位定义了系统当前需要进入的操作模式：

- 000：系统切换到 SLEEP 模式。
- 001：系统切换到 DOZE 模式。
- 01X：系统切换到 SLOW 模式。
- 1XX：系统切换到 NORMAL 模式。

说明

X 表示可以为 0 或者 1。

当系统模式被设置后，状态机将控制模式的自动切换，无需软件的干预。当前系统状态可通过读取 **SC_CTRL[modestatus]** 获得。这几位描述的系统当前状态不仅包括了上述的 4 个主要模式：NORMAL、SLOW、DOZE、SLEEP，还包括 4 个主要模式之间的几个中间态：SW from PLL、SW to PLL、PLLCTL、SW from XTAL、SW to XTAL、XTALCTL。

说明

NORMAL、SLOW、DOZE、SLEEP 四种模式切换，可配置为直接切换，如系统当前处于 NORMAL 模式，可通过配置寄存器 **SC_CTRL [modectrl]** 为“001”进入 DOZE 模式。但实际系统运行过程中，是经历了“SW from PLL”、“SLOW”、“SW from XTAL”等模式或中间态的。

上电复位后，系统控制器默认处于 SLOW 状态。



中断模式下，当 VIC 接收到中断输入，待切换模式由中断响应模式寄存器指定，而不是由 [SC_CTRL\[modectrl\]](#) 指定。

系统控制器和时钟模块配合完成系统时钟和系统模式的切换。当状态机状态发生迁移时，系统控制器发出时钟切换指示信号，时钟模块随之进行时钟切换，并向系统控制器反馈切换完成指示信号，系统控制器检测到切换完成指示信号，完成模式切换。

系统控制器状态机状态和系统时钟之间的关系如[表 3-26](#) 所示。

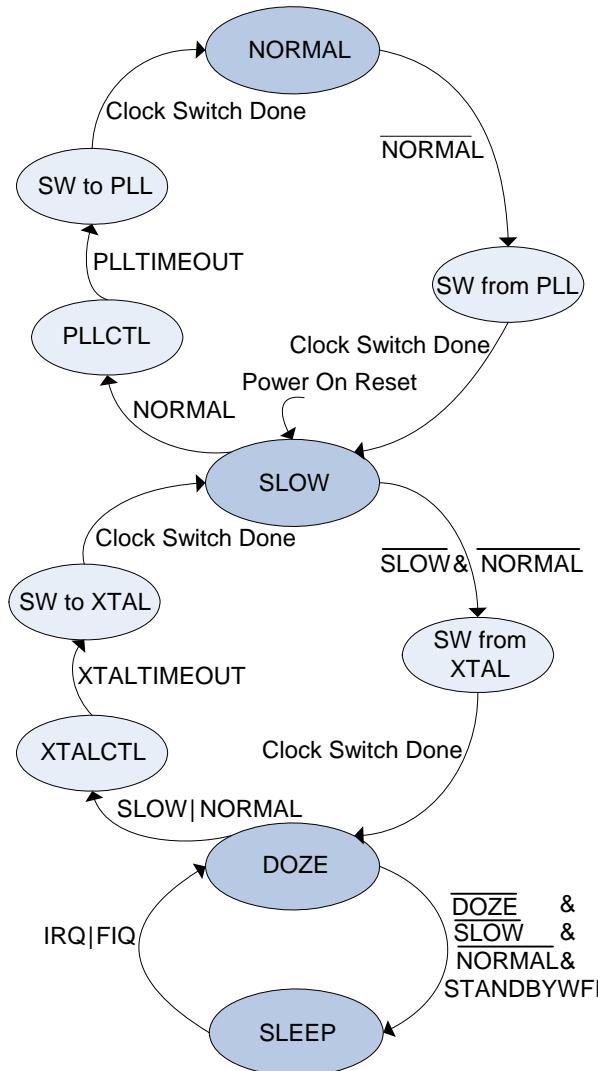
表3-26 系统控制器状态和时钟切换对应关系表

系统控制器状态	27MHz 晶振使能状态	PLL 使能状态	系统时钟状态
NORMAL	使能	使能	ARM 子系统的工作时钟来自 PLL 输出。
SLOW	使能	不使能	ARM 子系统的工作时钟来自 27MHz 晶振输入。
DOZE	使能	不使能	ARM 子系统的工作时钟都来自 45KHz 晶振分频时钟。
SLEEP	使能	不使能	除系统控制器和红外模块工作在 45KHz 外，其它模块的时钟都处于关闭状态。

系统的状态切换过程如[图 3-24](#) 所示。



图3-24 系统模式切换图



各种模式之间的切换涉及到的操作如下：

- **SLEEP 模式**

在 SLEEP 模式下，除系统控制器和 IR 模块时钟由低速 45kHz 晶振分频时钟驱动外，其它模块的时钟都被关闭。该模式下当有 FIQ 或者 IRQ 中断发生时，系统迁移到 DOZE 状态，并且 **SC_CTRL[modectrl]** 的值由 SLEEP 自动更新为 DOZE。

说明

从 SLEEP 模式切换到 DOZE 模式，是由中断触发的。因此要求在进入 SLEEP 模式之前保证中断控制器不会对相应的 IRQ 或 FIQ 中断进行屏蔽。当使用 GPIO 的外部中断输入来触发系统切换到 DOZE 时，GPIO 的外部中断输入必须是电平触发，不能使用边沿触发。

- **DOZE 模式**

在 DOZE 模式下，系统时钟和系统控制器时钟由 45kHz 晶振分频时钟驱动。该模式可能发生的状态迁移有：



- 如果 **SC_CTRL[modectrl]** 被设置为 SLOW 模式或者 NORMAL 模式，系统将进入晶振控制状态 **SC_XTALCTL**，对 27MHz 时钟晶振进行初始化。当晶振稳定后，系统迁移到 SW to XTAL 状态，将系统时钟从 45kHz 切换到 27MHz 时钟，进入 SLOW 模式。
- 如果 **SC_CTRL[modectrl]** 被设为 SLEEP 模式，并且 ARM926EJ-S 处于 Wait-for-interrupt 状态，系统进入 SLEEP 模式。

说明

- 系统控制器 **SC_XTALCTRL[18:3]** 定义了 27MHz 晶振的稳定时间，当晶振被使能时，超时计数器开始计数，用户可通过查询 **SC_XTALCTRL[2]** 判断 27MHz 时钟是否已经稳定。
- 用户可通过设置 ARM926EJ-S 系统控制协处理器 CP15 R7，使处理器进入低功耗(Wait-for-interrupt)状态。
- SLOW 模式
在 SLOW 模式下，ARM 子系统工作于 27MHz 时钟。该模式下可能发生的状态迁移有：
如果 **SC_CTRL[modectrl]** 被设为 NORMAL 模式，系统将进入 PLL 控制状态 **SC_PLLCTRL**，使能 PLL。当 PLL 稳定后，系统进入 SW to PLL 状态，将系统时钟切换到 PLL 时钟，切换完成后，进入 NORMAL 模式。
- NORMAL 模式
在 NORMAL 模式下，ARM 子系统工作于 PLL 的输出时钟。该模式下，如果 **SC_CTRL[modectrl]** 被设为非 NORMAL 模式，系统迁移到 SW from PLL 状态，将系统时钟切换到 27MHz 时钟，切换完成后，进入 SLOW 模式。

PLL 控制

系统控制器的系统状态机可用于控制片内 PLL 的使能，各种模式下 PLL 的状态如表 3-26 所示。

PLL 频率控制

系统控制器集成了 1 个 PLL 频率控制寄存器，用于定义 PLL 的倍频系数。具体请参见 **SC_PLLFCTRL**。

中断响应模式

中断响应模式用于定义中断发生后系统状态机所处的模式。中断响应模式由中断模式控制寄存器组进行控制，该组寄存器定义了如下功能：

- 中断响应模式是否使能。
- 中断发生后系统的工作模式。

- 触发中断响应模式的中断类型是 FIQ 还是 IRQ。
- 中断模式状态查询和清除机制。

说明

中断响应模式只支持系统运行频率从低速切换到高速，例如从 DOZE 模式切换到 NORMAL；不支持高速模式向低速模式点切换，例如从 NORMAL 模式切换到 SLOW 模式。

软复位

系统控制器支持对芯片全局以及局部模块进行软复位：

- 当配置全局软复位寄存器 [SC_SYSSTAT](#) 后，系统控制器将给片内复位模块发起请求，芯片将被复位。
- 当配置系统控制器的相应模块软复位控制位后，系统控制器将控制片内复位模块对这个模块进行复位。

对模块的一次软复位操作包括配置软复位和撤消软复位。例如要对 UART0 模块进行软复位，则需要首先对 [SC_PERCTRL0\[uart0_srst\]](#) 写 1 进行软复位，再写 0 撤消软复位。

说明

- 系统软复位通过对寄存器 [SC_SYSSTAT](#) 写任意值来实现。系统软复位的撤消自动完成，不需软件干预。
- 各模块的软复位控制见 [SC_PERCTRL0](#)。

系统地址重映射控制

系统控制器提供地址重映射控制信号，支持地址译码单元对系统存储地址空间进行重新映射和分配。上电复位后，芯片根据管脚 FUNSEL0 的值将 0 地址映射到不同的物理空间。FUNSEL0 等于 0 时，0 地址对应与 SMI 片选 0 所占的地址空间；FUNSEL0 等于 1 时，0 地址对应与 MDDRC 所占的地址空间。可通过系统控制器提供的 Remap 清除地址重映射。

说明

地址重映射清除以后，0 地址到 SMI 或 MDDRC 的存储器空间的映射被清除。此时，建议将 0x0000_0000 ~ 0x0000_0FFF 地址通过配置 CPU 分配给片内的 2KB 的 ITCM。

Watch Dog 和 TIMER 时钟使能控制

时钟使能可以使计数频率独立于系统时钟频率，即使系统时钟发生改变，计数器仍然会保持固定的计数频率，系统控制器时钟提供以下使能控制功能：

- 支持对输入的计数时钟进行采样，生成时钟使能信号，输出给 WatchDog 和 TIMER 模块。
- 可通过软件强制将 WatchDog 和 TIMER 计数时钟使能拉高，使其内部计数器按总线时钟计数，当系统处于 Debug 模式时，WatchDog 计数功能会被禁止。
- 支持对 TIMER 的计数时钟源进行选择。

管脚复用控制

系统控制器提供对管脚复用的控制。



在系统中，某些接口模块的管脚是静态复用的。如 UART1 模块和 GPIO 信号存在静态复用关系，因此在应用 UART1 的场景中，就无法使用对应的 GPIO。系统控制器在此时将芯片管脚“分配”给 UART1 模块（通过配置寄存器 [SC_PERCTRL1](#) 的某些比特位来实现）。



在使用某个模块时，如果此模块与其他模块存在管脚复用，则必须首先对管脚复用进行配置。

对关键寄存器的写保护

为防止软件对系统控制器的误操作对整个系统产生严重影响，系统控制器提供了对系统控制器的一些关键配置寄存器的写保护功能。包括：

- 模式切换的控制寄存器 [SC_CTRL](#)
- 系统全局软复位控制寄存器 [SC_SYSSTAT](#)
- 片内 ARMPPLL 的控制寄存器 [SC_PLLCTRL](#)
- 片内锁相环的频率设置寄存器 [SC_PLLFCTRL](#)

对这些关键寄存器进行写操作之前，必须配置寄存器 [SC_PERLOCK](#) 取消写保护。操作完成之后配置寄存器 [SC_PERLOCK](#) 打开写保护，让这些关键寄存器不会被软件随意改写。

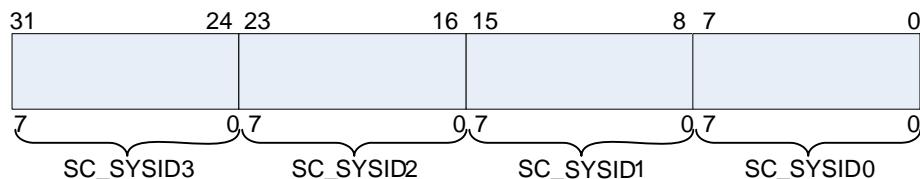


系统默认为复位后不对这些关键寄存器进行写保护处理。为启用此功能，建议在系统启动时利用该寄存器对这些关键寄存器（[SC_CTRL](#)、[SC_PLLCTRL](#)、[SC_PLLFCTRL](#)）进行写保护处理。

芯片的标识寄存器

系统控制器提供了芯片标识（ID）寄存器 [SC_SYSID](#)。这个标识寄存器是一个概念上的 32bit 的标识只读寄存器，实际上由 4 个 8bit 标识寄存器组成：[SC_SYSID3](#)、[SC_SYSID2](#)、[SC_SYSID1](#)、[SC_SYSID0](#)。读出这 4 个寄存器的值，通过组合得到芯片的 32 比特标识寄存器的值 0x3511_XXXX（XXXX 表示芯片的版本号），组合的方法如图 3-25 所示。

图3-25 芯片 ID 寄存器位分配图



3.10.4 寄存器概览

系统控制器寄存器概览如表 3-27 所示。

表3-27 系统控制器寄存器概览 (基址是 0x101E_0000)

偏移地址	名称	描述	页码
0x000	SC_CTRL	系统控制寄存器	3-69
0x004	SC_SYSSTAT	系统状态寄存器	3-69
0x008	SC_ITMCTRL	中断模式控制寄存器	3-69
0x00C	SC_IMSTAT	中断模式状态寄存器	3-69
0x010	SC_XTALCTRL	晶振控制寄存器	3-69
0x014	SC_PLLCTRL	PLL 控制寄存器	3-69
0x018	SC_PLLFCTRL	PLL 频率控制寄存器	3-69
0x01C	SC_PERCTRL0	外设控制寄存器 0 (IP 软复位控制)	3-69
0x020	SC_PERCTRL1	外设控制寄存器 1 (管脚复用控制)	3-69
0x024	SC_PEREN	外设时钟使能寄存器	3-69
0x028	SC_PERDIS	外设时钟禁止寄存器	3-69
0x02C	SC_PERCLKEN	外设时钟使能状态寄存器	3-69
0x030	RESERVED	保留	-
0x034	SC_PERCTRL2	外设控制寄存器 2 (IP 时钟分频控制)	3-69
0x038	SC_PERCTRL3	外设控制寄存器 3 (IP 时钟分频控制)	3-69
0x03C	SC_PERCTRL4	外设控制寄存器 4 (芯片工作模式控制)	3-69
0x040	RESERVED	保留	-
0x044	SC_PERLOCK	关键系统控制寄存器的锁定寄存器	3-69
0x006C~0xEDC	RESERVED	保留	-
0xEE0~0xEEC	SC_SYSID	系统标识寄存器	3-69

3.10.5 寄存器描述

SC_CTRL

SC_CTRL 为系统控制寄存器，用于指定需要系统完成的操作。



Offset Address			Register Name																		Total Reset Value																		
Bit	0x000			SC_CTRL																		0x0000_0212																	
Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0							
Bits	Access	Name	Description																																				
[31:24]	RW	reserved	保留。读时返回 0, 写时无影响。																																				
[23]	RW	wdogenov	WDG 计数时钟选择。 0: WDG 使用 3MHz 时钟进行计数。 1: WDG 使用总线时钟进行计数。																																				
[22]	RW	timeren3ov	Timer3 计数时钟选择。 0: 使能信号通过采用参考时钟得到, 参考时钟的选择由 [timeren3sel] 指定。 1: 由总线时钟进行计数。																																				
[21]	RW	timeren3sel	Timer3 计数时钟频率选择 (必须配置为 0)。 0: 选择 3MHz 时钟进行计数。 1: 保留。																																				
[20]	RW	timeren2ov	Timer2 计数时钟选择。 0: 使能信号通过采用参考时钟得到, 参考时钟的选择由 [TimerEn2Sel] 指定。 1: 由总线时钟进行计数。																																				
[19]	RW	timeren2sel	Timer2 计数时钟频率选择 (必须配置为 0)。 0: 选择 3MHz 时钟进行计数。 1: 保留。																																				
[18]	RW	timeren1ov	Timer1 计数时钟选择。 0: 使能信号通过采用参考时钟得到, 参考时钟的选择由 [timeren1sel] 指定。 1: 由总线时钟进行计数。																																				
[17]	RW	timeren1sel	Timer1 计数时钟频率选择 (必须配置为 0)。 0: 使用 3MHz 时钟进行计数。 1: 保留。																																				



[16]	RW	timeren0ov	Timer0 计数时钟选择。 0: 使能信号通过采用参考时钟得到, 参考时钟的选择由 [timeren0sel] 指定。 1: 由总线时钟进行计数。
[15]	RW	timeren0sel	Timer0 计数时钟频率选择 (必须配置为 0)。 0: 使用 3MHz 时钟进行计数。 1: 保留。
[14:12]	-	reserved	保留。
[11:10]	RW	reserved	保留。读时返回 0, 写时无影响。
[9]	RO	remapstat	地址重映射的状态。 0: 未进行地址重映射。 1: 进行地址重映射。具体如下所述: 当加载模式为自加载时, EBICS0N 被 Remap 到地址 0。 当加载模式为从加载时, DDRCSN 被 Remap 到地址 0。
[8]	RW	remapclear	地址重映射清除选择。 0: 保持 Remap 状态。 1: 清除 Remap。 Clear Remap 前后地址映射关系请参见 “ 3.3 处理器及存储器地址空间映射 ”。
[7]	RW	reserved	保留。读时返回 0, 写时无影响。
[6:3]	RO	modestatus	模式状态位。 这些位返回系统当前的操作模式。这 4 位定义如下: 0000: SLEEP。 0001: DOZE。 0010: SLOW。 0011: XTAL CTL。 0100: NORMAL。 0110: PLL CTL。 1001: SW from XTAL。 1010: SW from PLL。 1011: SW to XTAL。 1110: SW to PLL。 其余: 保留, 未使用。



[2:0]	RW	modectrl	<p>模式控制位。这些位定义了要求系统控制器进入的操作模式。这 3 位定义如下：</p> <p>000: SLEEP; 001: DOZE; 01X: SLOW; 1XX: NORMAL。</p>
-------	----	----------	--

注：这个寄存器被 [SC_PERLOCK](#) 寄存器写保护，只有禁用写保护，对这个寄存器的写操作才有效。

SC_SYSSTAT

SC_SYSSTAT 为系统状态寄存器。向该寄存器写入任何值都会使系统控制器向复位模块发出系统软复位请求，复位模块进行系统软复位。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	SC_SYSSTAT	0x0000_0002
Name	softresreq		
Reset	0 1 0		
Bits	Access	Name	Description
[31:0]	WO	softresreq	对该寄存器的任意写操作都会导致系统软复位。

注：这个寄存器被 [SC_PERLOCK](#) 寄存器写保护，只有打开写保护，对这个寄存器的写操作才有效。

SC_IMCTRL

SC_IMCTRL 为中断模式控制寄存器，用于控制中断发生时的系统模式。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	SC_IMCTRL	0x0000_0000
Name	reserved	inmdtype reserved itmdctrl itmden	
Reset	0 0		
Bits	Access	Name	Description
[31:8]	RW	reserved	保留。读时返回 0，写时无影响。
[7]	RW	inmdtype	设置触发系统进入中断模式的中断类型。 0: 仅有 FIQ 中断能使系统进入中断模式。

			1: FIQ 中断和 IRQ 中断都能使系统进入中断模式。
[6:4]	-	reserved	保留。
[3:1]	RW	itmctrl	设置中断模式下系统最低的工作模式, 该寄存器的值和 SC_CTRL[modectrl]的值相或后作为中断发生后系统所处的工作模式。定义见下。 000: SLEEP。 001: DOZE。 01X: SLOW。 1XX: NORMAL。
[0]	RW	itmden	中断模式使能。 0: 关闭中断模式。 1: 当中断发生时进入中断模式。

SC_IMSTAT

SC_IMSTAT 为中断模式状态寄存器, 用于监视系统是否处于中断模式, 同时也可以通过配置该寄存器强制系统进入中断模式。

	Offset Address	Register Name	Total Reset Value
	0x00C	SC_IMSTAT	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	itmstat
Reset	0 0		
Bits	Access	Name	Description
[31:1]	RW	reserved	保留。读时返回 0, 写时无影响。
[0]	RW	itmstat	中断模式状态。该位可用于软件控制直接进入中断模式。 读该寄存器时: 0: 当前未处于中断模式。 1: 当前处于中断模式。 写该寄存器时: 0: 软件不控制进入中断模式。 1: 软件控制进入中断模式。

注: 当中断服务程序结束时必须手动清除中断模式。

SC_XTALCTRL

SC_XTALCTRL 为晶振控制寄存器，用于控制初始化时钟模块的稳定等待时间，也就是从 XTAL CTL 中间态跳转到 SW to XTAL 中间态的等待时间。



说明

为了使系统从 DOZE 模式向 SLOW 模式切换更迅速, SC_XTALCTRL[xtaltime]的值需要配置为 0xFFFF, 即认为晶振上电后处于稳定状态。

SC_PLLCTRL

SC_PLLCTRL 为 PLL 控制寄存器，用于控制片内 ARM 锁相环 (ARMPLL) 的使能控制：软件控制使能，或由系统模式切换来控制使能。此外，该寄存器还用于设置 ARMPLL 锁相环稳定等待时间。

当处于“由系统模式切换来控制使能 ARMPLL 锁相环”（由 `SC_PLLCTRL[pllover]` 控制）时，在系统处于非 NORMAL 模式下，ARMPLL 被自动关闭；

当处于“由软件控制使能 ARMPPLL 锁相环”时，ARMPPLL 的使能受软件控制（由 SC_PLLCTRL[plen]控制），不受模式切换的影响。

ARMPPLL 的时钟频率由寄存器 [SC_PLLFCTRL](#) 中相应位控制。

Offset Address			Register Name	Total Reset Value																												
0x014			SC_PLLCTRL	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved	plltime																									reserved	reserved	pllover			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:28]	RW	reserved	保留。读时返回 0, 写时无影响。																													
[27:3]	RW	plltime	设置 ARMPLL 锁相环稳定等待时间。 这段时间用于等待 PLL 启动到 PLL 输出达到稳定的状态。也就是指定从系统模式切换时从 PLL CTL 状态跳转到 SW to PLL 状态的等待时间。超时时间值由下式计算得到 (T _{XIN} 为芯片外接晶振的时钟周期)： (33554432 – plltime) × T _{XIN}																													
[2]	RW	reserved	保留。读时返回 0, 写时无影响。																													
[1]	-	reserved	保留。																													
[0]	RW	pllover	允许 ARLPLL 锁相环直接受软件控制使能, 而不是受系统模式状态改变的控制。此位必须配置为 0。 0: 由系统模式切换来使能 ARMPLL 锁相环。 1: 保留。																													

注: 这个寄存器被 [SC_PERLOCK](#) 寄存器写保护, 只有禁止写保护, 对这个寄存器的写操作才有效。



说明

系统要求 PLL 在变更频率配置时, 需要等待 0.5ms 才能输出稳定的时钟。因此该寄存器的 plltime 的配置必须满足此要求。

SC_PLLFCTRL

SC_PLLFCTRL 为 PLL 频率控制寄存器, 用于控制 ARMPLL 的频率和使能旁路功能, 当需要旁路 ARMPLL 时, 设置 SC_PLLFCTRL[armpll_bypass]位为 1, 此时 ARMPLL 输出外接晶振时钟。

锁相环输出时钟频率由此公式计算得到:

$$F_{PLL} = F_{XIN} \times M / (N \times 2^{OD})$$

配置要求:

- $M \geq 2$
- $N \geq 2$



- $1\text{MHz} \leq F_{XIN}/N \leq 25\text{MHz}$
- $200\text{MHz} \leq F_{PLL} \times 2^{OD} \leq 1\text{GHz}$

F_{XIN} 为外接晶振的时钟频率。 F_{PLL} 为 PLL 输出时钟。M、N、OD 在寄存器中的相应位进行配置。



说明

外接 27MHz 的晶振时，按照默认的 M、N 和 OD 参数，ARMPPLL 输出 432MHz 时钟，经二分频，ARM 时钟为 216MHz。

	Offset Address	Register Name	Total Reset Value
	0x018	SC_PLLFCTRL	0x0000_0909
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved	armpll_bypass armpll_od armpll_m armpll_n	
Reset	0 1 0 0 1 0 0 0 0 1 0 0 0 1		
Bits	Access	Name	Description
[31:15]	RW	reserved	保留，读时返回写入值。
[14]	RW	armpll_bypass	ARMPPLL 时钟分频旁路 (bypass) 控制。 0: 非旁路 (no bypass)。 1: 旁路 (bypass)。
[13:12]	RW	armpll_od	ARM 时钟分频控制，OD，输出分频系数。
[11:4]	RW	armpll_m	ARM 时钟分频控制，M，倍频系数。
[3:0]	RW	armpll_n	ARM 时钟分频控制，N，分频系数。

注：这个寄存器被 [SC_PERLOCK](#) 寄存器写保护，只有禁止写保护，对这个寄存器的写操作才有效。

SC_PERCTRL0

SC_PERCTRL0 为外设控制寄存器 0，用于控制各模块的软复位。

Offset Address																Register Name										Total Reset Value										
0x01C																SC_PERCTRL0										0x0000_0000										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Name	reserved																reserved										vedu_srst									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Bits	Access		Name				Description																													
[31:26]	-		reserved				保留。																													
[25]	RW		i2c_srst				I ² C 软复位控制。 0: 撤消 I ² C 软复位。 1: I ² C 软复位。																													
[24]	RW		uart2_srst				UART2 软复位控制。 0: 撤消 UART2 软复位。 1: UART2 软复位。																													
[23]	RW		rtc_srst				RTC 模块软复位控制。 0: RTC 模块软复位。 1: 撤消 RTC 模块软复位。																													
[22]	RW		usb_srst				USB 1.1 HOST 模块软复位控制。 0: USB 1.1 HOST 模块软复位。 1: 撤消 USB 1.1 HOST 模块软复位。																													
[21]	RW		otg_srst				USB 2.0 OTG 模块软复位控制。 0: USB 2.0 OTG 模块软复位。 1: 撤消 USB 2.0 OTG 模块软复位。																													
[20]	RW		armcore_srst				ARM CORE 模块软复位控制。 0: ARM CORE 模块软复位。 1: 撤消 ARM CORE 模块软复位。 说明: <ul style="list-style-type: none">Hi3511/Hi3512 处于主动加载模式时, 此复位控制寄存器不能控制 ARM CORE 的复位状态。Hi3511/Hi3512 处于被动加载模式时, ARM CORE 上电后处于复位状态。此复位控制寄存器可以控制 ARM CORE 的复位状态。																													
[19]	RW		vo_srst				VO 软复位控制。 0: VO 电路软复位。 1: 撤消 VO 软复位。																													



[18]	RW	vi3_srst	VI3 模块接口时钟域电路的软复位控制。 0: VI3 模块接口时钟域电路软复位。 1: 撤消 VI3 模块接口时钟域电路软复位。
[17]	RW	vi2_srst	VI2 模块接口时钟域电路的软复位控制。 0: VI2 模块接口时钟域电路软复位。 1: 撤消 VI2 模块接口时钟域电路软复位。
[16]	RW	vi1_srst	VI1 模块接口时钟域电路的软复位控制。 0: VI1 模块接口时钟域电路软复位。 1: 撤消 VI1 模块接口时钟域电路软复位。
[15]	RW	vi0_srst	VI0 模块接口时钟域电路的软复位控制。 0: VI0 模块接口时钟域电路软复位。 1: 撤消 VI0 模块接口时钟域电路软复位。
[14]	RW	vi_srst	VI 模块 AHB 时钟域电路的软复位控制。 0: VI 模块 AHB 时钟域电路软复位。 1: 撤消 VI 模块 AHB 时钟域电路软复位。
[13]	RW	smi_srst	SMI 模块软复位控制。 0: 撤消 SMI 模块软复位。 1: SMI 模块软复位。
[12]	RW	eth_srst	ETH 模块软复位控制。 0: ETH 模块软复位。 1: 撤消 ETH 模块软复位。
[11]	RW	sio1_srst	SIO1 模块软复位控制。 0: 撤消 SIO1 模块软复位。 1: SIO1 模块软复位。
[10]	RW	sio0_srst	SIO0 模块软复位控制。 0: 撤消 SIO0 模块软复位。 1: SIO0 模块软复位。
[9]	RW	mmc_srst	MMC 模块软复位控制。 0: MMC 模块软复位。 1: 撤消 MMC 模块软复位。
[8]	RW	pci_srst	PCI 模块软复位控制。 0: 撤消 PCI 模块软复位。 1: PCI 模块软复位。



[7]	RW	ir_srst	IR 模块软复位控制。 0: 撤消 IR 模块软复位。 1: IR 模块软复位。
[6]	RW	ssp_srst	SSP 模块软复位控制。 0: 撤消 SSP 模块软复位。 1: SSP 模块软复位。
[5]	RW	uart1_srst	UART1 软复位控制。 0: 撤消 UART1 软复位。 1: UART1 电路软复位。
[4]	RW	uart0_srst	UART0 软复位控制。 0: 撤消 UART0 软复位。 1: UART0 电路软复位。
[3]	RW	dsu_srst	DSU 软复位控制。 0: 撤消 DSU 软复位。 1: DSU 软复位。
[2]	RW	cipher_srst	CIPHER 软复位控制。 0: 撤消 CIPHER 软复位。 1: CIPHER 软复位。
[1]	-	resvered	保留。
[0]	RW	vedu_srst	VEDU (Video Encoding Decoding Unit) 复位控制。 0: 撤消 VEDU 软复位。 1: VEDU 软复位。

SC_PERCTRL1

SC_PERCTRL1 为外设控制寄存器，用于管脚复用控制。



Offset Address																Register Name								Total Reset Value									
0x020																SC_PERCTRL1								0x0000_0000									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	pinmuxctrl31	pinmuxctrl30														reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																														
[31]	RW	pinmuxctrl31	SIO0 输入的发送帧指示复用控制。 0: 从管脚 SIO0XFS。 1: 从管脚 SIO0RFS。																														
[30]	RW	pinmuxctrl30	SIO0 输入的发送时钟复用控制。 0: 从管脚 SIO0XCK。 1: 从管脚 SIO0RCK。																														
[29:25]	-	reserved	保留。																														
[24]	RW	pinmuxctrl24	以太网半双工管脚 ERXERR/ECRS/ECOL 复用控制。 0: 以太网接口管脚。 1: GPIO3_0(GPIO3_1(GPIO3_2)。																														
[23]	RW	pinmuxctrl23	VI1 接口和 GPIO 复用控制。 0: VI1 接口。 1: GPIO6_4~GPIO6_7、GPIO7_0~GPIO7_3。																														
[22]	-	reserved	保留																														
[21]	RW	pinmuxctrl21	控制 EBI (External Bus Interface) 接口中片选信号 EBIADR24 和 GPIO 的复用控制。 0: EBI 接口的地址线。 1: GPIO1_2。																														
[20]	RW	pinmuxctrl20	控制 EBI 接口中片选信号 EBICS1N 和 GPIO 的复用控制。 0: GPIO1_1。 1: EBI 接口的片选信号。																														
[19]	RW	pinmuxctrl19	芯片输出可编程时钟管脚 ACKOUT 管脚 (可用于音频芯片工作时钟) 复用控制。 0: GPIO3_5。 1: 可编程时钟输出。																														

[18]	RW	pinmuxctrl18	SIO0 的 SIO0XFS 管脚和 GPIO 的管脚复用控制。 0: GPIO6_2。 1: SIO0XFS。
[17]	RW	pinmuxctrl17	IRRCV 和 GPIO 的管脚复用控制。 0: IRRCV。 1: GPIO2_3。
[16]	RW	pinmuxctrl16	UART1 接口复用控制。 0: GPIO2_4~GPIO2_7。 1: UART1 URXD1/UTXD1/URTSN1/UCTSN1 信号。
[15]	RW	pinmuxctrl15	DDR 的绕线信号的管脚复用控制。 0: GPIO1_0 和 GPIO0_7。 1: DDR 的绕线信号 DDRMRCVI 和 DDRMRCVO。
[14]	RW	pinmuxctrl14	控制 I ² C 信号和 GPIO 复用控制。 0: I ² C。 1: GPIO3_3 和 GPIO3_4。
[13]	RW	pinmuxctrl13	PCI 的 PCIREQ4 和 PCIGRANT4 与 GPIO 复用控制。 0: GPIO1_7 和 GPIO2_0。 1: PCI 的 PCIREQ4 和 PCIGRANT4。
[12]	RW	pinmuxctrl12	SPI 接口中片选的译码控制。 0: SPICSN0。 1: SPICSN1。
[11]	RW	pinmuxctrl11	SIO0 的 SIO0XCK 管脚和 GPIO 的管脚复用控制。 0: GPIO6_3。 1: SIO0XCK。
[10]	RW	pinmuxctrl10	GPIO 和 SPI 的管脚复用控制。 0: GPIO6_0、GPIO7_6、GPIO6_1 和 GPIO7_5。 1: SPI。
[9]	RW	pinmuxctrl9	MMC 和 GPIO 的管脚复用控制。 0: GPIO1_3~GPIO1_6、GPIO2_1 和 GPIO2_2。 1: MMC。



[8:7]	RW	pinmuxctrl8-7	VI2 接口的同步信号、端口、GPIO、和高清模式下的 VI2 输入的管脚复用控制。 00: GPIO0_5 和 GPIO0_6。 01: VI2 同步接口。 10: UART2。 11: 高清模式下的 10bit VIU 接口。
[6]	RW	pinmuxctrl6	VI3 接口的数据总线与 GPIO 的复用控制。 0: VI3 接口。 1: GPIO5_0~GPIO5_7。
[5]	RW	pinmuxctrl5	SIO1 的 SIO1DI、SIO1RFS、SIO1RCK 和 GPIO 的复用控制。 0: GPIO0_4、GPIO0_2、GPIO0_3。 1: SIO1 的 SIO1DI、SIO1RFS、SIO1RCK。
[4]	RW	pinmuxctrl4	EBIRDYN 和 GPIO 的复用控制。 0: EBIRDYN。 1: GPIO7_4。
[3]	RW	pinmuxctrl3	VI2 接口的数据总线与 GPIO 的复用控制。 0: VI2 接口。 1: GPIO3_6、GPIO7_7、GPIO3_7。
[2]	RW	pinmuxctrl2	VI0 的数据信号 VI0DAT2~VI0DAT9 和 GPIO4_0~GPIO4_7 的管脚复用控制。 0: VI0 的数据信号。 1: GPIO4_0~GPIO4_7。
[1:0]	RW	pinmuxctrl1-0	VI0 的同步信号 (VI0HS 和 VI0VS)、GPIO (GPIO0_0 和 GPIO0_1) 和 VI0 的高清模式数据输入 (VI0DAT0 和 VI0DAT1) 的复用控制。 00: GPIO0_0 和 GPIO0_1。 01: 第 0 路 VI 的同步信号。 10: 独立中断和 SPI 的片选 1。 11: 高清模式下的数据输入。

SC_PEREN

SC_PEREN 为外设时钟使能寄存器，只写寄存器，用于在外部时钟产生逻辑中产生外设时钟的使能信号。向该寄存器的某位写入 1 可打开对应模块的时钟使能，写入 0 无影响。

Offset Address 0x24																Register Name SC_PEREN								Total Reset Value 0xFFFF_FFFF								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved								uart2clken	usbclken	otgclken	sio1clken	sio0clken	ethclken	vi3clken	vi2clken	vi1clken	vihclken	mmcclken	pciclk	irclken	sspclken	uart1clken	dsuclken	smiclk	cipherclken	reserved	vedclken				
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
Bits	Access		Name				Description																									
[31:23]	-		reserved				保留。																									
[22]	WO		uart2clken				UART2 模块时钟使能控制。																									
[21]	WO		usbclken				USB 1.1 HOST 模块时钟使能控制。																									
[20]	WO		otgclken				USB 2.0 OTG 模块时钟使能控制。																									
[19]	WO		sio1clken				SIO1 模块 AHB 时钟使能控制。																									
[18]	WO		sio0clken				SIO0 模块 AHB 时钟使能控制。																									
[17]	WO		ethclken				ETH 模块时钟使能控制。																									
[16]	WO		vi3clken				VI3 模块接口时钟使能控制。																									
[15]	WO		vi2clken				VI2 模块接口时钟使能控制。																									
[14]	WO		vi1clken				VI1 模块接口时钟使能控制。																									
[13]	WO		vi0clken				VI0 模块接口时钟使能控制。																									
[12]	WO		vihclken				VI 模块 AHB 时钟使能控制。																									
[11]	WO		voclken				VO 时钟使能控制。																									
[10]	WO		mmcclken				MMC 模块时钟使能控制。																									
[9]	WO		pciclk				PCI 模块时钟使能控制。																									
[8]	WO		irclken				IR 模块时钟使能控制。																									
[7]	WO		sspclken				SSP 模块时钟使能控制。																									
[6]	WO		uart1clken				UART1 模块时钟使能控制。																									
[5]	WO		uart0clken				UART0 模块时钟使能控制。																									
[4]	WO		smiclk				SMI 模块时钟使能控制。																									
[3]	WO		dsuclken				DSU 模块时钟使能控制。																									
[2]	WO		cipherclken				CIPHER 模块时钟使能控制。																									
[1]	-		reserved				保留。																									



[0]	WO	veduelken	VEDU 模块时钟使能控制。
-----	----	-----------	----------------

SC_PERDIS

SC_PERDIS 为外设时钟禁止寄存器，只写寄存器，用于在外部时钟产生逻辑中将外设的时钟使能信号置为无效。向该寄存器的某位写入 1 可关断对应模块的时钟，写入 0 无影响。

寄存器 SC_PEREN 和寄存器 SC_PERDIS 必须配合使用才能实现对某个模块的时钟的打开或者关闭。如 SMI 模块的时钟当前状态为打开，那么对寄存器 SC_PERDIS 写入 0x0000_0010，则关闭 SMI 的时钟；此时如果需要再次打开 SMI 的时钟，就需要对寄存器 SC_PEREN 写入 0x0000_0010。判断对某个模块的时钟打开或关断的操作是否正常，可以通过读寄存器 SC_PERCLKEN 的相关位。如在打开 SMI 的时钟之后，可通过查询 [SC_PERCLKEN\[4\]](#) 是否为 1 判断操作是否成功。

Offset Address								Register Name								Total Reset Value																							
0x028								SC_PERDIS								0x0000_0000																							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	veduelken						
Name	reserved																								reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
Bits	Access	Name		Description																																			
[31:23]	-	reserved		保留。																																			
[22]	WO	uart2clkdis		UART2 模块时钟禁止控制。																																			
[21]	WO	usbclkdis		USB 1.1 HOST 模块时钟禁止控制。																																			
[20]	WO	otgclkdis		USB 2.0 OTG 模块时钟禁止控制。																																			
[19]	WO	sio1clkdis		SIO1 模块 AHB 时钟禁止控制。																																			
[18]	WO	sio0clkdis		SIO0 模块 AHB 时钟禁止控制。																																			
[17]	WO	ethclkdis		ETH 模块时钟禁止控制。																																			
[16]	WO	vi3clkdis		VI3 模块接口时钟禁止控制。																																			
[15]	WO	vi2clkdis		VI2 模块接口时钟禁止控制。																																			
[14]	WO	vi1clkdis		VI1 模块接口时钟禁止控制。																																			
[13]	WO	vi0clkdis		VI0 模块接口时钟禁止控制。																																			
[12]	WO	vihclkdis		VI 模块 AHB 时钟禁止控制。																																			
[11]	WO	voclkdis		VO 时钟禁止控制。																																			

[10]	WO	mmccclkdis	MMC 模块时钟禁止控制。
[9]	WO	pciclkdis	PCI 模块时钟禁止控制。
[8]	WO	irclkdis	IR 模块时钟禁止控制。
[7]	WO	sspclkdis	SSP 模块时钟禁止控制。
[6]	WO	uart1clkdis	UART1 模块时钟禁止控制。
[5]	WO	uart0clkdis	UART0 模块时钟禁止控制。
[4]	WO	smiclkdis	SMI 模块时钟禁止控制。
[3]	WO	dsuclkdis	DSU 模块时钟禁止控制。
[2]	WO	cipherclkdis	CIPHER 模块时钟禁止控制。
[1]	-	reserved	保留。
[0]	WO	veduclkdis	VEDU 模块时钟禁止控制。

SC_PERCLKEN

该只读寄存器是用于读出系统控制器内对各模块时钟使能的状态以检验写寄存器 SC_PEREN 和 SC_PERDIS 的操作是否生效。某位读为 0 表示相应的模块时钟关闭；某位读为 1 则表示相应的模块时钟打开。

	Offset Address	Register Name	Total Reset Value
Bit	0x02C	SC_PERCLKEN	0xFFFF_FFFF
Name	reserved		veduclklen reserved uart0clklen smiclklen dsuclklen cipherclklen veduclklen
Reset	1 1		veduclklen reserved uart0clklen smiclklen dsuclklen cipherclklen veduclklen
Bits	Access	Name	Description
[31:23]	-	reserved	保留。
[22]	RO	uart2clklen	UART2 模块时钟状态。
[21]	RO	usbclkden	USB 1.1 HOST 模块时钟状态。
[20]	RO	otgclklen	USB 2.0 OTG 模块时钟状态。
[19]	RO	sio1clklen	SIO1 模块 AHB 时钟状态。
[18]	RO	sio0clklen	SIO0 模块 AHB 时钟状态。
[17]	RO	ethclklen	ETH 模块时钟状态。



[16]	RO	vi3clken	VI3 模块接口时钟状态。
[15]	RO	vi2clken	VI2 模块接口时钟状态。
[14]	RO	vi1clken	VI1 模块接口时钟状态。
[13]	RO	vi0clken	VI0 模块接口时钟状态。
[12]	RO	vihclken	VI 模块 AHB 时钟状态。
[11]	RO	voclken	VO 时钟状态。
[10]	RO	mmcclken	MMC 模块时钟状态。
[9]	RO	pciclken	PCI 模块时钟状态。
[8]	RO	irclken	IR 模块时钟状态。
[7]	RO	sspclken	SSP 模块时钟状态。
[6]	RO	uart1clken	UART1 模块时钟状态。
[5]	RO	uart0clken	UART0 模块时钟状态。
[4]	RO	smiclken	SMI 模块时钟状态。
[3]	RO	dsuclken	DSU 模块时钟状态。
[2]	RO	cipherclken	CIPHER 模块时钟状态。
[1]	-	reserved	保留。
[0]	RO	veduclken	VEDU 模块时钟状态。

SC_PERCTRL2

SC PERCTRL2 为时钟控制和分频寄存器，控制相关模块的时钟分频和选择。

Offset Address								Register Name								Total Reset Value															
Bit	0x034								SC_PERCTRL2								0x0000_0000														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Name	reserved				pciclk_sel				vi3div_sel				vi1div_sel				reserved				mmeclk_sel				vi1_vio_sel						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access		Name				Description																								
[31:27]	-		reserved				保留。																								

[26]	RW	vedu_power_mode	VEDU 时钟门控低功耗模式控制。 0: VEDU 时钟门控正常模式。 1: VEDU 时钟门控低功耗模式。
[25]	RW	otg24_sel	USB 2.0 OTG 24MHz 时钟源选择。 0: 管脚输入的 24MHz 时钟。 1: 内部 PLL 分频产生 24MHz 时钟。
[24]	RW	smiclk_sel	SMI 模块工作时钟控制, 提供 AHB 总线的分频选择。 0: 1 分频。 1: 2 分频。
[23:20]	RW	pciclk_sel	PCI 模块的工作时钟控制, 提供 PLL 输出频率的 6 到 24 分频选择。 0000: 6 分频。 0001: 7 分频。 0010: 8 分频。 0011: 9 分频。 0100: 10 分频。 0101: 11 分频。 0110: 12 分频。 0111: 13 分频。 1000: 14 分频。 1001: 15 分频。 1010: 16 分频。 1011: 18 分频。 1100: 20 分频。 1101: 24 分频。 1110~1111: 保留。
[19:18]	RW	vi3div_sel	VI3 分频时钟控制。 00: VI3 端口时钟的 2 分频。 01: VI3 端口时钟的 4 分频。 10: VI3 端口时钟。 11: 保留。
17:16]	RW	vi2div_sel	VI2 分频时钟控制。 00: VI2 端口时钟的 2 分频。 01: VI2 端口时钟的 4 分频。 10: VI2 端口时钟。 11: 保留。



[15:14]	RW	vi1div_sel	VI1 分频时钟控制。 00: VI1 端口时钟的 2 分频。 01: VI1 端口时钟的 4 分频。 10: VI1 端口时钟。 11: 保留。
[13:12]	RW	vi0div_sel	VI0 分频时钟控制。 00: VI0 端口时钟的 2 分频。 01: VI0 端口时钟的 4 分频。 10: VI0 端口时钟。 11: 保留。
[11:10]	-	reserved	保留。
[9:8]	RW	usb48clk_sel	USB 1.1 HOST 48MHz 时钟频率控制。 00: 保留。 01: PLL 输出时钟的 10 分频。 10: PLL 输出时钟的 11 分频。 11: PLL 输出时钟的 12 分频。 注意: USB 1.1 HOST 的时钟频率必须为 48MHz。如果要得到准确的 48MHz 的时钟, PLL 输出时钟的频率必须为 48MHz 的整数倍。
[7]	RW	arm_hclk_sel	ARM HCLK 频率比控制。 0: ARM:HCLK =2:1。 1: ARM:HCLK =1:1。
[6]	RW	mmcsap_sel	MMC 模块采样卡数据的时钟正反相控制。 0: 采样时钟使用正向时钟。 1: 采样时钟使用反向时钟。
[5:4]	RW	mmcclk_sel	MMC 模块工作时钟频率控制。 00: PLL 输出时钟的 8 分频。 01: PLL 输出时钟的 10 分频。 10: PLL 输出时钟的 12 分频。 11: PLL 输出时钟的 14 分频。
[3]	RW	vi3_vi2_sel	VI3 端口时钟选择。 0: VI3 端口输入的时钟。 1: VI2 端口输入的时钟。
[2]	RW	vi2_vi0_sel	VI2 端口时钟选择。 0: VI2 端口输入的时钟。 1: VI0 端口输入的时钟。

[1]	RW	vi1_vi0_sel	VI1 端口时钟选择。 0: VI1 端口输入的时钟。 1: VI0 端口输入的时钟。
[0]	RW	voout_sel	VO 输出时钟正反相控制。 0: VO 输出反相时钟。 1: VO 输出正向时钟。

SC_PERCTRL3

SC_PERCTRL3 为 SIO 分频控制寄存器，配置该寄存器设置适当的 SIO 各时钟。

	Offset Address								Register Name								Total Reset Value															
Bit	0x038								SC_PERCTRL3								0x0000_0000															
Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access		Name				Description																									
[31:30]	-		reserved				保留。																									
[29:27]	RW		siolrclk_sel				SIO 采样时钟频率控制，提供 SIO 位流时钟的 2、4、8、16、32、48、64 分频选择。 000: 2 分频。 001: 4 分频。 010: 8 分频。 011: 16 分频。 100: 32 分频。 101: 48 分频。 110: 64 分频。 111: 保留。																									



[26:24]	RW	siobclk_sel	<p>SIO 位流时钟频率控制，提供 SIO 系统时钟的 1、2、4、8、16、32 分频选择。</p> <p>000: 1 分频。</p> <p>001: 2 分频。</p> <p>010: 4 分频。</p> <p>011: 8 分频。</p> <p>100: 16 分频。</p> <p>101: 32 分频。</p> <p>110、111: 保留。</p>
[23:0]	RW	sioclk_sel	<p>SIO 系统时钟频率控制，提供 PLL 输出时钟的任意分频选择。</p> <p>$K = \text{SC_PERCTRL3}[23:0]$</p> $F_{sio} = \frac{K}{2^{27}} F_{pll}$

SC_PERCTRL4

SC_PERCTRL4 为外设控制寄存器 4 (芯片工作模式控制)。

[26]	RW	rts_mode	UART1 的 RTS 信号输出模式。 0: 正常输出。 1: 取反输出。
[25]	-	reserved	保留。
[24]	RW	otg_for_srp	USB 2.0 OTG 选择是否使用电源检测 SRP 方案。 0: 不选择 SRP 方案。 1: 选择 SRP 方案。
[23:21]	RW	otg_for_glitch	USB 2.0 OTG 的工作模式。 000: 不去抖, 保持原来的工作模式。 001: 3us, 去除电压上升过程中的抖动。 010: 10us, 去除电压上升过程中的抖动。 011: 30us, 去除电压上升过程中的抖动。 100: 100us, 可以去除一些电源不稳的抖动。 101: 300us, 可以去除一些电源不稳的抖动。 110: 1ms, 基本消除电容充放电引起的抖动。 111: 3ms, 基本消除电容充放电引起的抖动。
[20]	RW	pciclk_mode	PCI 时钟模式。 0: 输入。 1: 输出。
[19:18]	-	reserved	保留。
[17]	RW	pci_bridge_mode	PCI 桥模式。 0: Device。 1: Host。
[16]	RW	pci_sim_fast	PCI 快速仿真选择。 0: 正常工作。 1: 快速仿真。
[15]	RW	uart1_mode	UART1 模式。 0: 普通串口。 1: 红外数据串口。
[14:10]	-	reserved	保留。
[9]	RW	sio1_clksel	SIO1 模块 SIO 时钟和同步信号选择。 0: SIO 模块内部分频和外部输入。 1: 内部 CRG 分频产生。
[8]	-	reserved	保留。



[7]	RW	sio0_clksel	SIO0 模块 SIO 时钟和同步信号选择。 0: SIO 模块内部分频和外部输入。 1: 内部 CRG 分频产生。
[6]	RW	sm_cancel_wait	SMI 模块的外部等待打断信号控制。 0: 正常等待。 1: 强行打断 SMI 接口操作。
[5]	-	reserved	保留。
[4:1]	RW	otg_tx_tune	USB 2.0 OTG PHY 发送信号电压和上升沿控制。 otg_tx_tune[1:0]表示 DP DM 高速电路的电源供电情况。 00: 缺省设计工作电压。 01: 缺省值减少 4.5%。 10: 缺省值增加 9%。 11: 缺省值增加 4.5%。 otg_tx_tune[3:2]表示 DP DM 高速信号的上升沿和下降沿的陡缓程度。 00: 设计缺省值。 01: 比缺省值快。 10: 比 01 表示的值快。 11: 比缺省值慢。
[0]	RW	i2c_delay_bypass	I ² C 信号延迟旁路指示。 0: I ² C 信号经过 delay 300ns。 1: I ² C 信号不经过 delay 300ns。

SC_PERLOCK

SC_PERLOCK 为关键系统控制寄存器的锁定寄存器。



SC_SYSID

SC_SYSID 由 SCSYSID0、SCSYSID1、SCSYSID2、SCSYSID3 寄存器组成，为芯片标识寄存器。



SCSYSID0[7:0]和SCSYSID1[7:0]共同表示芯片完整的版本号信息。

Offset Address				Register Name			Total Reset Value	
Bit	7	6	5	4	3	2	1	0
Name	sysid0							
Reset	0	0	0	0	0	0	0	0
Bits	Access	Name	Description					
[7:0]	RO	sysid0	读该寄存器返回 0xXX (XX 表示芯片版本号的低位部分)。					



Offset Address								Register Name	Total Reset Value
0xEE4								SCSYSID1	0xXX
Bit	7	6	5	4	3	2	1	0	
Name	sysid1								
Reset	0	0	0	0	0	0	0	1	
Bits	Access	Name	Description						
[7:0]	RO	sysid1	读该寄存器返回 0xXX (XX 表示芯片版本号的高位部分)。						

Offset Address								Register Name	Total Reset Value
0xEE8								SCSYSID2	0x11
Bit	7	6	5	4	3	2	1	0	
Name	sysid2								
Reset	0	0	0	1	0	0	0	1	
Bits	Access	Name	Description						
[7:0]	RO	sysid2	读该寄存器返回 0x11。						

Offset Address								Register Name	Total Reset Value
0xEEC								SCSYSID3	0x35
Bit	7	6	5	4	3	2	1	0	
Name	sysid3								
Reset	0	0	1	1	0	1	0	1	
Bits	Access	Name	Description						
7:0	RO	sysid3	读该寄存器返回 0x35。						

3.11 电源管理与低功耗模式控制

3.11.1 概述

芯片的低功耗模式用来有效的减少芯片的功耗，Hi3511/Hi3512 提供多种低功耗的控制来动态降低芯片的功耗：

- 系统工作模式控制



除了 NORMAL 模式之外，各种模式对功耗都有一定的减小作用，可以根据实际的功耗要求和功能要求选择不同的工作模式。

- 时钟门控和时钟频率调整

提供时钟关断功能，可以关闭没有必要的时钟，减少芯片的功耗。系统工作的时钟频率可以进行调整，在满足功能的情况下可以调节时钟频率，动态降低芯片功耗。

- 模块级低功耗控制

提供模块级的低功耗控制，可以在某模块不工作的情况下，关断该模块或使模块处于低功耗状态，以减少芯片的功耗。

- DDR 低功耗控制

DDR 的控制器和相关的管脚进行了动态的功耗控制功能，可以选择启动该功能，降低芯片功耗，还可以启动 DDR 的自刷新模式，来降低整个产品的功耗。

3.11.2 系统工作模式

系统提供四种工作模式，请参见“[3.10.3 功能描述](#)”中的“[系统运行模式控制](#)”。

3.11.3 时钟门控和时钟频率调整

系统提供以下模块的时钟门控功能，在模块空闲的时候，可以关闭相应的时钟，降低芯片功耗，操作流程可以参见以下模块时钟门控部分。

- VEDU
- PCI
- MMC
- VIU
- VOU
- DSU
- CIPHER
- ETH
- SIO
- UART
- SMI
- USB 2.0 OTG
- USB 1.1 HOST
- SSP
- IR

Normal 模式下，系统可以通过调整工作频率来降低芯片功耗，步骤如下：

步骤 1 关闭业务模块，使其不访问 DDR。

步骤 2 系统进入 Flash 或者 TCM (Tightly-Coupled Memory) 中运行。

步骤 3 配置 DDRC_DLL_CONFIG[dll_cali_en]为 1，重新校准 DLL。

步骤 4 配置 DDRC_CTRL[sr_req]为 1，请求进入自刷新模式。



- 步骤 5 查询 DDRC_STATUS[in_sr]位，直到其值为 1，则执行步骤 6。
- 步骤 6 配置 SC_PLLCTRL[27:3]为 PLL 的稳定时间，并配置 SC_PLLCTRL[1]为 1 选择 PLL。
- 步骤 7 配置 SC_PLLFCTRL，进行 PLL 分频比控制。切换 PLL 的时钟，即等待 PLL 时钟稳定，输出给 DDRC。
- 步骤 8 配置 DDRC_CTRL[sr_req]为 0，请求退出自刷新模式。
- 步骤 9 查询 DDRC_STATUS[in_sr]位，直到其值为 0。
- 步骤 10 配置 DDRC_DLL_CONFIG[dll_cali_en]为 0，禁止 DLL (Delay Locked Loop) 的重校准。
- 步骤 11 程序运行在 DDR 中，并且启用业务模块工作。

----结束

除了提供系统工作频率的调整外，还提供了以下模块的工作频率可单独调整的功能。调整这些模块的工作频率，也可以进一步降低系统的功耗。具体的操作参见以下模块时钟配置部分。

- PCI
- SMI
- MMC

3.11.4 模块级低功耗控制

芯片中的 USB 2.0 OTG 模块、VEDU、PLL 都提供低功耗的工作模式。

USB 2.0 OTG 模块的低功耗工作状态请参见“10.2 USB 2.0 OTG”的描述。

VEDU 低功耗模式配置方法如表 3-28 所示。

表3-28 VEDU 时钟低功耗模式配置

信号名	描述
vedu_power_mode	VEDU 时钟门控低功耗模式控制。低功耗模式使能之后，系统根据 VEDU 工作模式自动关断一些不工作模块的时钟，可通过配置 SC_PERCTRL2[26]控制该信号。

PLL 也提供低功耗的功能，如果不使用 PLL 的情况下可以关闭 PLL，使系统处于低功耗状态：如果不需要使用 ARMPPLL，则可以通过配置 SC_PLLCTRL[1]=0 禁止 ARMPPLL，使系统处于低功耗状态。

3.11.5 DDR 低功耗控制

提供对芯片中 DDR 控制器的管脚以及对端 DDR 芯片的动态功耗控制：

- 可通过配置 DDRC_CONFIG[pd_en]和 DDRC_CONFIG[pd_prd]，使能 DDR 低功耗自动进入功能，在总线没有访问时，DDRC 控制 DDR 自动进入低功耗模式，节省功耗。
- 可通过配置 DDRC_CTRL[sr_req]为 1，控制 DDR 进入自刷新模式，降低功耗。在要求进入自刷新模式时，需要满足下面的要求：系统在启动 DDR 的自刷新功能的时候，必须将程序放到片内的程序存储器或 flash 中进行运行，通过 DDR 控制器启动 DDR 的自刷新功能。
- 为了降低芯片中 DDR 管脚的功耗，可以配置 DDRC 中的寄存器，关闭 DDR 的管脚，以降低 IO 的功耗。
 - DDRC_PHY_CONFIG[pwr_ctrl]配置为 1，DDRC 将动态控制 DDRC 地址、数据等管脚的输出关闭。当芯片为 16bit 外部总线连接模式，DDRC 将关闭高 16bit 的数据总线输出，关闭高 2bit 的 DDRDQS 和 DDRDQM 输出。
 - DDRC_PHY_CONFIG[cmdpwr_ctrl]配置为 1，DDRC 将动态控制 DDRC 输出的 DDRASN、DDRCASN、DDRWEN 管脚的关闭和打开。



4 存储控制器

关于本章

本章描述内容如下表所示。

标题	内容
4.1 DDR 控制器	介绍 DDR 控制器的功能、特点、信号描述、功能描述、寄存器概览和寄存器描述。
4.2 SMI 控制器	介绍 SMI 控制器的功能、特点、信号描述、功能描述、寄存器概览和寄存器描述。



4.1 DDR 控制器

4.1.1 概述

DDRC (DDR2 SDRAM Controller) 对外提供 DDR2 接口, 用于完成 DDR2 SDRAM 的访问。

4.1.2 特点

DDRC 有如下特点:

- 提供 1 个片选的 DDR2 接口, 兼容数据位宽为 16bit 和 32bit 的 DDR2 SDRAM。
- 16bit DDR2 SDRAM 最大支持 256MB 存储器空间; 32bit DDR2 SDRAM 最大支持 512MB 存储器空间。
- 支持 DDR2 SDRAM 的 burst 为 4 的传输模式。
- 支持动态存储器的自刷新操作。
- 支持时序参数可配, 以适应不同器件不同频率的需求。
- 支持 CL (CAS Latency) 值可配置, 以适应不同器件的需求。
- 支持 DDR2 SDRAM 的工作频率 135MHz。

4.1.3 信号描述

DDRC 的接口信号描述如表 4-1 所示。

表4-1 DDRC 接口信号描述

信号名称	方向	描述	对应管脚
DDRCKP0	O	第 1 组正向差分时钟, 接 DDR2 SDRAM 0。	DDRCKP0
DDRCKN0	O	第 1 组反向差分时钟, 接 DDR2 SDRAM 0。	DDRCKN0
DDRCKP1	O	第 2 组正向差分时钟, 接 DDR2 SDRAM 1。	DDRCKP1
DDRCKN1	O	第 2 组反向差分时钟, 接 DDR2 SDRAM 1。	DDRCKN1
DDRCKE	O	DDR2 SDRAM 的时钟使能信号, 高电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。	DDRCKE0
DDRCSEN	O	输出到 DDR2 SDRAM 的片选信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。	DDRCSEN



信号名称	方向	描述	对应管脚
DDRRASN	O	输出到 DDR2 SDRAM 的行地址选通信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。	DDRRASN
DDRCASN	O	输出到 DDR2 SDRAM 的列地址选通信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。	DDRCASN
DDRWEN	O	输出到 DDR2 SDRAM 的写使能信号, 低电平有效。 该信号用来同时控制 DDR2 SDRAM 0 和 DDR2 SDRAM 1。	DDRWEN
DDRDM3	O	输出到 DDR2 SDRAM 1 的字节 Mask 信号, 对应数据总线 DDRDQ[31:24]。	DDRDM3
DDRDM2	O	输出到 DDR2 SDRAM 1 的字节 Mask 信号, 对应数据总线 DDRDQ[23:16]。	DDRDM2
DDRDM1	O	输出到 DDR2 SDRAM 0 的字节 Mask 信号, 对应数据总线 DDRDQ[15:8]。	DDRDM1
DDRDM0	O	输出到 DDR2 SDRAM 0 的字节 Mask 信号, 对应数据总线 DDRDQ[7:0]。	DDRDM0
DDRODT	O	输出到 DDR2 SDRAM 的终端匹配电阻使能信号。	DDRODT
DDRDQ[31:16]	I/O	DDR2 SDRAM 1 接口数据总线。	DDRDQ31 ~ DDRDQ16
DDRDQ[15:0]	I/O	DDR2 SDRAM 0 接口数据总线。	DDRDQ15 ~ DDRDQ0
DDRBA[2:0]	O	DDR2 SDRAM bank 选择信号。	DDRBA2 ~ DDRBA0
DDRADR[13:0]	O	DDR2 SDRAM 地址信号。	DDRADR13 ~ DDRADR0
DDRDQS3	I/O	DDR2 的 UDQS 信号, 对应数据总线 DDRDQ[31:24]。	DDRDQS3
DDRDQS2	I/O	DDR2 的 LDQS 信号, 对应数据总线 DDRDQ[23:16]。	DDRDQS2
DDRDQS1	I/O	DDR2 的 UDQS 信号, 对应数据总线 DDRDQ[15:8]。	DDRDQS1

信号名称	方向	描述	对应管脚
DDRDQS0	I/O	DDR2 的 LDQS 信号， 对应数据总线 DDRDQ[7:0]。	DDRDQS0

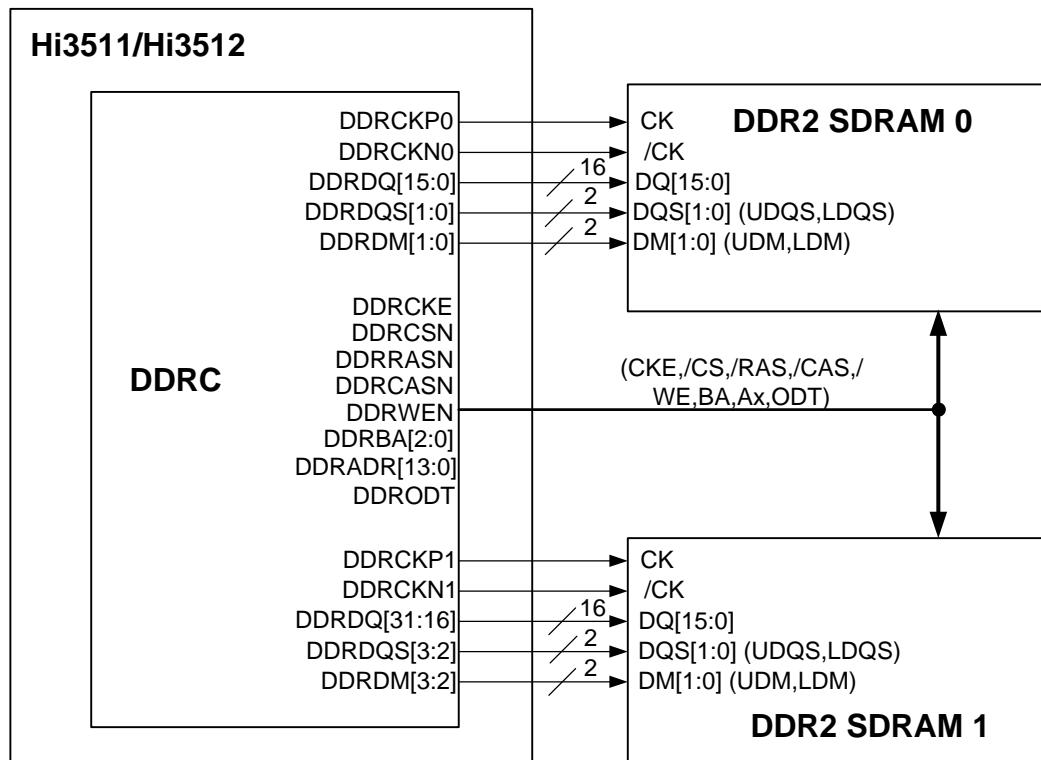
4.1.4 功能描述

应用框图

DDRC 用于实现内部系统总线与外部 DDR2 SDRAM 进行数据交换，它提供了可配置的 DDR 时序参数寄存器，可连接符合 JEDEC (JESD79E) 标准的 DDR2 SDRAM。DDRC 支持 2 种外部总线位宽的连接方式：32bit 模式和 16bit 模式。

32bit 外部总线连接方式如图 4-1 所示。

图4-1 DDRC 与 2 片 16bit DDR2 SDRAM 的连接示意图



连接说明：

16bit 的 DDR2 SDRAM 包含了 UDQS 和 LDQS，UDQS 对应图 4-1 中的 DDRDQS[1]，LDQS 对应图中的 DDRDQS[0]。DM 也一致。

DDRC 输出的控制信号组：DDRCKE、DDRCSE、DDRRASN、DDRCASN、DDRWEN、

DDRODT、DDRBA[2:0]、DDRADR[13:0]，同时连接 2 片 DDR2 SDRAM 的相应的控制信号。

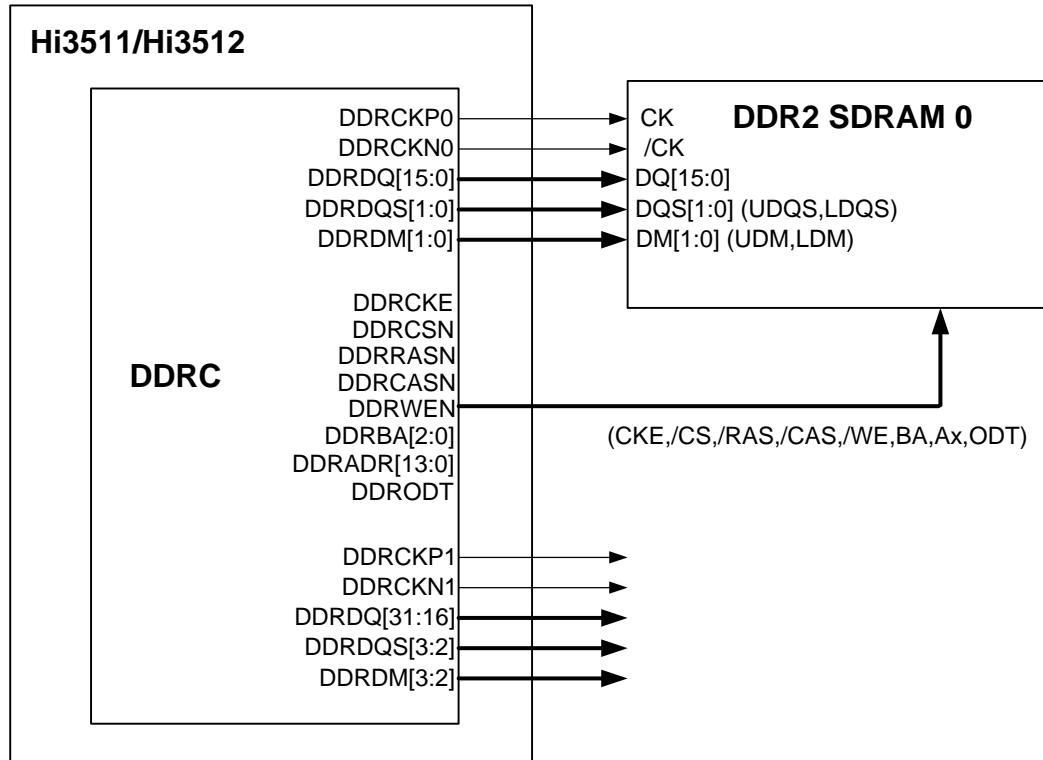
图 4-1 中 (CKE、/CS、/RAS、/CAS、/WE、BA、Ax、ODT) 表示 DDR2 SDRAM 的控制信号管脚。

小容量的 DDR2 SDRAM 没有 BA2 信号线，将该地址线悬空即可。



16bit 外部总线连接方式如图 4-2 所示。

图4-2 DDRC 与单片 16bit DDR2 SDRAM 的连接示意图



连接说明：

16bit 的 DDR SDRAM 包含了 UDQS 和 LDQS, UDQS 对应图 4-2 中的 DQS[1], LDQS 对应图中的 DQS[0]。DM 也一致。

DDRC 输出的控制信号组: DDRCKE, DDRCSN, DDRRASN, DDRCASN, DDRWEN, DDRBA[2:0]、DDRADR[13:0], 与 1 片 DDR SDRAM 的相应的控制信号相连。

图 4-2 中 (CKE, /CS, /RAS, /CAS, /WE, BA, Ax, ODT) 表示 DDR2 SDRAM 的控制信号管脚。

DDRC 的输出信号 DDRCKP1、DDRCKN1、DDRDQ[31:16]、DDRDQS[3:2]、DDRDM[3:2]不使用，悬空。

功能原理

DDRC 按照标准的 DDR2 SDRAM 的时序进行访问，在使用 DDRC 时，只需要根据实际使用的 DDR2 SDRAM 器件以及工作频率，配置 DDRC 的时序寄存器参数，DDRC 就可以正确访问 DDR2 SDRAM。

表 4-2 给出了 135MHz 时钟频率下，使用主流 DRAM 产商的 DDR2 SDRAM 器件时的寄存器配置值。TFAW 只在 1Gbit 容量及以上时才配置，示例中 TFAW 是在 DRAM 的页大小为 2K byte 的时序值。



表4-2 主流 DRAM 产商的 DDR2 SDRAM 器件的寄存器配置值 (135MHz)

时序参数	TRRD	TRCD	TRC	TWR	TRP	TFAW
DDRC 时序参数配置值	2	2	9	2	2	7
135MHz 对应的时间(ns)	15	15	67.5	15	15	52.5
micron(-3E)	10	12	54	15	12	-
micron(-3/-37/-37E/-5E)	12	15	55	15	15	-
micron(-25/-25E)	10	15	55	15	15	50
samsung(DDR2-667)	10	15	54	15	15	50
samsung(DDR2-533)	10	15	55	15	15	50
samsung(DDR2-400)	10	15	55	15	15	50
ELPIDA(GE DDR2 800)	7.5	12.5	57.5	15	12.5	-
ELPIDA(6E DDR2 667)	7.5	15	55	15	15	-
ELPIDA(5C DDR2 533)	7.5	15	60	15	15	-
ELPIDA(4A DDR2 400)	7.5	15	60	15	15	-
hynix(ddr667)	10	15	60	15	15	50
hynix(ddr533)	10	15	60	15	15	50
hynix(ddr400)	10	15	55	15	15	50

注：在 135MHz 情况下，配置的参数可以满足列出所有器件的需求。对于上表中没有列出的器件，其参数的配置，需要根据其器件手册上列出的参数值按照频率的大小转换成相应的周期数（这里需要向上取整），然后配置相应的寄存器。

时序参数 TRRD、TRCD、TRC、TWR、TRP 具体含义在 DDR SDRAM 器件的手册中有详细说明，本文不再介绍。在使用大于 1Gbit 容量的 DDR2 SDRAM 时，需要配置 TFAW 时序参数。表 4-2 列出了总线频率在 135MHz 时器件支持情况。如果系统不是工作在 135MHz，可根据实际需要，计算出相应频率对应的时间，然后配置相应的参数寄存器。

DDRC 支持 JEDEC 标准中 DDR2 的大多数命令。DDRC 支持的命令真值表如表 4-3 所示。

表4-3 DDRC 命令真值表

FUNCTION	DDR CKE	DDR CSN	DDR RASN	DDR CASN	DDR WEN	DDRADR			DDRBA
						11	AP(10)	9:0	
DESELECT	H	H	X	X	X	X	X	X	X
NOP	H	L	H	H	H	X	X	X	X
ACTIVE	H	L	L	H	H	V	V	V	V



FUNCTION	DDR CKE	DDR CSN	DDR RASN	DDR CASN	DDR WEN	DDRADR			DDRBA
						11	AP(10)	9:0	
READ	H	L	H	L	H	V	V	V	V
WRITE	H	L	H	L	L	V	V	V	V
PRECHARGE	H	L	L	H	L	X	L	X	V
PRECHARGE ALL	H	L	L	H	L	X	H	X	X
AUTO REFRESH	H	L	L	L	H	X	X	X	X
SELF REFRESH	L	L	L	L	H	X	X	X	X
MODE REGISTER SET	H	L	L	L	L	V	V	V	V

注 1: DDRADR 中的 AP 位用于区分当前的 PRECHARGE 命令为单 BANK 的 PRECHARGE 还是所有 BANK 的 PRECHARGE。对于 AP 的详细定义可参见 DDR 的 JEDEC 标准。

注 2: H 表示高电平; L 表示低电平; V 表示有效; X 表示不关心。

注 3: DDRC 只支持 DDR2 的 Burst Length 为 4 的工作方式, 不支持 DDR2 的 BURST TERMINATE 的操作。

注 4: DDRC 可以支持 A10 或者 A8 作为 AP 位。

4.1.5 工作方式

DDR2 初始化

系统上电之后, 需要对 DDR2 SDRAM 进行一系列的初始化操作, 保证系统能正确访问 DDR2 SDRAM。在进行初始化之前需要注意以下几点:

- 对 DDR2 SDRAM 进行上电操作时, 需要遵循 JEDEC 标准。即先提供 VDD, 然后提供 VDDQ, 最后提供 VREF 和 VTT。
- 该初始化过程需要在系统进入 NORMAL 模式后进行。

以存储结构为 2 片单片容量为 512Mbit、16bit 位宽的器件拼成 32bit 存储空间为例, DDR 的初始化步骤如下:

步骤 1 软件等待 200us 以上。

步骤 2 把 [DDRC_CTRL](#) 寄存器配置为 0x0000_0000, 退出自刷新状态。

步骤 3 软件等待 400ns 以上。

步骤 4 把 [DDRC_EMRS01](#) 寄存器设为 0x32, 把 [DDRC_EMRS23](#) 寄存器设为 0x00, 配置 DDR 器件的模式寄存器和扩展模式寄存器。这里把 DDR 的读延迟 (Cas Latency) 设

为 3, 突发长度 (Burst Length) 设定为 4。扩展模式寄存器根据实际需要, 可设定 DDR 器件的一些功能, 这里可设置为 0。

说明

- Cas Latency 必须和 **DDRC_TIMING1[tcl]** 设置成同一个值。
- **DDRC_EMRS01** 寄存器对应 DDR2 SDRAM 的模式寄存器 (MRS) 和扩展模式寄存器 1 (EMRS1)。配置该寄存器时, 只需要参考 DDR2 SDRAM 器件手册的模式寄存器的 A15-A0 的描述, 不需要配置模式寄存器的最高 3 位寄存器选择位, 即 bank 地址。
- DDR2 的扩展模式寄存器 1 (EMRS1) 需要特别注意: 将该模式寄存器中的 RDQS 和 **DQS** 都配置为禁止模式。

步骤 5 根据器件的行列地址位宽配置 **DDRC_CONFIG** 寄存器为 0x7000_7022, 表示器件的地址映射模式为 R-B-C-DW 模式, AP 为 A10, 列地址宽度为 10, 行地址宽度为 13, 初始化进行 7 次自动刷新操作 (AUTO REFRESH)。**DDRC_CONFIG[pd_en]** 采用上电复位值。

说明

低功耗配置默认不使能, 初始化过程中必须保证低功耗配置处理不使能状态, 在 DDRC 的正常模式下, 建议使能低功耗配置, 降低功耗。

步骤 6 根据系统需求配置 **DDRC_TIMING0**、**DDRC_TIMING1**、**DDRC_TIMING2** 和 **DDRC_TIMING3** 寄存器的值, 其中 tcl 值必须与 **DDRC_EMRS01** 模式寄存器中的配置一致。

步骤 7 根据系统需求配置 **DDRC_ODT_CONFIG** 寄存器。**注意:** 在普通模式下, 建议使用默认值, 即在对 DDR2 进行写操作时, odt 有效; 读操作时, odt 无效。

步骤 8 根据系统需求配置 **DDRC_DLL_CONFIG** 寄存器。建议使用默认值。

步骤 9 根据系统需求配置不同端口的优先级、QOS 值和 QOS 使能位。建议 QOS 值配置时大于 1, QOS 的配置范围为 0x2~0x3FF。

步骤 10 将 **DDRC_CTRL** 寄存器设置为 0x2, 启动初始化过程。

步骤 11 循环等待 **DDRC_STATUS[in_init]** 的值变为 1, 表示初始化完成。

----结束

完成以上步骤以后, DDR2 可以正常工作。

时钟门控

系统进入低功耗模式, 关闭 DDRC 的时钟, 需要按照一定的步骤进行操作, 以确保系统能正常关闭时钟和唤醒时钟。

关闭 DDRC 时钟步骤如下:

步骤 1 关闭业务模块, 使其不访问 DDR2。

步骤 2 系统进入 Flash 或者 TCM 中运行。

步骤 3 配置 **DDRC_CONFIG[sr_cc]** 为 1, 使能器件时钟控制。

步骤 4 配置 **DDRC_CTRL[sr_req]** 为 1, 请求进入自刷新模式。



步骤 5 查询 **DDRC_STATUS[in_sr]** 位，直到其值为 1。

步骤 6 关断 DDRC 时钟。

----结束

打开 DDRC 时钟步骤如下：

步骤 1 系统进入 NORMAL 模式，打开 DDRC 时钟。

步骤 2 配置 **DDRC_DLL_CONFIG[dll_cali_en]** 为 1，重新校准 DLL。

步骤 3 配置 **DDRC_CTRL[sr_req]** 为 0，请求退出自刷新模式。

步骤 4 查询 **DDRC_STATUS[in_sr]** 位，直到其值为 0，则执行步骤 5。

步骤 5 配置 **DDRC_DLL_CONFIG[dll_cali_en]** 为 0，禁止重新校准 DLL。

步骤 6 进入正常工作模式。

----结束

时钟频率切换

DDRC 内部的 DLL 用于延迟时钟以处理 DDR2 数据总线的采样。时钟频率切换时，为了让 DDRC 内部的 DLL 能跟上时钟的变换，需要按照下面的步骤来进行操作：

步骤 1 配置 **DDRC_CONFIG[sr_cc]** 为 1，使能器件时钟控制。

步骤 2 配置 **DDRC_DLL_CONFIG[dll_cali_en]** 为 1，重新校准 DLL。

步骤 3 配置 **DDRC_CTRL[sr_req]** 为 1，请求进入自刷新模式。

步骤 4 查询 **DDRC_STATUS[in_sr]** 位，直到其值为 1，则执行步骤 4。

步骤 5 切换 PLL 的时钟，并在新频率下等待 1000 个时钟周期。

步骤 6 配置 **DDRC_CTRL[sr_req]** 为 0，请求退出自刷新模式。

步骤 7 查询 **DDRC_STATUS[in_sr]** 位，直到其值为 0。

步骤 8 配置 **DDRC_DLL_CONFIG[dll_cali_en]** 为 0，禁止重新校准 DLL。

步骤 9 进入正常工作模式。

----结束

软复位

DDRC 不能进行单独的复位操作。只有在全局软复位时，才能复位 DDRC。复位之后，需要按照 **DDR2 初始化** 操作进行重新初始化。

4.1.6 寄存器概览

DDRC 寄存器概览如表 4-4 所示。DDRC 的可访问的存储地址范围为：0xE000_0000～0xFFFF_FFFF。



表4-4 DDRC 寄存器概览（基址是 0x1015_0000）

偏移地址	名称	描述	页码
0x000	DDRC_STATUS	DDRC 状态寄存器	4-11
0x004	DDRC_CTRL	DDRC 控制寄存器	4-11
0x008	DDRC_EMRS01	DDR2 SDRAM 模式寄存器 01	4-12
0x00C	DDRC_EMRS23	DDR2 SDRAM 模式寄存器 23	4-13
0x010	DDRC_CONFIG	DDRC 配置寄存器	4-13
0x014 ~ 0x01C	RESERVED	保留	-
0x020	DDRC_TIMING0	DDR2 SDRAM 时序参数寄存器 0	4-17
0x024	DDRC_TIMING1	DDR2 SDRAM 时序参数寄存器 1	4-18
0x028	DDRC_TIMING2	DDR2 SDRAM 时序参数寄存器 2	4-19
0x02C	DDRC_TIMING3	DDR2 SDRAM 时序参数寄存器 3	4-20
0x030 ~ 0x03C	RESERVED	保留	-
0x040	DDRC_ODT_CONFIG	DDR2 SDRAM ODT 使能寄存器	4-21
0x044 ~ 0x060	RESERVED	保留	-
0x078	DDRC_DLL_STATUS	DLL 状态寄存器	4-22
0x07C	DDRC_DLL_CONFIG	DLL 配置寄存器	4-23
0x090	RESERVED	保留	-
0x094	RESERVED	保留	-
0x098	DDRC_CH0_QOS	DDRC Channel0 QOS 寄存器	4-23
0x09C	DDRC_CH1_QOS	DDRC Channel1 QOS 寄存器	4-23
0x0A0	DDRC_CH2_QOS	DDRC Channel2 QOS 寄存器	4-23
0x0A4	DDRC_CH3_QOS	DDRC Channel3 QOS 寄存器	4-23
0x0A8	DDRC_CH4_QOS	DDRC Channel4 QOS 寄存器	4-23
0x0AC	DDRC_CH5_QOS	DDRC Channel5 QOS 寄存器	4-23
0x0B0	DDRC_CH6_QOS	DDRC Channel6 QOS 寄存器	4-23
0x0B4	DDRC_CH7_QOS	DDRC Channel7 QOS 寄存器	4-23



偏移地址	名称	描述	页码
0x0B8 ~ 0xFFF	RESERVED	保留	-

4.1.7 寄存器描述

寄存器的描述中，“时钟周期”的描述都是指总线时钟周期。

DDRC_STATUS

DDRC_STATUS 为 DDRC 状态寄存器。

Offset	Address	Register Name	Total Reset Value
	0x000	DDRC_STATUS	0x0000_000D
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	in_init in_sr reserved busy
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1		
Bits	Access	Name	Description
[31:4]	-	reserved	保留。
[3]	RO	in_init	初始化状态指示。 0: 初始化状态。 1: 正常工作状态。
[2]	RO	in_sr	自刷新 (SELF REFRESH) 状态指示。 0: 正常工作状态。 1: 自刷新状态。
[1]	-	reserved	保留。
[0]	RW	busy	DDRC 状态指示。 0: DDRC 空闲状态。 1: DDRC 正在执行命令状态。

DDRC_CTRL

DDRC_CTRL 为 DDRC 控制寄存器。

Offset Address																Register Name	Total Reset Value															
0x004																DDRC_CTRL	0x0000_0001															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																									init_req	sr_req					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
Bits	Access	Name	Description																													
[31:2]	-	reserved		保留。																												
[1]	RW	init_req	硬件初始化启动控制。 0: 正常工作状态。 1: 启动硬件初始化, 在初始化完成之后, 该位自动清零。																													
[0]	RW	sr_req		自刷新 (SELF REFRESH) 请求控制。 0: 正常工作状态, 或者当 DDRC 处于自刷新状态, 配置该位为 0 表示请求退出自刷新状态。 1: 请求进入自刷新状态。																												

DDRC_EMRS01

DDRC_EMRS01 为 DDR2 SDRAM 模式寄存器 01。

Offset Address																Register Name	Total Reset Value															
0x008																DDRC_EMRS01	0x0000_0000															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	emrs1																mrs															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description																													
[31:16]	RW	emrs1	DDR2 SDRAM 扩展模式寄存器 1。																													
[15:0]	RW	mrs	DDR2 SDRAM 模式寄存器。																													



说明

在配置 DDR2 SDRAM 的模式寄存器 (MRS, EMRS1) 时, 只需要将 DDR2 SDRAM 的模式寄存器中低 16 位的数值配置到 [DDRC_EMRS01](#) 寄存器中。



DDRC_EMRS23

DDRC_EMRS23 为 DDR2 SDRAM 模式寄存器 23。

Offset Address																Register Name								Total Reset Value								
0x00C																DDRC_DDRC CONFIG								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	emrs3																emrs2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name				Description																										
[31:16]	RW	emrs3				DDR2 SDRAM 扩展模式寄存器 3。																										
[15:0]	RW	emrs2				DDR2 SDRAM 扩展模式寄存器 2。																										



说明

在配置 DDR2 SDRAM 的模式寄存器 (EMRS2, EMRS3) 时, 只需要将 DDR2 SDRAM 的模式寄存器中低 16 位的数值配置到 [DDRC_EMRS23](#) 寄存器中。

DDRC_CONFIG

DDRC_CONFIG 为 DDRC 配置寄存器。

Offset Address																Register Name								Total Reset Value												
0x010																DDRC_CONFIG								0x0000_7022												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Name	init_arefnum				pd_prd				reserved				sr_cc				reserved				pd_en				mem_type				reserved				ap			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	1	0	
Bits	Access	Name				Description																														
[31:28]	RW	init_arefnum				硬件初始化需要发出的 AUTO REFRESH 个数。 0000~0010: 2 个。 0011~1111: n 个时钟周期, n 表示对应的十进制值。																														

[27:20]	RW	pd_prd	低功耗进入等待周期数, 当控制器连续 pr_prd 个总线周期没有收到传输请求就控制 DRAM 进入低功耗模式。 0x00: 在 pd_en 有效的情况下, 1 个时钟周期没有收到请求将自动进入低功耗模式。 0x01~0xn: 在 pd_en 有效的情况下, n 个时钟周期没有收到请求将自动进入低功耗模式。 如 0x03: 3 个时钟周期没有收到请求, 自动进入低功耗模式。
[19]	-	reserved	保留。
[18]	RW	sr_cc	自刷新时 DDR2 时钟控制。 0: 在自刷新时送给器件 ck 和反相 ck。 1: 在自刷新时停止送给器件 ck 和反相 ck。
[17]	-	reserved	保留。
[16]	RW	pd_en	低功耗使能。 0: 不使能。 1: 使能。
[15:12]	RW	mem_type	外部存储器类型。 0110: 16 bit DDR2 SDRAM。 0111: 32 bit DDR2 SDRAM。 其他: 保留。
[11:10]	-	reserved	保留。
[9]	RW	ap	DDR2 AP 功能位选择。 0: A10 作为功能。 1: A8 作为功能。
[8]	RW	mem_map	DDR2 的地址映射方式。 0: Row、Bank、Col、DW。 1: Bank、Row、Col、DW。
[7]	RW	mem_bank	DDR2 BANK 选择。 0: 4 Bank DDR2。 1: 8 Bank DDR2。



[6:4]	RW	mem_row	单片 DDR2 SDRAM 的行地址宽度。 000: 11。 001: 12。 010: 13。 011: 14。 100: 15。 101: 16。 其他: 保留。
[3]	-	reserved	保留。
[2:0]	RW	mem_col	单片 DDR2 SDRAM 的列地址宽度。 000: 8。 001: 9。 010: 10。 011: 11。 100: 12。 其他: 保留。

在配置

[DDRC_CONFIG\[mem_map\]](#)/[DDRC_CONFIG\[mem_row\]](#)/[DDRC_CONFIG\[mem_col\]](#)位时, 只需要按照使用的 DDR2 SDRAM 器件手册上描述的行地址宽度和列地址宽度进行配置即可。DDRC 根据配置情况自动进行地址的映射, 将内部总线的地址转换为 DDR2 的地址。

下面示例说明内部总线地址和 DDR2 地址的关系, 假设内部总线地址为 BUS_ADR[28:0], DDR2 的地址为 DDR_ADR[13:0], 当作为行地址时, 其地址宽度为 DDR_ROW[x-1:0], 作为列地址时, 其地址宽度为 DDR_COL[y-1:0], DDR 的 BANK 地址为 DDR_BA[z-1:0], 外部总线的数据宽度为 DW, 此时地址映射关系为:

- [DDRC_CONFIG\[mem_map\]](#) 为 0b0 时, RBC 映射方式:
 $BUS_ADR[m-1:0] = \{DDR_ROW[x-1:0], DDR_BA[z-1:0], DDR_COL[y-1:0], DW\}$
- [DDRC_CONFIG\[mem_map\]](#) 为 0b1 时, BRC 映射方式:
 $BUS_ADR[m-1:0] = \{DDR_BA[z-1:0], DDR_ROW[x-1:0], DDR_COL[y-1:0], DW\}$

在上面的表达式中, 参数的关系满足: $m = x + y + z + DW$ 。

当外部存储总线宽度为 16bit 模式时, DW 为 1; 当外部存储总线宽度为 32bit 模式时, DW 为 2。

下面以表格的方式描述了在 [DDRC_CONFIG\[mem_map\]](#) 为 0 时, A10 作为 DDR 的 AP 功能位时, 内部总线到 DDR2 地址总线的映射表。

外部存储总线宽度为 16bit 模式时, 地址映射如[表 4-5](#) 所示。外部存储总线宽度为 32bit 模式时, 地址的映射如[表 4-6](#) 所示。

表4-5 DDRC 地址映射表 (16bit 模式)

存储器类型	行地址 宽度	列地址 宽度	DDRBA			行地址 列地址	DDRADR						
			2	1	0		13	12	11	10/AP	9	8	[7:0]
256Mbit 4bank													
16 × 16	13	9	-	11	10	行地址	-	24	23	22	21	20	[19:12]
						列地址	-	-	-	AP	-	9	[8:1]
512Mbit 4bank													
32 × 16	13	10	-	12	11	行地址	-	25	24	23	22	21	[20:13]
						列地址	-	-	-	AP	10	9	[8:1]
1GMbit 8bank													
64 × 16	13	10	13	12	11	行地址	-	26	25	24	23	22	[21:14]
						列地址	-	-	-	AP	10	9	[8:1]
2GMbit 8bank													
128 × 16	14	9	12	11	10	行地址	26	25	24	23	22	21	[20:13]
						列地址	-	-	-	AP	-	9	[8:1]

表4-6 DDRC 地址映射表 (32bit 模式)

存储器类型	行地址 宽度	列地址 宽度	DDRBA			行地址 列地址	DDRADR						
			2	1	0		13	12	11	10/AP	9	8	[7:0]
256Mbit 4bank													
16 × 16	13	9	-	12	11	行地址	-	25	24	23	22	21	[20:13]
						列地址	-	-	-	AP	-	10	[9:2]
512Mbit 4bank													
32 × 16	13	10	-	13	12	行地址	-	26	25	24	23	22	[21:14]
						列地址	-	-	-	AP	11	10	[9:2]
1GMbit 8bank													
64 × 16	13	10	14	13	12	行地址	-	27	26	25	24	23	[22:15]
						列地址	-	-	-	AP	11	10	[9:2]
2GMbit 8bank													
128 × 16	14	9	13	12	11	行地址	28	27	26	25	24	23	[22:14]



存储器类型	行地址宽度	列地址宽度	DDRBA			行地址列地址	DDRADDR						
			2	1	0		13	12	11	10/AP	9	8	[7:0]
						列地址	-	-	-	AP	-	10	[9:2]



说明

DDRC 地址映射表中 AP 由 DDRC 自动产生，以 A10 或 A8 作为功能位，可以通过配置 [DDRC_CONFIG\[ap\]](#) 配置。

DDRC_TIMING0

DDRC_TIMING0 为 DDRC 时序参数寄存器 0。

Offset Address										Register Name										Total Reset Value												
0x020										DDRC_TIMING0										0x7FFF_3F1F												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved	tmrd		trrd		trp		trcd		reserved	trc		reserved		tras																	
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1			
Bits	Access	Name			Description																											
[31]	-	reserved			保留。																											
[30:28]	RW	tmrd			配置模式寄存器 (MODE REGISTER SET) 命令的等待延时。 000、001: 1 个时钟周期。 010~111: n 个时钟周期, n 表示对应的十进制值。 如 010: 2 个时钟周期。																											
[27:24]	RW	trrd			连续激活不同 bank 的等待延时。 0000、0001: 1 个时钟周期。 0010~1111: n 个时钟周期, n 表示对应的十进制值。 如 1111: 15 个时钟周期。																											
[23:20]	RW	trp			关闭 (PRECHARGE) 命令的等待延时。 0000、0001: 1 个时钟周期。 0010~1111: n 个时钟周期, n 表示对应的十进制值。 如 0111: 7 个时钟周期。																											

[19:16]	RW	trcd	同 bank 激活到读或写 (ACTIVE 到 READ 或 WRITE) 命令的等待延时。 0000、0001: 1 个时钟周期; 0010~1111: n 个时钟周期, n 表示对应的十进制值。 如 0011: 3 个时钟周期。
[15:14]	-	reserved	保留。
[13:8]	RW	trc	连续激活同 bank 不同行 (ACTIVE 到 ACTIVE) 的等待延时。 0x00、0x01: 1 个时钟周期。 0x02~0x3F: n 个时钟周期, n 表示对应的十进制值。 如 0x0F: 15 个时钟周期。
[7:5]	-	reserved	保留。
[4:0]	RW	tras	同 bank 激活到关闭 (ACTIVE 到 PRECHARGE) 命令的等待延时。 0x00、0x01: 1 个时钟周期。 0x02~0x1F: n 个时钟周期, n 表示对应的十进制值。 如 0x0F: 15 个时钟周期。

DDRC_TIMING1

DDRC_TIMING1 为 DDRC 时序参数寄存器 1。

	Offset Address								Register Name								Total Reset Value																
	0x024								DDRC_TIMING1								0xFF02_23FF																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	tsre								reserved				trl		twl		reserved		tcl		trfc												
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1		
Bits	Access	Name				Description																											
[31:24]	RW	tsre				退出自刷新 (SELF REFRESH) 命令到读命令的等待延时。 一般情况下器件要求 200 个时钟周期。 0x00~0xFF: n 个时钟周期, n 表示对应的十进制值。 如 0xFF: 255 个时钟周期。																											
[23:19]	-	reserved				保留。																											



[18:16]	RW	trl	读数据延迟，用于配置由于单板等外部环境因素引起的延时。在单板延时小于 1 个周期时，配置该值为 0b010，其他的情况下按照相应的延迟进行配置即可。 010: 3 个时钟周期（默认值）。 011: 4 个时钟周期。 101: 5 个时钟周期。 其他: 保留。
[15:12]	RW	twl	写命令到写数据的等待延时。 0000~1111: n 个时钟周期，n 表示对应的十进制值。 如 0011: 3 个时钟周期。 在 DDR2 模式下，twl 配置为 tcl - 1。 twl 配置时应满足 $twl - taond \geq 1$ 。
[11]	-	reserved	保留。
[10:8]	RW	tcl	DDR2 SDRAM 的读命令到读数据有效的延时值。 011: CL=3。 100: CL=4。 101: CL=5。 110: CL=6。 其他: 保留。
[7:0]	RW	trfc	自动刷新到激活（AUTO REFRESH 到 ACTIVE）命令的等待延时。 0x00、0x01: 1 个时钟周期。 0x02~0xFF: n 个时钟周期，n 表示对应的十进制值。 如 0x0F: 15 个时钟周期。

DDRC_TIMING2

DDRC_TIMING2 为 DDRC 时序参数寄存器 2。

Offset Address												Register Name												Total Reset Value											
0x028												DDRC_TIMING2												0x33F3_F7FF											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved	tcke	reserved	twtr	twr	reserved	tfaw	reserved	taref																										
Reset	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1					
Bits	Access	Name	Description																																
[31:30]	-	reserved	保留。																																
[29:28]	RW	tcke	DDR2 SDRAM 在低功耗模式下维持的最短时间。 00、01: 1 个时钟周期。 10、11: n 个时钟周期, n 表示对应的十进制值。																																
[27:26]	-	reserved	保留。																																
[25:24]	RW	twtr	写操作时最后一个写数据到读命令的等待延时。 00、01: 1 个时钟周期。 10、11: n 个时钟周期, n 表示对应的十进制值。 如 11: 3 个时钟周期。																																
[23:20]	RW	twr	写恢复的等待延时。 0000、0001: 1 个时钟周期。 0010~1111: n 个时钟周期, n 表示对应的十进制值。 如 0111: 7 个时钟周期。																																
[19:18]	-	reserved	保留。																																
[17:12]	RW	tfaw	连续 5 个激活命令的周期。 0x00~0x3F: n 个时钟周期。 如 0x14: 20 个时钟周期。																																
[11]	-	reserved	保留。																																
[10:0]	RW	taref	自动刷新周期。根据 SDRAM 实际工作频率决定。 0x00: 禁止自动刷新。 0x001~0x7FF: DDR2 SDRAM 刷新周期时间为 $16 \times n$ 时钟周期。n 表示对应的十进制值。 如 0x008: 128 个时钟周期 (16×8)。																																

DDRC_TIMING3

DDRC_TIMING3 为 DDRC 时序参数寄存器 3。



Offset Address								Register Name								Total Reset Value																	
0x02C								DDRC_TIMING3								0x0000_0F02																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved								tanod_taofd								reserved								txard		reserved				trtp		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0
Bits	Access	Name			Description																												
[31:22]	-	reserved			保留。																												
[21:20]	RW	tanod_taofd			DDR2 SDRAM 的终端电阻打开/关闭延迟（单位：时钟周期）。																												
					00: 2/2.5。 01: 3/3.5。 10: 4/4.5。 11: 5/5.5。																												
[19:12]	-	reserved			保留。																												
[11:8]	RW	txard			退出 DDR2 SDRAM 低功耗模式时等待的延迟延时（该参数为 txp、txard、txards 中的最大值）。																												
					0000、0001: 1 个时钟周期。 0010~1111: n 个时钟周期。 txp、txard、txards 为 DDR2 SDRAM 的时序参数，其定义参考 DDR2 SDRAM 的器件手册。																												
[7:3]	-	reserved			保留。																												
[2:0]	RW	trtp			读到关闭（PRECHARE）命令的等待延迟。																												
					000、010: 2 个时钟周期。 011~111: n 个时钟周期。 DDR2 的延迟计算公式为： AL + BL/2 + Max(trtp,2) - 2。																												

DDRC_ODT_CONFIG

DDRC_ODT_CONFIG 为 DDR2 SDRAM ODT 使能配置寄存器，建议使用默认值。

Offset Address																Register Name	Total Reset Value															
0x040																DDRC_ODT_CONFIG	0x0000_0001															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																rodt	wodt														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1				
Bits	Access	Name		Description																												
[31:2]	-	reserved		保留。																												
[1]	RW	rodt		读 ODT 使能。 0: 读时, 关闭 DDR2 SDRAM 芯片的 ODT。 1: 读时, 打开 DDR2 SDRAM 芯片的 ODT。																												
[0]	RW	wodt		写 ODT 使能。 0: 写时, 关闭 DDR2 SDRAM 芯片的 ODT。 1: 写时, 打开 DDR2 SDRAM 芯片的 ODT。																												

DDRC_DLL_STATUS

DDRC_DLL_STATUS 为 DDRC DLL 状态寄存器, 指示 DDRC 内部使用的 DLL 状态。

Offset Address																Register Name	Total Reset Value																				
0x78																DDRC_DLL_STATUS	0x0000_0000																				
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Name	reserved																overflow	dll_locked																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name		Description																																	
[31:2]	-	reserved		保留。																																	
[1]	RW	overflow		DLL 溢出标志位。 0: DLL 未溢出。 1: DLL 溢出。																																	
[0]	RW	dll_locked		DLL 锁定标志。 0: DLL 未锁定。 1: DLL 锁定。																																	



DDRC_DLL_CONFIG

DDRC DLL 配置寄存器。

Offset Address																Register Name								Total Reset Value								
0x07C																DDRC_DLL_CONFIG								0x0000_0009								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																									dll_cali_en	stop	tune				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1		
Bits	Access	Name		Description																												
[31:6]	-	reserved		保留。																												
[5]	RW	dll_cali_en		频率切换使能位，用于在频率切换时，重新校准 DLL。 0: 禁止。 1: 使能。																												
[4]	RW	stop		DLL 禁止工作使能。 0: 使能 DLL 工作。 1: 禁止 DLL 工作。																												
[3:0]	RW	tune		微调 DLL 的延迟参数。 tune[3]表示增加或者减少延迟。 0: 增加延迟单元数。 1: 减少延迟单元数。 tune[2:0]表示增加或减少的延迟单元数。 建议该值使用默认值。																												

DDRC_CHn_QOS



说明

DDRC_CHn_QOS 中 “n” 的取值范围是 0 ~ 7。

DDRC_CHn_QOS 为 DDRC Channeln QOS 寄存器。这 8 个寄存器为可读写寄存器，用于动态配置每个通道的优先级，同时可以配置每个通道的 QOS 值，以保证每个通道都可以在一个配置好的总线周期数目之内被服务。当通道请求有效后，该寄存器的值就会下载到一个递减计数器中。如果计数器的值为 0 而该端口还没有被服务，该端口的优先级就会增加，直到该端口的请求被服务。

该寄存器必须在初始化 DDRC 的过程中，或者没有任何 DDR 访问时配置。

各通道 QOS 功能如下：

- 通道 0 控制 VO 的响应延迟
- 通道 1 控制 VI 的响应延迟
- 通道 2 控制 DMA 等的响应延迟
- 通道 3 控制 EXP 的响应延迟
- 通道 4 控制 ARMD 的响应延迟
- 通道 5 控制 ARMI 的响应延迟
- 通道 6 控制 DSU 的响应延迟
- 通道 7 控制业务模块的响应延迟

Offset Address																Register Name								Total Reset Value								
0x098~0x0B4																DDRC_CHn_QOS								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																qos_en	reserved	qos								reserved	pri				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name			Description																											
[31:17]	-	reserved			保留。																											
[16]	RW	qos_en			QOS 使能信号。 0: QOS 禁止。 1: QOS 使能。																											
[15:14]	-	reserved			保留。																											
[13:4]	RW	qos			通道老化时间配置。 推荐该值配置为： 0x002~0x3FF。																											
[3]	-	reserved			保留。																											
[2:0]	RW	pri			通道优先级配置。 (数值越小，优先级越高) 000: 最高优先级。 001: 次高优先级。 111: 最低优先级。																											



4.2 SMI 控制器

4.2.1 概述

SMI (Static Memory Interface) 控制器对外提供异步静态存储器接口，可以连接 SRAM (Static Random Access Memory)、PSRAM (Pseudo Static Access Memory)、ROM (Read Only Memory)、NOR Flash 等异步静态存储器，用于实现系统启动、数据存储等功能。同时也可用于连接带异步静态存储器接口的控制芯片，实现主控功能。

4.2.2 特点

SMI 控制器具备以下特点：

- 支持异步静态存储器，包括 SRAM、PSRAM、ROM 和 NOR Flash 等。
- 支持异步 page 模式读操作。
- 支持控制器工作时钟可配置。
- 支持 2 个独立的存储器接口：0 通道和 1 通道。
- 支持从 0 通道外接存储器启动。
- 每个独立的存储器接口均可支持 8bit 数据位宽的外部存储器。
- 每个独立的存储器接口当外接 8bit 数据位宽存储器时，最大支持容量为 256Mbit。
- 支持读等待周期 (T_{WSTRD})、写等待周期 (T_{WSTWR}) 和 burst 读等待周期 ($T_{WSTBURSTRD}$) 可配置，最多可配置 31 个 SMI 工作时钟周期。
- 支持读使能等待周期 (T_{WSTOEN}) 和写使能等待周期 (T_{WSTWEN}) 可配置，最多可配为 15 个 SMI 工作时钟周期。
- 支持由读操作转变到写操作的等待周期 ($T_{WSTIDLY}$) 可配置，最多可配为 15 个 SMI 工作时钟周期。
- 支持异步等待信号输入，信号有效极性可配置。
- 支持时钟门控。
- 支持输出片选信号极性可配置。

4.2.3 信号描述

表4-7 SMI 控制器接口信号一览表

信号名称	方向	描述	对应管脚
SMI_CS0	O	SMI 控制器输出的 0 通道片选信号，可配置为高电平有效或者低电平有效（默认低电平有效）。	EBICS0N
SMI_CS1	O	SMI 控制器输出的 1 通道片选信号，可配置为高电平有效或者低电平有效（默认低电平有效）。与 GPIO 复用（复用时的配置信息请参见“4.2.5 工作方式”中的“管脚复用配置”）。	EBICS1N



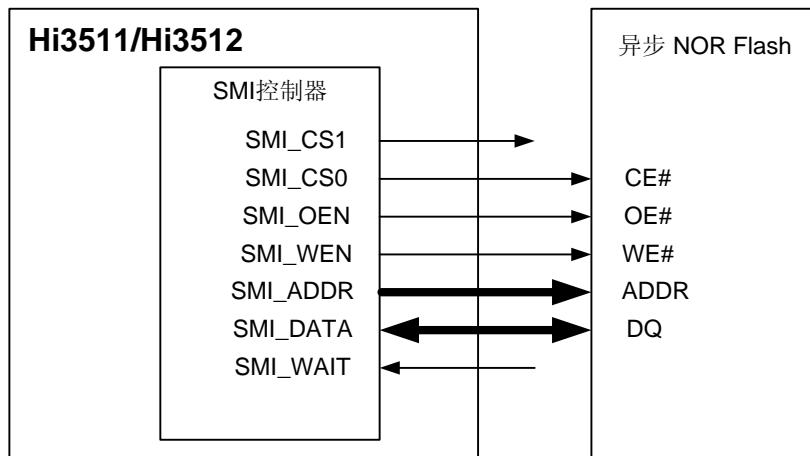
信号名称	方向	描述	对应管脚
SMI_OEN	O	SMI 控制器输出的读使能信号，低电平有效。	EBIOEN
SMI_WEN	O	SMI 控制器输出的写使能信号，低电平有效。	EBIWEN
SMI_ADDR[24:0]	O	SMI 控制器输出的地址总线。 其中 SMI_ADDR[24]与 GPIO 管脚复用（复用时的配置信息请参见“4.2.5 工作方式”中的“管脚复用配置”）。	EBIADR24～ EBIADR0
SMI_DATA[7:0]	I/O	SMI 控制器双向数据总线。	EBIDQ7～ EBIDQ0
SMI_WAIT	I	外部输入 SMI 控制器的异步等待信号，可配置为高电平有效或者低电平有效（默认低电平有效）。与 GPIO 管脚复用（复用时的配置信息请参见“4.2.5 工作方式”中的“管脚复用配置”）。	EBIRDYN

4.2.4 功能描述

应用框图

SMI 控制器用于控制内部系统总线与外部异步静态存储器总线之间数据交换，它提供了灵活的时序参数配置，适合连接各种异步静态存储器件，如图 4-3 所示。

图4-3 SMI 控制器与异步静态存储器连接示意图



连接说明：

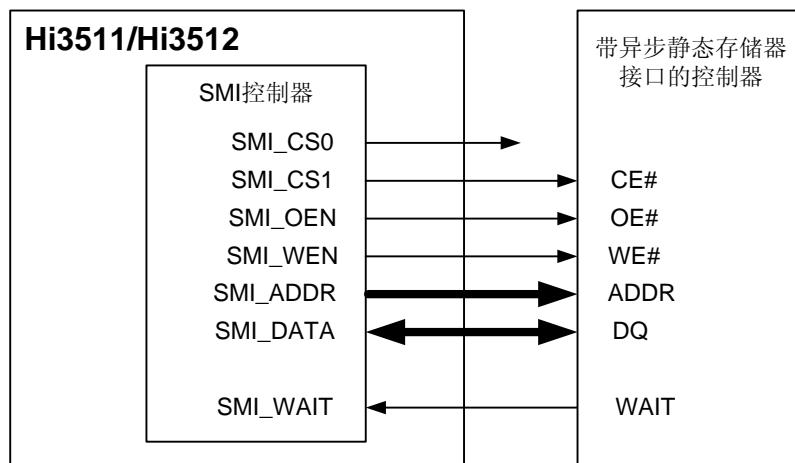


SMI 控制器有两个独立片选信号输出：SMI_CS0 和 SMI_CS1，其中支持从 SMI_CS0 外接 8bit 数据位宽存储器启动。

当 SMI 控制器外接异步静态存储器时，SMI_WAIT 信号未使用。

同时，SMI 控制器提供了异步等待机制，便于连接带有异步静态存储器接口的控制器件，如图 4-4 所示。

图4-4 SMI 控制器与带异步静态存储器接口的控制器件连接示意图



连接说明：

SMI 控制器有两个独立片选信号输出：SMI_CS0 和 SMI_CS1，其中支持从 SMI_CS0 外接 8bit 数据位宽存储器启动。

当 SMI 控制器外接带异步静态存储器接口的控制器件时，SMI_WAIT/WAIT 信号用于双方握手协商。

功能原理

SMI 控制器主要根据对接器件的工作时序参数，实现接口时序的转换。SMI 支持通过更改 SMI 控制器工作时钟，来增强 SMI 控制器的兼容性，以尽可能兼容各种静态存储器件。在不同的工作时钟下，相应的可配置的时序参数范围也随之不同（如表 4-8 所示）。

表4-8 SMI 控制器时序参数范围（总线时钟 $f_{BUSCLK}=135MHz$ ）

时序参数	$f_{SMICLK}=135MHz$	$f_{SMICLK}=67.5MHz$
读等待周期 $T_{WSTRD(max)}$	229ns	459ns
写等待周期 $T_{WSTWR(max)}$	229ns	459ns
burst 读等待周期 $T_{WSTBURSTRD(max)}$	229ns	459ns
读使能等待周期 $T_{WSTOEN(max)}$	111ns	222ns
写使能等待周期 $T_{WSTWEN(max)}$	111ns	222ns

时序参数	$f_{SMICLK}=135MHz$	$f_{SMICLK}=67.5MHz$
读写转向周期 $T_{WSTIDLY(max)}$	111ns	222ns

注：上表中所得时间为采用去尾法得到的计算结果。



注意

SMI 控制器只支持从通道 0 启动。

由于异步静态存储器接口的通用性，所以为了满足对接器件多样性的实际需求，SMI 控制器提供了两种工作模式：

- 时序参数模式

当 SMI 控制器提供的时序配置寄存器的配置范围可以满足对接器件时序要求时，SMI 控制器使用该模式来实现和对接器件的数据传送。SMI 控制器默认为工作在时序参数模式下。

- 异步等待模式

当 SMI 控制器提供的时序配置寄存器的配置范围不能满足对接器件时序要求，或者对接器件的时序不是一个明确的范围时，SMI 控制器使用该模式来实现和对接器件的数据传送。



说明

当使用 SMI 控制器的异步等待模式时，需要对接器件具有相应的异步等待控制信号。

时序参数模式下，典型的时序如图 4-5 所示。图 4-5 给出了在单次读写情况下，SMI 控制器接口上的时序关系。

图4-5 SMI 控制器时序参数模式时序图（读写）

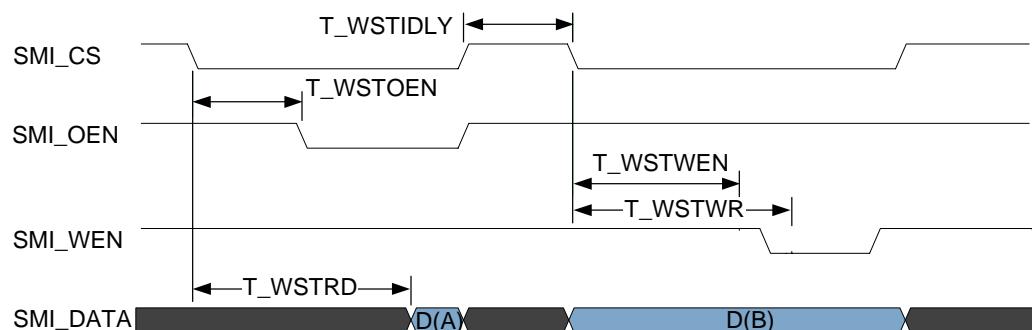


表 4-9 列出了 SMI 控制器读写时序参数。



表4-9 SMI 控制器读写时序参数表

符号	最小值	典型值	最大值	描述
T_WSTOEN	0	-	T_WSTOEN(max)	从 SMI 片选信号有效到读使能信号有效的时间
T_WSTRD	-	-	T_WSTRD(max)	从 SMI 片选信号有效到读数据有效的时间
T_WSTIDLY	0	-	T_WSTIDLY(max)	SMI 两次相邻读写操作间隔时间
T_WSTWEN	0	-	T_WSTWEN(max)	从 SMI 片选信号有效到写使能信号有效的时间
T_WSTWR	0	-	T_WSTWR(max)	从 SMI 片选信号有效到写数据有效的时间

图 4-6 给出了在 page 读情况下，SMI 控制器接口的时序关系。采用 page 读方式，可以大大提高读取数据速度。

图4-6 SMI 控制器时序参数模式时序图 (page 读)

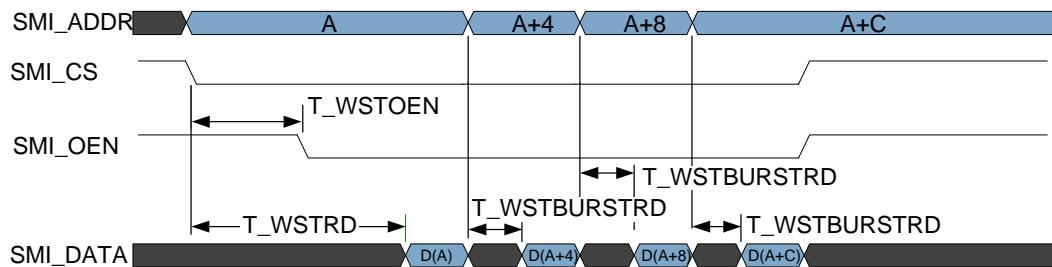


表 4-10 列出了 SMI 控制器 page 读时的时序参数。

表4-10 SMI 控制器 page 读时序参数表

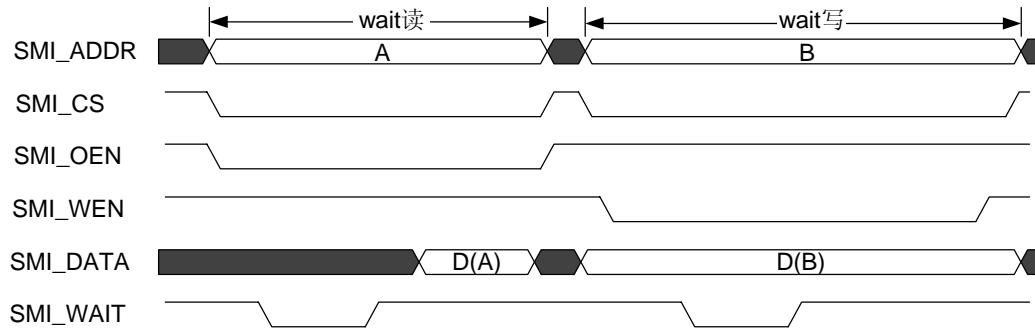
符号	最小值	典型值	最大值	描述
T_WSTOEN	0	-	T_WSTOEN(max)	从 SMI 片选信号有效到读使能信号有效的时间
T_WSTRD	-	-	T_WSTRD(max)	从 SMI 片选信号有效到读数据有效的时间
T_WSTBURSTRD	-	-	T_WSTBURSTRD(max)	SMI page 读过程中两次相邻读操作间隔时间

异步等待模式下时，典型的时序如图 4-7 所示。



图 4-7 给出了当 SMI 控制器外接其他控制芯片，通过 SMI_WAIT 信号进行握手协商时的读写时序关系。

图4-7 SMI 控制器异步等待模式时序图（wait 读写）



4.2.5 工作方式

管脚复用配置

SMI 控制器管脚与 GPIO 复用，使用 SMI 控制器之前，需要配置系统控制器，使能相应管脚的 SMI 功能，请参见 SC_PERCTRL1[pinmuxctrl4]、SC_PERCTRL1[pinmuxctrl20]和 SC_PERCTRL1[pinmuxctrl21]配置说明。

时钟门控

在软件完成当前数据传输，并且未启动新的数据传输情况下，向系统控制器 SC_PERDIS[smiclkdis]写 1，可以关断 SMI 控制器时钟。

在需要使用 SMI 控制器进行数据传输时，向系统控制器 SC_PEREN[smiclken]写 1，可以使能 SMI 控制器时钟。

对当前 SMI 控制器时钟状态的查询请参见系统控制器 SC_PERCLKEN。

时钟配置



寄存器 **SMI_CR** 中时钟比例的配置，必须与系统控制器 SC_PERCTRL2[smiclk_sel]时钟比例的配置一致。

在软件完成当前数据传输，并且未启动新的数据传输情况下，可以通过控制 SC_PERCTRL2[smiclk_sel]来配置 SMI 控制器的工作时钟，可配置为与总线时钟同频或者总线时钟的二分频。默认情况下，SMI 控制器的工作时钟等于总线时钟。



系统启动配置

系统只支持从 SMI 控制器 0 通道外接 8bit 数据位宽存储器启动。



注意

当系统采用从 SMI 控制器 0 通道外接存储器启动时，该外接存储器的片选信号必须为低电平有效。

SMI 控制器支持的存储器地址空间如表 4-11 所示。

表4-11 SMI 控制器支持的存储器地址空间

通道号	地址空间
0 通道	0x3400_0000~0x35FF_FFFF
1 通道	0x3000_0000~0x31FF_FFFF

时序参数模式下的 SMI 控制器初始化配置

需要根据 SMI 控制器外接存储器件的时序参数和特性，对 SMI 控制器进行正确配置，相关时序参数含义请参见“4.2.4 功能描述”中的“功能原理”。

时序参数模式下，初始化 SMI 控制器步骤如下（以配置 1 通道为例）：

- 步骤 1 配置寄存器 **SC_PERCTRL4[cs1_mode]**，根据对接芯片要求选择输出的片选信号极性。
- 步骤 2 配置寄存器 **SMI_CR**，设置 SMI 控制器工作时钟比例。
- 步骤 3 配置寄存器 **SMI_BIDCYR1[idcy]**，设置读写转向周期 $T_{WSTIDLY}$ 。
- 步骤 4 配置寄存器 **SMI_BWSTRDR1[wstrd]**，设置读等待周期 T_{WSTRD} 。
- 步骤 5 配置寄存器 **SMI_BWSTWRR1[wstwr]**，设置写等待周期 T_{WSTWR} 。
- 步骤 6 配置寄存器 **SMI_BWSTOENR1[wstoen]**，设置读使能等待周期 T_{WSTOEN} 。
- 步骤 7 配置寄存器 **SMI_BWSTWENR1[wstwen]**，设置写使能等待周期 T_{WSTWEN} 。
- 步骤 8 配置寄存器 **SMI_BWSTBRDR1[wstbrdr]**，设置 burst 读等待周期 $T_{WSTBURSTRD}$ 。
- 步骤 9 配置寄存器 **SMI_BCR**，设置存储器通道控制参数。

----结束

按照以上步骤进行配置后，CPU 就可以通过配置好的 SMI 控制器访问外接存储器。



注意

- 当 SMI 控制器外接异步静态存储器时，只需要正确配置 SMI 寄存器时钟比例 **SMI_CR[memclkratio]** 和外接存储器数据位宽 **SMI_BCR[mw]**，就可以对存储器进行正常数据传输。
- 在软件对每一个存储器通道进行配置时，必须满足： $T_{WSTOEN} \leq T_{WSTRD}$ 和 $T_{WSTWEN} \leq T_{WSTWR}$ 。

异步等待模式下的 SMI 控制器初始化配置

当 SMI 控制器外接带有等待信号输出的存储芯片或者控制芯片时，可以将 SMI 控制器切换到异步等待模式下，通过与外接芯片握手协商来进行数据传输。

异步等待模式下，初始化 SMI 控制器步骤如下（以配置 1 通道为例）：

- 步骤 1 配置系统控制器的 **SC_PERCTRL1[pinmuxctrl4]**，将 **EBIRDYN** 管脚设置为 **SMI_WAIT** 信号输入。
- 步骤 2 配置系统控制器的 **SC_PERCTRL4[cs1_mode]**，根据对接芯片要求选择输出的片选信号极性。
- 步骤 3 配置寄存器 **SMI_CR**，设置 SMI 控制器工作时钟比例。
- 步骤 4 配置寄存器 **SMI_BCR**，设置存储器通道控制参数。
- 步骤 5 根据外接芯片输出的“等待信号”有效极性，配置 **SMI_BCR[waitpol]**。
- 步骤 6 配置 **SMI_BCR[waiten]** 为 1，使能 SMI 控制器异步等待模式。

----结束

按照以上步骤进行配置后，SMI 控制器即可工作在异步等待模式。

4.2.6 寄存器概览

表4-12 SMI 控制器寄存器概览（基址是 0x1010_0000）

偏移地址	名称	描述	页码
0x000	SMI_BIDCYR1	SMI 控制器 1 通道读写转向周期寄存器	4-33
0x004	SMI_BWSTRDR1	SMI 控制器 1 通道读等待周期寄存器	4-34
0x008	SMI_BWSTWRR1	SMI 控制器 1 通道写等待周期寄存器	4-34
0x00C	SMI_BWSTOENR1	SMI 控制器 1 通道读使能等待周期寄存器	4-35
0x010	SMI_BWSTWENR1	SMI 控制器 1 通道写使能等待周期寄存器	4-35



偏移地址	名称	描述	页码
0x014	SMI_BCR1	SMI 控制器 1 通道控制寄存器	4-36
0x018	SMI_BSR1	SMI 控制器 1 通道状态寄存器	4-38
0x01C	SMI_BWSTBRDR1	SMI 控制器 1 通道突发读等待周期寄存器	4-38
0x020 ~ 0x0DC	RESERVED	保留	-
0x0E0	SMI_BIDCYR0	SMI 控制器 0 通道读写转向周期寄存器	4-39
0x0E4	SMI_BWSTRDR0	SMI 控制器 0 通道读等待周期寄存器	4-39
0x0E8	SMI_BWSTWRR0	SMI 控制器 0 通道写等待周期寄存器	4-40
0x0EC	SMI_BWSTOENR0	SMI 控制器 0 通道读使能等待周期寄存器	4-40
0x0F0	SMI_BWSTWENR0	SMI 控制器 0 通道写使能等待周期寄存器	4-41
0x0F4	SMI_BCR0	SMI 控制器 0 通道控制寄存器	4-41
0x0F8	SMI_BSR0	SMI 控制器 0 通道状态寄存器	4-43
0x0FC	SMI_BWSTBRDR0	SMI 控制器 0 通道突发读等待周期寄存器	4-43
0x100 ~ 0x1FC	RESERVED	保留	-
0x200	SMI_SR	SMI 控制器异步等待状态寄存器	4-44
0x204	SMI_CR	SMI 控制器工作时钟配置寄存器	4-45
0x208 ~ 0xFFC	RESERVED	保留	-

4.2.7 寄存器描述

SMI_BIDCYR1

SMI_BIDCYR1 为 SMI 控制器 1 通道读写转向周期寄存器，用来控制在读写操作之间的等待时间。

Offset Address																Register Name								Total Reset Value									
0x000																SMI_BIDCYR1								0x0000_000F									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved																									idcy							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1		
Bits	Access	Name			Description																												
[31:4]	-	reserved			保留。																												
[3:0]	RW	idcy			TWSTIDLY, 读写转向周期, 控制在读写操作之间的等待时间。 TWSTIDLY = idcy × (1/f _{SMICLK})。																												

SMI_BWSTRDR1

SMI_BWSTRDR1 为 SMI 控制器 1 通道读等待周期寄存器, 用来控制读操作的等待时间。

Offset Address																Register Name								Total Reset Value								
0x004																SMI_BWSTRDR1								0x0000_001F								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																									wstrd						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Bits	Access	Name			Description																											
[31:5]	-	reserved			保留。																											
[4:0]	RW	wstrd			TWSTRD, 控制读操作的等待时间。 对 non burst 读操作而言, 控制读操作的等待时间。 对 burst 读操作而言, 只控制第 1 次读操作的等待时间, 后续读操作等待时间由 SMI_BWSTBRDR1 进行配置。 TWSTRD = wstrd × (1/f _{SMICLK})																											

SMI_BWSTWRR1

SMI_BWSTWRR1 为 SMI 控制器 1 通道写等待周期寄存器, 用来控制写操作的等待时间。



Offset Address																Register Name	Total Reset Value															
0x008																SMI_BWSTWRR1	0x0000_001F															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																								wstwr							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1			
Bits	Access	Name		Description																												
[31:5]	-	reserved		保留																												
[4:0]	RW	wstwr		TWSTWR, 控制写操作的等待时间。 $T_{WSTWR} = wstwr \times (1/f_{SMICLK})$																												

SMI_BWSTOENR1

SMI_BWSTOENR1 为 SMI 控制器 1 通道读使能等待周期寄存器，用来控制从输出有效片选信号开始到输出有效读使能信号之间的等待时间。

Offset Address																Register Name	Total Reset Value															
0x00C																SMI_BWSTOENR1	0x0000_000F															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																									wstoen						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
Bits	Access	Name		Description																												
[31:4]	-	reserved		保留。																												
[3:0]	RW	wstoen		TWSTOEN, 控制从输出有效片选信号开始到输出有效读使能信号之间的等待时间。 $T_{WSTOEN} = wstoen \times (1/f_{SMICLK})$																												

SMI_BWSTWENR1

SMI_BWSTWENR1 为 SMI 控制器 1 通道写使能等待周期寄存器，用来控制从输出有效片选信号开始到输出有效写使能信号之间的等待时间。

Offset Address																Register Name	Total Reset Value															
0x010																SMI_BWSTWENR1	0x0000_000F															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																								wstwen							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1		
Bits	Access	Name			Description																											
[31:4]	-	reserved			保留。																											
[3:0]	RW	wstwen			T_{WSTWEN} , 控制从输出有效片选信号开始到输出有效写使能信号之间的等待时间。 $T_{WSTWEN} = wstwen \times (1/f_{SMICLK})$																											

SMI_BCR1

SMI_BCR1 为 SMI 控制器 1 通道控制寄存器, 用来配置 1 通道的各种传输参数。

Offset Address																Register Name	Total Reset Value															
0x014																SMI_BCR1	0x0030_3020															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																									reserved						
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name			Description																											
[31:22]	-	reserved			保留。																											
[21]	RW	reserved			保留, 此位只能设为 1。																											
[20]	RW	reserved			保留, 此位只能设为 1。																											
[19:18]	RW	burstlenwrite			burst 传输长度, 设定 burst 模式下一次 burst 写操作过程中传输数据的个数。 00: burst 4。 01: burst 8。 10、11: 保留。																											
[17]	RW	reserved			保留, 此位只能设为 0。																											



[16]	RW	bmwrite	写模式配置位。 0: 对外接器件进行 non burst 模式写。 1: 对外接器件进行 burst 模式写。
[15]	-	reserved	保留。
[14]	RW	reserved	保留, 此位只能设为 0。
[13]	RW	reserved	保留, 此位只能设为 1。
[12]	RW	reserved	保留, 此位只能设为 1。
[11:10]	RW	burstlenread	突发传输长度, 设定 burst 模式下一次 burst 读操作过程中传输数据的个数。 00: burst 4。 01: burst 8。 10: burst 16。 11: 保留。
[9]	RW	reserved	保留, 此位只能设为 0。
[8]	RW	bmread	读模式配置位。 0: 对外接器件进行 non burst 模式读。 1: 对外接器件进行 burst 模式读。
[7]	-	reserved	保留。
[6]	RW	reserved	保留, 此位只能设为 0。
[5:4]	RW	mw	SMI 控制器第 1 通道外接器件数据位宽。 00: 8bit。 其他: 保留。
[3]	RW	wp	外接器件写保护位, 控制是否对外接器件进行写保护。 0: 外接器件未被设为写保护, 例如 SRAM 等可写器件。 1: 外接器件为写保护状态, 例如 ROM 等只读器件。
[2]	RW	waiten	SMI 控制器 1 通道“等待信号”使能位。 0: 禁止, 此时 SMI 控制器第 1 通道不受外部输入的“等待信号”控制, 根据内部时序寄存器的配置参数进行数据传输。 1: 使能, 此时 SMI 控制器第 1 通道根据外部输入的“等待信号”进行数据传输。
[1]	RW	waitpol	选择外部输入的“等待信号”有效极性。 0: 低电平有效。 1: 高电平有效。
[0]	RW	reserved	保留, 此位只能设为 0。



SMI_BSR1

SMI_BSR1 为 SMI 控制器 1 通道状态寄存器，用来显示 1 通道当前状态。

Offset Address																Register Name								Total Reset Value								
0x018																SMI_BSR1								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												waittouterr			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:1]	-	reserved		保留。																												
[0]	RW	waittouterr		外部等待超时错误标志位。 读操作时： 0：没有错误。 1：外部等待超时错误。 写操作时： 0：无效。 1：清除外部等待超时错误状态标志。																												

SMI_BWSTBRDR1

SMI_BWSTBRDR1 为 SMI 控制器 1 通道 burst 读等待周期寄存器，用来控制 burst 读操作过程中非第 1 次读操作的等待时间。

Offset Address																Register Name								Total Reset Value								
0x01C																SMI_BWSTBRDR1								0x0000_001F								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												wstbrd			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1			
Bits	Access	Name		Description																												
[31:5]	-	reserved		保留。																												



[4:0]	RW	wstbrd	$T_{WSTBURSTRD}$, burst 读等待周期。 对于 non burst 读操作, 不使用该时序参数。 对 burst 读操作, 控制 burst 读操作过程中非第 1 次读操作的等待时间。 burst 读操作过程中第 1 次读操作的等待时间由 SMI_BWSTRDR1 进行配置。 $T_{WSTBURSTRD} = wstbrd \times (1/f_{SMICLK})$ 。
-------	----	--------	---

SMI_BIDCYR0

[SMI_BIDCYR0](#) 为 SMI 控制器 0 通道读写转向周期寄存器, 用来控制在读写操作之间的等待时间。

Offset Address																Register Name								Total Reset Value								
0x0E0																SMI_BIDCYR0								0x0000_000F								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																													idcy		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1		
Bits	Access	Name		Description																												
[31:4]	-	reserved		保留。																												
[3:0]	RW	idcy		$T_{WSTIDLY}$, 读写转向周期, 控制在读写操作之间的等待时间。 $T_{WSTIDLY} = idcy \times (1/f_{SMICLK})$																												

SMI_BWSTRDR0

[SMI_BWSTRDR0](#) 为 SMI 控制器 0 通道读等待周期寄存器, 用来控制读操作的等待时间。

Offset Address																Register Name								Total Reset Value									
0x0E4																SMI_BWSTRDR0								0x0000_001F									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved																																wstrd
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
Bits	Access	Name		Description																													
[31:5]	-	reserved		保留。																													

[4:0]	RW	wstrd	T_{WSTRD} , 控制读操作的等待时间。 对 non burst 读操作而言, 控制读操作的等待时间。 对 burst 读操作而言, 只控制第 1 次读操作的等待时间, 后续读操作等待时间由 SMI_BWSTBRDR0 进行配置。 $T_{WSTRD} = wstrd \times (1/f_{SMICLK})$
-------	----	-------	---

SMI_BWSTWRR0

[SMI_BWSTWRR0](#) 为 SMI 控制器 0 通道写等待周期寄存器, 用来控制写操作的等待时间。

	Offset Address 0x0E8																Register Name SMI_BWSTWRR0								Total Reset Value 0x0000_001F							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																										wstwr					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1			
Bits	Access	Name	Description																													
[31:5]	-	reserved	保留。																													
[4:0]	RW	wstwr	T_{WSTWR} , 控制写操作的等待时间。 $T_{WSTWR} = wstwr \times (1/f_{SMICLK})$																													

SMI_BWSTOENR0

[SMI_BWSTOENR0](#) 为 SMI 控制器 0 通道读使能等待周期寄存器, 用来控制从输出有效片选信号开始到输出有效读使能信号之间的等待时间。

	Offset Address 0x0EC																Register Name SMI_BWSTOENR0								Total Reset Value 0x0000_000F							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																										wstoen					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
Bits	Access	Name	Description																													
[31:4]	-	reserved	保留。																													



[3:0]	RW	wstoen	T_{WSTOEN} , 控制从输出有效片选信号开始到输出有效读使能信号之间的等待时间。 $T_{WSTOEN} = wstoen \times (1/f_{SMICLK})$
-------	----	--------	---

SMI_BWSTWENR0

SMI_BWSTWENR0 为 SMI 控制器 0 通道写使能等待周期寄存器, 用来控制从输出有效片选信号开始到输出有效写使能信号之间的等待时间。

Offset Address										Register Name										Total Reset Value												
0x0F0										SMI_BWSTWENR0										0x0000_000F												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												wstwen			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
Bits	Access	Name			Description																											
[31:4]	-	reserved			保留。																											
[3:0]	RW	wstwen			T_{WSTWEN} , 控制从输出有效片选信号开始到输出有效写使能信号之间的等待时间。 $T_{WSTWEN} = wstwen \times (1/f_{SMICLK})$																											

SMI_BCR0

SMI_BCR0 为 SMI 控制器 0 通道控制寄存器, 用来配置 0 通道的各种传输参数。

Offset Address										Register Name										Total Reset Value												
0x0F4										SMI_BCR0										0x0030_3000												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												reserved			
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name			Description																											
[31:22]	-	reserved			保留。																								waitpol			
[21]	RW	reserved			保留, 此位只能设为 1。																								reserved			
[20]	RW	reserved			保留, 此位只能设为 1。																								waiten			

[19:18]	RW	burstlenwrite	burst 传输长度, 设定 burst 模式下 (请参见图 4-6) 一次 burst 写操作过程中传输数据的个数。 00: burst 4。 01: burst 8。 10、11: 保留。
[17]	RW	reserved	保留, 此位只能设为 0。
[16]	RW	bmwrite	写模式配置位。 0: 对外接器件进行 non burst 模式写。 1: 对外接器件进行 burst 模式写。
[15]	-	reserved	保留。
[14]	RW	reserved	保留, 此位只能设为 0。
[13]	RW	reserved	保留, 此位只能设为 1。
[12]	RW	reserved	保留, 此位只能设为 1。
[11:10]	RW	burstlenread	burst 传输长度, 设定 burst 模式下一次 burst 读操作过程中传输数据的个数。 00: burst 4。 01: burst 8。 10: burst 16。 11: 保留。
[9]	RW	reserved	保留, 此位只能设为 0。
[8]	RW	bmread	读模式配置位。 0: 对外接器件进行 non burst 模式读。 1: 对外接器件进行 burst 模式读。
[7]	-	reserved	保留。
[6]	RW	reserved	保留, 此位只能设为 0。
[5:4]	RW	mw	SMI 控制器第 0 通道外接器件数据位宽。 00: 8bit。 其他: 保留。
[3]	RW	wp	外接器件写保护位, 控制是否对外接器件进行写保护。 0: 外接器件未被设为写保护, 例如 SRAM 等可写器件。 1: 外接器件为写保护状态, 例如 ROM 等只读器件。



[2]	RW	waiten	SMI 控制器第 0 通道“等待信号”使能位。 0: 禁止, 此时 SMI 控制器第 0 通道不受外部输入的“等待信号”控制, 根据内部时序寄存器的配置参数进行数据传输。 1: 使能, 此时 SMI 控制器第 0 通道根据外部输入的“等待信号”进行数据传输。
[1]	RW	waitpol	选择外部输入的“等待信号”有效极性。 0: 外部输入的“等待信号”低电平有效。 1: 外部输入的“等待信号”高电平有效。
[0]	RW	reserved	保留, 此位只能设为 0。

SMI_BSR0

SMI_BSR0 为 SMI 控制器 0 通道状态寄存器, 用来显示 0 通道当前状态。

Offset Address										Register Name										Total Reset Value													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Total Reset Value
Name	reserved																													waitouterr			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name										Description																					
[31:1]	-	reserved										保留。																					
[0]	RW	waitouterr										外部等待超时错误标志位。 读操作时: 0: 没有错误。 1: 外部等待超时错误。 写操作时: 0: 无效。 1: 清除外部等待超时错误状态标志。																					

SMI_BWSTBRDR0

SMI_BWSTBRDR0 为 SMI 控制器 0 通道 burst 读等待周期寄存器, 用来控制 burst 读操作过程中非第一次读操作的等待时间。

Offset Address																Register Name								Total Reset Value								
0x0FC																SMI_BWSTBRDR0								0x0000_001F								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																								wstbrd							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1			
Bits	Access	Name			Description																											
[31:5]	-	reserved			保留。																											
[4:0]	RW	wstbrd			$T_{WSTBURSTRD}$, burst 读等待周期。 对于 non burst 读操作, 不使用该时序参数。 对 burst 读操作, 控制 burst 读操作过程中非第 1 次读操作的 等待时间。 burst 读操作过程中第 1 次读操作的等待时间由 SMI_BWSTRDR0 进行配置。 $T_{WSTBURSTRD} = wstbrd \times (1/f_{SMICLK})$ 。																											

SMI_SR

SMI_SR 为 SMI 控制器异步等待状态寄存器, 用来表示异步等待模式下的异步等待状态。

Offset Address																Register Name								Total Reset Value								
0x200																SMI_SR								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																									waitstatus						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name			Description																											
[31:1]	-	reserved			保留。																											
[0]	RO	waitstatus			SMI 控制器异步等待状态位。 0: “等待信号” 无效。 1: “等待信号” 有效。																											

注: 在异步等待模式下, 该状态寄存器才表示异步等待状态。



SMI_CR

SMI_CR 为 SMI 控制器工作时钟配置寄存器，用来配置 SMI 控制器工作时钟与系统总线时钟的关系。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	SMI_CR	0x0000_0001
Name	reserved		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1		
Bits	Access	Name	Description
[31:3]	-	reserved	保留。
[2:1]	RW	memclkratio	配置 SMI 控制器工作时钟。 00: SMI 控制器工作时钟等于系统总线时钟。 01: SMI 控制器工作时钟等于系统总线时钟的一半。 10、11: 保留。
[0]	RW	reserved	保留，此位只能设为 1。



5 以太网接口

关于本章

本章描述内容如下表所示。

标题	内容
5.1 概述	介绍 ETH 模块的功能。
5.2 特点	介绍 ETH 模块的特点。
5.3 信号描述	介绍 ETH 模块的接口信号。
5.4 功能描述	介绍 ETH 模块的 MII 接口时序、MDIO 接口时序、收帧过程、发帧过程、中断管理、流量控制和 MAC 过滤。
5.5 工作方式	介绍 ETH 模块的管脚复用配置、时钟门控、软复位、初始化、中断收帧流程、查询收帧流程和发帧流程。
5.6 寄存器概览	概况介绍 ETH 模块的寄存器。
5.7 寄存器描述	详细描述 ETH 模块的寄存器。



5.1 概述

以太网接口 ETH (Ethernet MAC) 实现以太网接口数据的接收和发送，可以工作在 10Mbit/s 或者 100Mbit/s，支持全双工或者半双工工作模式，提供 MII (Media Independent Interface) 接口。ETH 模块提供可配置的 2 个 DMAC 地址过滤表，提供使用本机 MAC 进行帧过滤的功能，可以对网口的帧进行选择性过滤接收，ETH 模块还提供流量限制功能。

5.2 特点

ETH 模块具有以下特点：

- 支持端口 10Mbit/s 或 100Mbit/s 速率。
- 可工作在全双工或半双工模式。
- 支持 MII 接口。
- 支持半双工模式下的碰撞回退重传和 Late Collision 检测。
- 支持全双工模式下的 Pause 帧的发送。
- 支持帧长有效性检测，可丢弃超长帧和超短帧。
- 支持对输入帧进行 CRC (Cyclic Redundancy Check) 校验，可丢弃校验错的帧。
- 支持对输出帧添加 CRC 校验。
- 短帧填充功能：当发送数据不足 64byte 时，硬件可以自动填充至 64byte。
- 支持端口全双工模式下的内环回和外环回。
- 提供自适应功能，自动获取 PHY 芯片的工作状态。
- 提供 MDIO 接口，MDIO 时钟可配置为 2MHz 或者 1MHz。
- 支持对端口收发帧进行统计计数。
- 提供接收和发送各 2048byte 的缓冲区。
- 提供 64 个帧管理队列（接收和发送共用）。
- 支持 2 个可配置的 DMAC 地址过滤表。
- 支持根据本机 MAC 进行帧过滤，本机 MAC 可配置。
- 支持可配置是否接收广播帧、组播帧和单播帧。
- 提供总帧数流量限制，当超过限制时可根据配置选择丢弃广播帧、组播帧或者单播帧。

5.3 信号描述

ETH 模块的接口信号描述如表 5-1 所示。



表5-1 ETH 接口信号描述

信号名称	方向	描述	对应管脚
ETH_MDCK	O	MDIO 接口输出时钟。	MDCK
ETH_MDIO	I/O	MDIO 接口的输入/输出数据。	MDIO
ETH_TXD	O	MII 接口发送数据。	ETXD0~ETXD3
ETH_TXEN	O	MII 接口发送使能信号。	ETXEN
ETH_TXCK	I	MII 接口发送时钟。	ETXCK
ETH_RXD	I	MII 接口接收数据。	ERXD0~ERXD3
ETH_RXDV	I	MII 接口接收数据有效信号。	ERXDV
ETH_RXERR	I/O	MII 接口接收数据错误信号。与通用 GPIO 复用。(复用时的配置信息请参见“ 5.5.1 管脚复用配置 ”)。	ERXERR
ETH_RXCK	I	MII 接口接收时钟。	ERXCK
ETH_CRS	I/O	MII 载波侦听信号。与通用 GPIO 复用。(复用时的配置信息请参见“ 5.5.1 管脚复用配置 ”)。	ECRS
ETH_COL	I/O	MII 碰撞指示信号。与通用 GPIO 复用。(复用时的配置信息请参见“ 5.5.1 管脚复用配置 ”)。	ECOL

5.4 功能描述

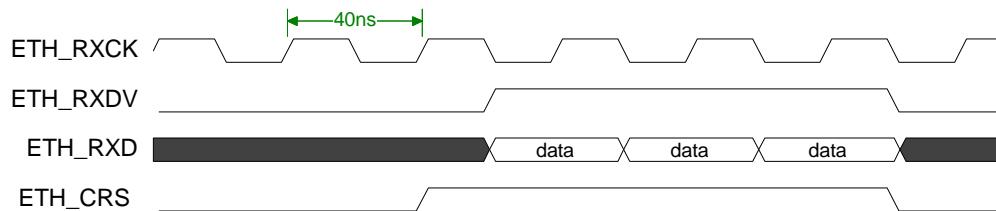
5.4.1 MII 接口时序

芯片提供标准的 MII 接口，连接 PHY (Physical Layer Entity Sublayer) 芯片，符合 MII 接口时序标准。

100Mbit/s 接收时序

MII 接口 100Mbit/s 接收时序如[图 5-1](#) 所示。

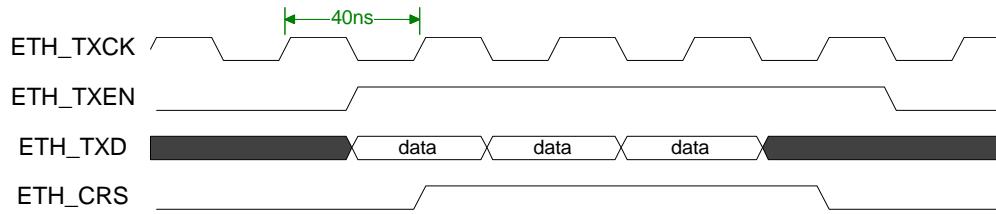
图5-1 MII 接口 100Mbit/s 接收时序



100Mbit/s 发送时序

MII 接口 100Mbit/s 发送时序如图 5-2 所示。

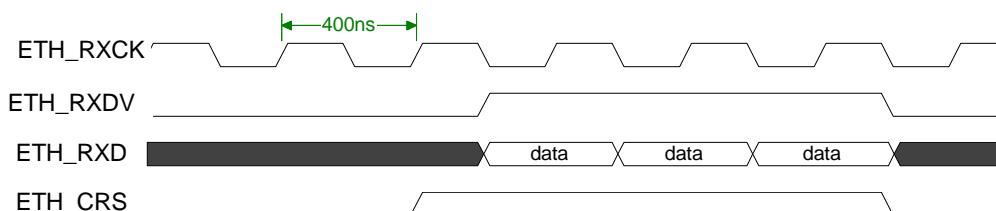
图5-2 MII 接口 100Mbit/s 发送时序



10Mbit/s 接收时序

MII 接口 10Mbit/s 接收时序如图 5-3 所示。

图5-3 MII 接口 10Mbit/s 接收时序

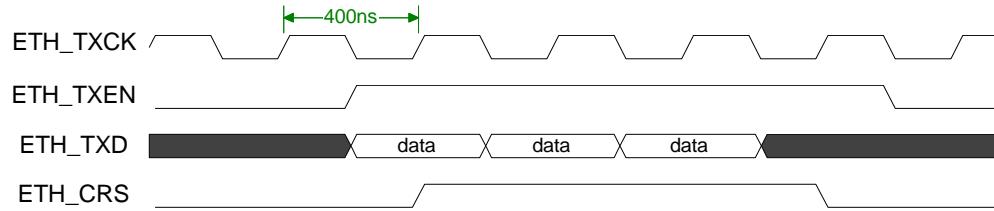


10Mbit/s 发送时序

MII 接口 10Mbit/s 发送时序如图 5-4 所示。



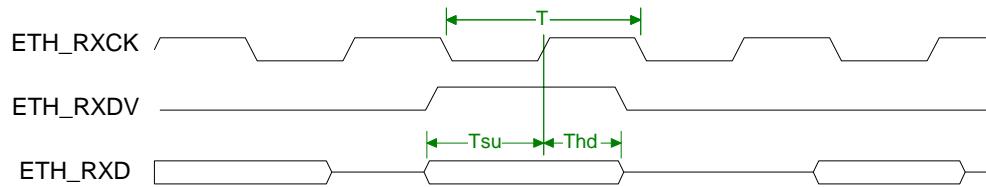
图5-4 MII 接口 10Mbit/s 发送时序



接收时序参数

MII 接口接收时序参数如图 5-5 所示。

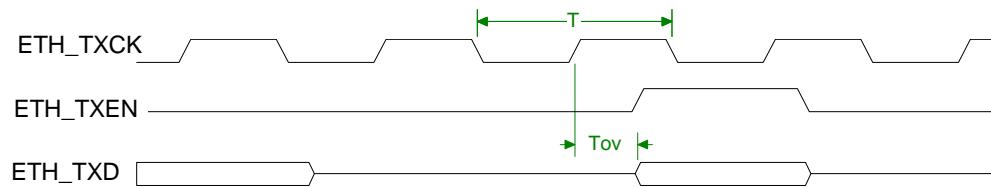
图5-5 MII 接口接收时序参数



发送时序参数

MII 接口发送时序参数如图 5-6 所示。

图5-6 MII 接口发送时序参数



时序参数说明

MII 接口时序参数说明如表 5-2 所示。

表5-2 MII 接口时序参数说明

参数	符号	信号	最小值	最大值	单位
MII 时钟周期	T	ETH_RXCK、ETH_TXCK	400(10Mbit/s)	400	ns
			40(10Mbit/s)	40	ns
MII 信号建立时间	Tsu (RX)	ETH_RXER、ETH_RXDV、ETH_RXD[3:0]	10	-	ns
MII 信号保持时间	Thd (RX)	ETH_RXER、ETH_RXDV、ETH_RXD[3:0]	10	-	ns
MII 输出信号延时	Tov (MIITX)	ETH_TXD[1:0]、ETH_TXEN	0	20	ns

5.4.2 MDIO 接口时序

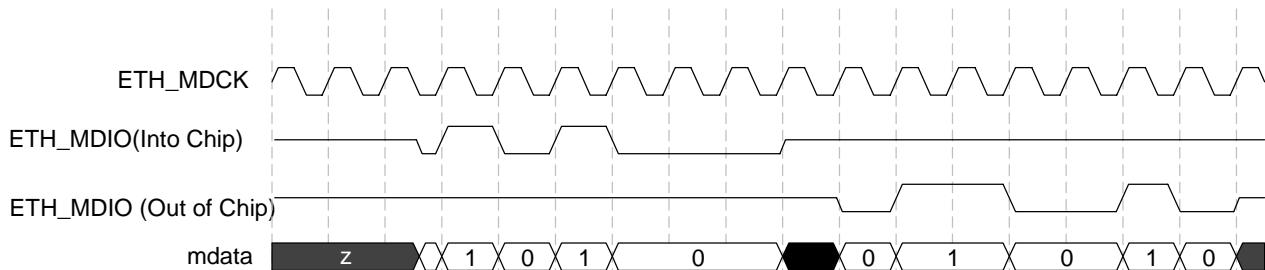
MDIO 接口实现对 PHY 芯片的读写控制。在软件操作时将 PHY 芯片的地址、数据寄存器地址和相关控制信息写入到 **MDIO_RWCTRL** 中，硬件将地址、数据和控制信息等转化成 MDIO 接口时序，然后查询到 **MDIO_RWCTRL[finish]** 为 1 表示硬件已经完成对 PHY 芯片的读写操作。读操作时，读回的数据存放在 **MAC_PORTSEL** 中，CPU 在查询到 **MDIO_RWCTRL[finish]** 为 1 后到 **MAC_PORTSEL** 中读取数据。

芯片可以通过 MDIO 接口自动获取 PHY 芯片的工作状态。用户指定 ETH 工作在自适应工作状态时，需要配置 **MAC_PORTSET** 和 **MAC_STAT_CHANGE** 寄存器中关于 PHY 芯片中相关状态寄存器的地址等信息。ETH 使用 MDIO 接口自动从指定的 PHY 芯片的相关寄存器中读出状态信息，存放在 **MAC_RO_STAT** 寄存器中。具体状态寄存器地址请参考相关 PHY 芯片文档。

读时序

MDIO 接口读时序如图 5-7 所示。

图5-7 MDIO 接口读时序

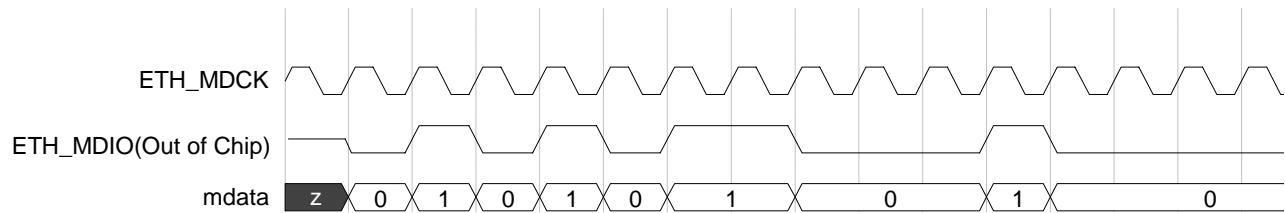




写时序

MDIO 接口写时序如图 5-8 所示。

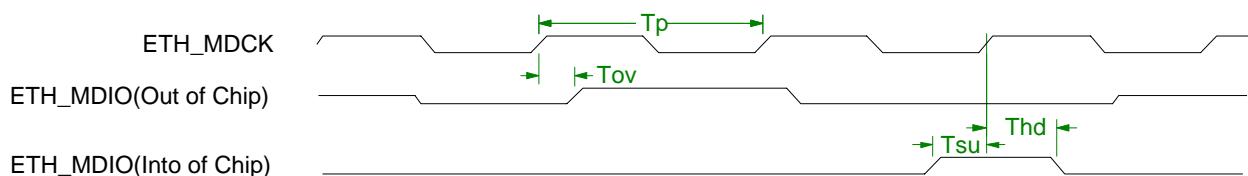
图5-8 MDIO 接口写时序



时序参数

MDIO 接口时序参数如图 5-9 所示。

图5-9 MDIO 接口时序参数



时序参数说明

MDIO 接口时序参数说明如表 5-3 所示。

表5-3 MDIO 接口时序参数

参数	符号	信号	最小值	最大值	单位
MDIO 发送数据延迟时间	Tov	MDIO	37	37	ns
MDIO 时钟周期	Tp	MDC	370 或 740	370 或 740	ns
MDIO 接收数据建立时间	Tsu	MDIO	10	-	ns
MDIO 接收数据保持时间	Thd	MDIO	10	-	ns

注：MDC 时钟周期 Tp 可通过调整 MDC 频率（[MDIO_RWCTRLfrq_dv](#)）进行改变，选择 ETH 工作时钟 135MHz 的 100 分频或者 50 分频。

5.4.3 收帧过程

当 CPU 获知有一帧数据需要接收时执行以下步骤：



步骤 1 读取寄存器 **GLB_IQFRM_DES** 中的帧描述子（Frame Descriptor，包含输入队列的地址索引和帧长度信息）。

步骤 2 根据帧描述子提供的帧存放的地址索引和帧长度信息，查找软件内部的地址索引表。

步骤 3 取出对应的帧缓冲区首地址，从该首地址开始读取帧数据，直到读取一个完整的帧后，并将 **GLB_IQDES_VLD[fd_vld_in]** 置 0（表示 CPU 收帧结束）。

----结束

软件收完一帧数据后需要重新申请一个的缓冲区，并将首地址重新写入到当前输入队列帧描述子中。否则，实际可以使用的输入队列的深度不等于 CPU 配置的数值，而等于 CPU 实际分配的缓冲区个数。



说明

申请的缓冲区满足最大帧长即可，最大值为 2KB。

CPU 接收帧描述子数据结构如表 5-4 所示。

表5-4 CPU 接收帧描述子数据结构

位	名称	描述
[31:17]	-	保留。
[16:11]	addr_index	当前帧索引值，软件根据该索引值查找内部的查找表，得到当前接收帧 Buffer 的首地址。
[10:0]	fm_len	接收帧长度，单位为 byte。

CPU 通过帧描述子得到的是帧存放的地址索引，软件需要查找内部的接收队列查找表得到当前接收帧 Buffer 的首地址。该查找表在初始化输入队列时建立，在每次新分配一个队列时需要软件更新。如果该查找表和 ETH 硬件的输入队列对应错误，将导致收帧产生错误，软件可以通过读 ETH 输入队列的当前读写地址进行校正或者进行软复位，重新建立索引表。



说明

查询 **GLB_ADDRQ_STAT** 可获得接收队列使用长度。

CPU 收帧可以采用中断方式或查询方式。

- 中断方式收帧

CPU 使能帧接收中断时，根据是否需要接收帧，硬件会产生帧接收中断 **int_cpu_rx** 通知 CPU 收帧。中断方式收帧流程如图 5-10 所示。

- 查询方式收帧

CPU 不使能帧接收中断 **GLB_IRQ_ENA[ien_cpu_rx]**，CPU 自动查询 **GLB_IQDES_VLD[fd_vld_in]**，为 1 表示有帧需要 CPU 接受。查询方式收帧流程如图 5-11 所示。



图5-10 中断方式收帧流程图

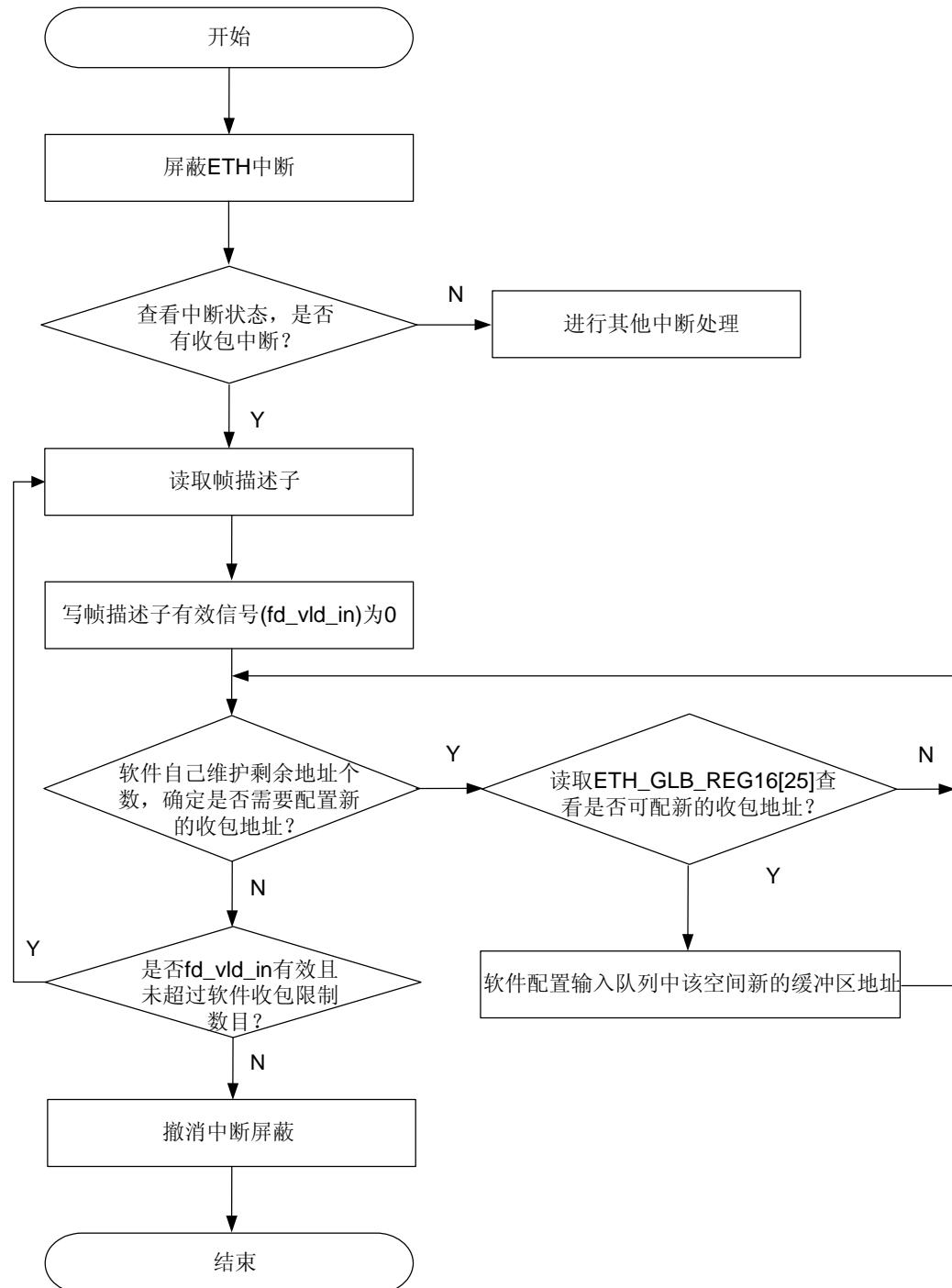
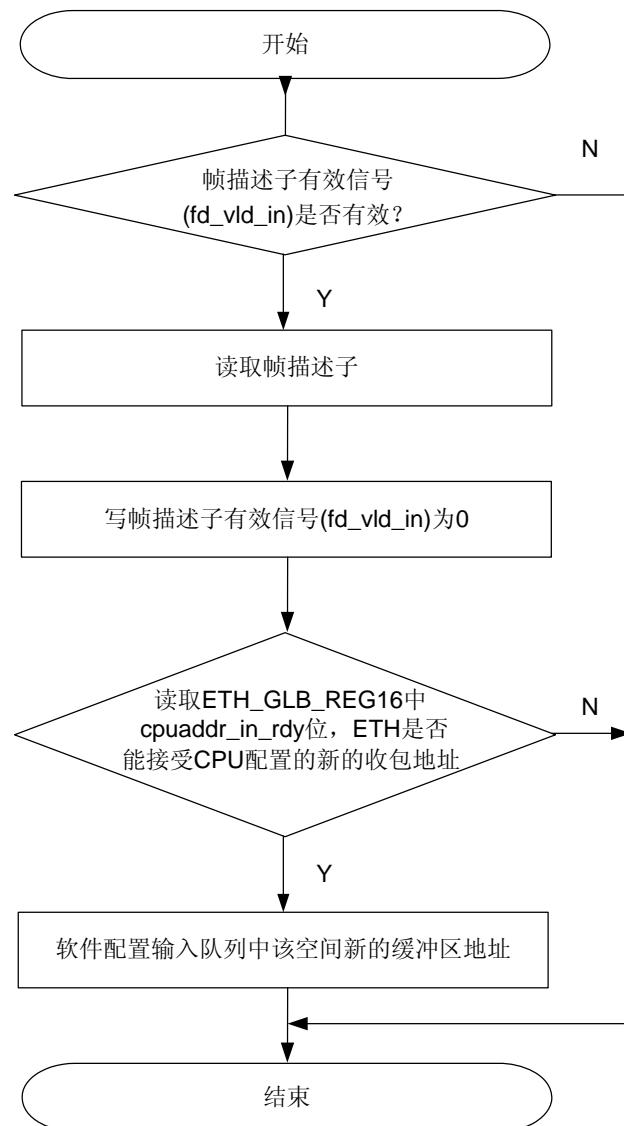


图5-11 查询方式收帧流程图



5.4.4 发帧过程

CPU 有帧待发送时，判断当前队列是否有空间发送。如果有空间发送，则分别写该帧 Buffer 的起始地址和帧长到输出队列帧描述子中。CPU 先写入起始地址，该地址必须保证 word 对齐（最低两位为 0），然后写入发送帧的帧长。写发送帧的帧长触发硬件将该输出帧的首地址和帧长信息写入到输出队列中等待发送。因此，软件应该控制不能任意写帧长寄存器，每写该寄存器一次会导致发送一个数据包。

帧格式如下：

目的 MAC (6 byte)	源 MAC (6 byte)	Type (2 byte)	Data (0 ~1500 byte)	FCS (4 byte)
--------------------	-------------------	------------------	------------------------	-----------------



CPU 发出的帧在 SDRAM 中缓存时, 不包括帧描述子。写帧描述子到 **GLB_EQDES_ADDR** 及 **GLB_EQDES_LEN**, 通知 ETH 将该帧入队。CPU 发送帧描述子数据结构如表 5-5 所示。

表5-5 CPU 发送帧描述子数据结构

位	名称	描述
[31:0]	start_addr_eq	发送一个帧的首地址。
[10:0]	fm_len	帧长度, 单位为 byte。

注: fm_len 小于 20byte 以及大于 1900byte 的帧将被丢弃, 即可发送范围在 20byte~1900byte 之间。



说明

查询 **GLB_ADDRQ_STAT** 可获得当前 CPU 发送队列使用情况。

CPU 发帧可以采用中断方式或查询方式。

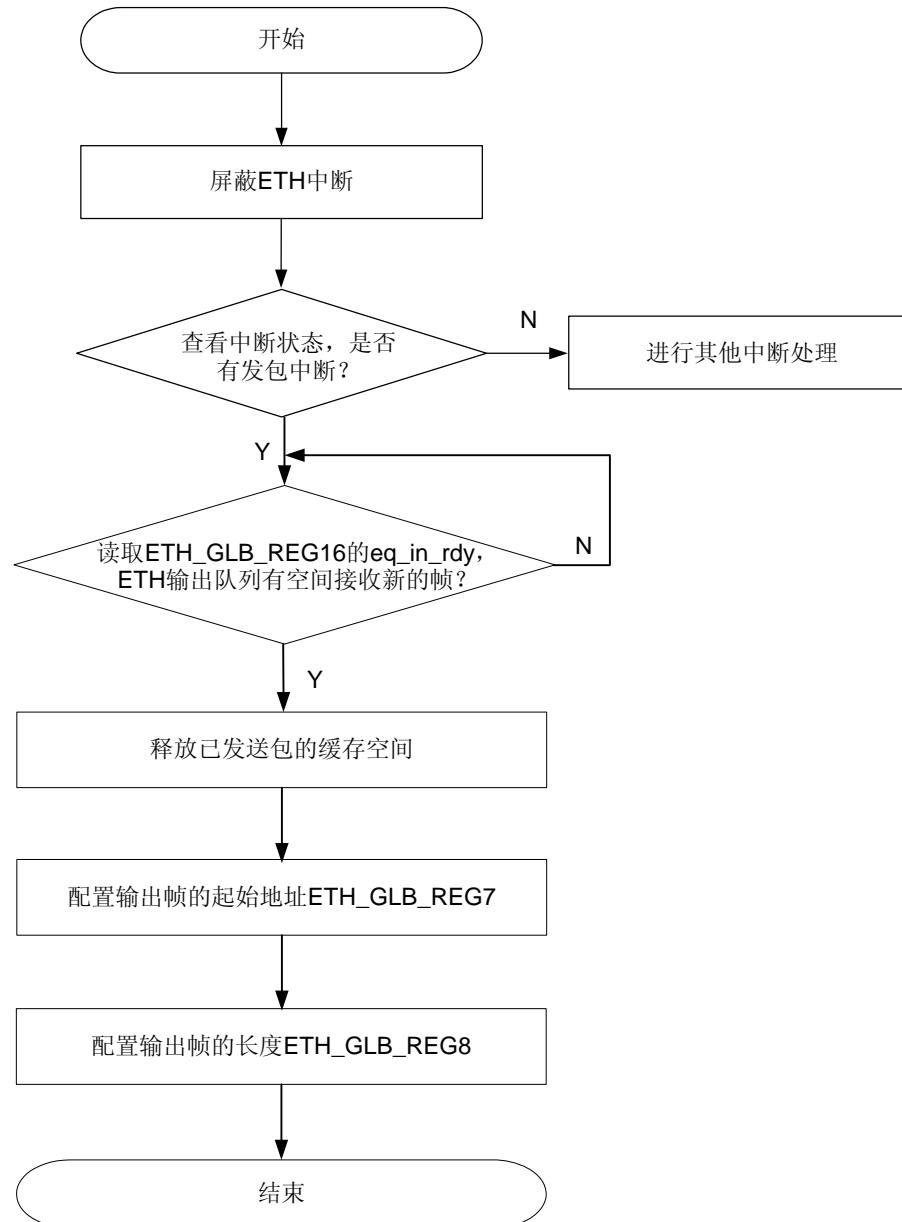
1. 中断方式发帧

CPU 使能 ETH 发包队列由不空到空中断 (**GLB_IRQ_STA[int_free_eq]**), 同时设置允许该中断通知 CPU。此时如果 ETH 输出队列由不空到空, 说明 ETH 可以发送帧, 硬件产生中断通知 CPU, 表示可以发帧。

如果软件需要发送帧, 但是当前输出队列已满 (**GLB_ADDRQ_STAT[eq_in_rdy]** = 0), 则软件可以使能该中断。当输出队列为空后, 产生中断, 通知软件发送等待的帧。软件可以使用该中断一次发送一组帧, 同时在该中断时释放上次发送的一组帧的缓冲区。

中断方式发帧流程如图 5-12 所示。

图5-12 中断方式发帧流程图

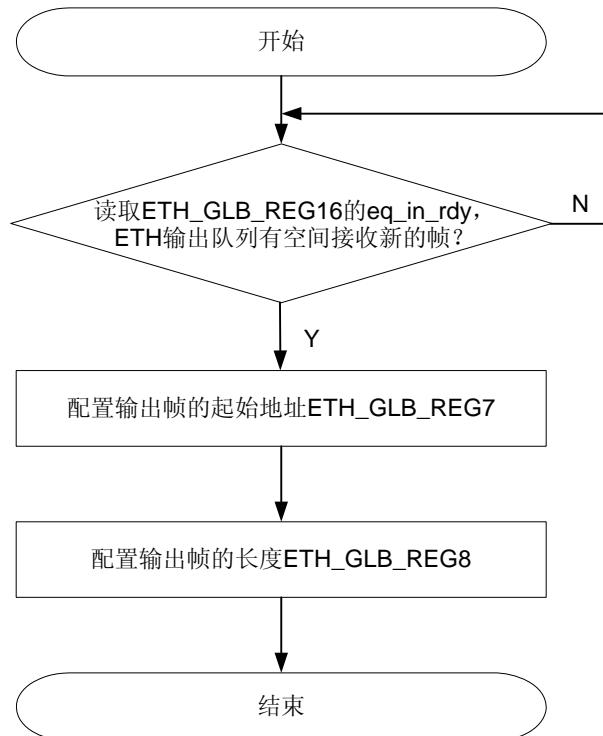


2. 查询方式发帧

软件查询自己内部的发帧计数，如果计数小于配置的输出队列长度，则可以直接将待发送的帧通知ETH，同时建立对应的发帧索引表。该表的索引地址为输出队列对应的物理地址，索引表项内容为写入输出队列该地址的输出帧首地址。当ETH发送完一帧后，通过输出队列的地址通知CPU释放对应的发送缓冲区，CPU通过该输出队列的地址查找到对应的发送缓冲区并释放。查询方式发帧流程如图 5-13 所示。



图5-13 查询方式发帧流程图



5.4.5 中断管理

ETH 中断状态寄存器

指示产生的中断类型，具体请参见“[GLB_IRQ_STA](#)”。

ETH 中断使能寄存器

控制是否产生相应的中断，具体请参见“[GLB_IRQ_ENA](#)”。如果使能某中断，其中断状态会写入到相应的中断状态寄存器中。

ETH 中断选择寄存器

设置某种类型的中断产生后是否允许传送给 CPU，具体请参见“[GLB_IRQ_SEL](#)”。如果选择某种中断不传送给 CPU，则该类型中断产生后，不会产生 CPU 中断，但是相应的中断状态位会置位。

5.4.6 流量控制

对于软件接收的帧，当单位时间间隔内接收到的帧超过软件配置上限时，则将后续接收的帧按配置进行选择性丢弃。软件可配置超过流量限制后丢弃广播帧、组播帧或者单播帧（通过[GLB_FC_DROPCTRL](#)配置）。流量限制阈值通过[GLB_FC_RXLIMIT](#)设置。当新一轮时间间隔开始时，收包计数器从 0 开始计数。



软件可通过 [GLB_FC_TIMECTRL](#) 配置流量限制的时间间隔。使用 1 个 10bit 时间间隔寄存器，最大可以是 1023 个时间粒度。时间间隔粒度采用 17bit 计数器，对主时钟进行计数，默认值是 100000，对于 100MHz 主时钟时间粒度为 1ms。软件可配置（20bit 寄存器）流量限制上限值的大小，若配置为 0 表示不进行流量限制全部接收。

5.4.7 MAC 过滤

配置寄存器 [GLB_FILTER_CFG](#) 可以对接收帧的目的 MAC 进行过滤。可以设定本机 MAC、MAC0、MAC1 地址的过滤使能、丢弃还是接收，以及广播帧、组播帧、单播帧的丢弃还是接收。本机 MAC、MAC0、MAC1 地址的配置方法如下：

- 本机 MAC 地址通过 [GLB_HOSTMAC_L32](#) 和 [GLB_HOSTMAC_H16](#) 配置。
- MAC0 地址通过 [GLB_MAC0_L32](#) 和 [GLB_MAC0_H16](#) 配置。
- MAC1 地址通过 [GLB_MAC1_L32](#) 和 [GLB_MAC1_H16](#) 配置。

5.5 工作方式

5.5.1 管脚复用配置

ETH 模块的部分管脚为复用管脚。使用 ETH 前，需要配置系统控制器的 SC_PERCTRL1 使能相应管脚的 ETH 功能。ERXERR、ECRS、ECOL 的复用由系统控制器的 SC_PERCTRL1[24]控制，设置为 0 表示选择 ETH 接口。

5.5.2 时钟门控

在不使用 ETH 模块时，可关断 ETH 时钟，以降低功耗。

关断 ETH 时钟的步骤如下：

- 步骤 1 关闭 ETH 的链路状态，使 ETH 无法收发包。
- 步骤 2 软件继续收包，直至收包队列清空，使 ETH 无收包中断上报。
- 步骤 3 软件下发复位 ETH 逻辑命令，复位不撤销。
- 步骤 4 关闭 ETH 时钟，将系统控制器的 SC_PERDIS[17]置 1，完成 ETH 时钟关闭操作。

----结束

打开 ETH 时钟的步骤如下：

- 步骤 1 保持复位不撤销。打开 ETH 时钟，将系统控制器的 SC_PEREN[17]置 1，使能 ETH 时钟。
- 步骤 2 如果需要使用半双工模式，将系统控制器的 SC_PERCTRL1[24]置 0，选择 ETH 接口；否则将系统控制器的 SC_PERCTRL1[24]置 1 跳入步骤 3。
- 步骤 3 撤销复位。
- 步骤 4 打开链路状态，ETH 恢复正常工作。



----结束

5.5.3 软复位

软复位的步骤如下：

- 步骤 1 关闭 ETH 的链路状态以及收包中断使能，使软件无法收包，同时软件不再发包。
- 步骤 2 等到软件处理完当前收发包后，清除收发包队列，同时队列长度保持软复位之前的值，相关指针、队列计数值均要回 0。
- 步骤 3 下发 ETH 软复位命令，将系统控制器的 SC_PERCTRL0[12]置 1。
- 步骤 4 撤销 ETH 软复位，将系统控制器的 SC_PERCTRL0[12]置 0。
- 步骤 5 对收包队列的帧首地址进行初始化配置。
- 步骤 6 打开链路状态，ETH 恢复正常工作。

----结束

5.5.4 初始化

初始化的步骤如下：

- 步骤 1 配置端口状态获取方式。

ETH 可以自适应 PHY 芯片工作状态或者由软件配置其工作状态，在初始化时通过配置 **MAC_PORTSEL[stat_ctrl]** 进行选择：

- 0：选择自适应获取工作状态。
- 1：选择软件配置端口工作状态。

复位时选择软件设置端口工作状态。

- 步骤 2 自适应工作状态。

如果配置端口通过自适应方式获取 PHY 芯片的工作状态，需要指明 PHY 芯片速度、双工模式和有无连接寄存器的地址以及这些状态位各自寄存器中的偏移地址。通过配置 **MDIO_ANEG_CTRL** 实现。

- 步骤 3 软件配置工作状态。

如果配置端口通过软件配置工作状态，则软件需要根据实际运用环境配置 **MAC_STAT_CHANGE** 中的速度、连接状态和双工模式状态信息，同时将这些信息配置到 PHY 芯片的相关寄存器中。

- 步骤 4 配置 PHY 芯片工作状态（可选）。

ETH 提供 MDIO 接口实现对 PHY 芯片的读写控制。软件操作时将数据、PHY 芯片的地址、寄存器地址和相关控制信息写入到 **MDIO_RWCTRL** 寄存器中，然后查询到 **MDIO_RWCTRL[finish]** 为 1，表示硬件完成对 PHY 芯片的读写操作。具体的配置内容需要参考相关 PHY 芯片的数据手册。

- 步骤 5 接收和发送帧队列深度设置。

设置寄存器 **GLB_FC_LEVEL** 中的输入队列深度和输出队列深度：

- 输入队列深度表示接收数据时可以缓存的帧的最大数量。
- 输出队列深度表示发送数据时可以缓存的帧的最大数量。

由于输入队列和输出队列共用 64 个管理空间，因此设置的输入队列深度和输出队列深度之和不能超过 64，输入队列深度和输出队列深度至少为 1。如果软件设置的输入队列深度和输出队列深度之和大于 64，则优先保证输入队列长度，另外一项则为 64 减去先配置项的结果。

步骤 6 初始化接收帧队列缓冲区。

复位后，软件需要申请与配置的输入队列深度个数相同的缓冲区，每个缓冲区大小为 2KB，然后将缓冲区首地址依次写入到帧输入队列中，写入的次数应刚好等于配置的输入队列的深度。软件需要保证配置的首地址是以 word 为单位，并且保证在收帧时不能释放该缓冲区。软件需要在内部建立一个索引表，索引地址为对应的输入队列的地址，索引项内容为分配给该输入队列地址的缓冲区的首地址。

步骤 7 模块软复位。

软件配置 ETH 模块软复位，可以对 ETH 内部的逻辑电路和帧管理队列进行复位，使 ETH 模块回到初始化状态。但是，ETH 内部所有的寄存器保留软复位前的值。软件解除软件复位后，需要重新申请收包缓冲区并对输入队列进行初始化，否则 ETH 不能接收网络包。

----结束



说明

ETH 软复位后，软件设置的寄存器是不变的。收发帧中与电路状态相关的寄存器会被软复位，其中：

- MDIO 控制寄存器只有 **MDIO_RO_DATA/MDIO_RO_STAT** 可以软复位。
- MAC 控制寄存器只有 **MAC_RO_STAT/MAC_STAT_CHANGE** 可以软复位。
- 全局控制寄存器只有 **GLB_IRQ_STA/GLB_QSTAT/GLB_IQFRM_DES/GLB_IQDES_VLD/GLB_ADDRQ_STAT** 可以软复位。
- 统计计数控制寄存器全部可以被软复位，但 **STS_RX_STA/STS_TX_STA** 必须清空统计计数器后才有意义。
- 统计计数器中 **RXDVRISE** (0x030C)、**COLLISIONS** (0x03A4)、**DOT3OUTPAUSE** (0x03A8)、**OCTETS_TX** (0x03AC) 可以被软复位，其他寄存器存储在 RAM (Random-Access Memory) 中，可以设置清空。相关控制请参见 **MAC_SET** (0x01BC) 的 **cntr_clr_all**。

5.5.5 中断收帧流程

中断收帧的步骤如下：

步骤 1 进入中断处理程序后，屏蔽 ETH 中断。

步骤 2 查看中断状态 **GLB_IRQ_STA[int_cpu_rx]**，看是否有收帧中断。如果有，转步骤 3；如果没有，则进行其他 ETH 中断处理。

步骤 3 读取帧描述子 **GLB_IQFRM_DES[fd_in_len]**，根据帧数据长度取数。



- 步骤 4 写帧描述子有效信号 **GLB_IQDES_VLD**[fd_vld_in] 为 0, 通知硬件接收完成。
- 步骤 5 软件根据自己维护的剩余可用地址个数, 确定是否需要配置新的收包地址。如果不需
要配置, 转步骤 8。完成一次收包。
- 步骤 6 读取 **GLB_ADDRQ_STAT**[cpu_addr_in_rdy], 查看是否可配新的收包地址。如果不可
配, 返回步骤 5。
- 步骤 7 软件通过 **GLB_IQ_ADDR** 将新的缓冲区地址配置给输入队列。返回步骤 5。
- 步骤 8 读取并判断 **GLB_IQDES_VLD**[fd_vld_in]。如果该位有效且软件可继续收包 (未超过
收包限制数目), 则返回步骤 3。
- 步骤 9 撤销 ETH 中断屏蔽。

----结束

5.5.6 查询收帧流程

不使能帧接收中断 **GLB_IRQ_ENA**[ien_cpu_rx], CPU 自动查询帧描述子有效信号
GLB_IQDES_VLD[fd_vld_in], 为 1 表示有帧需要接收。

查询收帧的步骤如下:

- 步骤 1 读取帧描述子有效信号 **GLB_IQDES_VLD**[fd_vld_in]。如果该位无效, 直接结束。
- 步骤 2 读取帧描述子 **GLB_IQFRM_DES**[fd_in_len], 根据帧数据长度取数。
- 步骤 3 写帧描述子有效信号 **GLB_IQDES_VLD**[fd_vld_in] 为 0, 通知硬件接收完成。
- 步骤 4 读取 **GLB_ADDRQ_STAT**[cpu_addr_in_rdy], 查看是否可配新的收包地址。如果无
效, 直接结束。
- 步骤 5 软件通过寄存器 **GLB_IQ_ADDR** 将新的缓冲区地址配置给输入队列。

----结束

5.5.7 发帧流程

发帧的步骤如下:

- 步骤 1 读取 **GLB_ADDRQ_STAT**[eq_in_rdy], 检查 ETH 输出队列是否有剩余空间接收新的发
送帧。如果没有, 继续等待并查询。
- 步骤 2 配置待输出帧的起始地址 **GLB_EQDES_ADDR**。
- 步骤 3 配置待输出帧的长度。完成一帧的发送配置 **GLB_EQDES_LEN**。

----结束



5.6 寄存器概览

ETH MDIO 控制寄存器

ETH MDIO 控制寄存器概览如表 5-6 所示。

表5-6 ETH MDIO 控制寄存器（基址是 0x9003_0000）

偏移地址	名称	描述	页码
0x0180	MDIO_RWCTRL	MDIO 命令字寄存器	5-22
0x0184	MDIO_RO_DATA	MDIO 读数据寄存器	5-23
0x0188	MDIO_PHYADDR	PHY 物理地址寄存器	5-23
0x018C	MDIO_RO_STAT	PHY 芯片状态寄存器	5-23
0x0190	MDIO_ANEG_CTRL	PHY 芯片各状态的偏移地址设置寄存器	5-24
0x0194	MDIO_IRQENA	端口状态变化扫描屏蔽寄存器	5-25
0x0198~0x019C	RESERVED	保留	-

ETH MAC 控制寄存器

ETH MAC 控制寄存器概览如表 5-7 所示。

表5-7 ETH MAC 控制寄存器（基址是 0x9003_0000）

偏移地址	名称	描述	页码
0x01A0	MAC_TX_IPGCTRL	发送帧间隙控制寄存器	5-26
0x01A4	MAC_PORTSEL	端口工作状态控制寄存器	5-27
0x01A8	RESERVED	保留	-
0x01AC	MAC_PORTSET	端口设定工作状态寄存器	5-28
0x01B0	MAC_RO_STAT	端口状态寄存器	5-28
0x01B4	MAC_STAT_CHANGE	端口状态改变指示寄存器	5-29
0x01B8	MAC_RX_IPGCTRL	接收帧间隙控制寄存器	5-30
0x01BC	MAC_SET	MAC 功能设置寄存器	5-30



ETH 统计计数控制寄存器

ETH 统计计数控制寄存器概览如表 5-8 所示。

表5-8 ETH 统计计数控制寄存器（基址是 0x9003_0000）

偏移地址	名称	描述	页码
0x01C4	STS_PORTCNT	端口部分状态计数器	5-31
0x01C8	STS_RX_STA	接收方向计数器状态指示寄存器	5-32
0x01CC	STS_TX_STA	发送方向计数器状态指示寄存器	5-34
0x01D0	STS_FLT_CFG	过滤和流量限制计数器设置寄存器	5-36
0x01D4~0x01FC	RESERVED	保留	-

ETH 全局控制寄存器

ETH 全局控制寄存器概览如表 5-9 所示。

表5-9 ETH 全局控制寄存器（基址是 0x9003_0000）

偏移地址	名称	描述	页码
0x0200	GLB_IRQ_STA	中断状态寄存器	5-39
0x0204	GLB_IRQ_ENA	中断使能寄存器	5-40
0x0208	GLB_IRQ_SEL	中断选择寄存器	5-42
0x020C	RESERVED	保留	-
0x0210	GLB_ENDIAN_MOD	大小端控制寄存器	5-43
0x0214	GLB_FILTER_CFG	MAC 过滤器配置寄存器	5-44
0x0218	GLB_QLEN_SET	队列长度配置寄存器	5-45
0x021C	GLB_FC_LEVEL	流控控制寄存器	5-46
0x0220	GLB_EQDES_ADDR	待发送帧的起始地址寄存器	5-46
0x0224	GLB_EQDES_LEN	待发送帧的帧长寄存器	5-46
0x0228	RESERVED	保留	-
0x022C	GLB_QSTAT	队列状态寄存器	5-47

偏移地址	名称	描述	页码
0x0230	GLB_IQFRM_DES	输入帧描述子寄存器	5-48
0x0234	GLB_IQDES_VLD	待接受帧描述子有效指示寄存器	5-48
0x0238	GLB_IQ_ADDR	输入帧首地址寄存器	5-49
0x023C	GLB_HOSTMAC_L32	本机 MAC 地址寄存器	5-49
0x0240	GLB_HOSTMAC_H16	本机 MAC 地址寄存器	5-49
0x0244	GLB_TXDROP_CFG	发送方向超时丢包配置寄存器	5-50
0x0248	GLB_ADDRQ_STAT	地址队列状态寄存器	5-50
0x024C	GLB_FC_TIMECTRL	流量限制时间配置寄存器	5-51
0x0250	GLB_FC_RXLIMIT	流量限制水线配置寄存器	5-52
0x0254	GLB_FC_DROPCTRL	流量限制丢包控制寄存器	5-52
0x0258	GLB_MAC0_L32	MAC 过滤器 0 寄存器	5-53
0x025C	GLB_MAC0_H16	MAC 过滤器 0 寄存器	5-53
0x0260	GLB_MAC1_L32	MAC 过滤器 1 寄存器	5-53
0x0264	GLB_MAC1_H16	MAC 过滤器 1 寄存器	5-54
0x0268~0x02FC	RESERVED	保留	-

ETH 统计结果寄存器

ETH 统计结果寄存器概览如表 5-10 所示。

表5-10 ETH 统计结果寄存器（基址是 0x9003_0000）

偏移地址	名称	描述	页码
0x0300	DROPEVENTS	ETH 统计结果寄存器	5-54
0x0304	CRCERR	ETH 统计结果寄存器	5-55
0x0308	ABNORMALSIZEPKTS	ETH 统计结果寄存器	5-55
0x030C	RXDVRISE	ETH 统计结果寄存器	5-56
0x0310	BROADCASTPKTS	ETH 统计结果寄存器	5-56
0x0314	MULTICASTPKTS	ETH 统计结果寄存器	5-56



偏移地址	名称	描述	页码
0x0318	IFINUCASTPKTS	ETH 统计结果寄存器	5-57
0x031C	IFINERRORS	ETH 统计结果寄存器	5-57
0x0320	DOT3ALIGNMENTERR	ETH 统计结果寄存器	5-57
0x0324	DOT3PAUSE	ETH 统计结果寄存器	5-58
0x0328	DOT3DRIBBLE	ETH 统计结果寄存器	5-58
0x032C	PKTS	ETH 统计结果寄存器	5-55
0x0330	FLT_DROP_CNT	ETH 统计结果寄存器	5-59
0x0334	LOCAL_MAC_MATCH	ETH 统计结果寄存器	5-59
0x0338	FLUX_FRAME_CNT	ETH 统计结果寄存器	5-60
0x033C	FLUX_DROP_CNT	ETH 统计结果寄存器	5-60
0x0340	FLTFLUX_REC_FRAME	ETH 统计结果寄存器	5-60
0x0344	FLTFLUX_DROP_FRAME	ETH 统计结果寄存器	5-61
0x0348	IFINOCTETS	ETH 统计结果寄存器	5-61
0x034C	OCTETS_RX	ETH 统计结果寄存器	5-62
0x0350~0x037C	RESERVED	保留	-
0x0380	BROADCASTPKTS_TX	ETH 统计结果寄存器	5-62
0x0384	MULTICASTPKTS_TX	ETH 统计结果寄存器	5-62
0x0388	RETRY_TIMES_TX	ETH 统计结果寄存器	5-63
0x038C	IFOUTUCASTPKTS_TX	ETH 统计结果寄存器	5-63
0x0390	DOT3COL_OK	ETH 统计结果寄存器	5-63
0x0394	DOT3LATECOL	ETH 统计结果寄存器	5-64
0x0398	DOT3EXCESSIVECOL	ETH 统计结果寄存器	5-64
0x039C	DOT3COLCNT	ETH 统计结果寄存器	5-64
0x03A0	PKTS_TX	ETH 统计结果寄存器	5-62
0x03A4	COLLISIONS	ETH 统计结果寄存器	5-65
0x03A8	DOT3OUTPAUSE	ETH 统计结果寄存器	5-66
0x03AC	OCTETS_TX	ETH 统计结果寄存器	5-66
0x03B0~0x03FC	RESERVED	保留	-

5.7 寄存器描述

5.7.1 ETH MDIO 控制寄存器

MDIO_RWCTRL

MDIO_RWCTRL 为 MDIO 命令字寄存器。该寄存器不支持软复位。

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	cpu_data_in																finish	reserved	rw	phy_exaddr	reserved	frq_dv	phy_inaddr									
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:16]	RW	cpu_data_in	用于 MDIO 模块对 PHY 进行写操作的数据寄存器。 写操作时，此处的 16bit 数据将通过 MDIO 写入 PHY 的对应寄存器中。																													
[15]	RW	finish	完成对 PHY 的读/写操作后，会置 1。 当要进行第 2 次读/写操作时，CPU 要先对该位进行清零。																													
[14]	-	reserved	保留。																													
[13]	RW	rw	用来表示对 PHY 的访问方式。 0：读操作。 1：写操作。																													
[12:8]	RW	phy_exaddr	对外部操作的 PHY 的对应的外部物理地址。 1 个 MDIO 可以对外面的多个 PHY 进行读写访问。每个 PHY 有 1 个相应的地址。当外部只连接一个 PHY 芯片时，其等价于 MDIO_PHYADDR[phy_addr] 。																													
[7:6]	-	reserved	保留。																													
[5]	RW	frq_dv	对外部 PHY 进行读写操作时，对 MDC (MDIO 接口时钟) 的分频系数，用来实现对 MDIO 的分频值。 frq_dv 与 MDC 频率对应关系（以主时钟频率 100MHz 为例）。 0：对 ETH 工作主时钟 50 分频，分频后的频率为 2MHz。 1：对 ETH 工作主时钟 100 分频，分频后的频率为 1MHz。																													
[4:0]	RW	phy_inaddr	对外部操作的 PHY 芯片的内部寄存器地址，用 5 位二进制数表示。																													



MDIO_RO_DATA

MDIO_RO_DATA 为 MDIO 读数据寄存器。

MDIO_PHYADDR

MDIO_PHYADDR 为 PHY 物理地址寄存器。该寄存器不支持软复位。

MDIO_RO_STAT

MDIO_RO_STAT 为 PHY 芯片状态寄存器。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	MDIO_RO_STAT	0x0000_0000
Name		reserved	reserved
reset	0 0		
Bits	Access	Name	Description
[31:3]	-	reserved	保留。
[2]	RO	speed_mdio2mac	从 MDIO 接口得到的端口速度工作状态，处于 10Mbit/s 或者 100Mbit/s 工作方式。 0: 10Mbit/s 工作方式。 1: 100Mbit/s 工作方式。
[1]	RO	link_mdio2mac	从 MDIO 接口得到的链接状态指示。 0: 无链接状态。 1: 有链接状态。
[0]	RO	duplex_mdio2mac	从 MDIO 接口得到的外部端口双工工作状态，处于半双工或全双工方式。 0: 半双工方式。 1: 全双工方式。

MDIO_ANEG_CTRL

MDIO_ANEG_CTRL 为 PHY 芯片各状态的偏移地址设置寄存器。该寄存器不支持软复位。



注：如 PHY 芯片速度状态位于其地址为 17 的寄存器的第 14bit，即可配置 internal_addr_speed 为 0x11，speed_index 为 0xE，则此时 ETH 可通过 MDIO 接口将该比特值读出作为 PHY 当前工作的速度模式信息。

MDIO_IRQENA

MDIO_IRQENA 为端口状态变化扫描屏蔽寄存器。该寄存器不支持软复位。

	Offset	Address	Register Name	Total Reset Value																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name			reserved																														
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																														
[31:4]	-	reserved	保留。																														
[3]	RW	link_partner_ch_m ask	端口 link partner 状态扫描变化屏蔽信号。 0: 屏蔽 link partner 扫描结果。 1: 允许 link partner 扫描结果产生中断。																														
[2]	RW	speed_ch_mask	端口速度模式扫描变化屏蔽信号。 0: 屏蔽速度模式扫描结果。 1: 允许速度模式扫描结果产生中断。																														
[1]	RW	link_ch_mask	端口连接模式扫描变化屏蔽信号。 0: 屏蔽连接模式扫描结果。 1: 允许连接模式扫描结果产生中断。																														
[0]	RW	duplex_ch_mask	端口双工模式扫描变化屏蔽信号。 0: 屏蔽双工模式扫描结果。 1: 允许双工模式扫描结果产生中断。																														

说明

- 对于端口所连接 PHY 状态信息无法通过配置 [MDIO_ANEG_CTRL](#) 进行扫描获取的，用户可通过 [MDIO_IRQENA](#) 扫描 PHY 状态寄存器来获取端口状态是否发生了变化，并产生中断通知软件进行处理。
- link_partner 状态变化是指 PHY 状态的 link、speed、duplex 任意 1 位发生变化。

5.7.2 ETH MAC 控制寄存器

ETH MAC 控制寄存器为端口控制类寄存器，当端口状态有效时配置其改变后需要进行一次软件复位。

MAC_TX_IPGCTRL

MAC_TX_IPGCTRL 为发送帧间隙控制寄存器。该寄存器不支持软复位。



MAC PORTSEL

MAC_PORTSEL 为端口工作状态控制寄存器。该寄存器不支持软复位。

MAC_PORTSET

MAC_PORTSET 为端口设定工作状态寄存器。该寄存器不支持软复位。

	Offset Address	Register Name	Total Reset Value
	0x01AC	MAC_PORTSET	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	
reset	0 0		
Bits	Access	Name	Description
[31:3]	-	reserved	保留。
[2]	RW	speed_stat_dio	CPU 设定外部端口速度模式。 0: 10Mbit/s 模式。 1: 100Mbit/s 模式。
[1]	RW	link_stat_dio	CPU 设定链接状态。 0: 无链接。 1: 有链接。
[0]	RW	duplex_stat_dio	CPU 设定双工模式。 0: 半双工模式。 1: 全双工模式。

MAC_RO_STAT

MAC_RO_STAT 为端口状态寄存器。

	Offset Address	Register Name	Total Reset Value
	0x01B0	MAC_RO_STAT	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	
reset	0 0		
Bits	Access	Name	Description
[31:3]	-	reserved	保留。



[2]	RO	speed_stat	端口当前速度模式。 0: 10Mbit/s 模式。 1: 100Mbit/s 模式。
[1]	RO	link_stat	端口当前连接状态。 0: 无链接。 1: 有链接。
[0]	RO	duplex_stat	端口当前双工模式。 0: 半双工模式。 1: 全双工模式。

MAC_STAT_CHANGE

MAC_STAT_CHANGE 为端口状态改变指示寄存器。

	Offset Address			Register Name	Total Reset Value																											
	0x01B4			MAC_STAT_CHANGE	0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																											speed_stat_ch	link_stat_ch	duplex_stat_ch		
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:3]	-	reserved	保留。																													
[2]	W1C	speed_stat_ch	速度模式改变指示信号。 0: 速度模式没有改变。 1: 速度模式改变。																													
[1]	W1C	link_stat_ch	链接状态改变指示信号。 0: 链接状态没有改变。 1: 链接状态改变。																													
[0]	W1C	duplex_stat_ch	双工模式改变指示信号。 0: 双工模式没有改变。 1: 双工模式改变。																													



MAC_RX_IPGCTRL

MAC_RX_IPGCTRL 为接收帧间隙控制寄存器。该寄存器不支持软复位。

注 1: rx_ipg_en 设置有效时, 若帧间隙与前导码之和小于配置的 rx_ipg_cfg, 则该帧将会直接被丢掉。

注 2: rx_jpg_en 设置无效时, 若输入帧帧间隙过短(小于规定 96bit 值), 有可能导致错包(帧间隙+前导码小于 24BT)或丢包(帧间隙+前导码介于 24BT~96BT)。BT 是 bittime 的简写。

MAC_SET

MAC_SET 为 MAC 功能设置寄存器。该寄存器不支持软复位。



[29]	RW	add_pad_en	发送时自动添加 PAD 使能。 0: 不添加 PAD。 1: 自动添加 PAD。
[28]	RW	crcgen_dis	CRC 生成禁止控制寄存器, 1 表示输出帧不计算 CRC, 缺省为 0 (重计 CRC)。
[27]	RW	cntr_rdclr_en	端口统计计数器读清空使能信号, 1 有效, 缺省为 0 (非读清)。
[26]	RW	cntr_clr_all	端口统计计数器清空信号, 1 有效, 缺省为 0。
[25]	RW	cntr_roll_dis	端口统计非循环计数使能信号, 1 有效, 缺省为 0 (可循环计数)。
[24:21]	RW	colthreshold	端口冲突次数统计阈值, 缺省为 1, 表示发送出现一次冲突的帧的个数。
[20]	RW	in_loop_en	端口内环回使能信号, 1 有效。
[19]	RW	ex_loop_en	端口外环回使能信号, 1 有效。
[18]	RW	pause_en	端口流控帧发送使能信号, 1 有效, 缺省为 1。
[17]	RW	rx_shframe_en	端口短帧接收使能信号, 1 有效, 缺省为 1。
[16:11]	RW	rx_min_thr	端口允许接收的最小帧长, 缺省为 42。取值范围为 42byte~63byte。
[10:0]	RW	len_max	端口允许接收的最大帧长, 缺省为 1518。

外环回和内环回不可同时配置使能, 即不支持内、外环回同时使能。

当网口处于正常工作状态时, 内、外环回配置改变之后需对 ETH 模块进行软件复位。各 bit 位的配置规则如下:

- 若 rx_shframe_en 配置为 1 时, 端口允许接收的最小帧长为 rx_min_thr 配置帧长。
- 若 rx_shframe_en 配置为 0 时, 端口允许接收的最小帧长缺省为 64byte (含 CRC)。
- 若 rx_min_thr 配置值小于 42, 将取值为 42。
- 若 len_max 配置大于 2000, 将取值 2000。
- 若 len_max 配置小于 256, 将取值 256。一般配置 1518byte~1535byte。
- 若 cntr_clr_all 配置为 1 时, 在下次全清时需要置 0 后再置为 1 才会执行全清操作。

5.7.3 ETH 统计计数控制寄存器

STS_PORTCNT

STS_PORTCNT 为端口部分状态计数器。



STS_RX_STA

STS_RX_STA 为接收方向计数器状态指示寄存器。



[18]	RO	cntr_stat_rx[18]	IFINOCTETS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[17]	RO	cntr_stat_rx[17]	FLTFLUX_DROP_FRAME 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[16]	RO	cntr_stat_rx[16]	FLTFLUX_REC_FRAME 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[15]	RO	cntr_stat_rx[15]	FLUX_DROP_CNT 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[14]	RO	cntr_stat_rx[14]	FLUX_FRAME_CNT 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[13]	RO	cntr_stat_rx[13]	LOCAL_MAC_MATCH 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[12]	RO	cntr_stat_rx[12]	FLT_DROP_CNT 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[11]	RO	cntr_stat_rx[11]	PKTS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[10]	RO	cntr_stat_rx[10]	DOT3DRIBBLE 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[9]	-	reserved	保留。
[8]	RO	cntr_stat_rx[8]	DOT3ALIGNMENTERR 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[7]	RO	cntr_stat_rx[7]	IFINERRORS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。

[6]	RO	cntr_stat_rx[6]	IIFINUCASTPKTS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[5]	RO	cntr_stat_rx[5]	MULTICASTPKTS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[4]	RO	cntr_stat_rx[4]	BROADCASTPKTS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[3]	RO	cntr_stat_rx[3]	RXDVRISE 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[2]	RO	cntr_stat_rx[2]	ABNORMALSIZEPKTS 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[1]	RO	cntr_stat_rx[1]	CRCERR 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[0]	RO	cntr_stat_rx[0]	DROPEVENTS 计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。

STS_TX_STA

STS_TX_STA 为发送方向计数器状态指示寄存器。



[3]	RO	cntr_stat_tx[3]	IFOUTUCASTPKTS_TX 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[2]	-	reserved	保留。
[1]	RO	cntr_stat_tx[1]	MULTICASTPKTS_TX 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。
[0]	RO	cntr_stat_tx[0]	BROADCASTPKTS_TX 寄存器计数状态。 0: 计数值不足一半。 1: 计数值已超过一半。

STS_FLT_CFG

STS_FLT_CFG 为过滤和流量限制计数器设置寄存器。

	Offset Address		Register Name	Total Reset Value																													
	0x01D0		STS_FLT_CFG	0x0000_0000																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	local_mac_rec
Name	reserved																																
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description																									local_mac_rec					
[31:25]	-	reserved	保留。																														
[24]	RW	flux_mutil_drop	利用 FLTFLUX_DROP_FRAME 计数器对流量限制中丢弃的正确组播帧进行计数。 0: 不进行计数。 1: 进行计数。																														



[23]	RW	flux_uni_drop	利用 FLTFLUX_DROP_FRAME 计数器对流量限制中丢弃的正确单播帧进行计数。 0: 不进行计数。 1: 进行计数。
[22]	RW	flux_broad_drop	利用 FLTFLUX_DROP_FRAME 计数器对流量限制中丢弃的正确广播帧进行计数。 0: 不进行计数。 1: 进行计数。
[21]	RW	flt_mutil_drop	利用 FLTFLUX_DROP_FRAME 计数器对过滤表组播帧丢弃的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[20]	RW	flt_uni_drop	利用 FLTFLUX_DROP_FRAME 计数器对过滤表单播帧丢弃的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[19]	RW	flt_broad_drop	利用 FLTFLUX_DROP_FRAME 计数器对过滤表广播帧丢弃的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[18]	RW	mac1_drop	利用 FLTFLUX_DROP_FRAME 计数器对符合 MAC1 丢弃的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[17]	RW	mac0_drop	利用 FLTFLUX_DROP_FRAME 计数器对符合 MAC0 丢弃的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[16]	RW	local_mac_drop	利用 FLTFLUX_DROP_FRAME 计数器对符合本机 MAC 丢弃的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[15:9]	-	reserved	保留。
[8]	RW	flux_mutil_rec	利用 FLTFLUX_REC_FRAME 计数器对流量限制中接收的正确组播帧进行计数。 0: 不进行计数。 1: 进行计数。



[7]	RW	flux_uni_rec	利用 FLTFLUX_REC_FRAME 计数器对流量限制中接收的正确单播帧进行计数。 0: 不进行计数。 1: 进行计数。
[6]	RW	flux_broad_rec	利用 FLTFLUX_REC_FRAME 计数器对流量限制中接收的正确广播帧进行计数。 0: 不进行计数。 1: 进行计数。
[5]	RW	flt_mutil_rec	利用 FLTFLUX_REC_FRAME 计数器对过滤表组播帧接收的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[4]	RW	flt_uni_rec	利用 FLTFLUX_REC_FRAME 计数器对过滤表单播帧接收的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[3]	RW	flt_broad_rec	利用 FLTFLUX_REC_FRAME 计数器对过滤表广播帧接收的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[2]	RW	mac1_rec	利用 FLTFLUX_REC_FRAME 计数器对符合 MAC1 接收的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[1]	RW	mac0_rec	利用 FLTFLUX_REC_FRAME 计数器对符合 MAC0 接收的正确帧进行计数。 0: 不进行计数。 1: 进行计数。
[0]	RW	local_mac_rec	利用 FLTFLUX_REC_FRAME 计数器对符合本机 MAC 接收的正确帧进行计数。 0: 不进行计数。 1: 进行计数。

注: 由于 **FLTFLUX_REC_FRAME** 只能指定一种情况进行计数, 所以 **STS_FLT_CFG[8:0]** 最多只能有 1bit 配置为 1, 否则无意义。同样, **FLTFLUX_DROP_FRAME** 也只能指定一种情况进行计数, 所以 **STS_FLT_CFG[24:16]** 最多只能有 1bit 配置为 1。



5.7.4 ETH 全局控制寄存器

GLB_IRQ_STA

GLB_IRQ_STA 为中断状态寄存器。

[4]	WC	int_duplex_ch	ETH 双工模式变化中断指示。 0: ETH 双工模式没有变化。 1: ETH 双工模式发生变化, 产生中断。 进入该中断程序后, 查询 MAC_STAT_CHANGE [duplex_stat_ch], 判断是否发生双工状态改变。
[3]	WC	int_speed_ch	ETH 速度模式变化中断指示。 0: ETH 速度模式没有变化。 1: ETH 速度模式发生变化, 产生中断。 进入该中断程序后, 查询 MAC_STAT_CHANGE [speed_stat_ch], 判断是否发生速度状态改变。
[2]	WC	int_link_ch	ETH 连接状态变化中断指示。 0: ETH 连接状态没有变化。 1: ETH 连接状态发生变化, 产生中断。 进入该中断程序后, 查询 MAC_STAT_CHANGE [link_stat_ch], 判断是否发生连接状态改变。
[1]	WC	int_tx_cpu	ETH 发送完来自 CPU 的 1 帧数据指示。 0: ETH 未发送完。 1: ETH 已发送完, 产生中断。 当发生该中断后, 需要查询 GLB_QSTAT 中的输出队列当前的出队地址 eq_out_index, 判断是否释放输出帧的缓存空间。
[0]	WC	int_cpu_rx	ETH 有帧等待 CPU 接收中断指示。 0: 中断无效。 1: 中断有效, 输入队列中有帧等待 CPU 接收。 当发生该中断后, 需要查询 GLB_IQDES_VLD [fd_vld_in]信号是否有效判断是否有帧等待接收。

GLB_IRQ_ENA

GLB_IRQ_ENA 为中断使能寄存器。该寄存器不支持软复位。



[0]	RW	ien_cpu_rx	ETH 有帧等待 CPU 接收使能。 0: 不使能该中断。 1: 使能该中断。
-----	----	------------	---

GLB_IRQ_SEL

GLB_IRQ_SEL 为中断选择寄存器。该寄存器不支持软复位。

	Offset	Address	Register Name	Total Reset Value
		0x0208	GLB_IRQ_SEL	0x0000_01FF
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name		reserved		isel_cpu_rx isel_tx_cpu isel_link_ch isel_speed_ch isel_duplex_ch isel_mdio_finish isel_err_bus isel_mdio_statch isel_free_eq
reset	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			
Bits	Access	Name	Description	
[31:9]	-	reserved	保留。	
[8]	RW	isel_free_eq	输出队列由不空到空中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。	
[7]	RW	isel_mdio_statch	端口状态改变中断信号允许送给 CPU 指示。 0: 不允许。 1: 允许。	
[6]	RW	isel_err_bus	总线操作出错指示中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。	
[5]	RW	isel_mdio_finish	MDIO 完成 CPU 操作指示中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。	



[4]	RW	isel_duplex_ch	ETH 双工模式变化中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。
[3]	RW	isel_speed_ch	ETH 速度模式变化中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。
[2]	RW	isel_link_ch	ETH 连接状态变化中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。
[1]	RW	isel_tx_cpu	ETH 发送完来自 CPU 的一帧数据指示中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。
[0]	RW	isel_cpu_rx	ETH 有帧等待 CPU 接收中断信号允许输出给 CPU 指示。 0: 不允许。 1: 允许。

GLB_ENDIAN_MOD

GLB_ENDIAN_MOD 为大小端控制寄存器。该寄存器不支持软复位。

	Offset Address	Register Name	Total Reset Value
	0x0210	GLB_ENDIAN_MOD	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	in_endian out_endian
reset	0 0		
Bits	Access	Name	Description
[31:2]	-	reserved	保留。
[1]	RW	in_endian	ETH 写 SDRAM 大小端配置。 0: big endian 模式。 1: little endian 模式。 数据按字节翻转。

[0]	RW	out_endian	ETH 读 SDRAM 大小端配置。 0: big endian 模式。 1: little endian 模式。
-----	----	------------	---

GLB_FILTER_CFG

GLB_FILTER_CFG 为 MAC 过滤器配置寄存器。该寄存器不支持软复位。



[3]	RW	mac1_ctrl	mac1_en 使能时, 对接收帧的目的 MAC 为 MAC1 配置帧的控制。 0: 丢弃。 1: 接收。
[2]	RW	broad_ctrl	对于目的 MAC 地址为 48'hFF_FF_FF_FF_FF_FF 的广播帧接收控制。 0: 丢弃。 1: 接收。
[1]	RW	uni_ctrl	对于目的 MAC 地址最高 byte 为偶数的单播帧接收控制。 0: 丢弃。 1: 接收。
[0]	RW	multi_ctrl	对于目的 MAC 地址最高 byte 为奇数的组播帧接收控制。 0: 丢弃。 1: 接收。

GLB_QLEN_SET

GLB_QLEN_SET 为队列长度配置寄存器。该寄存器不支持软复位。

	Offset Address			Register Name			Total Reset Value																									
	0x0218			GLB_QLEN_SET			0x0000_2020																									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															iq_len			reserved			eq_len										
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:14]	-	reserved	保留。																													
[13:8]	RW	iq_len	输入队列长度配置。																													
[7:6]	-	reserved	保留。																													
[5:0]	RW	eq_len	输出队列长度配置。																													

注 1: iq_len 和 eq_len 不得配置 0, 否则将会强行置 1。

注 2: 配置的 iq_len 和 eq_len 之和不得大于 64, 否则优先满足 iq_len 的值 (非 0), eq_len 取 64 - iq_len。

GLB_FC_LEVEL

GLB_FC_LEVEL 为流控控制寄存器。该寄存器不支持软复位。

	Offset Address 0x021C																															Total Reset Value 0x0000_1E1D			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved																								reserved	qlimit_down									
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	0	1				
Bits	Access	Name	Description																																
[31:14]	-	reserved	保留。																																
[13:8]	RW	qlimit_up	接收队列流控上限。当使用的输入队列数大于或等于该上限值时，发送流控信息。																																
[7:6]	-	reserved	保留。																																
[5:0]	RW	qlimit_down	接收队列解除流控下限。当使用的输入队列个数小于该下限值同时正处于接收流控状态时，解除当前流控状态。																																

注 1：建议不要将 qlimit_up 和 qlimit_down 配置为 0。若 qlimit_up 配置为 0，将始终处于流控状态；若 qlimit_down 配置为 0，将始终不能解除流控。

注 2：上限值 qlimit_up 要比下限值 qlimit_down 大，并且上限值 qlimit_up 不可大于配置的输入队列长度，否则无意义。

GLB_EQDES_ADDR

GLB_EQDES_ADDR 为待发送帧的起始地址寄存器。该寄存器不支持软复位。

	Offset Address 0x0220																															Total Reset Value 0x0000_0000										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
Name	start_addr_eq																																									
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description																																							
[31:0]	RW	start_addr_eq	CPU 添加到输出队列的发送帧的起始地址。																																							

GLB_EQDES_LEN

GLB_EQDES_LEN 为待发送帧的帧长寄存器。该寄存器不支持软复位。



Register Name																																	
0x0224																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved																																
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																														
[31:11]	-	reserved	保留。																														
[10:0]	RW	fm_len	CPU 添加到输出队列的发送帧的帧长。 配置该寄存器触发硬件将发送帧的起始地址和帧长写入到输出队列中等待发送，软件发送帧时必须先写入帧的起始地址，然后再写入该帧的帧长。																														

注：fm_len 小于 20byte 以及大于 1900byte 的帧将被丢弃，即可发送范围在 20byte~1900byte 之间。

GLB_QSTAT

GLB_QSTAT 为队列状态寄存器。

Offset Address																																	
0x022C																																	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved																																
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																														
[31:30]	-	reserved	保留。																														
[29:24]	RO	iq_in_index	输入队列入队 index。																														
[23:22]	-	reserved	保留。																														
[21:16]	RO	cpuw_index	输入队列帧首地址入队 index。																														
[15:14]	-	reserved	保留。																														
[13:8]	RO	eq_in_index	输出队列描述子入队 index。																														
[7:6]	-	reserved	保留。																														
[5:0]	RO	eq_out_index	输出队列描述子出队 index。																														

GLB_IQFRM_DES

GLB_IQFRM DES 为输入帧描述子寄存器。

GLB_IQDES_VLD

GLB_IQDES_VLD 为待接受帧描述子有效指示寄存器。

注：CPU 收帧采用写清方式表明收帧完毕（写 `glb_reg11`，清有效信号 `fd_vld_in`），然后才能配置新的帧起始地址（写 `glb_reg12`，`startaddr_iq`）。即不能超前配置，否则，清该位置描述子之前配置的该位置的帧起始地址将无效。



GLB_IQ_ADDR

GLB_IQ_ADDR 为输入帧首地址寄存器。该寄存器不支持软复位。

Offset Address																Register Name								Total Reset Value								
0x0238																GLB_IQ_ADDR								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	startaddr_iq																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Bits	Access	Name			Description																										
	[31:0]	RW	startaddr_iq			CPU 配置输入队列的每个空间对应的存储空间的起始地址。接收帧根据此地址来申请总线。																										

注：收发帧软件分配的地址必须是 word 对齐，否则起止处有可能取出错误的数据（硬件不对其进行 word 对齐调整）。

GLB_HOSTMAC_L32

GLB_HOSTMAC_L32 为本机 MAC 地址寄存器。该寄存器不支持软复位。

GLB_HOSTMAC_H16

GLB_HOSTMAC_H16 为本机 MAC 地址寄存器。该寄存器不支持软复位。

	Offset Address																													Total Reset Value		
	0x0240																												0x0000_0000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															local_mac[47:32]																
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15:0]	RW	local_mac[47:32]	本机 MAC 的高 16 位。																													

GLB_TXDROP_CFG

GLB_TXDROP_CFG 为发送方向超时丢包配置寄存器。该寄存器不支持软复位。

	Offset Address																													Total Reset Value		
	0x0244																												0x0000_0000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															fd_drop_index															fd_drop_en	
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:7]	-	reserved	保留。																													
[6:1]	RW	fd_drop_index	CPU 配置输出队列 (eq) 中某帧无效的索引值。配置值需在当前输出对列范围之内 0~eq_len-1。																													
[0]	RW	fd_drop_en	该索引值对应的帧是否丢弃。 0: 不丢弃。 1: 丢弃。																													

GLB_ADDRQ_STAT

GLB_ADDRQ_STAT 为地址队列状态寄存器。



注: `cpuaddr_in_rdy`、`eq_in_rdy` 复位时为 0, 但 reset 后将立即被电路置为 1。即复位后 iq 地址队列和 eq 描述子队列是可配的。

GLB_FC_TIMECTRL

GLB_FC_TIMECTRL 为流量限制时间配置寄存器。该寄存器不支持软复位。



[26:17]	RW	flux_timer_cfg	流量限制时间间隔计数器，对 flux_timer_inter 产生的分频时钟进行计数。配置为 0 不进行流量限制。
[16:0]	RW	flux_timer_inter	流量限制时间粒度计数器，对主时钟进行计数，默认为 100000，在 100MHz 主时钟时为 1ms。

GLB_FC_RXLIMIT

GLB_FC_RXLIMIT 为流量限制水线配置寄存器。该寄存器不支持软复位。

GLB_FC_DROPCTRL

GLB_FC_DROPCTRL 为流量限制丢包控制寄存器。该寄存器不支持软复位。



[1]	RW	flux_multi	流量限制超过上限值时丢弃组播包配置。 0: 不丢弃组播包。 1: 丢弃组播包。
[0]	RW	flux_broad	流量限制超过上限值时丢弃广播包配置。 0: 不丢弃广播包。 1: 丢弃广播包。

GLB_MAC0_L32

GLB_MAC0_L32 为 MAC 过滤器 0 寄存器。该寄存器不支持软复位。

	Offset	Address	Register Name	Total Reset Value
		0x0258	GLB_MAC0_L32	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name			flt_mac0[31:0]	
reset	0 0			
Bits	Access	Name	Description	
[31:0]	RW	flt_mac0[31:0]	过滤表 MAC0 的低 32 位。	

GLB_MAC0_H16

GLB_MAC0_H16 为 MAC 过滤器 0 寄存器。该寄存器不支持软复位。

	Offset	Address	Register Name	Total Reset Value
		0x025C	GLB_MAC0_H16	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name		reserved	flt_mac0[47:31]	
reset	0 0			
Bits	Access	Name	Description	
[31:16]	-	reserved	保留。	
[15:0]	RW	flt_mac0[47:32]	过滤表 MAC0 的高 16 位。	

GLB_MAC1_L32

GLB_MAC1_L32 为 MAC 过滤器 1 寄存器。该寄存器不支持软复位。

GLB_MAC1_H16

GLB_MAC1_H16 为 MAC 过滤器 1 寄存器。该寄存器不支持软复位。

5.7.5 ETH 统计结果寄存器

统计结果可配置读清零和只读 2 种模式（[MAC_SET\[cntr_rdclr_en\]](#)设置为 1 表示为读清零模式，设置为 0 表示为只读模式）。

现在所列地址为相对地址（统计接收结果的地址），实际地址为统计计数缓存地址与相对地址的和。

DROPEVENTS

该寄存器不支持软复位。



Offset Address			Register Name			Total Reset Value																										
0x0300			DROPEVENTS			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dropevents																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dropevents	帧接收的过程中，RXFIFO 溢出事件的累计次数。																													

CRCERR

该寄存器不支持软复位。

Offset Address			Register Name			Total Reset Value																										
0x0304			CRCERR			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	crcerr																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	crcerr	接收帧的帧长有效（非超短、超长帧），但其 CRC 或 Alignment 检查出错的帧的个数。																													

ABNORMALSIZEPKTS

该寄存器不支持软复位。

Offset Address			Register Name			Total Reset Value																										
0x0308			ABNORMALSIZEPKTS			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	abnormalsizepkts																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	abnormalsizepkts	帧长无效（小于设定的最小有效帧长，或者帧长大于设定的最大有效帧长）的个数（超短帧、超长帧）。																													

RXDVRISE

该寄存器不支持软复位。

BROADCASTPKTS

该寄存器不支持软复位。

Offset Address		Register Name	Total Reset Value
0x0310		BROADCASTPKTS	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	broadcastpkts		
reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	broadcastpkts	帧长有效并且 CRC 正确的广播帧的计数，但不包括流控帧、传输错的帧。

MULTICASTPKTS

该寄存器不支持软复位。

Offset Address		Register Name	Total Reset Value
0x0314		MULTICASTPKTS	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	multicastpkts		
reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	multicastpkts	帧长有效并且 CRC 正确的组播帧的计数，但不包括流控帧、传输错的帧。



IFINUCASTPKTS

该寄存器不支持软复位。

	Offset Address																														Total Reset Value		
	0x0318																													0x0000_0000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	ifinucastpkts																																
reset	0 0																																
Bits	Access	Name	Description																														
[31:0]	RO	ifinucastpkts	帧长有效并且 CRC 正确的单播帧的计数，但不包括流控帧、传输错的帧。																														

IFINERRORS

该寄存器不支持软复位。

	Offset Address																															Total Reset Value	
	0x031C																														0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	ifinerrors																																
reset	0 0																																
Bits	Access	Name	Description																														
[31:0]	RO	ifinerrors	接收的所有错帧计数，包括 CRC 错的帧、超短帧、超长帧和传输错的帧。																														

DOT3ALIGNMENTERR

该寄存器不支持软复位。

	Offset Address																													Total Reset Value		
	0x0320																												0x0000_0000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3alignmenterr																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3alignmenterr	接收到的奇数个 nibble 的 CRC 错的帧。																													

DOT3PAUSE

该寄存器不支持软复位。

	Offset Address																														Total Reset Value	
	0x0324																													0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3pause																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3pause	接收的流控帧数。																													

DOT3DRIBBLE

该寄存器不支持软复位。

	Offset Address																														Total Reset Value	
	0x0328																													0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3dribble																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3dribble	接收的奇数个 nibble 的 CRC 正确的帧。																													



PKTS

该寄存器不支持软复位。

FLT_DROP_CNT

该寄存器不支持软复位。

Offset Address		Register Name	Total Reset Value
0x0330		FLT_DROP_CNT	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	flt_drop_cnt		
reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	flt_drop_cnt	查过滤表丢弃的正确帧计数，不包括超短帧、超长帧、CRC错的帧、流控帧和传输错的帧。

LOCAL_MAC_MATCH

该寄存器不支持软复位。

			不包括超短帧、超长帧、CRC 错的帧、流控帧和传输错的。
--	--	--	------------------------------

FLUX_FRAME_CNT

该寄存器不支持软复位。

	Offset Address	Register Name	Total Reset Value
Bit	0x0338	FLUX_FRAME_CNT	0x0000_0000
reset	0 0		
Name	flux_frame_cnt		
Bits	Access	Name	Description
[31:0]	RO	flux_frame_cnt	流量限制输入正确帧的总数计数，不包括超短帧、超长帧、CRC 错的帧、流控帧和传输错的帧。

FLUX_DROP_CNT

该寄存器不支持软复位。

	Offset Address	Register Name	Total Reset Value
Bit	0x033C	FLUX_DROP_CNT	0x0000_0000
reset	0 0		
Name	flux_drop_cnt		
Bits	Access	Name	Description
[31:0]	RO	flux_drop_cnt	流量限制丢弃正确帧的计数，不包括超短帧、超长帧、CRC 错的帧、流控帧和传输错的帧。

FLTFLUX_REC_FRAME

该寄存器不支持软复位。



FLTFLUX_DROP_FRAME

该寄存器不支持软复位。

IFIN OCTETS

该寄存器不支持软复位。

		未检测到有效 SFD (Start of Frame Delimiter) 的帧不进行统计。
--	--	--

OCTETS_RX

该寄存器不支持软复位。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	OCTETS_RX	0x0000_0000
Name	octets_rx		
reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	octets_rx	接收字节计数，包括正确帧和错误帧，但不包括 Preamble 字节。未检测到有效 SFD 的帧不进行统计。

BROADCASTPKTS_TX

该寄存器不支持软复位。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	BROADCASTPKTS_TX	0x0000_0000
Name	broadcastpkts_tx		
reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	broadcastpkts_tx	发送成功的广播帧的计数（不含重传）。

MULTICASTPKTS_TX

该寄存器不支持软复位。



RETRY_TIMES_TX

该寄存器不支持软复位。

IFOUTUCASTPKTS TX

该寄存器不支持软复位。

Register Name																Total Reset Value																
0x038C																0x0000_0000																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ifoutucastpkts_tx																															
reset	0 0																															
Bits	Access	Name			Description																											
[31:0]	RO	ifoutucastpkts_tx			发送成功的单播帧的计数（不含重传）。																											

DOT3COL OK

该寄存器不支持软复位。

	Offset Address																													Total Reset Value		
	0x0390																												0x0000_0000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3col_ok																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3col_ok	发生冲突后发送成功的帧数目。																													

DOT3LATECOL

该寄存器不支持软复位。

	Offset Address																														Total Reset Value	
	0x0394																													0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3latecol																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3latecol	发生 Late collision 的帧数目。																													

DOT3EXCESSIVECOL

该寄存器不支持软复位。

	Offset Address																														Total Reset Value	
	0x0398																										0x0000_0000					
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3excessivecol																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3excessivecol	由于重传次数大于 15 而丢弃的帧数目。																													

DOT3COLCNT

该寄存器不支持软复位。



	Offset Address																													Total Reset Value		
	0x039C																												0x0000_0000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dot3colcnt																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	dot3colcnt	冲突次数等于冲突阈值的帧数，由 MAC_SET [colthreshold] 设定。																													

PKTS_TX

该寄存器不支持软复位。

	Offset Address																														Total Reset Value	
	0x03A0																													0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	pkts_tx																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	pkts_tx	所有配置发送帧的计数，不包含超时丢弃帧以及 GLB_EQDES_LEN 配置长度不在有效范围之内的发送帧。																													

COLLISIONS

该寄存器不支持软复位。

	Offset Address																														Total Reset Value	
	0x03A4																													0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	collisions																															
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	collisions	Collision 发生的次数。																													

DOT3OUTPAUSE

该寄存器不支持软复位。

	Offset Address																															Register Name	Total Reset Value				
	0x03A8																															DOT3OUTPAUSE	0x0000_0000				
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Name	dot3outpause																																				
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description																																		
[31:0]	RO	dot3outpause	发送的流控帧数目。																																		

OCTETS_TX

该寄存器不支持软复位。

	Offset Address																															Register Name	Total Reset Value				
	0x03AC																															OCTETS_TX	0x0000_0000				
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Name	octets_tx																																				
reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description																																		
[31:0]	RO	octets_tx	发送字节计数，含重发帧以及正确帧和错误帧，但 Preamble 字节不计在内。																																		



6 视频接口

关于本章

本章描述内容如下表所示。

标题	内容
6.1 视频输入单元	介绍 VI 模块的功能、特点、工作方式、寄存器概览和寄存器描述。
6.2 视频输出单元	介绍 VO 模块的功能、特点、工作方式、寄存器概览和寄存器描述。

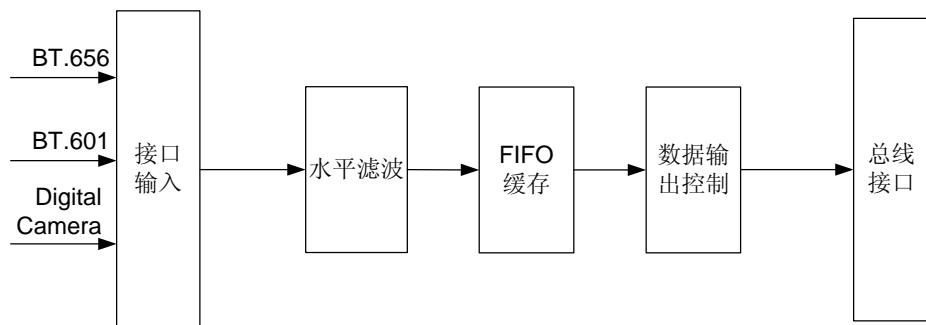


6.1 视频输入单元

6.1.1 概述

视频输入单元 VIU (Video Input Unit) 可以通过 BT.656/601 接口和 DC (Digital Camera) 接口接收视频数据，存入指定的内存区域。在此过程中，VIU 可以对视频图像数据进行水平 1/2 的缩小 (Down Scaling) 和色度重采样 (Chrominance Resampling)。VIU 的功能框图如图 6-1 所示。

图6-1 VIU 功能框图



6.1.2 特点

VIU 有以下特点：

- 支持 BT.656/BT.601 标准。
- 通道 0 和通道 2 支持 BT.601 模式的输入。
- 支持同时 4 通道的视频输入。
- 支持视频数据时分复用模式输入。
- 支持最多 8 路数据同时接收和处理，每一路可以独立配置。
- 支持场接收模式和帧接收模式。
- 输入数据支持 CbYCrY、CrYCbY、YCbYCr 和 YCrYCb4 种顺序。
- 通道 0 和通道 2 支持直接 digital camera 连接（最大支持 300 万像素）。
- 支持 1/2 水平 down scaling。
- 支持色度重新采样，且支持 co-sited 格式到 interspersed 格式的水平转换。
- 支持 8bit 和 10bit 的数据位宽。
- 支持在一个特定窗口内获取数据。
- 支持图像块填充。
- 输出支持存储模式：
 - package YCbCr 4:2:2 模式。
 - SPYCbCr 4:2:0 和 SPYCbCr 4:2:2 模式。



- planar YCbCr 4:2:0 和 planar YCbCr 4:2:2 模式。
- 输出数据格式支持 raw data 存储模式，该模式只用在高清模式下。
- 支持图象亮度统计。
- 提供 AHB Master 接口，用于直接把数据写入 DDR 中。
- 提供 AHB Slave 接口，用于软件对 VIU 的寄存器配置和状态信息读取。

6.1.3 信号描述



说明

此处以 Hi3511 芯片为例进行介绍，Hi3512 芯片的相关内容请参见“14 Hi3511 与 Hi3512 差异说明”。

表 6-1 描述了视频输入接口的外部输入输出管脚信号。

表6-1 视频输入接口信号

信号名	方向	描述	对应管脚
vi_datain_ch0[9]	I	视频通道 0 输入数据。	VI0DAT9
vi_datain_ch0[8]			VI0DAT8
vi_datain_ch0[7]			VI0DAT7
vi_datain_ch0[6]			VI0DAT6
vi_datain_ch0[5]			VI0DAT5
vi_datain_ch0[4]			VI0DAT4
vi_datain_ch0[3]			VI0DAT3
vi_datain_ch0[2]			VI0DAT2
vi_datain_ch1[7]	I	视频通道 1 输入数据。	VI1DAT7
vi_datain_ch1[6]			VI1DAT6
vi_datain_ch1[5]			VI1DAT5
vi_datain_ch1[4]			VI1DAT4
vi_datain_ch1[3]			VI1DAT3
vi_datain_ch1[2]			VI1DAT2
vi_datain_ch1[1]			VI1DAT1
vi_datain_ch1[0]			VI1DAT0
vi_datain_ch2[9]	I	视频通道 2 输入数据。	VI2DAT9
vi_datain_ch2[8]			VI2DAT8
vi_datain_ch2[7]			VI2DAT7
vi_datain_ch2[6]			VI2DAT6

信号名	方向	描述	对应管脚
vi_datain_ch2[5]			VI2DAT5
vi_datain_ch2[4]			VI2DAT4
vi_datain_ch2[3]			VI2DAT3
vi_datain_ch2[2]			VI2DAT2
vi_datain_ch3[7]	I	视频通道 3 输入数据。	VI3DAT7
vi_datain_ch3[6]			VI3DAT6
vi_datain_ch3[5]			VI3DAT5
vi_datain_ch3[4]			VI3DAT4
vi_datain_ch3[3]			VI3DAT3
vi_datain_ch3[2]			VI3DAT2
vi_datain_ch3[1]			VI3DAT1
vi_datain_ch3[0]			VI3DAT0
clk_vi_ch0	I	通道 0 视频输入时钟。	VI0CK
clk_vi_ch1	I	通道 1 视频输入时钟。	VI1CK
clk_vi_ch2	I	通道 2 视频输入时钟。	VI2CK
clk_vi_ch3	I	通道 3 视频输入时钟。	VI3CK
vi_hsync_vd_ch0	I	第 0 通道水平同步脉冲或者数据有效信号。 VI_CH_CFG[9] 控制该信号是同步脉冲或者数据有效电平。 ch0_hsync=0: 数据有效电平, 高电平或者低电平有效可配。 ch0_hsync=1: 同步脉冲, 高电平或者低电平有效可配。	VI0HS



信号名	方向	描述	对应管脚
vi_hsync_vd_ch2	I	<p>第 2 通道水平同步脉冲或者数据有效信号。</p> <p>VI_CH_CFG[8]控制该信号是同步脉冲或者数据有效电平。</p> <p>ch2_hsync=0: 数据有效电平, 高电平或者低电平有效可配。</p> <p>ch2_hsync=1: 同步脉冲, 高电平或者低电平有效可配。</p>	VI2HS
vi_vsync_field_ch0	I	<p>第 0 通道垂直同步脉冲或者场指示信号。</p> <p>VI_CH_CFG[13]控制该信号是同步脉冲或者场指示信号。</p> <p>ch0_vsync=0: 场指示信号, 高电平或者低电平有效可配。</p> <p>ch0_vsync=1: 同步脉冲, 高电平或者低电平有效可配。</p>	VI0VS
vi_vsync_field_ch2	I	<p>第 2 通道垂直同步脉冲或者场指示信号。</p> <p>VI_CH_CFG[12]控制该信号是同步脉冲或者场指示信号。</p> <p>ch2_vsync=0: 场指示信号。</p> <p>ch2_vsync=1: 同步脉冲。</p>	VI2VS

注:

第 0 通道和第 2 通道支持的数据格式: 单路 BT.601、单路 BT.656、两路时分复用 BT.656、四路时分复用 BT.656、单路 digital camera。

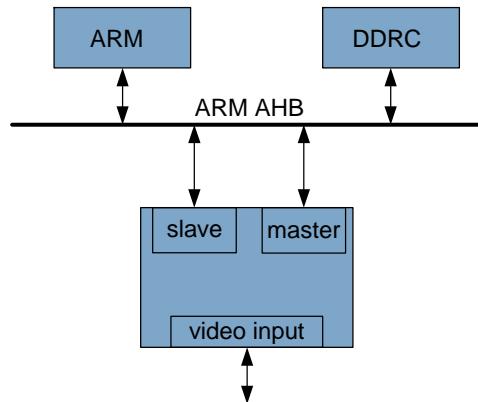
第 1 通道和第 3 通道支持的数据格式: 单路 BT.656, 两路时分复用 BT.656。

6.1.4 功能描述

6.1.4.1 典型应用

VIU 典型应用如图 6-2 所示。

图6-2 VIU 典型应用图



VIU 是一个支持多种时序输入的视频输入采集单元，将采集到的视频数据存储到 DDR 中，系统可以配置不同的功能模式，使之可以灵活的适应不同的外部输入视频接口，支持多种外部输入设备。

6.1.4.2 功能原理

VIU 模块支持以下 2 组输入时序：

- BT.656/601 YCbCr 4:2:2 标准
 - 像素时序
 - 行时序
 - 帧时序
 - 数字摄像头
 - 水平时序
 - 垂直时序

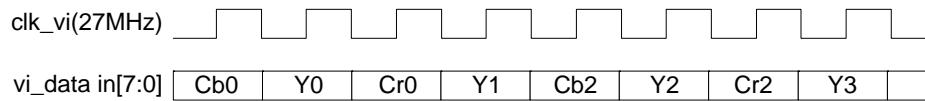
BT.656/601 YCbCr 4:2:2

1. 像素时序

在 BT.656/601 YCbCr 4:2:2 建议中：

- 亮度信号与色度信号的采样比例为 2:1, 2 个亮度信号共用 1 个色度 CbCr 信号。
 - 推荐每行亮度采样 720 个有效图像像素点, 色度采样 360 个有效图像像素点, 色度在对应偶数像素点采样 (起始采样点为 0 像素点), 以 co-sited 格式采样。
 - 亮度色度信号在同一个 8bit 通道内传输的情况下, 传输顺序为: Cb0 Y0 Cr0 Y1 Cb2 Y2 Cr2 Y3……Cb718 Y718 Cr718 Y719。时钟频率为 27MHz。图 6-3 为像素输入时序关系图。

图6-3 像素输入时序

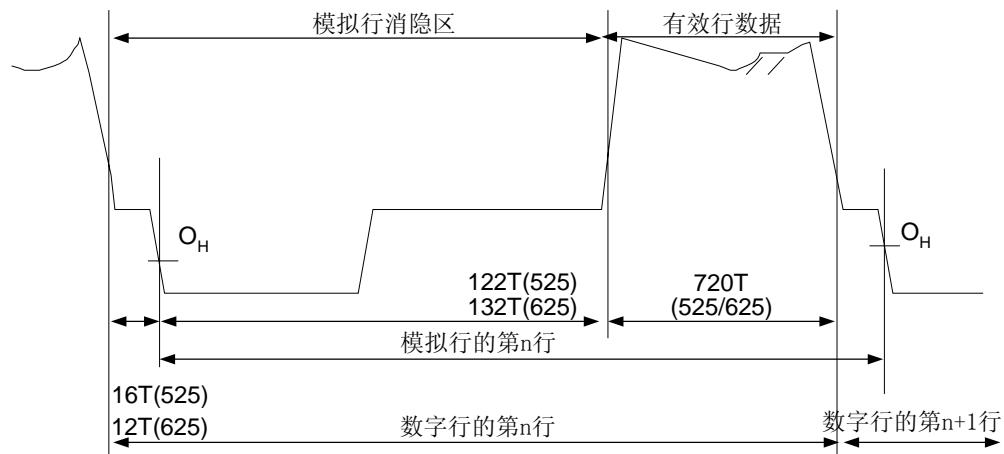




2. 行时序

BT.656/601 建议规定了 PAL (Phase-Alternation Line) 制式和 NTSC (National Television System Committee) 制式一行的整个采样数 (包括行消隐区), PAL 制采样数为 864, NTSC 制采样数为 858。数据有效行的 720 个亮度取样与 PAL (525 行)、NTSC (625 行) 制式的模拟同步基准间的关系如图 6-4 所示 (其中 T 表示像素周期, O_H 表示行同步前沿, 半幅度基准)。

图6-4 模拟全电视信号对应的数字行采样时序

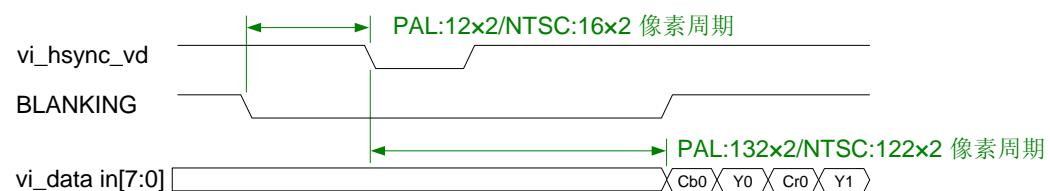


BT.656/601 采用不同的方法标志行同步信号:

- BT.601 建议

行同步由行同步信号 HSYNC 产生, HSYNC 的下降沿 (上升沿、下降沿可编辑) 表示新的一行开始, HSYNC 同步脉冲在图 6-4 的 O_H 处产生。行同步脉冲的宽度可编辑。ITU-R BT.601 的行时序如图 6-5 所示。图中“BLANKING”表示行消隐, VI 中无此管脚, 仅用此信号来表示有效数据。

图6-5 ITU-R BT. 601 行时序图



- BT.656 建议

同步信号集成在数据流中, 数据流中的特殊字节 SAV (Start of Active Video) 和 EAV (End of Active Video) 分别表示有效行数据的开始和结束。在视频数据流中, 由 FF 00 00 (FF、00 为图像编码数据的保留字节, 为非图像数据) 构成的定时基准码字的码头来标志紧接着的一个字节为 SAV 或者 EAV, BT.656 的行数据流格式如图 6-6 所示。

SAV 和 EAV 依据 SAV/EAV 字节内的特殊位“H”区分。SAV/EAV 还包含了垂直消隐区“V”和场号“F”。SAV/EAV 的具体描述如表 6-2 所示。

在 BT.656 建议中，采用了 8 个有效保留位来定义有效的 SAV 和 EAV，4 个有效保留位起纠错的作用（发生 1bit 错误，可以纠错；发生 2bit 错误，可以报错）。有效的 SAV/EAV 值如表 6-3 所示。

图6-6 BT.656 YCbCr 4:2:2 行数据格式

行消隐期		定时基准码				720 有效像素 YCbCr 4:2:2										定时基准码				行消隐期	
...	10	FF	00	00	SAV	Cb0	Y0	Cr0	Y1	...	Y719	FF	00	00	EAV	80	...				

表6-2 SAV/EAV Format

Bit7	Bit6 (F)	Bit5 (V)	Bit4 (H)	Bit[3:0]
1 (固定值)	场指示位 F=0: 1st field F=1: 2nd field	垂直消隐位 V=0: Active video V=1: VBI (Vertical Blanking Interval)	H=0 in SAV H=1 in EAV	保留

表6-3 有效 SAV/EAV 值

编码	二进制值	场号	垂直消隐期
SAV	1000 0000	1	否
EAV	1001 1101	1	否
SAV	1010 1011	1	是
EAV	1011 0110	1	是
SAV	1100 0111	2	否
EAV	1101 1010	2	否
SAV	1110 1100	2	是
EAV	1111 0001	2	是

3. 帧时序

在 BT.601 和 BT.656 建议中，规定 PAL 和 NTSC 制式 TV 图像帧为隔行扫描。



表6-4 PAL 和 NTSC 制式 TV 图像帧对比

制式	行/帧	帧/秒
PAL 制	625 行	每秒 25 帧, 每帧分奇偶场
NTSC 制	525 行	每秒 30 帧, 每帧分奇偶场

对于 BT.601 建议, 把 SYNC、FIELD 信号作为垂直同步信号、场同步信号。

对于 BT.656 建议, 垂直同步信号包含在 SAV 和 EAV 内。

- ITU-R BT.601 垂直时序

ITU-R BT.601 建议: 信号 VSYNC/FIELD 作为垂直同步信号。VSYNC 的脉冲或者 FIELD 的跳变标志奇偶场的开始。

图 6-7 和图 6-8 分别为 VI 在 NTSC 制式和 PAL 制式的垂直时序关系图。其中 vi_hsync_vd 为水平同步脉冲, vi_vsync_field 在 vsync=1 时为垂直同步脉冲; 在 vsync=0 时为场同步信号。

在 NTSC 隔行扫描制式下, 第 1 场的垂直同步信号在第 4 行的起始位置变为低电平, 持续 3 行低电平后, 在第 7 行的起始位置变为高电平。VIU 接收从第 22 行开始到第 261 行结束的 240 行数据。第 2 场的垂直同步信号在第 266 行的中间位置变为低电平, 持续 3 行低电平后, 在第 269 行中间位置变为高电平。VIU 接收从第 285 行开始第 524 行结束的 240 行数据。

在 PAL 隔行扫描制式下, 第 1 场的垂直同步信号在第 1 行起始位置变为低电平, 持续 2.5 行低电平后, 在第 3 行的中间位置变为高电平。VIU 接收从第 24 行开始到第 310 行结束的 288 行数据。第 2 场的垂直同步信号在第 313 行的中间位置变为低电平, 持续 2.5 行低电平后, 在第 316 行的起始位置变为高电平。VIU 接收从第 336 行开始到第 623 行结束的 288 行数据。在所有行中, 有效行数据如表 6-5 所述。

图6-7 NTSC 制式垂直同步时序图

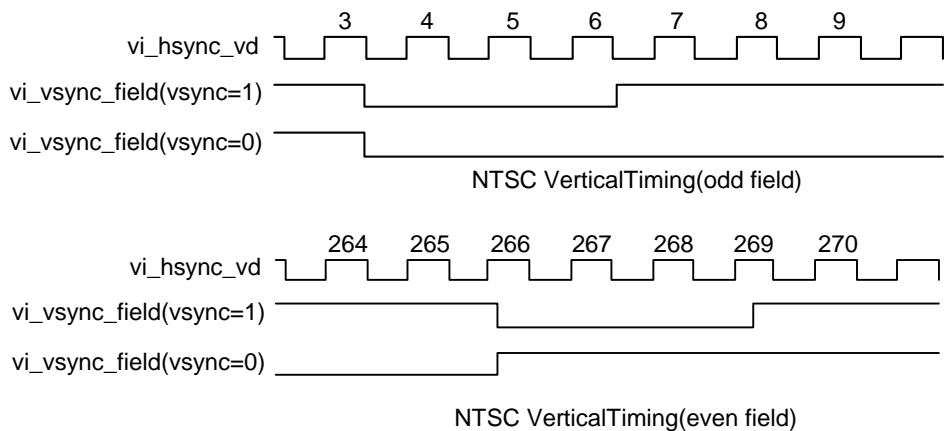




图6-8 PAL 制式垂直同步时序图

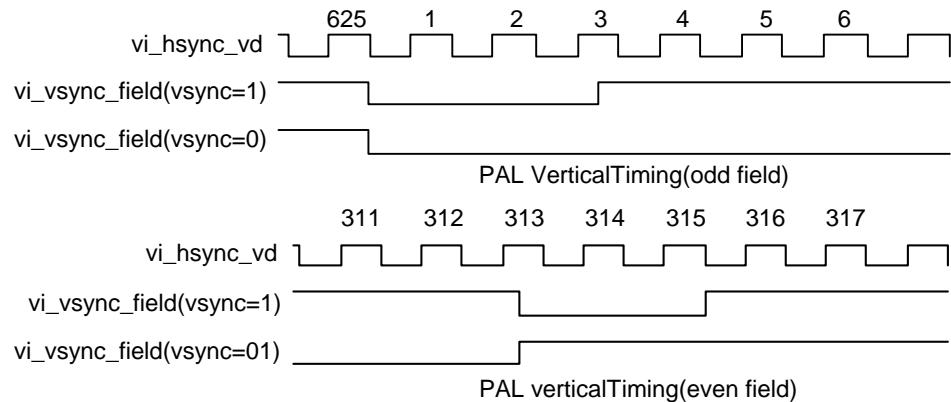


表6-5 BT.601 一帧有效行数据

制式	场	有效行
NTSC	奇数场	22~261
	偶数场	285~524
PAL	奇数场	23~310
	偶数场	336~623

- BT.656 垂直时序

BT.656 隔行扫描帧时序定义如表 6-6 所示。F=1 表示第二场（偶数场），F=0 表示第一场（奇数场）。V=1 表示垂直消隐区，V=0 表示有效行数据区。

表6-6 BT.656 帧时序

行号		F 位	V 位	描述
525/60				
1~3	624~625	1	1	第二场垂直消隐。
4~19	1~22	0	1	第一场垂直消隐，SAV/EAV 改变为显示第一场。
20~263	23~310	0	0	第一场有效数据行。
264~265	311~312	0	1	第一场垂直消隐。
266~282	313~335	1	1	第二场垂直消隐，SAV/EAV 改变为显示第二场。
283~525	336~623	1	0	第二场有效数据行。



4. BT.656 多路时分复用时序

VIU 除了支持单路标准 BT.656 时序外, 还支持两路或四路 BT.656 通过时分复用, 使用一个通道进行传输的时序, 具体如图 6-9 和图 6-10 所示。

图6-9 两路 BT.656 时分复用时序

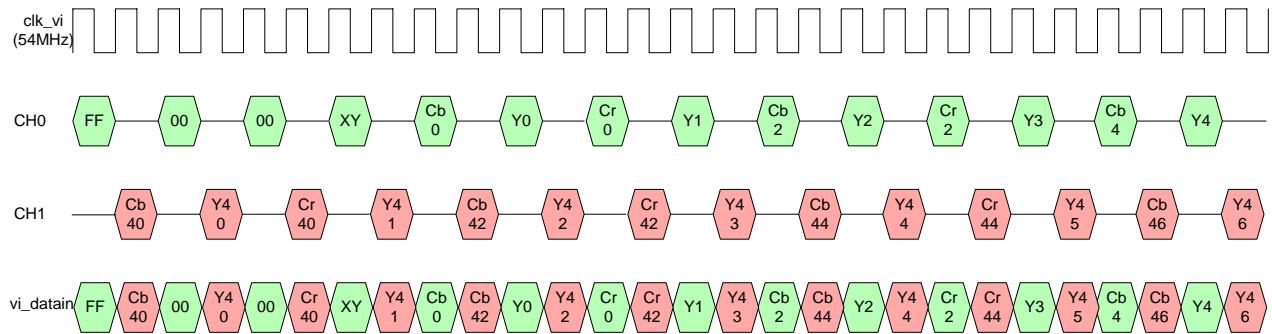
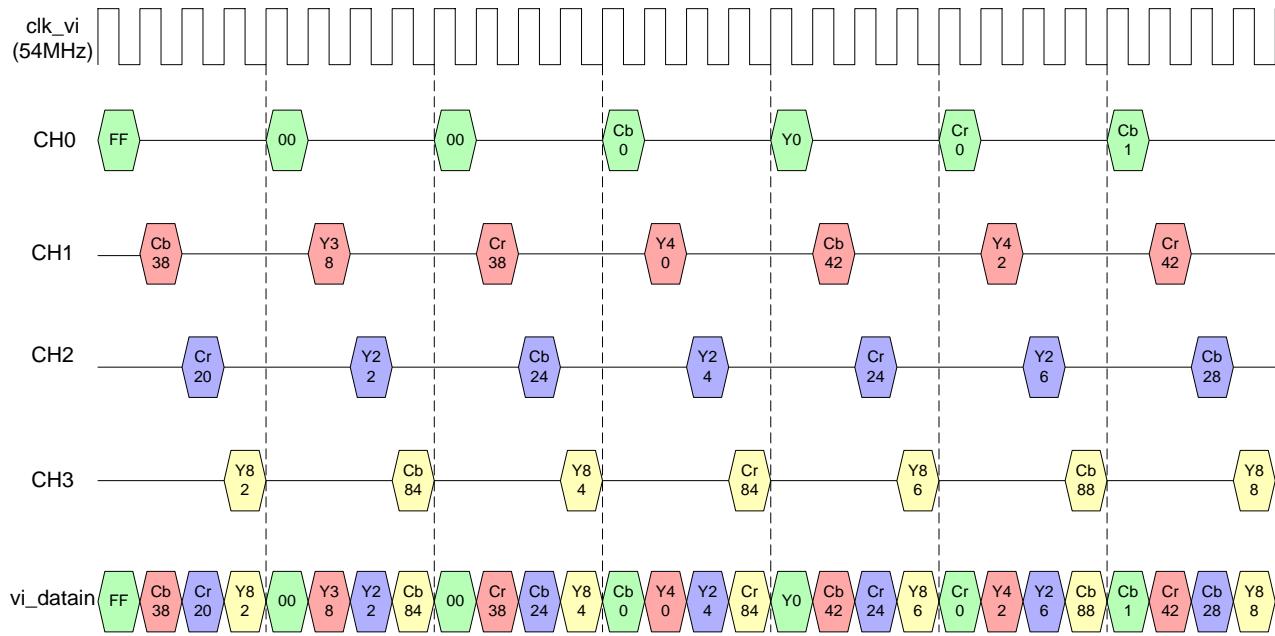


图6-10 四路 BT.656 时分复用时序

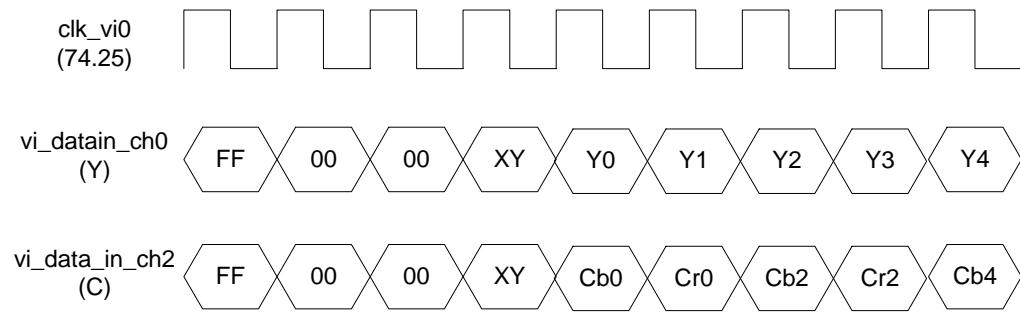


当输入数据为时分复用格式时, SAV/EAV 中纠错位的位置则用来表示输入数据属于哪路的信息。

高清接口

VIU 支持 720p 高清接口输入时序, 通道 0 连接亮度通道, 通道 2 连接色度通道, 具体时序如图 6-11 所示。

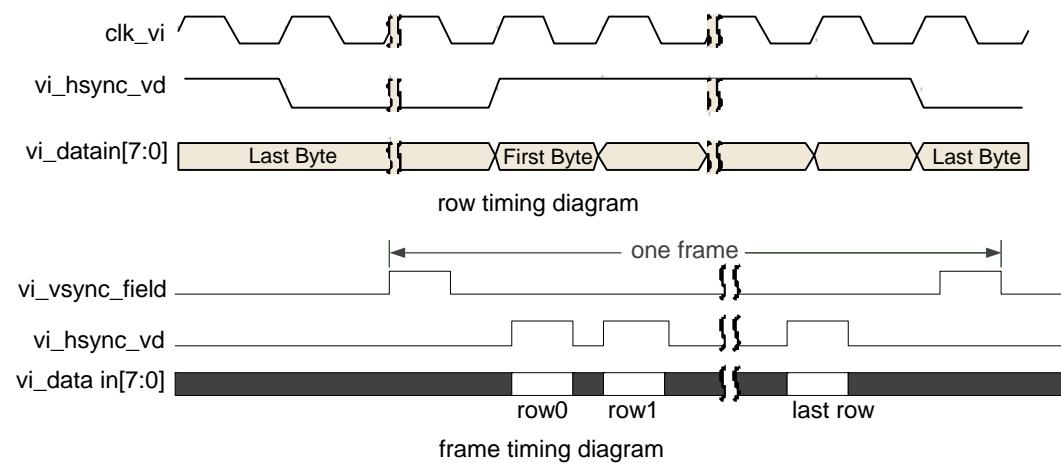
图6-11 高清接口输入时序



数字摄像头接口

VIU 支持分辨率最大为 QXGA (Quantum Extended Graphics Array) (2048x1536) 的摄像头数据传输。数字摄像头支持的水平和垂直时序如图 6-12 所示。

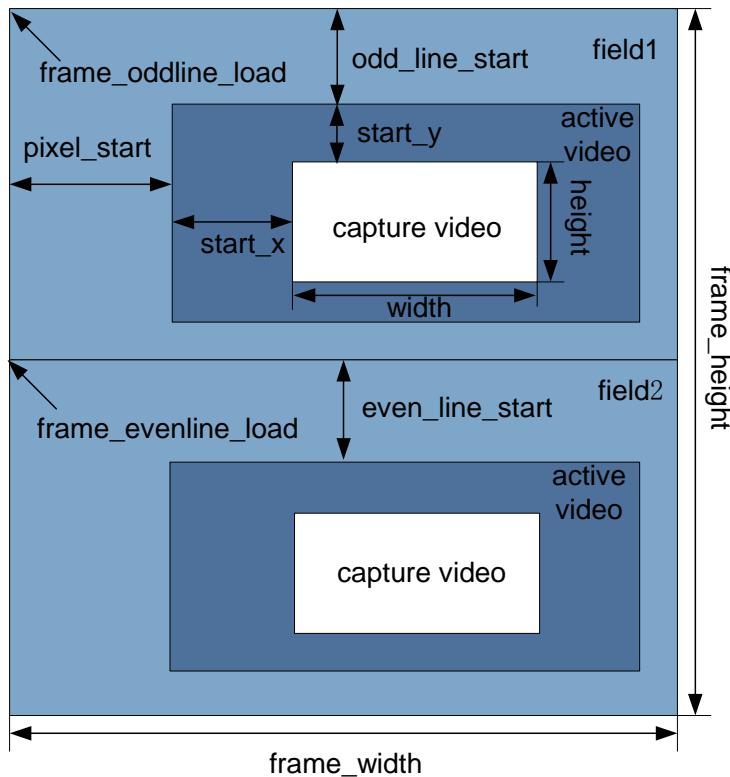
图6-12 数字摄像头支持的水平、垂直时序图



视频有效范围

有效视频范围如图 6-13 所示，开始于水平消隐和垂直消隐之后。而实际显示的视图区域常常包含在有效视频范围之内，与有效视频的边界有一点点缩小，其目的是避免边缘效应。图 6-13 中的参数详细定义请参见寄存器详细描述。

图6-13 有效图像区域与水平垂直消隐关系图



图像数据采集

VIU 模块的主要任务就是把视频数据流采集并存储到 DDR 中；VIU 内部，在水平方向上支持对亮度以及色度的 1/2 缩小。同时也支持色度空间的 co-sited 格式到 interspersed 格式的转换。

1. 两种 YCbCr 4:2:2 的色度采样方式

在水平方向上 co-sited 和 interspersed 格式色度关系如图 6-14 和图 6-15 所示。

图6-14 YCbCr 4:2:2 co-sited 采样格式

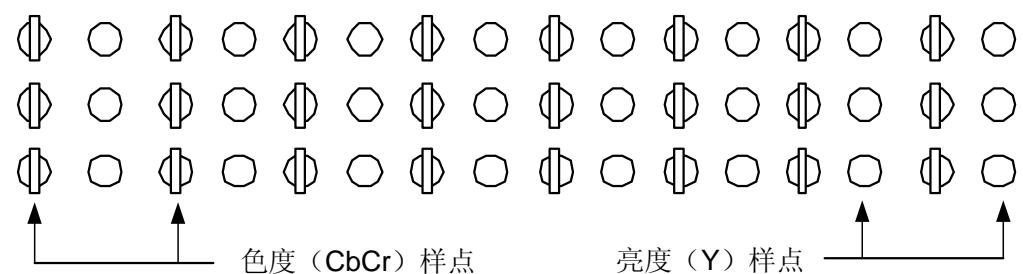




图6-15 YCbCr 4:2:2 interspersed 采样格式



2. co-sited 格式到 interspersed 格式的转换

BT.656/BT.601 YCbCr 4:2:2 标准接口输入的是 co-sited 格式数据。VIU 支持 co-sited 方式直接采集存储和转换成 interspersed 方式存储。

- co-sited 采样

直接将输入图像的亮度和色度分量写入内存，对亮度和色度分量的采样位置不作任何修改，因此，缓存中的图像采样的亮度和分量和图 6-14 相同，仍然是 YCbCr 4:2:2 co-sited 格式。

- interspersed 采样

色度信号被重新采样，使得色度分量采样在空间上位于亮度分量的中间，如图 6-15 所示 interspersed 格式。

3. 水平方向 1/2 缩小

水平方向 1/2 缩小时，VIU 将输入图像的水平分辨率 (Y、Cb、Cr) 降低一半：

- 对于亮度 (Y) 分量，用 7 阶 FIR 滤波器进行插值滤波。
- 对于色度分量 (Cb, Cr)，用 4 阶 FIR 滤波器进行插值滤波。

水平方向 1/2 缩小时，可同时进行 co-sited 格式到 interspersed 格式的转换。

图像存储模式

图像存储模式包括：

- semi-planar YCbCr 存储
- planar YCbCr 存储
- package 存储
- raw data 存储

1. semi-planar YCbCr 存储

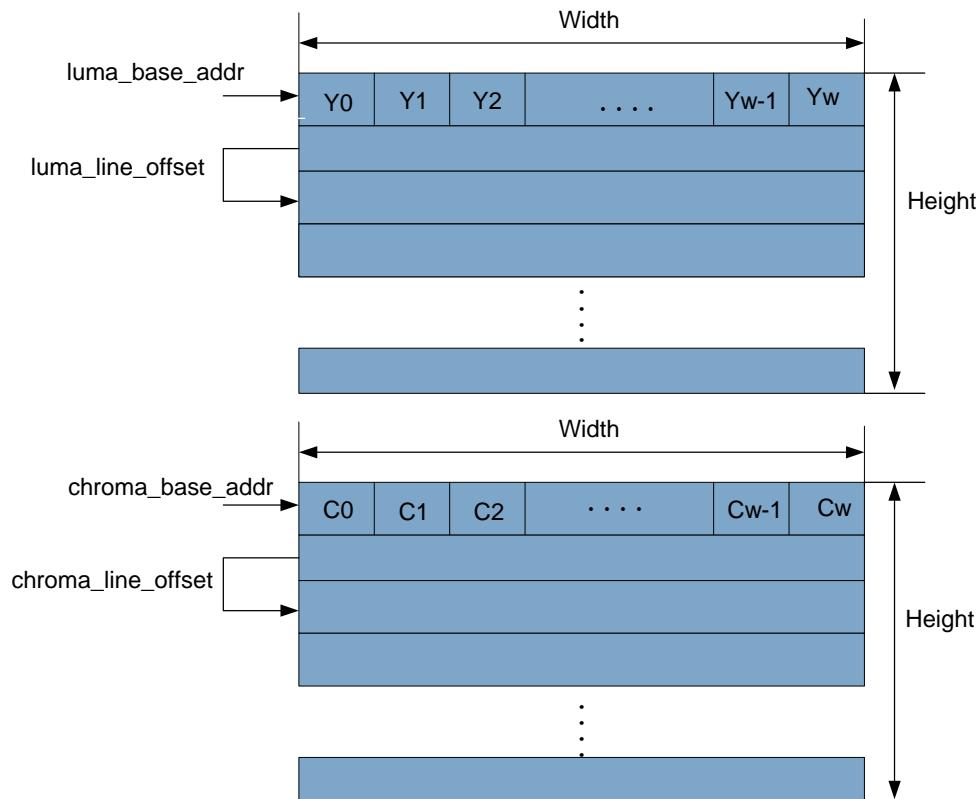
系统设定了视图区域后，对读入数据按照 semi-planar 方式存储，即亮度分量和色度分量分别存储在 DDR 中的亮度存储空间和色度存储空间。

- 在 1 行内，亮度、色度分量各自连续存储。



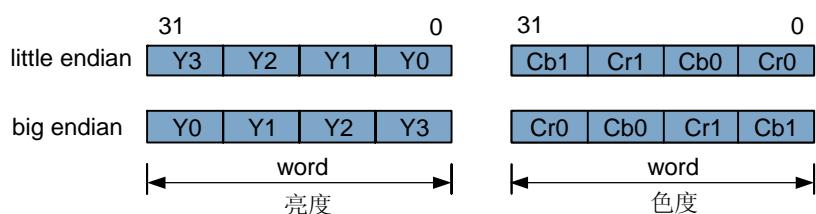
- 连续 2 行之间的存储，可以通过系统定义的行首与行首之间的存储间隔参数 offset 定义。亮度和色度分量在 DDR 中的存储位置由起始地址 base_addr 来指示。全分辨率采集的 YCbCr4:2:2 存储结构如图 6-16 所示。

图6-16 YCbCr4:2:2 的存储模式



在 DDR，数据的存储是以 word (32bit) 为单位。由 4 个 8bit 像素组成一个 32bit 的 word，在 4 个字节构成一个 word 时有 2 种方式：big endian 和 little endian。图 6-17 是以亮度和色度分量为例来说明 big endian 和 little endian 的存储方式。

图6-17 big endian 和 little endian 图像存储模式

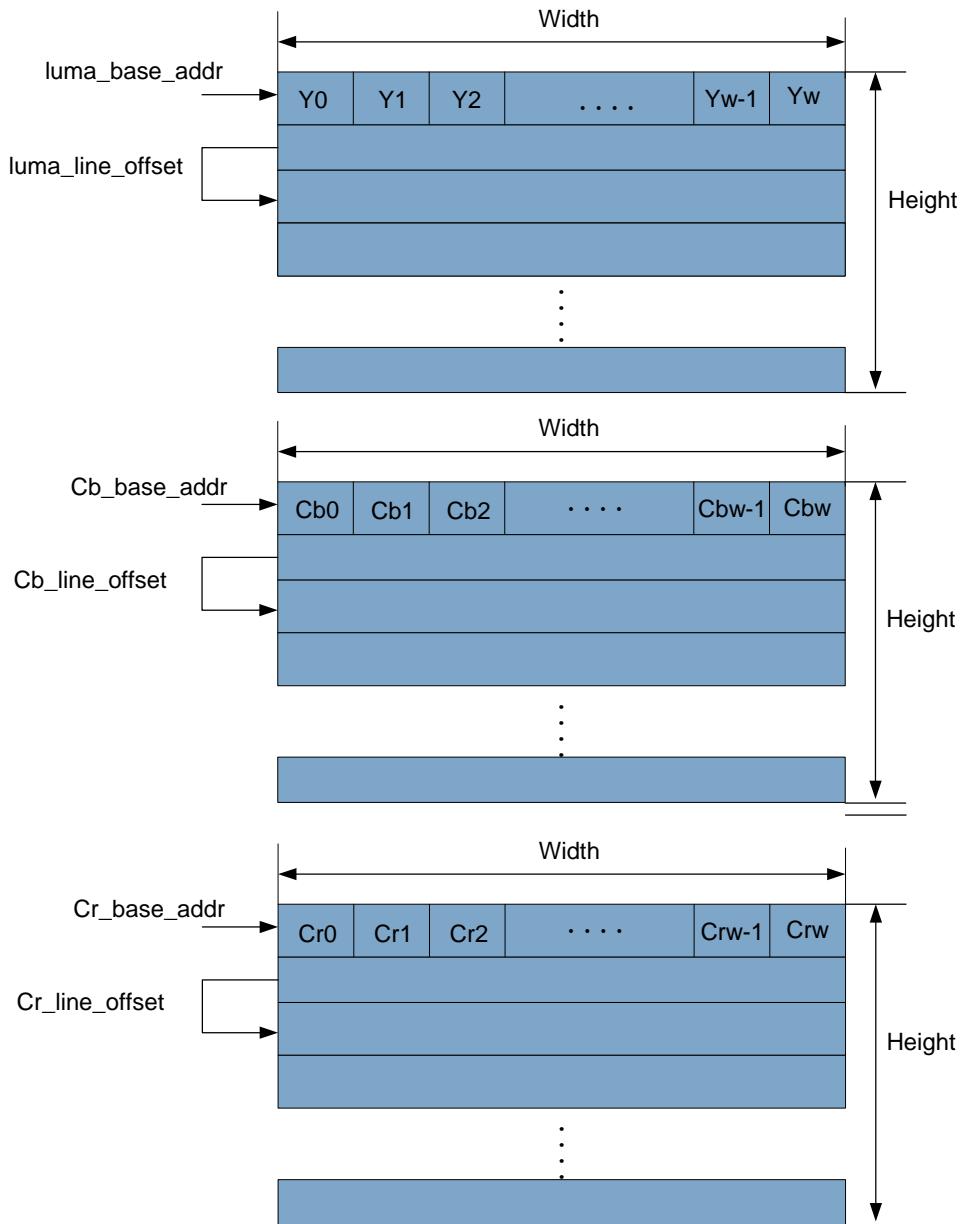


VIU 只支持采用 little endian 方式存储数据的 DDR。

2. planar YCbCr 存储

planar Y/Cb/Cr 存储方式将图象的 Y/Cb/Cr 三个分量分别存储。

图6-18 Y/Cb/Cr 图像存储模式

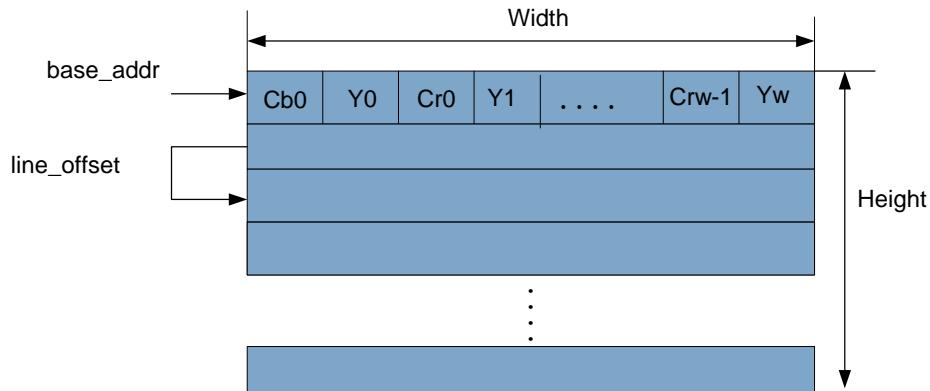


3. package 存储

package 方式存储是将 4:2:2 Y/Cb/Cr 数据按图 6-19 的方式存储。



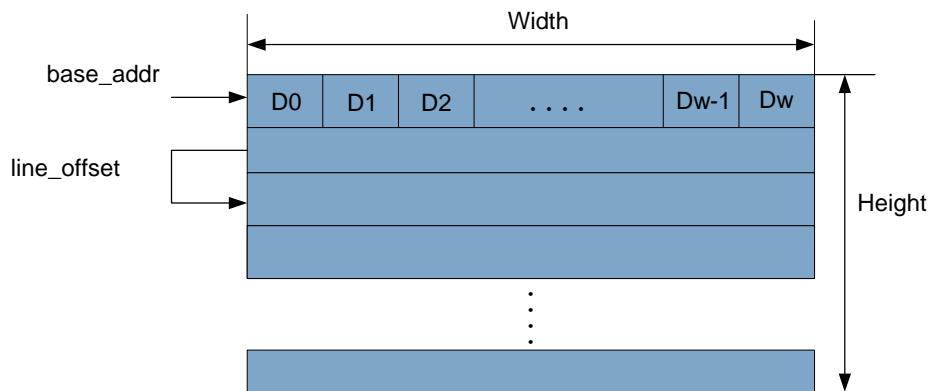
图6-19 图象存储 package 模式



4. raw data 存储

raw data 方式存储是将数据按数据顺序依次存放到一个 word 中。由于在 DDR 中，1 个 word 由 4 个 8bit 组成，当数据为 8bit 时，其存储方式如图 6-20 所示。

图6-20 raw data 8bit 存储模式



亮度统计

在 VIU 模块中，亮度统计是对当前图象平均亮度的估计，用于判断摄像头是否被遮蔽。其算法结构为：用一个 32 位的加法器来对输入亮度分量累加，当接收完一幅图象数据后，送出亮度的累加值。

6.1.5 工作方式

6.1.5.1 管脚复用配置

管脚复用配置请参见系统控制器章节的 SC_PERCTRL1[23]、SC_PERCTRL1[8:7]、SC_PERCTRL1[6]、SC_PERCTRL1[3]、SC_PERCTRL1[2]和 SC_PERCTRL1[1:0]。



6.1.5.2 软复位

通过 SC_PERCTRL0[18:15]控制 VI 的第 0~3 通道接口时钟域电路的软复位，通过 SC_PERCTRL0[14]控制 VI 模块 AHB 时钟域电路的软复位，请参见系统控制器章节的 SC_PERCTRL 寄存器。

6.1.5.3 中断

VIU 在系统中的中断号为“25”，在 VIU 中，每个内部通道有 8 个中断源，在 VIU 的寄存器中，每个通道有一个表示中断使能的寄存器 [VI_INT_MASK](#) 和经过使能控制后产生的中断状态寄存器 [VI_INT_STATUS](#)。

6.1.5.4 使用指南

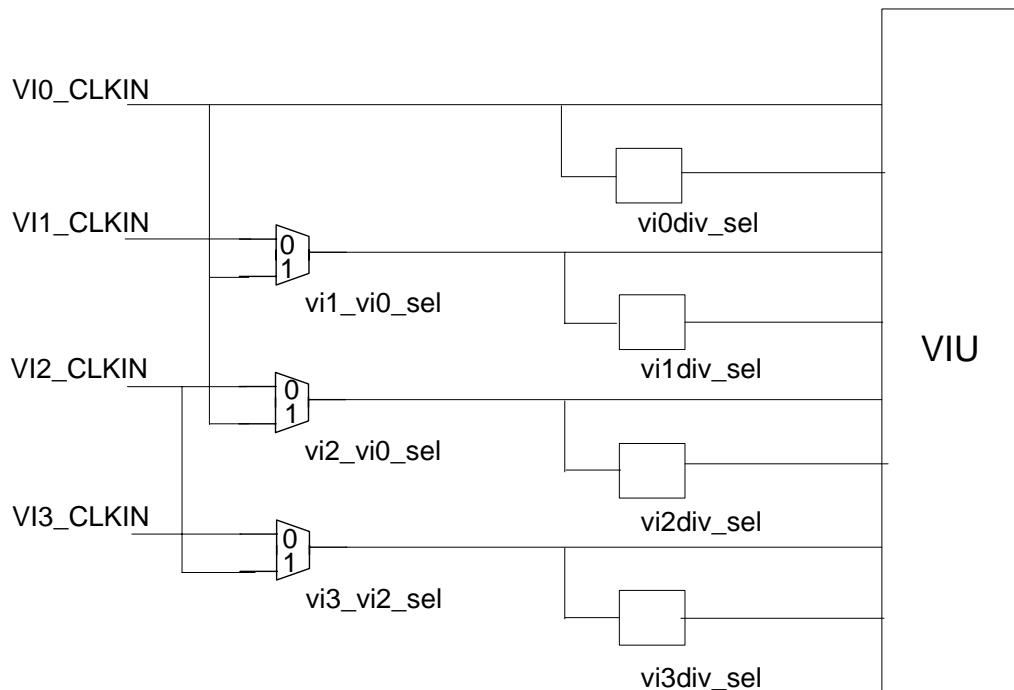
时钟配置方法

VIU 时钟配置主要在系统控制寄存器 SC_PERCTRL2（地址为 0x101E_0034）中完成，SC_PERCTRL2 的相关比特位的含义如下：

- vi1_vi0_sel、vi2_vi0_sel、vi3_vi2_sel 为端口时钟选择位，用来选择各个通道的时钟来源。**说明：**由于通道 0 的时钟始终来源于 VI0，因此通道 0 时钟来源无需配置；
- vi0_div_sel、vi1_div_sel、vi2_div_sel、vi3_div_sel 为分频时钟控制位，用来控制分频倍数。

VIU 的时钟选择框图如图 6-21 所示。

图6-21 VIU 的时钟选择框图





VIU 时钟配置方法如下：

- 单路 656 配置方法
 - 端口时钟选择位 (vi1_vi0_sel、vi2_vi0_sel、vi3_vi2_sel) 配置为 0, 即选择本通道时钟;
 - 分频时钟控制位 (vi0_div_sel、vi1_div_sel、vi2_div_sel、vi3_div_sel) 配置为 10, 即选择不分频。

以通道 1 输入为单路 656 为例, 则需配置 vi1_vi0_sel 为 0, 即选择 VI1_CLKIN, 配置 vi1div_sel 为 10, 即选择不分频。

- 2 路 656 配置方法
 - 端口时钟选择位 (vi1_vi0_sel、vi2_vi0_sel、vi3_vi2_sel) 配置为 0, 即选择本通道时钟;
 - 分频时钟控制位 (vi0_div_sel、vi1_div_sel、vi2_div_sel、vi3_div_sel) 配置为 00, 即选择 2 分频。

以通道 1 输入为 2 路 656 为例, 则需配置 vi1_vi0_sel 为 0, 即选择本通道输入时钟 VI1_CLKIN, 配置 vi1div_sel 为 00, 即 2 分频。

- 4 路 656 配置方法
 - 通道 0 输入为 4 路 656, 则 vi1_vi0_sel 配置为 1, 即通道 1 时钟来源于通道 0 的时钟 VI0_CLKIN, vi0div_sel 和 vi1div_sel 均配置为 01, 即选择 4 分频;
 - 通道 2 输入为 4 路 656, 则 vi2_vi0_sel 配置为 0, 即选择本通道时钟 VI2_CLKIN, vi3_vi2_sel 配置为 1, 即通道 3 时钟来源于通道 2 的时钟 VI2_CLKIN, vi2div_sel 和 vi3div_sel 均配置为 01, 即选择 4 分频。
- 601/Camera 模式配置方法

601/Camera 模式只能存在于通道 0 和通道 2, 时钟配置方法与单路 656 一样, 即接口时钟选择位 (vi2_vi0_sel) 配置选择本通道时钟; 分频时钟控制位 (vi0_div_sel、vi1_div_sel、vi2_div_sel、vi3_div_sel) 配置为 10, 即选择不分频。
- 高清模式配置方法

高清模式需要占用通道 0 和通道 2, 时钟配置方法为 vi2_vi0_sel 配置为 1, 即通道 2 时钟来源于通道 0 的时钟 VI0_CLKIN, vi0div_sel 和 vi2div_sel 配置为 10, 即选择不分频。



注意

- 使用 VIU 各个通道时, 还需在系统控制寄存器 SC_PERCTRL0 (地址为 0x101E_001C) 中, 将相应通道的软复位撤销掉, 默认值为软复位状态。
- 通道 0 为 4 路 656 时, 通道 0 和通道 1 的软复位都要撤销。同样, 通道 2 为 4 路 656 时, 通道 2 和通道 3 的软复位也都要撤销。
- 高清模式下通道 0 和通道 2 的软复位都要撤销。



VIU 的使能

VIU 只有在 vi_en 等于 1 的情况下才会工作。软件在使能 VIU 模块之前，必须先复位 VIU。

VIU 的 reg_newer 功能

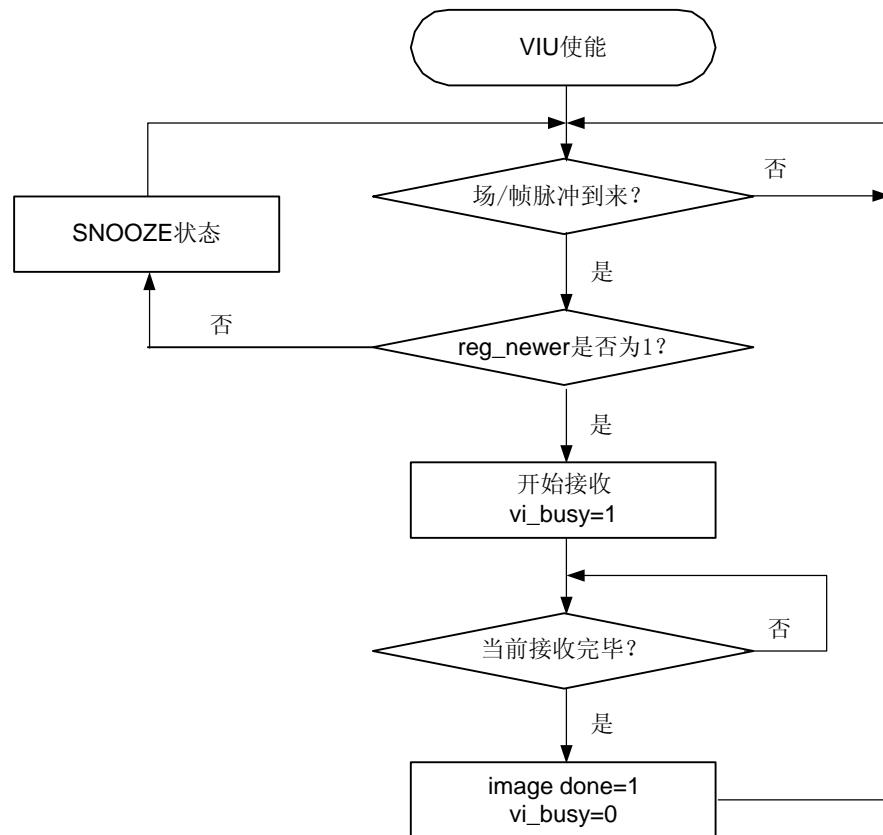
- 在软件使能 VIU 之前，软件应该完成以下操作：
 - 完成对 VIU 的图像寄存器的写操作。
 - 写 reg_newer 位为“1”，通知 VIU 模块当前的寄存器已经准备就绪。
- 当使能 VIU 后，VIU 逻辑将开始工作。

当一场/帧到来的时候，则：

 - 如果 reg_newer 为 0，则 VIU 将不会接收数据，置硬件状态为 SNOOZE，等待下一场/帧的数据的到来。
 - 如果 reg_newer 为 1，则开始接收数据，同时给出寄存器更新中断 (reg_update_int)，并设置硬件状态为 busy。
- 当接收完毕当前数据后，清除硬件 busy 状态。等到下一场/帧到来的时候，则：
 - 如果 reg_newer 为 0，则放弃下一场/帧数据的接收。
 - 如果 reg_newer 为 1，则可以紧接着前一次数据继续接收下一场/帧的数据。

VIU 的硬件工作流程如图 6-22 所示。

图6-22 VIU 的硬件工作流程



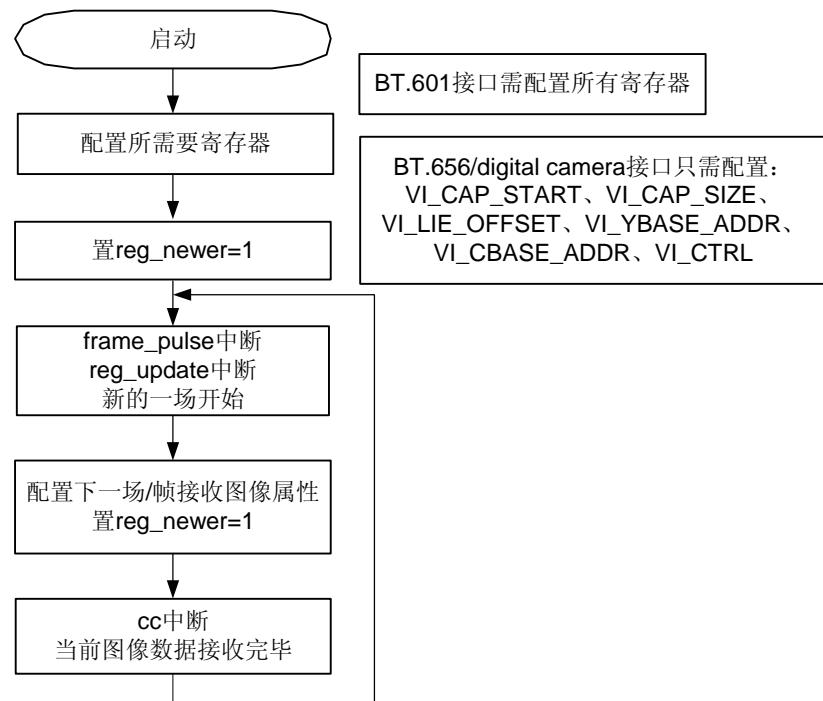


在 BT.656/601 和 digital camera 模式下, 每接收完一场/帧规定数据, 在下一场的到来时将 VIU 检测 reg_newer 位, 如果 reg_newer 位为 1 (表示软件已经更新或者确认 VIU 的寄存器), VIU 将自动 load 软件所配置的寄存器值到工作寄存器 (工作寄存器软件不可访问), 然后将 reg_newer 位清 0, 并开始接收下一场/帧数据, 否则只有等到 reg_newer 为 1 且新的一场/帧到来时开始接收数据。

软件配置流程

图 6-23 描述了在中断方式下, 软件的操作流程。

图6-23 软件操作流程



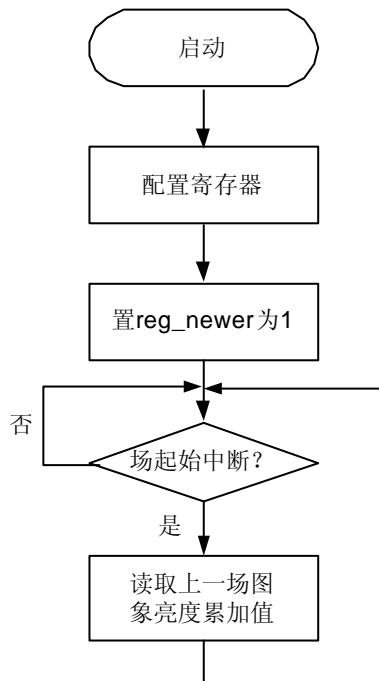
- 在 BT.656 模式和 digital camera 接口情况下, 只需要配置图 6-23 所示的寄存器, 而不需要配置时序寄存器。
- 在 BT.601 模式下, 时序寄存器必须配置, 以 PAL 制为例, 时序寄存器的值为寄存器默认值。

说明

时序寄存器包括垂直同步寄存器和水平同步寄存器。

亮度统计值软件读取操作流程

图6-24 亮度统计值软件读取流程



6.1.6 寄存器概览

VIU 寄存器概览如表 6-7 所示。

表6-7 VIU 寄存器概览 (基址是 0x9006_0000)

偏移地址	名称	描述	页码
0x00	VI0_CFG	第 0 路配置寄存器 ^b	6-33
0x04	VI0_CAP_START	第 0 路图像获取起始位置寄存器 ^{ab}	6-36
0x08	VI0_CAP_SIZE	第 0 路图像获取大小寄存器 ^{ab}	6-37
0x0C	VI0_LINE_OFFSET	第 0 路图像存储行间距寄存器 ^{ab}	6-38
0x10	VI0_YBASE_ADDR	第 0 路 Y 基地址寄存器 ^{ab}	6-39
0x14	VI0_UBASE_ADDR	第 0 路 Cb 基地址寄存器 ^{ab}	6-39
0x18	VI0_VBASE_ADDR	第 0 路 Cr 基地址寄存器 ^{ab}	6-40
0x1C	VI0_CTRL	第 0 路控制寄存器 ^b	6-40
0x20	VI0_INT_MASK	第 0 路中断使能寄存器 ^b	6-42
0x24	VI0_INT_STATUS	第 0 路中断状态寄存器 ^b	6-43



偏移地址	名称	描述	页码
0x28	VI0_RAW_INT	第 0 路原始中断状态寄存器 ^b	6-44
0x2C	VI0_STATUS	第 0 路状态寄存器 ^b	6-46
0x30	VI0_Y_STORESIZE	第 0 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x34	VI0_U_STORESIZE	第 0 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48
0x38	VI0_V_STORESIZE	第 0 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x3C	VI0_LUM_ADDER	第 0 路亮度累加寄存器	6-48
0x40	VI0_BLOCK0_STONE_START	第 0 路图象块 0 填充起始地址寄存器	6-48
0x44	VI0_BLOCK0_STONE_SIZE	第 0 路图象块 0 填充大小寄存器	6-48
0x48	VI0_BLOCK0_STONE_COLOR	第 0 路图象块 0 填充颜色寄存器	6-48
0x4C	VI_CH_CFG	通道配置寄存器 ^b	6-48
0x50	VI_CH0_VSYNC1	通道 0 垂直同步寄存器 1	6-48
0x54	VI_CH0_VSYNC2	通道 0 垂直同步寄存器 2	6-48
0x58	VI_CH0_HSYNC	通道 0 行同步寄存器	6-48
0x05C	PRIO_CONFIG	各通道总线申请绝对优先级配置寄存器	6-48
0x070	VI0_BLOCK1_STONE_START	第 0 路图象块 1 填充起始地址寄存器	6-48
0x074	VI0_BLOCK1_STONE_SIZE	第 0 路图象块 1 填充大小寄存器	6-48
0x078	VI0_BLOCK1_STONE_COLOR	第 0 路图象块 1 颜色寄存器	6-48
0x080	VI0_BLOCK2_STONE_START	第 0 路图象块 2 填充起始地址寄存器	6-48
0x084	VI0_BLOCK2_STONE_SIZE	第 0 路图象块 2 填充大小寄存器	6-48
0x088	VI0_BLOCK2_STONE_COLOR	第 0 路图象块 2 颜色寄存器	6-48
0x090	VI0_BLOCK3_STONE_START	第 0 路图象块 3 填充起始地址寄存器	6-48
0x094	VI0_BLOCK3_STONE_SIZE	第 0 路图象块 3 填充大小寄存器	6-48

偏移地址	名称	描述	页码
0x098	VI0_BLOCK3_STONE_COLOR	第 0 路图象块 3 颜色寄存器	6-48
0x0A0	VI0_LUM_STRH	第 0 路亮度拉伸寄存器	6-48
0x0A4	VI0_LUM_DIFF_ADDER	第 0 路亮度差值累加和寄存器	6-48
0x100	VI1_CFG	第 1 路配置寄存器 ^b	6-33
0x104	VI1_CAP_START	第 1 路图像获取起始位置寄存器 ^{ab}	6-36
0x108	VI1_CAP_SIZE	第 1 路图像获取大小寄存器 ^{ab}	6-37
0x10C	VI1_LINE_OFFSET	第 1 路图像存储行间距寄存器 ^{ab}	6-38
0x110	VI1_YBASE_ADDR	第 1 路 Y 基地址寄存器 ^{ab}	6-39
0x114	VI1_UBASE_ADDR	第 1 路 Cb 基地址寄存器 ^{ab}	6-39
0x118	VI1_VBASE_ADDR	第 1 路 Cr 基地址寄存器 ^{ab}	6-40
0x11C	VI1_CTRL	第 1 路控制寄存器 ^b	6-40
0x120	VI1_INT_MASK	第 1 路中断使能寄存器 ^b	6-42
0x124	VI1_STATUS_INT	第 1 路中断状态寄存器 ^b	6-43
0x128	VI1_RAW_INT	第 1 路原始中断状态寄存器 ^b	6-44
0x12C	VI1_STATUS	第 1 路状态寄存器 ^b	6-46
0x130	VI1_Y_STORESIZE	第 1 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x134	VI1_U_STORESIZE	第 1 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48
0x138	VI1_V_STORESIZE	第 1 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x13C	VI1_LUM_ADDER	第 1 路亮度累加寄存器	6-48
0x140	VI1_BLOCK0_STONE_START	第 1 路图象块 0 填充起始地址寄存器	6-48
0x144	VI1_BLOCK0_STONE_SIZE	第 1 路图象块 0 填充大小寄存器	6-48
0x148	VI1_BLOCK0_STONE_COLOR	第 1 路图象块 0 填充颜色寄存器	6-48
0x170	VI1_BLOCK1_STONE_START	第 1 路图象块 1 填充起始地址寄存器	6-48
0x174	VI1_BLOCK1_STONE_SIZE	第 1 路图象块 1 填充大小寄存器	6-48



偏移地址	名称	描述	页码
0x178	VI1_BLOCK1_STONE_COLOR	第 1 路图象块 1 颜色寄存器	6-48
0x180	VI1_BLOCK2_STONE_START	第 1 路图象块 2 填充起始地址寄存器	6-48
0x184	VI1_BLOCK2_STONE_SIZE	第 1 路图象块 2 填充大小寄存器	6-48
0x188	VI1_BLOCK2_STONE_COLOR	第 1 路图象块 2 颜色寄存器	6-48
0x190	VI1_BLOCK3_STONE_START	第 1 路图象块 3 填充起始地址寄存器	6-48
0x194	VI1_BLOCK3_STONE_SIZE	第 1 路图象块 3 填充大小寄存器	6-48
0x198	VI1_BLOCK3_STONE_COLOR	第 1 路图象块 3 颜色寄存器	6-48
0x1A0	VI1_LUM_STRH	第 1 路亮度拉伸寄存器	6-48
0x1A4	VI1_LUM_DIFF_ADDER	第 1 路亮度差值累加和寄存器	6-48
0x200	VI2_CFG	第 2 路配置寄存器 ^b	6-33
0x204	VI2_CAP_START	第 2 路图像获取起始位置寄存器 ^{ab}	6-36
0x208	VI2_CAP_SIZE	第 2 路图像获取大小寄存器 ^{ab}	6-37
0x20C	VI2_LINE_OFFSET	第 2 路图像存储行间距寄存器 ^{ab}	6-38
0x210	VI2_YBASE_ADDR	第 2 路 Y 基地址寄存器 ^{ab}	6-39
0x214	VI2_UBASE_ADDR	第 2 路 Cb 基地址寄存器 ^{ab}	6-39
0x218	VI2_VBASE_ADDR	第 2 路 Cr 基地址寄存器 ^{ab}	6-40
0x21C	VI2_CTRL	第 2 路控制寄存器 ^b	6-40
0x220	VI2_INT_MASK	第 2 路中断使能寄存器 ^b	6-42
0x224	VI2_STATUS_INT	第 2 路中断状态寄存器 ^b	6-43
0x228	VI2_RAW_INT	第 2 路原始中断状态寄存器 ^b	6-44
0x22C	VI2_STATUS	第 2 路状态寄存器 ^b	6-46
0x230	VI2_Y_STORESIZE	第 2 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x234	VI2_U_STORESIZE	第 2 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48



偏移地址	名称	描述	页码
0x238	VI2_V_STORESIZE	第 2 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x23C	VI2_LUM_ADDER	第 2 路亮度累加寄存器	6-48
0x240	VI2_BLOCK0_STONE_START	第 2 路图象块 0 填充起始地址寄存器	6-48
0x244	VI2_BLOCK0_STONE_SIZE	第 2 路图象块 0 填充大小寄存器	6-48
0x248	VI2_BLOCK0_STONE_COLOR	第 2 路图象块 0 填充颜色寄存器	6-48
0x270	VI1_BLOCK1_STONE_START	第 2 路图象块 1 填充起始地址寄存器	6-48
0x274	VI2_BLOCK1_STONE_SIZE	第 2 路图象块 1 填充大小寄存器	6-48
0x278	VI2_BLOCK1_STONE_COLOR	第 2 路图象块 1 颜色寄存器	6-48
0x280	VI2_BLOCK2_STONE_START	第 2 路图象块 2 填充起始地址寄存器	6-48
0x284	VI2_BLOCK2_STONE_SIZE	第 2 路图象块 2 填充大小寄存器	6-48
0x288	VI2_BLOCK2_STONE_COLOR	第 2 路图象块 2 颜色寄存器	6-48
0x290	VI2_BLOCK3_STONE_START	第 2 路图象块 3 填充起始地址寄存器	6-48
0x294	VI2_BLOCK3_STONE_SIZE	第 2 路图象块 3 填充大小寄存器	6-48
0x298	VI2_BLOCK3_STONE_COLOR	第 2 路图象块 3 颜色寄存器	6-48
0x2A0	VI2_LUM_STRH	第 2 路亮度拉伸寄存器	6-48
0x2A4	VI2_LUM_DIFF_ADDER	第 2 路亮度差值累加和寄存器	6-48
0x300	VI3_CFG	第 3 路配置寄存器 ^b	6-33
0x304	VI3_CAP_START	第 3 路图像获取起始位置寄存器 ^{ab}	6-36
0x308	VI2_CAP_SIZE	第 3 路图像获取大小寄存器 ^{ab}	6-37
0x30C	VI3_LINE_OFFSET	第 3 路图像存储行间距寄存器 ^{ab}	6-38
0x310	VI3_YBASE_ADDR	第 3 路 Y 基地址寄存器 ^{ab}	6-39
0x314	VI3_UBASE_ADDR	第 3 路 Cb 基地址寄存器 ^{ab}	6-39



偏移地址	名称	描述	页码
0x318	VI3_VBASE_ADDR	第3路Cr基地址寄存器 ^{ab}	6-40
0x31C	VI3_CTRL	第3路控制寄存器 ^b	6-40
0x320	VI3_INT_MASK	第3路中断使能寄存器 ^b	6-42
0x324	VI3_STATUS_INT	第3路中断状态寄存器 ^b	6-43
0x328	VI3_RAW_INT	第3路原始中断状态寄存器 ^b	6-44
0x32C	VI3_STATUS	第3路状态寄存器 ^b	6-46
0x330	VI3_Y_STORESIZE	第3路Y分量数据存储大小寄存器 ^{ab}	6-47
0x334	VI3_U_STORESIZE	第3路Cb分量数据存储大小寄存器 ^{ab}	6-48
0x338	VI3_V_STORESIZE	第3路Cr分量数据存储大小寄存器 ^{ab}	6-48
0x33C	VI3_LUM_ADDER	第3路亮度累加寄存器	6-48
0x340	VI3_BLOCK0_STONE_START	第3路图象块0填充起始地址寄存器	6-48
0x344	VI3_BLOCK0_STONE_SIZE	第3路图象块0填充大小寄存器	6-48
0x348	VI3_BLOCK0_STONE_COLOR	第3路图象块0填充颜色寄存器	6-48
0x370	VI3_BLOCK1_STONE_START	第3路图象块1填充起始地址寄存器	6-48
0x374	VI3_BLOCK1_STONE_SIZE	第3路图象块1填充大小寄存器	6-48
0x378	VI3_BLOCK1_STONE_COLOR	第3路图象块1颜色寄存器	6-48
0x380	VI3_BLOCK2_STONE_START	第3路图象块2填充起始地址寄存器	6-48
0x384	VI3_BLOCK2_STONE_SIZE	第3路图象块2填充大小寄存器	6-48
0x388	VI3_BLOCK2_STONE_COLOR	第3路图象块2颜色寄存器	6-48
0x390	VI3_BLOCK3_STONE_START	第3路图象块3填充起始地址寄存器	6-48
0x394	VI3_BLOCK3_STONE_SIZE	第3路图象块3填充大小寄存器	6-48
0x398	VI3_BLOCK3_STONE_COLOR	第3路图象块3颜色寄存器	6-48



偏移地址	名称	描述	页码
0x3A0	VI3_LUM_STRH	第 3 路亮度拉伸寄存器	6-48
0x3A4	VI3_LUM_DIFF_ADDE_R	第 3 路亮度差值累加和寄存器	6-48
0x400	VI4_CFG	第 4 路配置寄存器 ^b	6-33
0x404	VI4_CAP_START	第 4 路图像获取起始位置寄存器 ^{ab}	6-36
0x408	VI4_CAP_SIZE	第 4 路图像获取大小寄存器 ^{ab}	6-37
0x40C	VI4_LINE_OFFSET	第 4 路图像存储行间距寄存器 ^{ab}	6-38
0x410	VI4_YBASE_ADDR	第 4 路 Y 基地址寄存器 ^{ab}	6-39
0x414	VI4_UBASE_ADDR	第 4 路 Cb 基地址寄存器 ^{ab}	6-39
0x418	VI4_VBASE_ADDR	第 4 路 Cr 基地址寄存器 ^{ab}	6-40
0x41C	VI4_CTRL	第 4 路控制寄存器 ^b	6-40
0x420	VI4_INT_MASK	第 4 路中断使能寄存器 ^b	6-42
0x424	VI4_STATUS_INT	第 4 路中断状态寄存器 ^b	6-43
0x428	VI4_RAW_INT	第 4 路原始中断状态寄存器 ^b	6-44
0x42C	VI4_STATUS	第 4 路状态寄存器 ^b	6-46
0x430	VI4_Y_STORESIZE	第 4 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x434	VI4_U_STORESIZE	第 4 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48
0x438	VI4_V_STORESIZE	第 4 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x43C	VI4_LUM_ADDER	第 4 路亮度累加寄存器	6-48
0x440	VI4_BLOCK_STONE_START	第 4 路图象块 0 填充起始地址寄存器	6-48
0x444	VI4_BLOCK_STONE_SIZE	第 4 路图象块 0 填充大小寄存器	6-48
0x448	VI4_BLOCK_STONE_COLOR	第 4 路图象块 0 填充颜色寄存器	6-48
0x450	VI_CH2_VSYNC1	通道 2 垂直同步寄存器 1	6-48
0x454	VI_CH2_VSYNC2	通道 2 垂直同步寄存器 2	6-48
0x458	VI_CH2_HSYNC	通道 2 行同步寄存器	6-48
0x470	VI4_BLOCK1_STONE_START	第 4 路图象块 1 填充起始地址寄存器	6-48



偏移地址	名称	描述	页码
0x474	VI4_BLOCK1_STONE_SIZE	第 4 路图象块 1 填充大小寄存器	6-48
0x478	VI4_BLOCK1_STONE_COLOR	第 4 路图象块 1 颜色寄存器	6-48
0x480	VI4_BLOCK2_STONE_START	第 4 路图象块 2 填充起始地址寄存器	6-48
0x484	VI4_BLOCK2_STONE_SIZE	第 4 路图象块 2 填充大小寄存器	6-48
0x488	VI4_BLOCK2_STONE_COLOR	第 4 路图象块 2 颜色寄存器	6-48
0x490	VI4_BLOCK3_STONE_START	第 4 路图象块 3 填充起始地址寄存器	6-48
0x494	VI4_BLOCK3_STONE_SIZE	第 4 路图象块 3 填充大小寄存器	6-48
0x498	VI4_BLOCK3_STONE_COLOR	第 4 路图象块 3 颜色寄存器	6-48
0x4A0	VI4_LUM_STRH	第 4 路亮度拉伸寄存器	6-48
0x4A4	VI4_LUM_DIFF_ADDER	第 4 路亮度差值累加和寄存器	6-48
0x500	VI5_CFG	第 5 路配置寄存器 ^b	6-33
0x504	VI5_CAP_START	第 5 路图像获取起始位置寄存器 ^{ab}	6-36
0x508	VI5_CAP_SIZE	第 5 路图像获取大小寄存器 ^{ab}	6-37
0x50C	VI5_LINE_OFFSET	第 5 路图像存储行间距寄存器 ^{ab}	6-38
0x510	VI5_YBASE_ADDR	第 5 路 Y 基地址寄存器 ^{ab}	6-39
0x514	VI5_UBASE_ADDR	第 5 路 Cb 基地址寄存器 ^{ab}	6-39
0x518	VI5_VBASE_ADDR	第 5 路 Cr 基地址寄存器 ^{ab}	6-40
0x51C	VI5_CTRL	第 5 路控制寄存器 ^b	6-40
0x520	VI5_INT_MASK	第 5 路中断使能寄存器 ^b	6-42
0x524	VI5_STATUS_INT	第 5 路中断状态寄存器 ^b	6-43
0x528	VI5_RAW_INT	第 5 路原始中断状态寄存器 ^b	6-44
0x52C	VI5_STATUS	第 5 路状态寄存器 ^b	6-46
0x530	VI5_Y_STORESIZE	第 5 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x534	VI5_U_STORESIZE	第 5 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48



偏移地址	名称	描述	页码
0x538	VI5_V_STORESIZE	第 5 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x53C	VI5_LUM_ADDER	第 5 路亮度累加寄存器	6-48
0x540	VI5_BLOCK0_STONE_START	第 5 路图象块 0 填充起始地址寄存器	6-48
0x544	VI5_BLOCK0_STONE_SIZE	第 5 路图象块 0 填充大小寄存器	6-48
0x548	VI5_BLOCK0_STONE_COLOR	第 5 路图象块 0 填充颜色寄存器	6-48
0x570	VI5_BLOCK1_STONE_START	第 5 路图象块 1 填充起始地址寄存器	6-48
0x574	VI5_BLOCK1_STONE_SIZE	第 5 路图象块 1 填充大小寄存器	6-48
0x578	VI5_BLOCK1_STONE_COLOR	第 5 路图象块 1 颜色寄存器	6-48
0x580	VI5_BLOCK2_STONE_START	第 5 路图象块 2 填充起始地址寄存器	6-48
0x584	VI5_BLOCK2_STONE_SIZE	第 5 路图象块 2 填充大小寄存器	6-48
0x588	VI5_BLOCK2_STONE_COLOR	第 5 路图象块 2 颜色寄存器	6-48
0x590	VI5_BLOCK3_STONE_START	第 5 路图象块 3 填充起始地址寄存器	6-48
0x594	VI5_BLOCK3_STONE_SIZE	第 5 路图象块 3 填充大小寄存器	6-48
0x598	VI5_BLOCK3_STONE_COLOR	第 5 路图象块 3 颜色寄存器	6-48
0x5A0	VI5_LUM_STRH	第 5 路亮度拉伸寄存器	6-48
0x5A4	VI5_LUM_DIFF_ADDER	第 5 路亮度差值累加和寄存器	6-48
0x600	VI6_CFG	第 6 路配置寄存器 ^b	6-33
0x604	VI6_CAP_START	第 6 路图像获取起始位置寄存器 ^{ab}	6-36
0x608	VI6_CAP_SIZE	第 6 路图像获取大小寄存器 ^{ab}	6-37
0x60C	VI6_LINE_OFFSET	第 6 路图像存储行间距寄存器 ^{ab}	6-38
0x610	VI6_YBASE_ADDR	第 6 路 Y 基地址寄存器 ^{ab}	6-39
0x614	VI6_UBASE_ADDR	第 6 路 Cb 基地址寄存器 ^{ab}	6-39



偏移地址	名称	描述	页码
0x618	VI6_VBASE_ADDR	第 6 路 Cr 基地址寄存器 ^{ab}	6-40
0x61C	VI6_CTRL	第 6 路控制寄存器 ^b	6-40
0x620	VI6_INT_MASK	第 6 路中断使能寄存器 ^b	6-42
0x624	VI6_STATUS_INT	第 6 路中断状态寄存器 ^b	6-43
0x628	VI6_RAW_INT	第 6 路原始中断状态寄存器 ^b	6-44
0x62C	VI6_STATUS	第 6 路状态寄存器 ^b	6-46
0x630	VI6_Y_STORESIZE	第 6 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x634	VI6_U_STORESIZE	第 6 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48
0x638	VI6_V_STORESIZE	第 6 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x63C	VI6_LUM_ADDER	第 6 路亮度累加寄存器	6-48
0x640	VI6_BLOCK0_STONE_START	第 6 路图象块 0 填充起始地址寄存器	6-48
0x644	VI6_BLOCK0_STONE_SIZE	第 6 路图象块 0 填充大小寄存器	6-48
0x648	VI6_BLOCK0_STONE_COLOR	第 6 路图象块 0 填充颜色寄存器	6-48
0x670	VI6_BLOCK1_STONE_START	第 6 路图象块 1 填充起始地址寄存器	6-48
0x674	VI6_BLOCK1_STONE_SIZE	第 6 路图象块 1 填充大小寄存器	6-48
0x678	VI6_BLOCK1_STONE_COLOR	第 6 路图象块 1 颜色寄存器	6-48
0x680	VI6_BLOCK2_STONE_START	第 6 路图象块 2 填充起始地址寄存器	6-48
0x684	VI6_BLOCK2_STONE_SIZE	第 6 路图象块 2 填充大小寄存器	6-48
0x688	VI6_BLOCK2_STONE_COLOR	第 6 路图象块 2 颜色寄存器	6-48
0x690	VI6_BLOCK3_STONE_START	第 6 路图象块 3 填充起始地址寄存器	6-48
0x694	VI6_BLOCK3_STONE_SIZE	第 6 路图象块 3 填充大小寄存器	6-48
0x698	VI6_BLOCK3_STONE_COLOR	第 6 路图象块 3 颜色寄存器	6-48

偏移地址	名称	描述	页码
0x6A0	VI6_LUM_STRH	第 6 路亮度拉伸寄存器	6-48
0x6A4	VI6_LUM_DIFF_ADDE R	第 6 路亮度差值累加和寄存器	6-48
0x700	VI7_CFG	第 7 路配置寄存器 ^b	6-33
0x704	VI7_CAP_START	第 7 路图像获取起始位置寄存器 ^{ab}	6-36
0x708	VI7_CAP_SIZE	第 7 路图像获取大小寄存器 ^{ab}	6-37
0x70C	VI7_LINE_OFFSET	第 7 路图像存储行间距寄存器 ^{ab}	6-38
0x710	VI7_YBASE_ADDR	第 7 路 Y 基地址寄存器 ^{ab}	6-39
0x714	VI7_UBASE_ADDR	第 7 路 Cb 基地址寄存器 ^{ab}	6-39
0x718	VI7_VBASE_ADDR	第 7 路 Cr 基地址寄存器 ^{ab}	6-40
0x71C	VI7_CTRL	第 7 路控制寄存器 ^b	6-40
0x720	VI7_INT_MASK	第 7 路中断使能寄存器 ^b	6-42
0x724	VI7_STATUS_INT	第 7 路中断状态寄存器 ^b	6-43
0x728	VI7_RAW_INT	第 7 路原始中断状态寄存器 ^b	6-44
0x72C	VI7_STATUS	第 7 路状态寄存器 ^b	6-46
0x730	VI7_Y_STORESIZE	第 7 路 Y 分量数据存储大小寄存器 ^{ab}	6-47
0x734	VI7_U_STORESIZE	第 7 路 Cb 分量数据存储大小寄存器 ^{ab}	6-48
0x738	VI7_V_STORESIZE	第 7 路 Cr 分量数据存储大小寄存器 ^{ab}	6-48
0x73C	VI7_LUM_ADDER	第 7 路亮度累加寄存器	6-48
0x740	VI7_BLOCK0_STONE_S TART	第 7 路图象块 0 填充起始地址寄存器	6-48
0x744	VI7_BLOCK0_STONE_S IZE	第 7 路图象块 0 填充大小寄存器	6-48
0x748	VI7_BLOCK0_STONE_ COLOR	第 7 路图象块 0 填充颜色寄存器	6-48
0x770	VI7_BLOCK1_STONE_S TART	第 7 路图象块 1 填充起始地址寄存器	6-48
0x774	VI7_BLOCK1_STONE_S IZE	第 7 路图象块 1 填充大小寄存器	6-48
0x778	VI7_BLOCK1_STONE_ COLOR	第 7 路图象块 1 颜色寄存器	6-48



偏移地址	名称	描述	页码
0x780	VI7_BLOCK2_STONE_START	第 7 路图象块 2 填充起始地址寄存器	6-48
0x784	VI7_BLOCK2_STONE_SIZE	第 7 路图象块 2 填充大小寄存器	6-48
0x788	VI7_BLOCK2_STONE_COLOR	第 7 路图象块 2 颜色寄存器	6-48
0x790	VI7_BLOCK3_STONE_START	第 7 路图象块 3 填充起始地址寄存器	6-48
0x794	VI7_BLOCK3_STONE_SIZE	第 7 路图象块 3 填充大小寄存器	6-48
0x798	VI7_BLOCK3_STONE_COLOR	第 7 路图象块 3 颜色寄存器	6-48
0x7A0	VI7_LUM_STRH	第 7 路亮度拉伸寄存器	6-48
0x7A4	VI7_LUM_DIFF_ADDER	第 7 路亮度差值累加和寄存器	6-48

注：

- a: 非及时性寄存器，只有在新的一帧或一场的起始时刻才会被载入到工作寄存器；
- b: 在直接 digital camera 模式下有效的寄存器。

6.1.7 寄存器描述

本节详细描述了 VIU 的相关寄存器。

VI_CFG



说明

以下寄存器 VI_n_ABC 中涉及的 “n” 表示 VI 的通道号，满足 0≤n≤7，且 n 为整数。

VI_CFG 为配置寄存器。

Offset Address		Register Name	Total Reset Value
0x00 + n × 100		VIn_CFG	0x0001_4088
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved	chn_id	data_width
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0	store_mode	store_mode
		chn_id	little_endian
		odd_line_sel	cap_seq
		even_line_sel	cap_sel
		reserved	
Bits	Access	Name	Description
[31:24]	-	reserved	保留。
[23]	RW	seav_f_neg	BT.656 定时基准码场指示位 (F) 极性。 0: 1st field: F=0; 2nd field: F=1 (标准)。 1: 1st field: F=1; 2nd field: F=0 (非标准)。 注: 与 2815 对接时, 该 bit 需配置为 0。
[22]	RW	chn_id_en	通道号检测使能。 0: 不使能。 1: 使能。 仅在 4CIF 和 2D1 的情况下有效。
[21:20]	RW	chn_id	通道号。 当通道号检测使能时, 只有等于该通道号的数据才能够进入该通道。 00: VIU0/VIU4。 01: VIU1/VIU5。 10: VIU2/VIU6。 11: VIU3/VIU7。
[19:17]	-	reserved	保留。
[16:15]	RW	even_line_sel	偶场图像色度数据获取行选择。 00: 仅对奇数行采集。 01: 仅对偶数行采集。 10: 对奇数行和偶数行都采集。 11: 保留。 说明 行号是从 0 开始计数的; 此隔行获取功能只对色度分量有效, 亮度奇偶行都采集, 用于获得 YCbCr 420 的数据; 输入数据为帧模式 (DC 模式和高清模式) 时由 even_line_sel 控制色度丢行; 输出为帧模式时, 不允许配置成奇场只采集奇行, 偶场只采集偶行, 推荐配置为奇场只采集偶行, 偶场只采集奇行; 高清模式下只需配置通道 2。



[14:13]	RW	odd_line_sel	<p>奇场图像色度数据获取行选择。 00: 仅对奇数行采集。 01: 仅对偶数行采集。 10: 对奇数行和偶数行都采集。 11: 保留。</p> <p>说明 行号是从 0 开始计数的；此隔行获取功能只对色度分量有效，亮度奇偶行都采集，用于获得 YcbCr 420 的数据；输入数据为帧模式（DC 模式和高清模式）时 odd_line_sel 无效；输出为帧模式时，不允许配置成奇场只采集奇行，偶场只采集偶行，推荐配置为奇场只采集偶行，偶场只采集奇行。</p>
[12]	RW	correct_en	<p>SAV (Start of Active Video) /EAV (End of Activevideo) 数据校验使能。 0: 不使能。 1: 使能。</p>
[11]	RW	down_scaling	<p>水平 1/2 缩放。 0: 不使能。 1: 使能。</p> <p>说明 此功能有效，需要同时使能 fir_en。</p>
[10]	RW	chroma_resample	<p>色度重新采样。 0: 不使能。 1: 使能色度重新采样及 co-sited 到 interspersed 转换。</p> <p>说明 此功能有效，需要同时使能 fir_en。</p>
[9:8]	RW	store_method	<p>存储方式 (Store Method) 。</p> <p>00: planar Y/Cb/Cr。 01: semi-planar YCbCr。 10: package YCbCr4:2:2。 11: raw data。</p>
[7:6]	RW	cap_sel	<p>图像数据获取场选择。 00: 仅对奇数场 (顶场) 采集。 01: 仅对偶数场 (底场) 采集。 10: 对奇数场和偶数场都采集。 11: 保留。</p>
[5:4]	RW	cap_seq	<p>YCbCr 输入顺序寄存器。 00: CbYCrY。 01: CrYCbY。</p>

			10: YCbYCr。 11: YCrYCb。
[3]	RW	little_endian	little endian 存储方式。 0: big endian 方式存储。 1: little endian 方式存储。
[2]	RW	store_mode	帧模式接收或场模式接收指示。 0: 场接收模式。 1: 帧接收模式。
[1:0]	RW	data_width	数据位宽 (Data Width)。 00: 8bit。 其他: 保留。 说明 此版本只支持系统工作在 8bit 位宽模式下，并且此两位用户不能配置。

VI_CAP_START

VI_CAP_START 为数据获取起始位置寄存器，配置数据获取的起始位置。

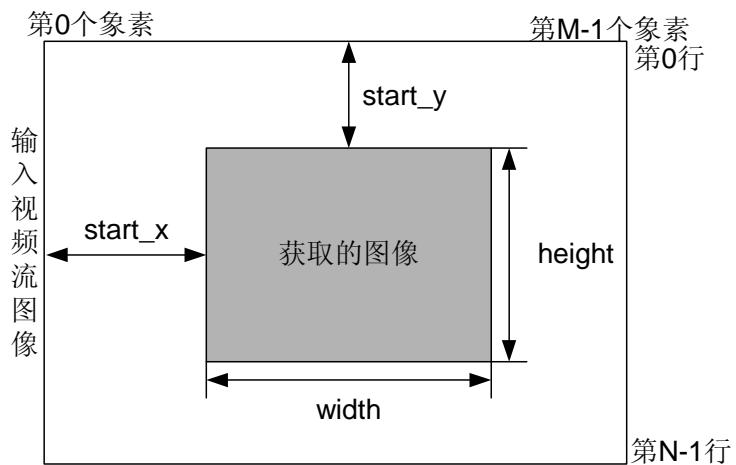
VI_CAP_SIZE 为数据获取大小寄存器，配置数据获取的大小。

VI_CAP_START 和 VI_CAP_SIZE 描述了从输入视频图像中获取矩形图像，如图 6-25 所示。

- VI_CAP_START 寄存器描述了相对于输入图像数据流中获取图像起始坐标，start_y 和 start_x。
- VI_CAP_SIZE 描述了获取图像的大小，宽度 (width) 和高度 (height)。

width 和 start_x 以输入像素为单位 (即亮度像素为单位)；在 YCbCr 4:2:2 图像数据获取时，height 和 start_y 以行为单位。

图6-25 图像获取参数示意图



注: start_y 和 start_x 在 VI_CAP_START 寄存器中分别占用 12 位宽度。

width 和 height 在 VI_CAP_SIZE 寄存器中分别占用 12 位宽度。

Offset Address																Register Name										Total Reset Value								
0x004 + n × 100																VIn_CAP_START										0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved										start_y										start_x													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
	Bits	Access	Name			Description																												
	[31:24]	-	reserved			保留。																												
	[23:12]	RW	start_y			开始获取图像的行号。																												
	[11:0]	RW	start_x			开始获取图像的像素号。																												

VI CAP SIZE

VI CAP SIZE 为图像获取大小寄存器。

Offset Address												Register Name												Total Reset Value											
0x008 + n × 100												VIn_CAP_SIZE												0x0016_077F											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												height												width										
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1					

Bits	Access	Name	Description
[31:24]	-	reserved	保留。
[23:12]	RW	height	获取图像的高度（以行为单位）。 帧模式时，获取图像高度为实际高度的 1/2，VI 捕获两场组成一帧图像。 VI 支持的最小捕获高度为 1 行。
[11:0]	RW	width	获取图像一行的宽度（以像素为单位）。 VI 支持的最小捕获宽度为 4 个像素。

VI_LINE_OFFSET

VI_LINE_OFFSET 为图像存储行间距寄存器，配置获取数据存储行偏移的大小。

	Offset Address		Register Name	Total Reset Value																												
	0x00C + n × 100		VIn_LINE_OFFSET	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	yline_offset										uline_offset										vline_offset											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:20]	RW	yline_offset	以 word 为单位。（实际偏移为 yline_offset 左移 2bit）。 在 raw data 存储模式下，仅该 stride 有效。 在 package YCbCr4:2:2 存储模式下，仅该 stride 有效。 在 planar Y/C 或者 Y/Cb/Cr 存储模式下，表示 Y 的行 stride。																													
[19:10]	RW	uline_offset	以 word 为单位。（实际偏移为 uline_offset 左移 2bit）。 在 raw data 存储模式下，无效。 在 package YCbCr4:2:2 存储模式下，无效。 在 planar Y/C 存储模式下，表示 C 分量的行 stride。 在 planar Y/Cb/Cr 存储模式下，表示 Cb 分量行 stride。																													
[9:0]	RW	vline_offset	以 word 为单位。（实际偏移为 vline_offset 左移 2bit）。 在 raw data 存储模式下，无效。 在 package YCbCr4:2:2 存储模式下，无效。 在 planar Y/C 存储模式下，无效。 在 planar Y/Cb/Cr 存储模式下，表示 Cr 分量行 stride。																													



VI YBASE ADDR

VI_YBASE_ADDR 为 Y 通道基地址寄存器，配置 Y 或者 G 分量的存放开始地址。以 word 为单位，最后 2bits 为 0。

VI_UBASE_ADDR

VI_UBASE_ADDR 为 Cb 通道基地址寄存器，配置 Cb 分量的存放开始地址。以 word 为单位，最后 2bits 为 0。

VI_VBASE_ADDR

VI_VBASE_ADDR 为 Cr 通道基地址寄存器，配置 Cr 分量的存放开始地址。以 word 为单位，最后 2bits 为 0。

Offset Address												Register Name												Total Reset Value											
0x018 + n × 100												VIIn_VBASE_ADDR												0x0000_0000											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	vi_vbase_addr																																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Bits	Access	Name	Description																																
[31:0]	RW	vi_vbase_addr	Cr 通道地址。 最低 2 位为 0, 64byte 对齐。 在 raw data 存储模式下, 无效。 在 package YCbCr4:2:2 存储模式下, 无效。 在 planar Y/C 存储模式下, 无效。 在 planar Y/Cb/Cr 存储模式下, 表示 Cr 分量的首地址。																																

VI CTRL

VI_CTRL 为控制寄存器，控制数据获取的开始和结束。VI_CTRL 可实现 VIU 的使能和 reg_newer 和中断控制。



[8]	RW	block2_stone_en	块屏蔽 2 使能。 0: 不使能。 1: 使能。
[7]	RW	block1_stone_en	块屏蔽 1 使能。 0: 不使能。 1: 使能。
[6]	RW	blcok0_stone_en	块屏蔽 0 使能。 0: 不使能。 1: 使能。
[5]	RW	debug_en	测试模式使能。 0: 不使能。 1: 使能。 测试使能未打开时测试寄存器回读值为 0。
[4]	RW	lum_add_en	图象亮度统计使能。 0: 不使能。 1: 使能。
[3]	RW	int_pulse_select	中断方式选择信号。 0: 以电平方式中断。 1: 以脉冲方式中断。 说明 仅第一路寄存器中此位有效，其他路寄存器此位保留。
[2]	RW	fir_en	滤波器使能信号。 0: 不使能。 1: 使能。 说明: 在 fir_en 使能的情况下，配置寄存器的 down_scaling 和 chrom_resample 不能全为 0。
[1]	RW	reg_newer	下一场/帧寄存器准备完毕。 0: 下一需要接收的场/帧寄存器未准备好，硬件将放弃下一场/帧的接收。 1: 下一需要接收的场/帧寄存器准备好，在检测到下一场/帧的场/帧开始时，硬件开始接收下一场数据。 说明 此位在 store_mode 为帧模式时，VIU 硬件在自动更新内部工作寄存器后，将自动清零该位。
[0]	RW	vi_en	VIU 模块使能。 0: 不使能。

		1: 使能。 只支持静态切换。任何模式切换（多路切换或者 BT.656 模式到 BT.601 模式切换等）时，要先将该位置 0，切换完毕再置 1。
--	--	--

VI_INT_MASK

VI_INT_MASK 为中断屏蔽寄存器。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	VIn_INT_MASK	0x0000_0000
Name	reserved		cc_int_en buf_ovf_int_en field_throw_int_en err_int_en proc_err_int_en reg_update_int_en frame_pulse_int_en ntsc_pal_trans_int_en chdiv_err_int_en
Reset	0 0		
Bits	Access	Name	Description
[31:9]	-	reserved	保留。
[8]	RW	chdiv_err_int_en	通道分配错误指示中断使能。 0: 屏蔽中断。 1: 使能中断。
[7]	RW	ntsc_pal_trans_int_en	制式转化中断使能。 0: 屏蔽中断。 1: 使能中断。
[6]	RW	frame_pulse_int_en	场起始中断使能。 0: 屏蔽中断。 1: 使能中断。
[5]	RW	reg_update_int_en	寄存器更新中断使能。 0: 屏蔽中断。 1: 使能中断。
[4]	RW	proc_err_int_en	BT.656 模式下，保护比特位错误中断使能。 0: 屏蔽中断。 1: 使能中断。



[3]	RW	err_int_en	总线错误中断使能。 0: 屏蔽中断。 1: 使能中断。
[2]	RW	field_throw_int_en	场/帧丢失中断使能。 0: 屏蔽中断。 1: 使能中断。
[1]	RW	buf_ovf_int_en	内部 FIFO 溢出错误中断使能。 0: 屏蔽中断。 1: 使能中断。
[0]	RW	cc_int_en	数据获取完毕中断使能。 0: 一场图像获取完毕中断。 1: 一帧图像获取完毕中断。

VI_INT_STATUS

VI_INT_STATUS 为中断状态寄存器，对中断状态寄存器的相应位写 1，则清除该中断位。各通道分别控制，用于软硬交互，可屏蔽。

	Offset Address		Register Name	Total Reset Value																																					
	0x024 + n × 100		VIn_INT_STATUS	0x0000_0000																																					
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cc_int	buf_ovf_int	field_throw_int	error_int	proc_err_int	reg_update_int	frame_pulse_int	ntsc_pal_trans_int	chdiv_err_int
Name	reserved																																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
Bits	Access	Name	Description																																						
[31:9]	-	reserved	保留。																																						
[8]	WC	chdiv_err_int	通道分配错误指示中断状态。 0: 无中断。 1: 有中断。																																						
[7]	WC	ntsc_pal_trans_int	制式变换中断状态。 0: 无中断。 1: 有中断。 说明 制式变换中断，制式发生变化就会上报，不仅限于 N 制和 P 制，只要																																						



			输入数据有效行数发生变换即认为制式变换。
[6]	WC	frame_pulse_int	场起始中断状态。 0: 无中断。 1: 有中断。
[5]	WC	reg_update_int	工作寄存器更新中断状态。 0: 无中断。 1: 有中断。 当 store_mode 为 1 时, 表示一帧图像获取完毕中断。 当 store_mode 为 0 时, 表示一场图像获取完毕中断。
[4]	WC	proc_err_int	保护位错误中断状态 (BT.656 模式)。 0: 无中断。 1: 有中断。
[3]	WC	error_int	AHB 总线错误中断状态。 0: 无中断。 1: 有中断。
[2]	WC	field_throw_int	场数据丢失中断状态。 0: 无中断。 1: 有中断。 说明 配置为只获取奇场或者只获取偶场时, 硬件不会上报场数据丢失。
[1]	WC	buf_ovf_int	内部缓冲 FIFO 溢出中断状态。 0: 无中断。 1: 有中断。
[0]	WC	cc_int	当前图像数据获取完毕中断状态。 0: 无中断。 1: 有中断。 当 store_mode 为 1 时, 表示一帧图像获取完毕中断。 当 store_mode 为 0 时, 表示一场图像获取完毕中断。

VI_RAW_INT

VI_RAW_INT 为原始中断状态寄存器。清除为对中断状态寄存器的相应 bit 写 1, 则清除该中断比特位, 各通道分别控制, 用于 DEBUG, 不可屏蔽。



Offset Address										Register Name										Total Reset Value												
0x028 + n × 100										VIn_RAW_INT										0x0000_0000												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																													cc_raw_int		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:9]	-	reserved	保留。																										buf_ovf_raw_int			
[8]	RO	chdiv_err_raw_int	通道分配错误指示原始中断状态。 0: 无原始中断。 1: 有原始中断。																										field_throw_raw_int			
[7]	RO	ntsc_pal_trans_raw_int	制式变换原始中断状态。 0: 无原始中断。 1: 有原始中断。 说明 制式变换中断，制式发生变化就会上报，不仅限于 N 制和 P 制，只要输入数据有效行数发生变换即认为制式变换。																										proc_err_raw_int			
[6]	RO	frame_pulse_raw_int	场起始原始中断状态。 0: 无原始中断。 1: 有原始中断。																										reg_update_raw_int			
[5]	RO	reg_update_raw_int	工作寄存器更新原始中断状态。 0: 无原始中断。 1: 有原始中断。																										frame_pulse_raw_int			
[4]	RO	proc_err_raw_int	保护位错误原始中断状态 (BT.656 模式)。 0: 无原始中断。 1: 有原始中断。																										ntsc_pal_trans_raw_int			
[3]	RO	error_raw_int	AHB 总线错误原始中断状态。 0: 无原始中断。 1: 有原始中断。																										chdiv_err_raw_int			
[2]	RO	field_throw_raw_int	场数据丢失原始中断状态。 0: 无原始中断。 1: 有原始中断。																										buf_ovf_raw_int			

[1]	RO	buf_ovf_raw_int	内部缓冲 FIFO 溢出原始中断状态。 0: 无原始中断。 1: 有原始中断。
[0]	RO	cc_raw_int	当前图像数据获取完毕原始中断状态 (cc: capture completion)。 0: 一场图像获取完毕原始中断。 1: 一帧图像获取完毕原始中断。 当 int_mode 为 1 时, 表示一帧图像获取完毕中断。 当 int_mode 为 0 时, 表示一场图像获取完毕中断。

VI_STATUS

VI_STATUS 为状态寄存器。

	Offset	Address	Register Name	Total	Reset	Value
		0x02C + n × 100	VIn_STATUS			0x0000_0020
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0					
Name	reserved		act_height		vi_busy	field2
Reset	0 0			proc_err	snooze	buf_ovf
				frame_loss	bus_err	image_done
Bits	Access	Name	Description			
[31:20]	-	reserved	保留。			
[19:8]	RO	act_height	检测到的一场中有效像素数据的行数。			
[7]	RO	vi_busy	VIU 当前工作状态。 0: VIU 空闲。 1: VIU 忙。			
[6]	RO	field2	当前接收为偶数场。 0: 奇数场。 1: 偶数场。			
[5]	RO	snooze	当前 VIU 处于睡眠状态。 0: VIU 处于非睡眠状态。 1: VIU 处于睡眠状态。			



[4]	RO	proc_err	保护位错误状态。 0: 保护位正确。 1: 保护位错误。
[3]	RO	bus_err	总线错误状态。 0: 总线正确。 1: 总线错误。
[2]	RO	frame_loss	VIU 丢失一场数据。 0: 未丢失。 1: 丢失。
[1]	RO	buf_ovf	VIU 内部 buffer 溢出。 0: 未溢出。 1: 溢出。
[0]	RO	image_done	VIU 接收完毕当前场数据。 0: 未接受完毕。 1: 接受完毕。

VI_Y_STORESIZE

VI_Y_STORESIZE 为 Y 分量数据存储大小寄存器。

	Offset Address		Register Name	Total Reset Value																												
	0x030 + n × 100		VIn_Y_STORESIZE	0x000F_00A0																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		y_height		y_width																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:12]	RW	y_height	Y 分量数据存储高度（以行为单位）。																													
[11:0]	RW	y_width	Y 分量数据存储宽度（以 word 为单位，不够一个 word，以一个 word 算）。																													



VI_U_STORESIZE

VI_U_STORESIZE 为 Cb 分量数据存储大小寄存器。在 package 存储模式时该寄存器不适用；semi-planar YCbCr 数据格式时该寄存器表示色度存储大小；planar YCbCr 数据格式时该寄存器表示 Cb 分量存储大小。

Offset Address			Register Name	Total Reset Value
0x034 + n × 100			VIn_U_STORESIZE	0x000F_00A0
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name	reserved		u_height	u_width
Reset	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0			
Bits	Access	Name	Description	
[31:24]	-	reserved	保留。	
[23:12]	RW	c_height	semi-planar YCbCr 模式下，色度分量数据存储高度（以行为单位）。 planar YCbCr 模式下，Cb 分量数据存储高度（以行为单位）。	
[11:0]	RW	c_width	semi-planar YCbCr 模式下，色度分量数据存储宽度（以 word 为单位，不够一个 word，以一个 word 算）。 planar YCbCr 模式下，Cb 分量数据存储宽度（以 word 为单位，不够一个 word，以一个 word 算）。 在 package 存储模式和 raw data 接收模式下无效。	

VI_V_STORESIZE

VI_V_STORESIZE 为 Cr 分量数据存储大小寄存器。

Offset Address			Register Name	Total Reset Value
0x038 + n × 100			VIn_V_STORESIZE	0x000F_00A0
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name	reserved		v_height	v_width
Reset	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0			
Bits	Access	Name	Description	
[31:24]	-	reserved	保留。	
[23:12]	RW	v_height	Cr 分量数据存储高度（以行为单位）。	
[11:0]	RW	v_width	Cr 分量数据存储宽度（以 word 为单位，不够一个 word，以一个 word 算）。 在 semi-planar YCbCr 存储模式、packet YCbCr 存储模式和	



			raw data 接收模式下无效。
--	--	--	-------------------

当接收 4:2:2 Y/Cb/Cr 数据时，数据存储大小计算步骤如下：

步骤 1 判断 cap_width 的奇偶。若偶数不变，奇数则减 1。

```
cap_width = cap_width - cap_width%2
```

步骤 2 判断是否 1/2 缩放。

```
down_scaling ?  
yes: cap_width = cap_width/2  
no: cap_width = cap_width
```

步骤 3 不同存储模式下，数据存储大小的计算。

1. 存储模式为 Y/Cb/Cr。 (数据位宽为 8bit)

```
y_width = if(cap_width%4 == 0)  
          cap_width/4  
        else  
          cap_width/4 + 1  
c_width = if(cap_width%8 == 0)  
          cap_width/8  
        else  
          cap_width/8 + 1  
v_width = if(cap_width%8 == 0)  
          cap_width/8  
        else  
          cap_width/8 + 1
```

2. 存储模式为 planar Y/C。 (数据位宽为 8bit)

```
y_width = if(cap_width%4 == 0)  
          cap_width/4  
        else  
          cap_width/4 + 1  
c_width = if(cap_width%4 == 0)  
          cap_width/4  
        else  
          cap_width/4 + 1
```

3. 存储模式为 package 模式。 (数据位宽为 8bit)

```
y_width = cap_width/2 + cap_width%2
```

4. 存储模式为 raw data。 (数据位宽为 8bit)

```
y_width = if(cap_width%4 == 0)  
          cap_width/4  
        else
```

cap_width/4 + 1

5. 存储模式为 raw data。 (数据位宽为 10bit)

y_width = cap_width/2 + cap_width%2

VI_LUM_ADDER

LUM_ADDER 为亮度统计寄存器，统计整个活动图像的亮度信息，而不是获取图像的区域的亮度。

	Offset Address	Register Name	Total Reset Value
	0x03C + n × 100	VIn_LUM_ADDER	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	lum_adder		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	vi_lum_adder	亮度数值累加。

注：当打开图像块屏蔽功能后，亮度统计寄存器对被屏蔽的图像块亮度不敏感。

VI_BLOCK0_STONE_START

VI_BLOCK0_STONE_START 为图像块 0 填充位置寄存器，配置填充的起始位置。

	Offset Address	Register Name	Total Reset Value
	0x040 + n × 100	VIn_BLOCK0_STONE_START	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved block0_stone_starty block0_stone_startx		
Reset	0 0		
Bits	Access	Name	Description
[31:24]	-	reserved	保留。
[23:12]	RW	block0_stone_starty	开始图像块 0 填充的行号。
[11:0]	RW	block0_stone_startx	开始图像块 0 填充的像素号。

VI_BLOCK1_STONE_START

VI_BLOCK1_STONE_START 为图像块 1 填充位置寄存器，配置填充的起始位置。



Offset Address			Register Name			Total Reset Value																										
0x070 + n × 100			VIn_BLOCK1_STONE_START			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved			block1_stone_starty			block1_stone_startx																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:12]	RW	block1_stone_starty	开始图像块 1 填充的行号。																													
[11:0]	RW	block1_stone_startx	开始图像块 1 填充的像素号。																													

VI_BLOCK2_STONE_START

VI_BLOCK2_STONE_START 为图像块 2 填充位置寄存器，配置填充的起始位置。

Offset Address			Register Name			Total Reset Value																										
0x080 + n × 100			VIn_BLOCK2_STONE_START			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved			block2_stone_starty			block2_stone_startx																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:12]	RW	block2_stone_starty	开始图像块 2 填充的行号。																													
[11:0]	RW	block2_stone_startx	开始图像块 2 填充的像素号。																													

VI_BLOCK3_STONE_START

VI_BLOCK3_STONE_START 为图像块 3 填充位置寄存器，配置填充的起始位置。

Offset Address			Register Name	Total Reset Value																												
0x090 + n × 100			VIn_BLOCK3_STONE_START	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved			block3_stone_starty			block3_stone_startx																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	name		description																												
[31:24]	-	reserved		保留。																												
[23:12]	RW	block3_stone_starty		开始图像块 3 填充的行号。																												
[11:0]	RW	block3_stone_startx		开始图像块 3 填充的像素号。																												

VI_BLOCK0_STONE_SIZE

VI_BLOCK0_STONE_SIZE 为图像块 0 屏蔽大小寄存器，配置填充数据块的大小。

Offset Address			Register Name	Total Reset Value																												
0x044 + n × 100			VIn_BLOCK0_STONE_SIZE	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved			block0_stone_height			block0_stone_width																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		description																												
[31:24]	-	reserved		保留。																												
[23:12]	RW	block0_stone_height		图像块 0 填充的高度（以行为单位）。																												
[11:0]	RW	block0_stone_width		图像块 0 填充的宽度（以像素为单位）。																												

VI_BLOCK1_STONE_SIZE

VI_BLOCK1_STONE_SIZE 为图像块 1 屏蔽大小寄存器，配置填充数据块的大小。



VI_BLOCK2_STONE_SIZE

VI_BLOCK2_STONE_SIZE 为图像块 2 屏蔽大小寄存器，配置填充数据块的大小。

Offset Address		Register Name	Total Reset Value
0x084 + n × 100		VIn_BLOCK2_STONE_SIZE	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved	block2_stone_height	block2_stone_width
Reset	0 0		
Bits	Access	Name	Description
[31:24]	-	reserved	保留。
[23:12]	RW	block2_stone_height	图像块 2 填充的高度（以行为单位）。
[11:0]	RW	block2_stone_width	图像块 2 填充的宽度（以像素为单位）。

VI_BLOCK3 STONE SIZE

VI_BLOCK3_STONE_SIZE 为图像块 3 屏蔽大小寄存器，配置填充数据块的大小。

Offset Address		Register Name	Total Reset Value
0x094 + n × 100		VIn_BLOCK3_STONE_SIZE	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved	block3_stone_height	block3_stone_width
Reset	0 0		
Bits	Access	Name	Description
[31:24]	-	reserved	保留。
[23:12]	RW	block3_stone_height	图像块 3 填充的高度（以行为单位）。
[11:0]	RW	block3_stone_width	图像块 3 填充的宽度（以像素为单位）。

VI_BLOCK0_STONE_COLOR

VI_BLOCK0_STONE_COLOR 为图像块 0 填充颜色寄存器。

填充颜色时需要注意以下两点：

- 填充图像颜色必须为 YCbCr 格式数据。
 - 当 VIU 配置为多个块遮挡，各个块之间有叠加部分时，输出叠加部分的颜色按照 0、1、2、3 的优先级覆盖。即填充块 0 的优先级最高，填充块 3 的优先级最低。

VI_BLOCK1_STONE_COLOR

VI_BLOCK1_STONE_COLOR 为图像块 1 填充颜色寄存器。



VI_BLOCK2_STONE_COLOR

VI_BLOCK2_STONE_COLOR 为图像块 2 填充颜色寄存器。

VI BLOCK3 STONE COLOR

VI BLOCK3 STONE COLOR 为图像块填充颜色寄存器。

VI LUM STRH

VI_LUM_STRH 为亮度拉伸寄存器，用于调整输入图像亮度值，调节公式如下：

$$luma_out = (sign(luma_in - m0) * (k * |luma_in - m0| + 64) \gg 7) + m0$$

其中：

- $m0$ 为上一帧或场的平均亮度, 用 `VI_LUM_ADDER` 除以输入图像像素个数得到
(注: 输入图像像素个数为接口接收到的像素个数, 不是捕获图像像素个数)。
 - k 为拉伸系数, 正常范围为 64~196, 64 是缩小相邻像素亮度差, 196 是放大相邻
像素亮度差, 128 是保持原值不变。
 - 该寄存器中的 $m1$ 建议配置成与 $m0$ 相同的值。

Offset Address								Register Name								Total Reset Value																
0x0A0 + n × 100								VIn_LUM_STRH								0x0000_0000																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved	m1								reserved	m0								k													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name			Description																											
[31:30]	-	reserved			保留。																											
[29:20]	RW	m1			亮度均值 1。																											
[19:18]	-	reserved			保留。																											
[17:8]	RW	m0			亮度均值 0。																											



[7:0]	RW	k	拉度拉伸系数。
-------	----	---	---------

VI_LUM_DIFF_ADDER

`VI_LUM_DIFF_ADDER` 为亮度差值累加和寄存器，输入图像亮度与亮度拉伸寄存器的亮度均值 m_1 之差的绝对值之和；亮度差值累加和统计整个活动图像的亮度信息，而不仅是获取图像的区域的亮度。帧存储模式时，该寄存器按帧进行统计；场存储模式时，该寄存器按场进行统计。输入图像亮度为缩放之前的亮度。可参考该寄存器来决定亮度拉伸功能是否打开，当该寄存器值较大时，说明上一帧或场各个像素之间亮度差较大，若要平滑各个像素之间的亮度差，可将 `VI_LUM_STRH` 中的 k 值配置成小于 128 的值，相反，当该寄存器值较小时，说明上一帧或场各个像素之间亮度差较小，若要增加各个像素之间的亮度差，可将 `VI_LUM_STRH` 中的 k 值配置成大于 128 的值。

VI CH CFG

VI_CH_CFG 为通道配置寄存器，用于配置各个通道的参数。

Offset Address										Register Name										Total Reset Value														
Bit	0x04C										VI_CH_CFG										0x0000_0000													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved	chrom_swap	hsv_mode	bus_mode	ch0_cap_mode	ch2_cap_mode	ch0_mode	ch1_mode	ch2_mode	ch3_mode	ch0_ctrl_mode	ch2_ctrl_mode	ch0_vsync_neg	ch2_vsync_neg	ch0_hsync	ch2_hsync	ch0_hsync_en	ch2_hsync_en	ch0_master_mode	ch2_master_mode	ch0_hsync_neg	ch2_hsync_neg	ch0_vsync	ch2_vsync	ch0_vsync_en	ch2_vsync_en	ch0_en	ch2_en	ch1_en	cl3_en	cl2_en	cl1_en	cl0_en	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits		Access	Name										Description																					
[31:30]		-	reserved										保留。																					

[29]	RW	chrom_swap	通道 2 的数据 swap 使能。 0: 按照正常方式输出。 1: halfword 的 byte 倒序。例如: Y0Y1Y2Y3 变成 Y1Y0Y3Y2。 只在高清模式下, 通道 2 用于传输色度, 此时色度的存储顺序可调整为: 0: 色度存储顺序为 Cb1Cr1Cb0Cr0。 1: 色度存储顺序为 Cr1Cb1Cr0Cb0。
[28:27]	RW	htv_mode	输入视频模式。 00: 标清 BT.656 模式。 01: 高清 720p 模式。 10: 逐行 BT.1358 模式。 11: 保留。 只有 ch0_cap_mode 和 ch1_cap_mode 配置为 BT.656 模式, 高清 720p 模式才有效。只有 ch0_cap_mode 配置为 BT.656 模式, 逐行 BT.1358 模式才有效。
[26]	RW	bus_mode	总线工作模式。 0: 64 位工作模式。 1: 32 位工作模式。
[25:24]	RW	ch0_cap_mode	通道 0 数据接收模式。 00: BT.656 模式。 01: BT.601 模式。 10: 数字摄像头模式。 11: 保留。
[23:22]	RW	ch2_cap_mode	通道 2 数据接收模式。 00: BT.656 模式。 01: BT.601 模式。 10: 数字摄像头模式。 11: 保留。 说明 通道 1 和通道 3 的数据接收模式只能是 BT.656 模式, 所以此处不用配置。
[21:20]	RW	ch0_mode	通道 0 工作模式 (当数据接收模式为 BT.656 模式时, 该两位有效)。 00: 输入数据为 1 路 D1。 01: 输入数据为 2 路 D1 时分复用数据。 10: 输入数据为 4 路 half D1 时分复用数据。 11: 保留。



			<p>说明</p> <p>当 ch0_mode 配置为 “00” 时占用内部第 0 路通道；当 ch0_mode 配置为 “01” 时占用内部第 0、1 路通道；当 ch0_mode 配置成 “10” 模式时，占用内部第 0、1、2 和 3 路通道，此时则无论 ch1_mode 配置成何值，内部第 2、3 路都连接到第 0 通道的数据，通道 1 不工作。</p>
[19]	RW	ch1_mode	<p>通道 1 工作模式（当数据接收模式为 BT.656 模式时，该位有效）。</p> <p>0: 输入数据为 1 路 D1。</p> <p>1: 输入数据为 2 路 DI 时分复用数据。</p> <p>说明</p> <p>当 ch1_mode 配置为 “0” 时占用内部第 2 路通道；当 ch1_mode 配置为 “1” 时占用内部第 2、3 路通道。</p>
[18:17]	RW	ch2_mode	<p>通道 2 工作模式（当数据接收模式为 BT.656 模式时，该位有效）。</p> <p>00: 输入数据为 1 路 D1。</p> <p>01: 输入数据为 2 路 DI 时分复用数据。</p> <p>10: 输入数据为 4 路 half D1 时分复用数据。</p> <p>11: 保留。</p> <p>说明</p> <p>当 ch2_mode 配置为 “00” 时占用内部第 4 路通道；当 ch2_mode 配置为 “01” 时占用内部第 4、5 路通道；当 ch0_mode 配置成 “10” 模式时，占用内部第 4、5、6 和 7 路通道，此时则无论 ch3_mode 配置成何值，内部第 6、7 路都连接到第 2 通道的数据，通道 3 不工作。</p>
[16]	RW	ch3_mode	<p>通道 3 工作模式（当数据接收模式为 BT.656 模式时，该位有效）。</p> <p>0: 输入数据为 1 路 D1。</p> <p>1: 输入数据为 2 路 DI 时分复用数据。</p> <p>说明</p> <p>当 ch3_mode 配置为 “0” 时占用内部第 6 路通道；当 ch3_mode 配置为 “1” 时占用内部第 6、7 路通道。</p>
[15]	RW	ch0_ctrl_mode	<p>通道 0 中断控制模式。</p> <p>0: 4 路通道单独上报中断。</p> <p>1: 4 路通道同时上报数据捕获完毕中断 (cc_int) 和工作寄存器更新中断 (reg_update_int)。</p> <p>说明</p> <p>只有通道 0 工作在 BT.656 模式 (ch0_cap_mode=00)，输入数据为 4 路时分复用 (ch0_mode=10) 时该位才有效。</p>
[14]	RW	ch2_ctrl_mode	<p>通道 2 中断控制模式。</p> <p>0: 4 路通道单独上报中断。</p> <p>1: 4 路通道同时上报数据捕获完毕中断 (cc_int) 和工作寄存器更新中断 (reg_update_int)。</p> <p>说明</p>

			只有通道 2 工作在 BT.656 模式 (ch2_cap_mode=00), 输入数据为 4 路时分复用 (ch2_mode=10) 时该位才有效。
[13]	RW	ch0_vsync	管脚 ch0_vi_vsync_field 配置信号。 0: 场号 (奇场或者偶场) 或者行有效信号。在 BT.601 模式下表示场号; 在 camera 接口情况下表示行有效信号。 1: 垂直同步脉冲。
[12]	RW	ch2_vsync	管脚 ch2_vi_vsync_field 配置信号。 0: 场号 (奇场或者偶场) 或者行有效信号。在 BT.601 模式下表示场号; 在 camera 接口情况下表示行有效信号。 1: 垂直同步脉冲。
[11]	RW	ch0_vsync_neg	管脚 ch0_vi_vsync_field 极性配置。 0: 高电平有效。 在脉冲情况下 (vsync=1), 正脉冲表示同步脉冲。 在场号模式下, 高电平表示偶数场, 低电平表示奇数场。 在行有效情况下, 高电平表示行有效。 1: 低电平有效。 在脉冲情况下 (vsync=1), 负脉冲表示同步脉冲。 在场号模式下, 低电平表示偶数场, 高电平表示奇数场。 在行有效情况下, 低电平表示行有效。
[10]	RW	ch2_vsync_neg	管脚 ch2_vi_vsync_field 极性配置。 0: 高电平有效。 在脉冲情况下 (vsync=1), 正脉冲表示同步脉冲。 在场号模式下, 高电平表示偶数场, 低电平表示奇数场。 在行有效情况下, 高电平表示行有效。 1: 低电平有效。 在脉冲情况下 (vsync=1), 负脉冲表示同步脉冲。 在场号模式下, 低电平表示偶数场, 高电平表示奇数场。 在行有效情况下, 低电平表示行有效。
[9]	RW	ch0_hsync	管脚 ch0_vi_hsync_vd 配置信号。 0: 数据有效。 1: 水平同步脉冲。
[8]	RW	ch2_hsync	管脚 ch2_vi_hsync_vd 配置信号。 0: 数据有效。 1: 水平同步脉冲。
[7]	RW	ch0_hsync_neg	管脚 ch0_vi_hsync_vd 极性配置。 0: 高电平有效。 在脉冲情况下 (hsync=1), 正脉冲表示同步脉冲。



			在数据有效情况下 (hsync=0) , 高电平表示数据有效。 1: 低电平有效。 在脉冲情况下 (hsync=1) , 负脉冲表示同步脉冲。 在数据有效情况下 (hsync=0) , 低电平表示数据有效。
[6]	RW	ch2_hsync_neg	管脚 ch2_vi_hsync_vd 极性配置。 0: 高电平有效。 在脉冲情况下 (hsync=1) , 正脉冲表示同步脉冲。 在数据有效情况下 (hsync=0) , 高电平表示数据有效。 1: 低电平有效。 在脉冲情况下 (hsync=1) , 负脉冲表示同步脉冲。 在数据有效情况下 (hsync=0) , 低电平表示数据有效。
[5]	RW	ch0_master_mode	第 0 通道 BT.601 主从工作模式选择。 0: 从模式。 1: 主模式。 说明 此位只能配置成从模式。
[4]	RW	ch2_master_mode	第 2 通道 BT.601 主从工作模式选择。 0: 从模式。 1: 主模式。 说明 此位只能配置成从模式。
[3]	RW	ch0_en	通道 0 使能。 0: 不使能。 1: 使能。
[2]	RW	ch1_en	通道 1 使能。 0: 不使能。 1: 使能。
[1]	RW	ch2_en	通道 2 使能。 0: 不使能。 1: 使能。
[0]	RW	ch3_en	通道 3 使能。 0: 不使能。 1: 使能。

VI_CH0_VSYNC1

VI_CH0_VSYNC1 为第 1 场垂直同步配置寄存器，仅第 0 路和第 4 路有效，其他路不支持。

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	act1_voff				act1_vbb				reserved				act1_height																				
Reset	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1	1
Bits	Access	Name	Description																														
[31:24]	RW	act1_voff	第 1 场开始到活动图像行距离（以行为单位）。 配置值为实际行数减 1。																														
[23:16]	RW	act1_vbb	第 1 场活动图像结束到第 2 场开始的行距（以行为单位）。 配置值为实际行数减 1。																														
[15:12]	RW	reserved	保留。																														
[11:0]	RW	act1_height	第 1 场活动图像的高度（以行为单位）。 配置值为实际行数减 1。																														

VI_CH2_VSYNC1

VI_CH2_VSYNC1 为第 1 场垂直同步配置寄存器，仅第 0 路和第 4 路有效，其他路不支持。

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	act1_voff				act1_vbb				reserved				act1_height																			
Reset	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	1	1	1
Bits	Access	Name	Description																													
[31:24]	RW	act1_voff	第 1 场开始到活动图像行距离（以行为单位）。 配置值为实际行数减 1。																													
[23:16]	RW	act1_vbb	第 1 场活动图像结束到第 2 场开始的行距（以行为单位）。 配置值为实际行数减 1。																													
[15:12]	RW	reserved	保留。																													
[11:0]	RW	act1_height	第 1 场活动图像的高度（以行为单位）。 配置值为实际行数减 1。																													



			配置值为实际行数减 1。
--	--	--	--------------

VI_CH0_VSYNC2

VI_CH0_VSYNC2 为第 2 场垂直同步配置寄存器，仅第 0 路和第 4 路有效，其他路不支持。

Offset Address			Register Name	Total Reset Value																												
Bit	0x0054			VI_CH0_VSYNC2	0x1600_811F																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	act2_voff act2_vbb hsyn_width_msb act2_height																															
Reset	0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1																															
Bits	Access	Name	Description																													
[31:24]	RW	act2_voff	第 2 场开始到活动图像行距离（以行为单位）。 配置值为实际行数减 1。																													
[23:16]	RW	act2_vbb	第 2 场活动图像结束到第 1 场开始的行距（以行为单位）。 配置值为实际行数减 1。																													
[15:12]	RW	hsyn_width_msb	水平同步信号宽度的高 4 位（仅限于 BT.601 模式），水平同步脉冲宽度默认为 128。																													
[11:0]	RW	act2_height	第 2 场活动图像的高度（以行为单位）。 配置值为实际行数减 1。																													

VI_CH2_VSYNC2

VI_CH2_VSYNC2 为第 2 场垂直同步配置寄存器，仅第 0 路和第 4 路有效，其他路不支持。

Offset Address			Register Name	Total Reset Value																												
Bit	0x0454			VI_CH2_VSYNC2	0x1600_811F																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	act2_voff act2_vbb hsyn_width_msb act2_height																															
Reset	0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1																															
Bits	Access	Name	Description																													
[31:24]	RW	act2_voff	第 2 场开始到活动图像行距离（以行为单位）。 配置值为实际行数减 1。																													

[23:16]	RW	act2_vbb	第 2 场活动图像结束到第 1 场开始的行距（以行为单位）。 配置值为实际行数减 1。
[15:12]	RW	hsyn_width_msb	水平同步信号宽度的高 4 位（仅限于 BT.601 模式），水平同步脉冲宽度默认为 128。
[11:0]	RW	act2_height	第 2 场活动图像的高度（以行为单位）。 配置值为实际行数减 1。

VI_CH0_HSYNC

VI_CH0_HSYNC 为第 2 场垂直同步配置寄存器，仅第 0 路和第 4 路有效，其他路不支持。

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	act_hoff				act_hbb				hsyn_width_ls_b				act_width																			
Reset	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1		
Bits	Access	Name	Description																													
[31:24]	RW	act_hoff	上一行结束到本行活动数据区域的距离（以时钟为单位）。 BT.656/601 模式下，配置值为像素的 2 倍减 1。																													
[23:16]	RW	act_hbb	活动区域结束到本行的结束距离（以时钟为单位）。 配置值为像素的 2 倍减 1。																													
[15:12]	RW	hsyn_width_lsb	水平同步信号宽度的低 4 位（仅限于 BT.601 模式）。 水平同步脉冲宽度默认为 128。																													
[11:0]	RW	act_width	活动图像宽度（以时钟为单位）。 BT.656/601 模式下，配置为像素的 2 倍减 1。 默认为 1439（720 个像素）。																													

VI_CH2_HSYNC

VI_CH2_HSYNC 为第 2 场垂直同步配置寄存器，仅第 0 路和第 4 路有效，其他路不支持。

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset Address	0x0458																								Total Reset Value							
Reset	0	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1		



Name	act_hoff				act_hbb				hsyn_width_lsb				act_width												
Reset	0	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	1	0	1	1	1	1
Bits	Access	Name	Description																						
[31:24]	RW	act_hoff	上一行结束到本行活动数据区域的距离（以时钟为单位）。 BT.656/601 模式下，配置值为像素的 2 倍减 1。																						
[23:16]	RW	act_hbb	活动区域结束到本行的结束距离（以时钟为单位）。 配置值为像素的 2 倍减 1。																						
[15:12]	RW	hsyn_width_lsb	水平同步信号宽度的低 4 位（仅限于 BT.601 模式）。 水平同步脉冲宽度默认为 128。																						
[11:0]	RW	act_width	活动图像宽度（以时钟为单位）。 BT.656/601 模式下，配置为像素的 2 倍减 1。 默认为 1439（720 个像素）。																						

PRIO_CONFIG

PRIO CONFIG 为各通道总线申请绝对优先级配置寄存器。

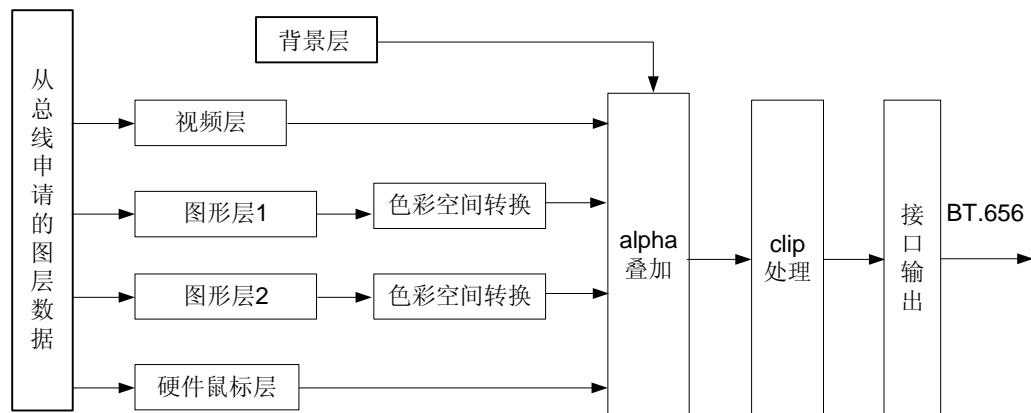
6.2 视频输出单元

6.2.1 概述

视频输出单元 VOU (Video Output Unit) 可以通过 BT.656 接口把存入到指定内存空间的视频数据送给芯片外模块。并且支持背景层、视频层、图形层和硬件鼠标层；视频层和图形层支持 alpha 叠加和色度 keying。视频层、图形层的大小以及在显示屏的起始位置可任意调节。

VOU 功能框图如图 6-26 所示。

图6-26 VOU 功能框图



6.2.2 特点

VOU 有以下特点：

- 支持 8 位 BT.656, YCbCr 4:2:2 标准输出接口 (PAL 制式/NTSC 制式@27MHz)。
- 视频层数据支持 SPYCbCr 4:2:0 和 SPYCbCr 4:2:2 数据格式，亮度色度分开存储，其中色度分量按字节间插存储。
- 图形层支持 package YCbCr 4:4:4、package YCbCr 4:2:2、aRGB1555 和 aRGB8888 数据格式。
- 支持 32×32 像素大小的硬件鼠标叠加。
- 支持视频图像层、图形层 1、图形层 2 和硬件鼠标的四层叠加，其中图形层 1 和图形层 2 的叠加支持 129 层 alpha 叠加，支持 chroma keying 和 mask。
- 支持 24 BPP (Bit Per Pixel) 的背景颜色层。
- 支持在 BT.656 模式下，对输出像素值 clip 处理。
- 支持内部 FIFO Underflow 告警。



6.2.3 信号描述

表6-8 VOU 视频接口信号

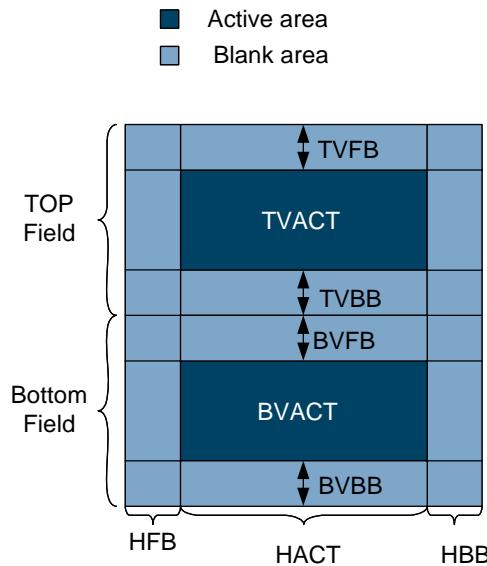
信号名	方向	描述	对应管脚
CLK_VO	O	芯片输出的 VOU 时钟，27MHz，正反相可以配置，具体请参见“ 6.2.5 工作方式 ”中“ VOU 时钟 ”的内容。	VOCK
VO_PDATA[7]	O	芯片输出的 BT.656 接口信号。	VODAT7
VO_PDATA[6]			VODAT6
VO_PDATA[5]			VODAT5
VO_PDATA[4]			VODAT4
VO_PDATA[3]			VODAT3
VO_PDATA[2]			VODAT2
VO_PDATA[1]			VODAT1
VO_PDATA[0]			VODAT0

6.2.4 功能描述

6.2.4.1 视频接口时序

VOU 只有一个 BT.656 接口，可实现 PAL 或 NTSC 制输出。

图6-27 VOU 输出图像结构示意图



注:

T/BVBB: top/bottom vertical back porch (顶/底场垂直后消隐)
T/BVACT: top/bottom vertical active area (顶/底场垂直活动有效区域)
T/BVFB: top/bottom vertical front porch (顶/底场垂直前消隐)
H/VBB: horizontal/vertical back porch (水平/垂直后消隐)
H/VACT: horizontal/vertical active area (水平/垂直活动有效区域)
H/VFB: horizontal/vertical front porch (水平/垂直前消隐)
H/VPW: horizontal/vertical pulse width (水平/垂直脉冲宽度)

BT.656 输出符合 ITU-R BT656 协议, PAL 和 NTSC 制式的时序参数请参见 [VO_VSYNC1](#)、[VO_VSYNC2](#) 和 [VO_HSYNC](#)。

6.2.4.2 VOU 数据格式

表 6-9 和表 6-10 介绍了图形层和视频层的数据格式 (32bit), 硬件鼠标层的数据格式请参见 “6.2.4 功能描述” 中 “[硬件鼠标层的处理方式](#)” 的内容。

表6-9 VOU 数据格式在 SDRAM 中的存放方式 (高 16 位)

数据格式		高 16bit														
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
SPYCbCr4 22	Y	Y3										Y2				
	CbCr	Cb2										Cr2				
SPYCbCr4 20	Y	Y3										Y2				
	CbCr	Cb2										Cr2				
PackageYCbCr444 a		a										Y0				



数据格式	高 16bit															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PackageYCbCr422 α	Y1								Cb0							
RGB555 α	α 1	R1				G1				B1						
RGB888 α	α								R							

表6-10 VOU 数据格式在 SDRAM 中的存放方式 (低 16 位)

数据格式	低 16bit																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SPYCbCr4 22	Y	Y1								Y0							
	CbCr	Cb0								Cr0							
SPYCbCr4 20	Y	Y1								Y0							
	CbCr	Cb0								Cr0							
PackageYCbCr444 α	Cb0								Cr0								
PackageYCbCr422 α	Y0								Cr0								
RGB555 α	α 0	R0				G0				B0							
RGB888 α	G								B								

6.2.4.3 数据读取方式

VOU 中的数据读取方式有 2 种:

- 帧模式数据读取
- 场模式数据读取

视频层和图形层 (包括硬件鼠标层) 可分别控制数据读取方式。中断产生模式也分为场模式和帧模式, 当配置为场模式时, 部分中断按场产生, 当配置为帧模式时, 部分中断按帧产生。

6.2.4.4 图像参数

图像参数包括 2 类:



- SDRAM 中的参数, 如表 6-11 所示。
- 图像显示的参数, 如表 6-12 所示。

表6-11 VOU 存储图像参数

图像参数	含义
addr	图像的起始地址
stride	存放在 SDRAM 中相邻两行行首的距离
width	图像的宽度
height	图像的高度
dataformat	图像的数据格式
帧场模式	数据读取模式, 按帧读取或按场读取

表6-12 VOU 显示图像参数

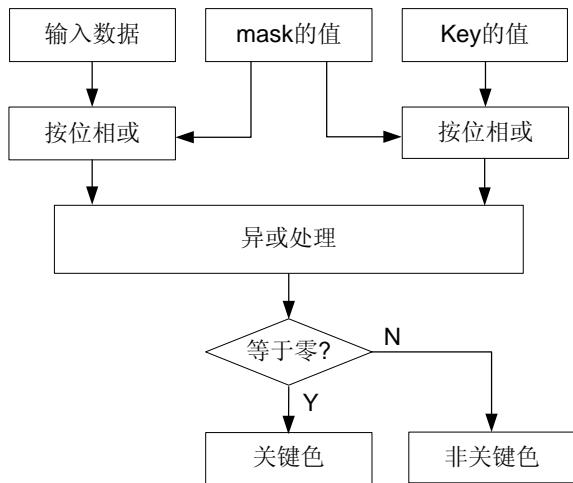
图像参数	含义
start_line	显示图层的起始行数
start_pixel	显示图层的起始列号
width	图像的宽度
height	图像的高度

6.2.4.5 关键色处理

在 VOU 中, 图形层 1 和图形层 2 进行关键色处理。如果是关键色, 对其做透明处理; 否则进行叠加。在关键色处理前, 需要进行对数据进行扩展成 a YCbCr8888 或 a RGB8888 的数据格式, 其处理如图 6-28 所示。



图6-28 关键色处理流程图



6.2.4.6 RGB 到 YCbCr 转换

由于 4 层叠加在 YCbCr 颜色空间域进行处理的，对于图形层的 RGB 数据在叠加之前需要进行颜色空间转换，RGB 到 YCbCr 的转换公式如下所示：

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} coef00 & coef01 & coef02 \\ coef10 & coef11 & coef12 \\ coef20 & coef21 & coef22 \end{bmatrix} \cdot \begin{bmatrix} R'255 \\ G'255 \\ B'255 \end{bmatrix}$$

公式中的 RGB 的范围 0~255, Y 的值 16~235, CbCr 的值 16~240。其系数如下，

coef00=0.257; coef01=0.504; coef02=0.098;

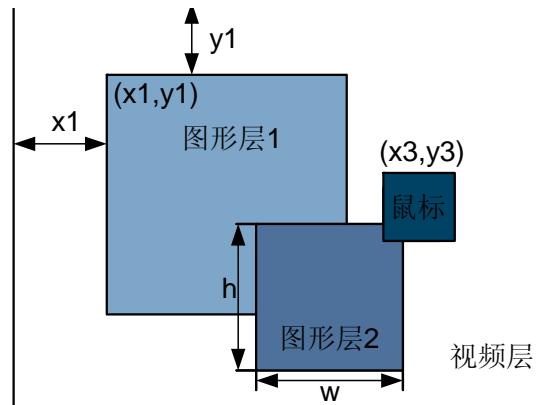
coef10=0.148; coef11=0.291; coef12=0.439;

coef20=0.439; coef21=0.368; coef22=0.071;

6.2.4.7 图层叠加

VOU 支持 4 层图像叠加。优先级最高为硬件鼠标，其次为图形层 2、图形层 1、视频层。即叠加层次依次为视频层，图形层 1，图形层 2 和硬件鼠标。图像的分层如图 6-29 所示。图层叠加是按照 129 级叠加， α 值用 8bit、0~128 表示，0 表示当前层做透明处理，128 表示当前层做全覆盖处理。同时对叠加过的图像数据进行 CLIP 处理，使 Y 的值限制在寄存器 `VO_CLIP` 的 `low_luma` 和 `high_luma` 之间，低于 `low_luma` 的值，被嵌位到 `low_luma`，大于 `high_luma` 的值，被嵌位到 `high_luma`，CbCr 的值限制在寄存器 `VO_CLIP` 的 `low_chroma` 和 `high_chroma` 之间，低于 `low_chroma` 的值，被嵌位到 `low_chroma`，大于 `high_chroma` 的值，被嵌位到 `high_chroma`。

图6-29 图像分层示意图



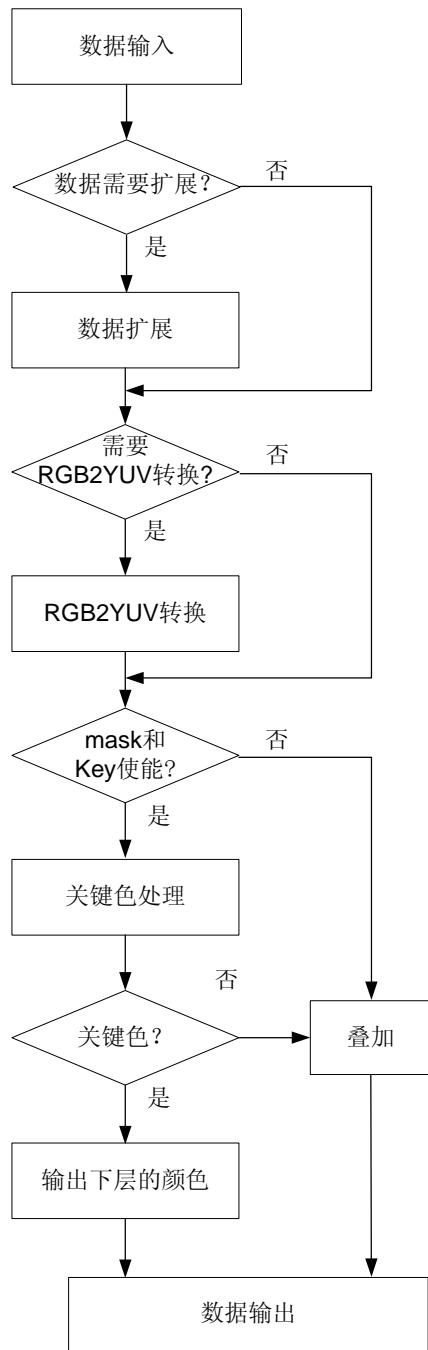
图形层和硬件鼠标层在视频层中的位置由图像坐标 (x, y) 以及图形层和硬件鼠标层的宽 (w) 和高 (h) 唯一确定。

6.2.4.8 图形层、视频层的处理方式

图形层的处理流程如图 6-30 所示，而视频层直接把图像数据送给叠加器进行叠加。



图6-30 图形层的处理流程示意图



注：

当图形层的数据格式是 α RGB1555 时，数据需要扩展。

当图形层的数据格式是 α RGB1555 或 α RGB8888 时，数据需要 RGB 到 YCbCr 转换。

6.2.4.9 硬件鼠标层的处理方式

VOU 支持硬件鼠标层处理，硬件鼠标层的大小为 32×32 的大小。硬件鼠标有 3 种模式，由控制寄存器配置。



32×32×2bpp 双色和透明模式

该模式用于支持鼠标的数据格式。每一个像素用 2 位表示，代表如表 6-13 所示的 4 种颜色。

- 鼠标颜色 0 和鼠标颜色 1 用于画一个鼠标。
- 鼠标颜色 2 用于透明处理（允许在鼠标后面的视频层被显示）。
- 鼠标颜色 3 用于反向透明处理（鼠标后面的视频层被显示，但是其显示颜色与原视频层颜色相反）。

表6-13 32×32×2bpp 双色和透明模式列表

bit/pixel	在相应位置的颜色显示
00	鼠标颜色 0。
01	鼠标颜色 1。
10	透明处理。在当前硬件鼠标后的视频层像素被显示。
11	反向透明处理。在当前硬件鼠标后的视频层被反色显示。

32×32×2bpp 四色模式

该模式提供 4 种颜色来显示鼠标。每一个像素用 2 位表示，代表如表 6-14 所示的 4 种颜色。

表6-14 32×32×2bpp 四色模式列表

bit/pixel	在相应位置的颜色显示
00	鼠标颜色 0。
01	鼠标颜色 1。
10	鼠标颜色 2。
11	鼠标颜色 3。

32×32×2bpp 三色和透明模式

该模式提供 3 种颜色来显示鼠标，其中一种颜色对视频层透明处理（覆盖在鼠标后的像素被显示）。每一个像素用 2 位表示，4 种颜色的显示如表 6-15 所示。

表6-15 2×32×2bpp 三色和透明模式列表

bit/pixel	在相应位置的颜色显示
00	鼠标颜色 0。



bit/pixel	在相应位置的颜色显示
01	鼠标颜色 1。
10	鼠标颜色 2。
11	透明处理。覆盖在硬件鼠标后的像素被显示。

6.2.5 工作方式

6.2.5.1 VOU 时钟

VOU 模块有 2 个时钟：

- hclk 是总线时钟域，负责数据读取。
- clk_vo 是 VOU 的工作时钟，负责 VOU 的图像处理和接口时序的产生。

hclk 和 clk_vo 的门控、反相和工作频率的相关内容如表 6-16 所示。

表6-16 hclk 和 clk_vo 的比较

内容	hclk	clk_vo
门控	SC_PEREN[11]: 0: 不改变原来状态。 1: VOU 时钟使能。 SC_PERDIS[11]: 0: 不改变原来状态。 1: VOU 时钟禁止。 SC_PERCLKEN[11]: 0: VOU 的时钟禁止状态。 1: VOU 的时钟使能状态。	SC_PEREN[11]: 0: 不改变原来状态。 1: VOU 时钟使能。 SC_PERDIS[11]: 0: 不改变原来状态。 1: VOU 时钟禁止。 SC_PERCLKEN[11]: 0: VOU 的时钟禁止状态。 1: VOU 的时钟使能状态。
反向	不可以进行反相控制	当作为 VOU 的输出时钟时，可通过配置系统控制器进行反相控制。 SC_PERCTRL2[0]: 0: 反相输出。 1: 正相输出。
工作频率	135MHz	27MHz

6.2.5.2 VOU 的复位

VOU 有 3 个复位信号：

- RST_N



- VOU_BRST_N
- VOU_VRST_N

具体的复位描述如表 6-17 所示。

表6-17 VOU 复位描述

复位信号	RST_N	VOU_BRST_N	VOU_VRST_N
所在时钟域	hclk	hclk	clk_vo
功能	复位总线时钟域的配置寄存器	复位总线时钟域的工作寄存器	复位接口时钟域的所用寄存器、状态机等
软复位	支持, 配置 SC_PERCTRL0[19]	支持, 配置 SC_PERCTRL0[19]	支持, 配置 SC_PERCTRL0[19]
硬复位	支持	支持	支持

6.2.5.3 VOU 的中断

VOU 在系统中的中断号为“16”，在 VOU 子系统中有 8 个中断源，在 VOU 的寄存器中有一个表示中断使能的寄存器 [VO_INT_MASK](#) 和经过使能控制后产生的中断状态寄存器 [VO_INT_STATUS](#)。

6.2.5.4 背景色输出

当视频层、图形层和硬件鼠标层都不使能时，VOU 工作使能时 ([VO_CTRL\[31\]=1](#))，VOU 只输出背景色，配置相应的接口时序，BT.656 模式输出时具体配置如下：

步骤 1 通过 [VO_BG_COLOR](#) 配置背景色。

步骤 2 通过 [VO_CLIP](#) 配置 clip。

步骤 3 通过 [VO_VSYNC1](#)、[VO_VSYNC2](#) 和 [VO_HSYNC](#) 配置 PAL 和 NTSC 时序。

步骤 4 配置 [VO_CTRL\[31\]](#) 为 1，VOU 开始输出背景色。

----结束

6.2.5.5 视频层输出

VOU 的视频层输出配置步骤如下：

步骤 1 通过 [VO_CTRL\[30\]](#) 配置视频层是帧模式或场模式。

步骤 2 通过 [表 6-18](#) 所示的寄存器配置视频层数据存储参数。

步骤 3 通过 [表 6-19](#) 所示的寄存器配置视频层显示参数。

步骤 4 通过 [VO_INT_MASK\[2:1\]](#) 配置视频层相应中断使能。

步骤 5 通过 [VO_CTRL\[27\]](#) 配置视频层使能，



步骤 6 配置视频输出, 详细内容请参见“[6.2.5.4 背景色输出](#)”。

----结束

表6-18 视频层存储参数表

存储参数	视频层	
	亮度信号	色度信号
首地址	VO_MLADDR[31:0]	VO_MCADDR[31:0]
STRIDE	VO_MOFFSET[31:16]	VO_MOFFSET[15:0]
数据格式	VO_CTRL[26]	
图像的宽度	VO_IMAGE[11:0]	
图像的高度	VO_IMAGE[23:12]	

表6-19 视频层显示参数

显示参数	视频层
图像显示行开始	VO_IMAGE_OFF[23:12]
图像显示列开始	VO_IMAGE_OFF[11:0]
图像显示宽度	VO_IMAGE[11:0]
图像显示高度	VO_IMAGE[23:12]

6.2.5.6 图形层输出

VOU 的图形层输出配置步骤如下:

步骤 1 通过如[表 6-20](#) 所示的寄存器配置图形层的数据存储参数。

步骤 2 通过如[表 6-21](#) 所示的寄存器配置图形层的显示参数。

步骤 3 通过 [VO_OVL1_KEY](#)/[VO_OVL2_KEY](#) 和 [VO_MASK](#) 配置图形层的关键色。

步骤 4 通过 [VO_INT_MASK\[5:4\]](#) 和 [VO_INT_MASK\[0\]](#) 配置图形层相应的中断使能。

步骤 5 通过 [VO_CTRL\[25:24\]](#) 配置图形层使能。

步骤 6 配置视频输出, 详细内容请参见“[6.2.5.4 背景色输出](#)”。

----结束



表6-20 图形层数据存储参数表

存储参数	图形层 1	图形层 2
首地址	VO_OVL1ADDR[31:0]	VO_OVL2ADDR[31:0]
Stride	VO_OVLOFFSET[31:16]	VO_OVLOFFSET[15:0]
数据格式	VO_CTRL[23:22]	VO_CTRL[19:18]
图像的宽度	VO_OVL1_IMAGE[11:0]	VO_OVL2_IMAGE[11:0]
图像的高度	VO_OVL1_IMAGE[23:12]	VO_OVL2_IMAGE[23:12]

表6-21 图形层显示参数

显示参数	图形层 1	图形层 2
图像显示行开始	VO_OVL1_START[23:12]	VO_OVL2_START[23:12]
图像显示列开始	VO_OVL1_START[11:0]	VO_OVL2_START[11:0]
图像显示宽度	VO_OVL1_IMAGE[11:0]	VO_OVL2_IMAGE[11:0]
图像显示高度	VO_OVL1_IMAGE[23:12]	VO_OVL2_IMAGE[23:12]

6.2.5.7 硬件鼠标层显示输出

VOU 的硬件鼠标层显示输出配置步骤如下：

- 步骤 1 通过如表 6-22 所示的寄存器配置硬件鼠标层的存储参数。
- 步骤 2 通过如表 6-23 所示的寄存器配置硬件鼠标层的显示参数。
- 步骤 3 通过 VO_HC_COLOR0、VO_HC_COLOR1、VO_HC_COLOR2、VO_HC_COLOR3 配置硬件鼠标层的颜色寄存器。
- 步骤 4 通过 VO_INT_MASK[4]配置硬件鼠标层数据读取完毕中断使能。
- 步骤 5 通过 VO_CTRL[15]配置硬件鼠标层使能。
- 步骤 6 配置视频输出，详细内容请参见“6.2.5.4 背景色输出”。

----结束

表6-22 硬件鼠标层的存储参数

存储参数	硬件鼠标层
首地址	VO_HCADDR
数据格式	VO_CTRL[14:13]



存储参数	硬件鼠标层
图像的宽度	硬件鼠标层的大小为 32×32, 宽度和高度软件不可配置。
图像的高度	

表6-23 硬件鼠标层的显示参数

显示参数	视频层
图像显示行开始	VO_HC_START[23:12]
图像显示列开始	VO_HC_START[11:0]
图像显示宽度	硬件鼠标层的大小为 32×32, 宽度和高度软件不可配置。
图像显示高度	

6.2.6 寄存器概览

VOU 寄存器概览如表 6-24 所示。

表6-24 VOU 寄存器概览 (基址是 0x9005_0000)

偏移地址	名称	描述	页码
0x00	VO_CTRL	VOU 控制寄存器 ^a	6-48
0x04	VO_INT_MASK	VOU 中断使能寄存器 ^b	6-48
0x08	VO_INT_STATUS	VOU 中断状态寄存器 ^a	6-48
0x0C	VO_STATUS	VOU 状态寄存器	6-48
0x10	VO_VSYNC1	帧时序寄存器 1 ^a	6-48
0x14	VO_VSYNC2	帧时序寄存器 2	6-48
0x18	VO_HSYNC	水平时序寄存器	6-48
0x1C	VO_IMAGE	图像寄存器 ^b	6-48
0x20	VO_OVL1_IMAGE	图形层 1 寄存器	6-48
0x24	VO_OVL2_IMAGE	图形层 2 寄存器	6-48
0x28	RESERVED	保留	-
0x2C	VO_IMAGE_OFF	图像偏移寄存器 ^b	6-48
0x30	VO_BG_COLOR	背景颜色寄存器	6-48



偏移地址	名称	描述	页码
0x34	VO_CLIP	Clipping 值寄存器 ^b	6-48
0x38	VO_MASK	Keying 的 mask 值寄存器	6-48
0x3C	VO_OVL1_KEY	图形层 1 的 key 值寄存器	6-48
0x40	VO_OVL2_KEY	图形层 2 的 key 值寄存器	6-48
0x44	VO_HC_COLOR0	硬件鼠标颜色 0 寄存器	6-48
0x48	VO_HC_COLOR1	硬件鼠标颜色 1 寄存器	6-48
0x4C	VO_HC_COLOR2	硬件鼠标颜色 2 寄存器	6-48
0x50	VO_HC_COLOR3	硬件鼠标颜色 3 寄存器	6-48
0x54	VO_OVL1_START	图形层 1 起始位置寄存器	6-48
0x58	VO_OVL2_START	图形层 2 起始位置寄存器	6-48
0x5C	VO_HC_START	硬件鼠标起始位置寄存器	6-48
0x60	VO_OVL1_ALPHA	图形层 1 的 alpha 值寄存器	6-48
0x64	VO_OVL2_ALPHA	图形层 2 的 alpha 值寄存器	6-48
0x68	VO_MLADDR	视频层亮度分量地址寄存器	6-48
0x6C	VO_MCADDR	视频层色度分量地址寄存器	6-48
0x70	RESERVED	保留	-
0x74	VO_OVL1ADDR	图形层 1 地址寄存器 ^b	6-48
0x78	VO_OVL2ADDR	图形层 2 地址寄存器	6-48
0x7C	VO_HCADDR	硬件鼠标地址寄存器	6-48
0x80	VO_MOFFSET	视频层的亮度和色度行偏移量寄存器	6-48
0x84	VO_OVLOFFSET	图形层行偏移量寄存器	6-48

注：

a: 及时性寄存器，软件写该寄存器将及时生效。

b: 非及时性寄存器，软件写入的新值只有在 reload 中断到来时才会生效。

没有标识 a 或 b 表示该寄存器不可写。

6.2.7 寄存器描述

本节详细描述了 VOU 的寄存器。



VO_CTRL

VO_CTRL 为控制寄存器，控制 VOU 的整个工作流程以及一些配置信息。

[24]	RW	ovl2_en	图形层 2 使能。 0: 不使能。 1: 使能。
[23:22]	RW	ovl1_type	图形层 1 的类型。 00: packageYCbCr4:2:2+ α 。 01: packageYCbCr4:4:4+ α 。 10: RGB5:5:5+ α 。 11: RGB8:8:8+ α 。
[21]	RW	ovl1_key_en	图形层 1 叠加 keying 使能。 0: 不使能。 1: 使能。
[20]	RW	ovl1_mask_en	图形层 1 keying mask 使能。 0: 不使能。 1: 使能。
[19:18]	RW	ovl2_type	图形层 2 的类型。 具体描述同 ovl1_type。
[17]	RW	ovl2_key_en	图形层 2 叠加 keying 使能。 0: 不使能。 1: 使能。
[16]	RW	ovl2_mask_en	图形层 2 叠加 keying mask 使能。 0: 不使能。 1: 使能。
[15]	RW	hc_en	硬件鼠标使能。 0: 不使能。 1: 使能。
[14:13]	RW	hc_mode	硬件鼠标模式。 00: 双色和透明模式; 01: 4 色模式; 10: 3 色和透明模式; 11: 保留。 模式相对于颜色值的使用请参见“ 6.2.4.9 硬件鼠标层的处理方式 ”。
[12:0]	-	reserved	保留。



VO_INT_MASK

VO_INT_MASK 为中断使能寄存器，用于对各个中断使能。当中断使能位为 1 时，相应的中断使能，否则该中断被屏蔽。

Offset Address			Register Name			Total Reset Value																										
0x04			VO_INT_MASK			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												osd_reload_en			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:8]	-	reserved	保留。																													
[7]	RW	bus_err_int_en	AHB master 读 AHB 总线错误中断使能。 0: 屏蔽中断。 1: 使能中断。																													
[6]	RW	lbw_int_en	带宽分配太低出现内部 FIFO underflow 中断使能。 0: 屏蔽中断。 1: 使能中断。																													
[5]	RW	ovl1_rint_en	图形层 1 数据读取完毕中断使能。 0: 屏蔽中断。 1: 使能中断。																													
[4]	RW	ovl2_rint_en	图形层 2 数据读取完毕中断使能。 0: 屏蔽中断。 1: 使能中断。																													
[3]	RW	hc_rint_en	硬件鼠标数据读取完毕中断使能。 0: 屏蔽中断。 1: 使能中断。																													
[2]	RW	main_rint_en	视频层数据读取完毕中断使能。 0: 屏蔽中断。 1: 使能中断。																													
[1]	RW	main_reload_en	视频层参数更新完毕中断。 0: 屏蔽中断。 1: 使能中断。																													

[0]	RW	osd_reload_en	图形层参数更新完毕中断。 0: 屏蔽中断。 1: 使能中断。
-----	----	---------------	--------------------------------------

VO_INT_STATUS

VO_INT_STATUS 为中断状态寄存器, 若某一位产生中断, 并且该位对应的中断使能位为 1 时, 相应的中断状态位置 1。该寄存器写 1 清 0。

	Offset	Address	Register Name	Total Reset Value
	Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	VO_INT_STATUS	0x0000_0000
Name		reserved		
Reset		0 0		
Bits	Access	Name	Description	
[31:8]	-	reserved	保留。	
[7]	WC	bus_err_int	AHB master 读 AHB 总线错误中断。 0: 未产生中断。 1: 已产生中断。	
[6]	WC	lbw_int	带宽分配太低出现内部 FIFO underflow 中断。 0: 未产生中断。 1: 已产生中断。	
[5]	WC	ovl1_rint	图形层 1 数据读取完毕中断。 0: 未产生中断。 1: 已产生中断。 注: 该中断与帧模式或场模式读取有关, 帧模式时一帧产生一个中断, 场模式时一场产生一个中断。	
[4]	WC	ovl2_rint	图形层 2 数据读取完毕中断。 0: 未产生中断。 1: 已产生中断。 注: 该中断与帧模式或场模式读取有关, 帧模式时一帧产生一个中断, 场模式时一场产生一个中断。	
[3]	WC	hc_rint	硬件鼠标数据读取完毕中断。	



			0: 未产生中断。 1: 已产生中断。 注: 该中断与帧模式或场模式读取有关, 帧模式时一帧产生一个中断, 场模式时一场产生一个中断。
[2]	WC	main_rint	视频层数据读取完毕中断。 0: 未产生中断。 1: 已产生中断。 注: 该中断与帧模式或场模式读取有关, 帧模式时一帧产生一个中断, 场模式时一场产生一个中断。
[1]	WC	main_reload_int	视频层参数更新完毕中断。 0: 未产生中断。 1: 已产生中断。 注: 该中断与帧模式或场模式读取有关, 帧模式时一帧产生一个中断, 场模式时一场产生一个中断。
[0]	WC	osd_reload_int	图形层参数更新完毕中断 0: 未中断产生。 1: 已产生中断。 注: 该中断与帧模式或场模式读取有关, 帧模式时一帧产生一个中断, 场模式时一场产生一个中断。

VO_STATUS

VO_STATUS 为状态寄存器, 反映硬件当前的工作状态。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	VO_STATUS	0x0000_0000
Name	reserved		field2
Reset	0 0		
Bits	Access	Name	Description
[31:12]	-	reserved	保留。
[11]	RO	main_done	视频层数据读取完毕。 0: 数据未读取完毕。 1: 数据读取完毕。

[10]	RO	ovl1_done	图形层 1 数据读取完毕。 0: 数据未读取完毕。 1: 数据读取完毕。
[9]	RO	ovl2_done	图形层 2 数据读取完毕。 0: 数据未读取完毕。 1: 数据读取完毕。
[8]	RO	hc_done	硬件鼠标数据读取完毕。 0: 数据未读取完毕。 1: 数据读取完毕。
[7]	RO	ml_under_flow	视频层亮度分量内部 FIFO underflow。 0: FIFO 不空。 1: FIFO 空。
[6]	RO	mc_under_flow	视频层色度分量内部 FIFO underflow。 0: FIFO 不空。 1: FIFO 空。
[5]	-	reserved	保留。
[4]	RO	ovl1_under_flow	图形层 1 内部 FIFO underflow。 0: FIFO 不空。 1: FIFO 空。
[3]	RO	ovl2_under_flow	图形层 2 内部 FIFO underflow。 0: FIFO 不空。 1: FIFO 空。
[2]	RO	bus_err	Master 读取总线错误状态。 0: 正确。 1: 错误。
[1]	-	reserved	保留。
[0]	RO	field2	第 2 场指示信号。 0: 当前场非第 2 场。 1: 当前场是第 2 场。

VO_VSYNC1

VO_VSYNC1 为 TV 接口情况下第 1 场垂直同步配置寄存器。该寄存器配置后立即生效，影响到输出的时序。默认为 PAL 制式，也可以配置为 NTSC 制式。



VO_VSYNC2

VO_VSYNC2 为 TV 接口情况下第 2 场垂直同步配置寄存器。该寄存器配置后立即生效，影响到输出的时序。默认为 PAL 制式，也可以配置为 NTSC 制式。

[15:12]	-	reserved	保留。
[11:0]	RW	bvact	第 2 场活动图像的高度（以行为单位）。 配置值为实际值减 1。 默认 PAL 制式配置为 287。

VO_HSYNC

VO_HSYNC 为 TV 接口情况下水平同步配置寄存器。该寄存器配置后立即生效，影响到输出的时序。默认为 PAL 制式，也可以配置为 NTSC 制式。

	Offset Address	Register Name	Total Reset Value
Bit	0x18	VO_HSYNC	0x41D7_059F
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	act_hoff	act_hbb	reserved
Reset	0 1 0 0 0 0 0 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1	act_width	
Bits	Access	Name	Description
[31:22]	RW	hfb	上一行结束到本行活动数据区域的距离（以时钟为单位）。 在 BT.656 模式下，配置值为像素的 2 倍减 1。 默认 PAL 制式配置为 263（实际距离为 132 个像素）。
[21:16]	RW	hbb	活动区域结束到本行的结束距离（以时钟为单位）。 配置值为实际值减 1。 默认 PAL 制式配置为 23。
[15:12]	-	reserved	保留。
[11:0]	RW	hact	活动图像宽度（以时钟为单位）。 在 BT.656 模式下，配置值为像素的 2 倍减 1。 默认 PAL 制式配置为 1439（720 个像素）。

VO_IMAGE

VO_IMAGE 为视频图像层宽高属性寄存器。

- 视频层配置为帧模式下，显示图像的实际高度为 image_height+1。
- 视频层配置为场模式下，显示图像的实际高度为(image_height +1)× 2。



说明

帧模式的时候，图像的实际高度需要配置为偶数，配置为奇数会被截断为偶数处理。以下各层的高度意义相同。

VO_OVL1_IMAGE

VO_OVL1_IMAGE 为图形层 1 宽高属性寄存器。

- 叠加层配置为帧模式时, 图形层 1 实际的高度为 $ovl1_height+1$ 。
 - 叠加层配置为场模式时, 图形层 1 实际的高度为 $(ovl1_height+1) \times 2$ 。

[11:0]	RW	ovl1_width	图形层 1 的宽度（以像素为单位）。 配置值为实际值减 1。
--------	----	------------	-----------------------------------

VO_OVL2_IMAGE

VO_OVL2_IMAGE 为图形层 2 宽高属性寄存器。

- 叠加层配置为帧模式时，图形层 2 实际的高度为 ovl2_height+1。
- 叠加层配置为场模式时，图形层 2 实际的高度为 $(ovl2_height+1) \times 2$ 。

	Offset Address		Register Name	Total Reset Value																												
	0x24		VO_OVL2_IMAGE	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		ovl2_height		ovl2_width																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:12]	RW	ovl2_height	图形层 2 的高度（以行为单位）。 配置值为实际值减 1。																													
[11:0]	RW	ovl2_width	图形层 2 的宽度（以像素为单位）。 配置值为实际值减 1。																													

VO_IMAGE_OFF

VO_IMAGE_OFF 为视频图像叠加起始坐标属性寄存器。

由于 TV 显示是隔行的，所以有：

- 视频图像层配置为帧模式，显示图像的实际开始行坐标为 image_voff+1。
- 视频图像层配置为场模式，显示图像的实际开始行坐标为 $(image_voff+1) \times 2$ 。

说明

- 视频图像的起始行距与图像的高度之和不能大于时序寄存器中配置的图像高度。
- 视频图像的起始像素与图像的宽度之和不能大于时序寄存器中配置的图像宽度，即不能超出显示设备的最大分辨率。



Offset Address			Register Name			Total Reset Value																										
0x2C			VO_IMAGE_OFF			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved			image_voff			image_hoff																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:12]	RW	image_voff	实际传输图像第 1 行相对于活动图形开始行的距离（以行为单位）。																													
[11:0]	RW	image_hoff	实际传输图像开始对于活动图形开始的距离（以像素为单位）。																													

VO_BG_COLOR

VO_BG_COLOR 为背景颜色寄存器，背景颜为 YCbCr 数据格式。

Offset Address			Register Name			Total Reset Value																											
0x30			VO_BG_COLOR			0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved			bg_y			bg_u			bg_v																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																														
[31:24]	-	reserved	保留。																														
[23:16]	RW	bg_y	背景颜色 Y 分量。																														
[15:8]	RW	bg_u	背景颜色 Cb 分量。																														
[7:0]	RW	bg_v	背景颜色 Cr 分量。																														

VO_CLIP

VO_CLIP 为 clip 配置寄存器，在 TV 显示需要 clip 时，clip 值寄存器规定了亮度和色度的范围。

VO MASK

VO_MASK 为色度 keying mask 寄存器，在做色度 keying 时，为了提高色度 key 值的范围，对每一个色度 key 值增加了低位的 mask 位。被屏蔽的位不参与比较。

VO_OVL1_KEY

VO_OVL1_KEY 为图形层 1 色度 key 值寄存器。



Offset Address			Register Name	Total Reset Value																												
0x3C			VO_OVL1_KEY	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved							ovl1_key_y							ovl1_key_u							ovl1_key_v										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:16]	RW	ovl1_key_y	图形层 1 的亮度 Y (或者 R 分量) keying 比较值。																													
[15:8]	RW	ovl1_key_u	图形层 1 的色度 Cb (或者 G 分量) keying 比较值。																													
[7:0]	RW	ovl1_key_v	图形层 1 的色度 Cr (或者 B 分量) keying 比较值。																													

VO_OVL2_KEY

VO_OVL2_KEY 为图形层 2 色度 key 值寄存器。

Offset Address			Register Name	Total Reset Value																												
0x40			VO_OVL2_KEY	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved							ovl2_key_y							ovl2_key_u							ovl2_key_v										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:16]	RW	ovl2_key_y	图形层 2 的亮度 Y (或者 R 分量) keying 比较值。																													
[15:8]	RW	ovl2_key_u	图形层 2 的色度 Cb (或者 G 分量) keying 比较值。																													
[7:0]	RW	ovl2_key_v	图形层 2 的色度 Cr (或者 B 分量) keying 比较值。																													

VO_HC_COLOR0

VO_HC_COLOR0 为硬件鼠标颜色 0 寄存器。



说明

硬件鼠标层只支持 YCbCr 的数据格式。

Offset Address			Register Name	Total Reset Value																												
0x44			VO_HC_COLOR0	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved							hc_y0							hc_u0							hc_v0										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:16]	RW	hc_y0	硬件鼠标色度 0 的亮度 Y 值。																													
[15:8]	RW	hc_u0	硬件鼠标色度 0 的色度 Cb 值。																													
[7:0]	RW	hc_v0	硬件鼠标色度 0 的色度 Cr 值。																													

VO_HC_COLOR1

VO_HC_COLOR1 为硬件鼠标颜色 1 寄存器。

Offset Address			Register Name	Total Reset Value																												
0x48			VO_HC_COLOR1	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved							hc_y1							hc_u1							hc_v1										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:16]	RW	hc_y1	硬件鼠标色度 1 的亮度 Y 值。																													
[15:8]	RW	hc_u1	硬件鼠标色度 1 的色度 Cb 值。																													
[7:0]	RW	hc_v1	硬件鼠标色度 1 的色度 Cr 值。																													

VO_HC_COLOR2

VO_HC_COLOR2 为硬件鼠标颜色 2 寄存器。



	Offset Address		Register Name		Total Reset Value																											
	0x4C		VO_HC_COLOR2		0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		hc_y2		hc_u2		hc_v2																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:16]	RW	hc_y2	硬件鼠标色度 2 的亮度 Y 值。																													
[15:8]	RW	hc_u2	硬件鼠标色度 2 的色度 Cb 值。																													
[7:0]	RW	hc_v2	硬件鼠标色度 2 的色度 Cr 值。																													

VO_HC_COLOR3

VO_HC_COLOR3 为硬件鼠标颜色 3 寄存器。

	Offset Address		Register Name		Total Reset Value																											
	0x50		VO_HC_COLOR3		0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		hc_y3		hc_u3		hc_v3																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:16]	RW	hc_y3	硬件鼠标色度 3 的亮度 Y 值。																													
[15:8]	RW	hc_u3	硬件鼠标色度 3 的色度 Cb 值。																													
[7:0]	RW	hc_v3	硬件鼠标色度 3 的色度 Cr 值。																													

VO_OVL1_START

VO_OVL1_START 为图形层 1 起始坐标属性寄存器。

由于 TV 显示是隔行的，所以有：

- 叠加层配置为帧模式，图形层 1 的相对行坐标为 ovl1_start_line+1。
- 叠加层配置为场模式，图形层 1 的相对行坐标为(ovl1_start_line+1)×2。

 说明

- 图形层 1 的起始行距与图形层 1 的高度之和不能大于时序寄存器中配置的图像高度。
- 图形层 1 的起始像素与图形层 1 的宽度之和不能大于时序寄存器中配置的图像宽度，即不能超出显示设备的最大分辨率。

	Offset Address		Register Name		Total Reset Value																											
	0x54		VO_OVL1_START		0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		ovl1_start_line		ovl1_start_pixel																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:24]	-	reserved	保留。																													
[23:12]	RW	ovl1_start_line	图形层 1 相对于实际传输图像开始叠加时的行坐标，从 0 开始。 配置值为实际值减 1。																													
[11:0]	RW	ovl1_start_pixel	图形层 1 相对于实际传输图像开始叠加时的像素坐标，从 0 开始。 例如 (0, 0) (第 1 行的第 1 个像素开始叠加) 为图像顶点。 配置值为实际值减 1。																													

VO_OVL2_START

VO_OVL2_START 为图形层 2 起始坐标属性寄存器。

由于 TV 显示是隔行的，所以有：

- 叠加层配置为帧模式，图形层 2 实际开始行坐标为 $ovl2_start_line + 1$ 。
- 叠加层配置为场模式，图形层 2 实际开始行坐标为 $(ovl2_start_line + 1) \times 2$ 。

 说明

- 图形层 2 的起始行距与图形层 1 的高度之和不能大于时序寄存器中配置的图像高度。
- 图形层 2 的起始像素与图形层 1 的宽度之和不能大于时序寄存器中配置的图像宽度，即不能超出显示设备的最大分辨率。

	Offset Address		Register Name		Total Reset Value																											
	0x58		VO_OVL2_START		0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		ovl2_start_line		ovl2_start_pixel																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													



[31:24]	-	reserved	保留。
[23:12]	RW	ovl2_start_line	<p>图形层 2 相对于实际传输图像开始叠加时的行坐标，从 0 开始。</p> <p>配置值为实际值减 1。</p>
[11:0]	RW	ovl2_start_pixel	<p>图形层 2 相对于实际传输图像开始叠加时的像素坐标，从 0 开始。</p> <p>例如 (0, 0) (第 1 行的第 1 个像素开始叠加) 为图像顶点。</p> <p>配置值为实际值减 1。</p>

VO_HC_START

VO_HC_START 为硬件鼠标起始坐标属性寄存器。

由于 TV 显示是隔行的，所以有：

- 硬件鼠标层配置为帧模式，硬件鼠标实际开始行坐标为 `hc_start_line`+1。
 - 硬件鼠标层配置为场模式，硬件鼠标实际开始行坐标为 $(hc_start_line+1) \times 2$ 。



说明

- 鼠标的起始行距与鼠标的高度之和不能大于时序寄存器中配置的图像高度。
 - 鼠标的起始像素与鼠标的宽度之和不能大于时序寄存器中配置的图像宽度，即不能超出显示设备的最大分辨率。

[11:0]	RW	hc_start_pixel	硬件鼠标相对于实际传输图像, 开始叠加时的像素偏移量 (以像素为单位)。 配置值为实际值减 1。
--------	----	----------------	--

VO_OVL1_ALPHA

VO_OVL1_ALPHA 为图形层 1 alpha 值寄存器。

	Offset Address			Register Name			Total Reset Value																									
	0x60			VO_OVL1_ALPHA			0x0000_0000																									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved												ovl1_alpha0				ovl1_alpha1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15:8]	RW	ovl1_alpha0	图形层 1 的 alpha 值 0。 在 YCbCr 4:2:2+ α 中的 α 位来选择该寄存器值作为 alpha 叠加的 alpha 值。																													
[7:0]	RW	ovl1_alpha1	图形层 1 的 alpha 值 1。 在 YCbCr 4:2:2+ α 中的 α 位来选择该寄存器值作为 alpha 叠加的 alpha 值。																													

VO_OVL2_ALPHA

VO_OVL2_ALPHA 为图形层 2 alpha 值寄存器。

	Offset Address			Register Name			Total Reset Value																									
	0x64			VO_OVL2_ALPHA			0x0000_0000																									
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved												ovl2_alpha0				ovl2_alpha1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													



[15:8]	RW	ovl2_alpha0	图形层 2 的 alpha 值 0。 在 YCbCr 4:2:2+ α 中的 α 位来选择该寄存器值作为 alpha 叠加的 alpha 值。
[7:0]	RW	ovl2_alpha1	图形层 2 的 alpha 值 1。 在 YCbCr 4:2:2+ α 中的 α 位来选择该寄存器值作为 alpha 叠加的 alpha 值。

VO_MLADDR

VO_MLADDR 为视频图像亮度分量地址属性寄存器。

	Offset Address																															Register Name	Total Reset Value
	0x68																														VO_MLADDR	0x0000_0000	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	vo_mladdr																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																														
[31:0]	RW	vo_mladdr	视频层亮度存储 SDRAM 首地址。																														

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置低 3 位为 0。

如果配置低位不为 0，则设计按照为低位 0 计算。

VO_MCADDR

VO_MCADDR 为视频图像色度分量地址属性寄存器。

	Offset Address																															Register Name	Total Reset Value
	0x6C																														VO_MCADDR	0x0000_0000	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	vo_mcaddr																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																														
[31:0]	RW	vo_mcaddr	视频层色度存储 SDRAM 首地址。																														

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置低 3 位为 0。

如果配置低位不为 0，则设计按照为低位 0 计算。

VO_OVL1ADDR

VO_OVL1ADDR 为图形层 1 地址属性寄存器。

Offset Address		Register Name	Total Reset Value
0x74		VO_OVL1ADDR	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	vo_ovl1addr		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	vo_ovl1addr	图形层 1 存储 SDRAM 首地址。

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置低 3 位为 0。

如果配置低位不为 0，则设计按照为低位 0 计算。

VO OVL2ADDR

VO_OVL2ADDR 为图形层 2 地址属性寄存器。

Offset Address		Register Name	Total Reset Value
0x78		VO_OVL2ADDR	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	vo_ovl2addr		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RW	vo_ovl2addr	图形层 2 存储 SDRAM 首地址。

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置低 3 位为 0。

如果配置低位不为 0，则设计按照为低位 0 计算。

VO HCADDR

VO HCADDR 为硬件鼠标地址寄存器。



Offset Address																																
0x7C																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	vo_hcaddr																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:0]	RW	vo_hcaddr	硬件鼠标存储 SDRAM 首地址。																													

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置低 3 位为 0。

如果配置低位不为 0，则设计按照为低位 0 计算。

VO_MOFFSET

VO_MOFFSET 为视频图像行间距属性寄存器。

Offset Address																																
0x80																																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	mluma_offset																mchroma_offset															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	RW	mluma_offset	视频图像亮度信号存储在 SDRAM 的行首与行首之间的地址距离。																													
[15:0]	RW	mchroma_offset	视频图像色度信号存储在 SDRAM 的行首与行首之间的地址距离。																													

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置低 3 位为 0。

如果配置低位不为 0，则设计按照为低位 0 计算。

当视频图像配置为帧模式，视频层的 offset 只有低 15bits 有效。

VO_OVLOFFSET

VO_OVLOFFSET 为图形层行间距属性寄存器。

Offset Address			Register Name	Total Reset Value																												
0x84			VO_OVLOFFSET	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ovl2_offset															ovl1_offset																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	RW	ovl2_offset	图形层 2 存储在 SDRAM 的行首与行首之间的地址距离, 该值为 WORD 地址。																													
[15:0]	RW	ovl1_offset	图形层 1 存储在 SDRAM 的行首与行首之间的地址距离, 该值为 WORD 地址。																													

注：在启动总线 32bits 模式的时候配置最低 2 位为 0；启动总线 64bits 模式的时候配置最低 3 位为 0。

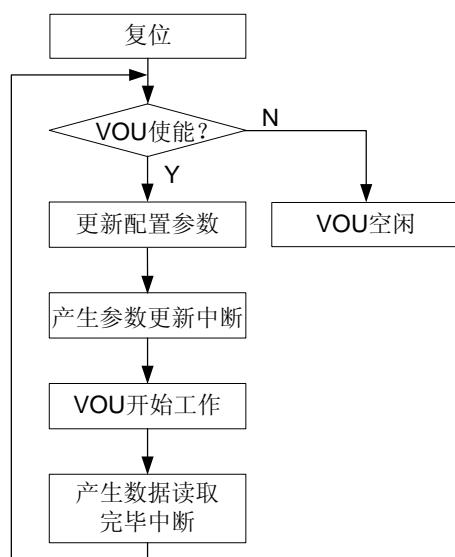
如果配置低位不为 0，则设计按照为低位 0 计算。

当视频图像配置为帧模式，视频层的 offset 只有低 15bits 有效。

6.2.8 使用指南

硬件工作流程

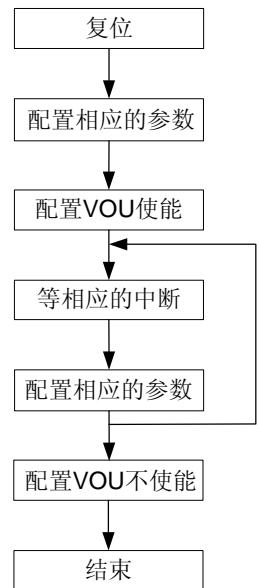
图6-31 VOU 的硬件工作流程





软件工作流程

图6-32 VOU 的软件工作流程





7 音频接口

关于本章

本章描述内容如下表所示。

标题	内容
7.1 概述	介绍音频接口的功能。
7.2 特点	介绍音频接口的特点。
7.3 信号描述	介绍音频接口的信号。
7.4 功能描述	介绍音频接口的典型应用和功能原理。
7.5 工作方式	介绍音频接口的管脚复用配置、时钟门控、时钟配置、软复位、音频播放和录制。
7.6 寄存器概览	概况介绍音频接口的寄存器。
7.7 寄存器描述	详细描述音频接口的寄存器。



7.1 概述

音频输入输出接口 SIO (Sonic Input/Output)，用于和片外 Audio CODEC 芯片连接，完成声音的播放及录制。Hi3511/Hi3512 提供 2 组音频接口 SIO0 和 SIO1。SIO0 配合 TW2815 可以完成 8 路音频的输入和输出，SIO1 配合 TW2815 只支持 8 路的输入。

7.2 特点

SIO 接口支持 PCM (Pulse Code Modulation)、I²S 两种模式。

- PCM 接口
主要用于语音通道，比如 VOIP 电话。
- I²S 接口
主要用于配合 AUDIO CODEC 完成音频播放和录制。

SIO 接口对外提供主时钟 ACKOUT，其频率可以设置，当 AUDIO CODEC 为从模式时，ACKOUT 可以作为 AUDIO CODEC 的工作时钟。SIO 接口还支持 DMA 操作。

7.2.1 PCM 接口特点

PCM 接口有如下特点：

- 支持 16 位线性 PCM 编码。
- 支持由芯片内部产生位时钟以及帧同步信号，也可支持外接时钟及帧同步信号。
- PCM 接口帧同步信号只支持短脉冲同步信号（同步信号的持续时间为 1 个 SIO 时钟周期）。
- 支持发送和接收通道单独使能。
- 接收通道和发送通道具有独立的 FIFO，FIFO 深度为 8×32 bit，FIFO 水线可调。
- 帧数据在时钟上升沿或下降沿发送/接收。
- 支持同步信号和数据对齐的方式。

7.2.2 I²S 接口特点

I²S 接口有如下特点：

- 支持主从模式。
- 支持 8/16/20/24/32 bit 数据位宽。
- 支持 8KHz~48KHz 采样率。
- I²S 工作模式只支持 MSB first 模式。
- I²S 接收通道和发送通道具有独立的 FIFO，并且每个通道的左声道和右声道均有独立的 FIFO，其 FIFO 深度为 8×32 bit，FIFO 水线可调。
- I²S 支持 FIFO Disable 功能，在 FIFO Disable 状态下，接收和发送数据均不经过 FIFO，直接在一个内置 buffer 中缓存。



- I²S 支持发送和接收通道单独使能。
- 在 I²S 接口 16 位数据宽度的传输模式下，支持左右声道接收数据合并成一个 32bit 数据在接收 FIFO 中存储，支持左右声道发送数据合并成一个 32bit 数据写入发送 FIFO，提高 FIFO 的缓冲能力。
- 在 I²S 工作模式下，在传输时支持由 SIO 模块自身产生位时钟以及帧同步信号，也可支持外接位时钟及同步信号；在接收模式下只支持外接位时钟及同步信号。
- I²S 支持高位符号扩展。

7.3 信号描述

SIO 模块提供 2 组 SIO 接口：SIO0 和 SIO1。这 2 组接口的信号描述如表 7-1 和表 7-2 所示。

表7-1 SIO0 接口信号描述

信号名称	方向	描述	对应管脚
SIO0_DI	I	数据输入。	SIO0DI
SIO0_DO	O	数据输出。	SIO0DO
SIO0_XFS	I/O	I ² S 发送左右声道选择信号（与 DAC 接口），或 PCM 帧同步信号。 与 GPIO 接口复用。（复用时的配置信息请参见“ 7.5.1 管脚复用配置 ”。）	SIO0XFS
SIO0_XCK	I/O	I ² S/PCM 发送位流时钟。 与 GPIO 接口复用。（复用时的配置信息请参见“ 7.5.1 管脚复用配置 ”。）	SIO0XCK
SIO0_RFS	I/O	I ² S 接收左右声道选择信号（与 ADC 接口），或 PCM 帧同步信号。	SIO0RFS
SIO0_RCK	I/O	I ² S/PCM 接收位流时钟。	SIO0RCK
ACKOUT	O	I ² S 或 PCM 接口主时钟，可作为 CODEC 的工作时钟。 与 GPIO 接口复用。（复用时的配置信息请参见“ 7.5.1 管脚复用配置 ”。）	ACKOUT

有一些 AUDIO CODEC 的发送左右声道选择信号和接收左右声道选择信号是同一个信号，与这些 CODEC 对接时，只需连接 SIO0_RFS (SIO1_RFS)，SIO0 (SIO1) 内部 XFS 信号通过配置 SC_PERCTRL1 可以由 RFS 获得，这样 SIO0_XFS (SIO1_XFS) 可被用作 GPIO。



表7-2 SIO1 接口信号描述

信号名称	方向	描述	对应管脚
SIO1_DI	I	数据输入。 与 GPIO 复用。（复用时的配置信息请参见“ 7.5.1 管脚复用配置 ”）	SIO1DI
SIO1_RFS	I/O	I ² S 接收左右声道选择信号（与 ADC 接口），或 PCM 帧同步信号。 与 GPIO 复用。（复用时的配置信息请参见“ 7.5.1 管脚复用配置 ”）	SIO1RFS
SIO1_RCK	I/O	I ² S/PCM 接收位流时钟。 与 GPIO 复用。（复用时的配置信息请参见“ 7.5.1 管脚复用配置 ”）	SIO1RCK

7.4 功能描述

7.4.1 典型应用

SIO0 和 SIO1 的功能和特性基本相同，SIO1 只能作为音频输入接口，SIO0 可作为音频输入输出接口。下面以 SIO0 为例，对 I²S 接口的典型连接进行说明。

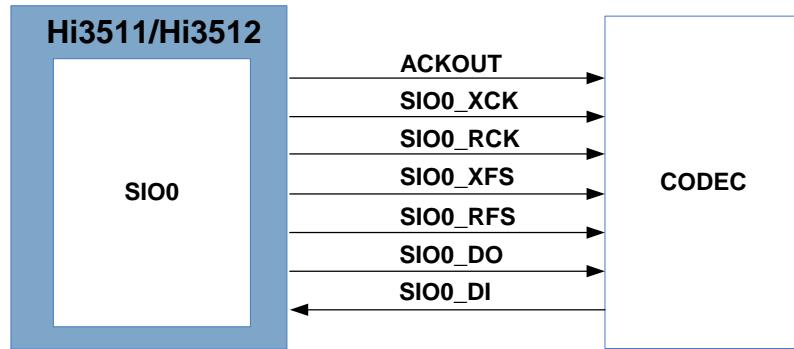


注意

- 不建议使用主模式。
 - 在主模式下，AUDIO CODEC 的工作时钟要使用芯片提供的主时钟（ACKOUT）信号，而不能使用外接的晶振，否则，可能导致声音失真。
-
- 主模式下，如果 CODEC 能同时提供接收和发送位流时钟、帧同步 4 根管脚信号，则连接方式如[图 7-1](#) 所示。

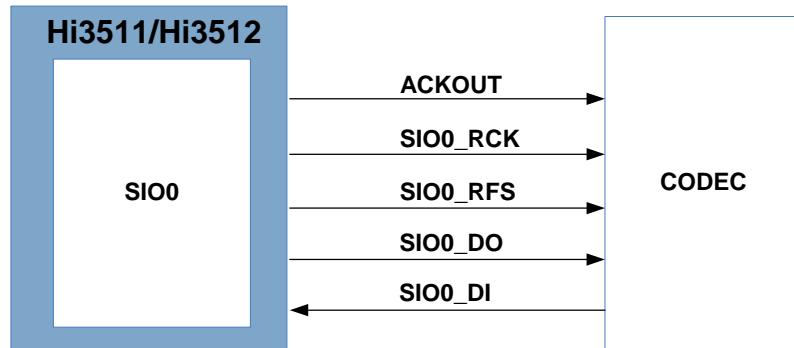


图7-1 I²S 接口主模式连接示意图一



2. 主模式下, 如果 CODEC 能同时提供位流时钟、帧同步 2 根管脚信号 (连接 SIO0_RFS 和 SIO0_RCK), 则连接方式如图 7-2 所示。

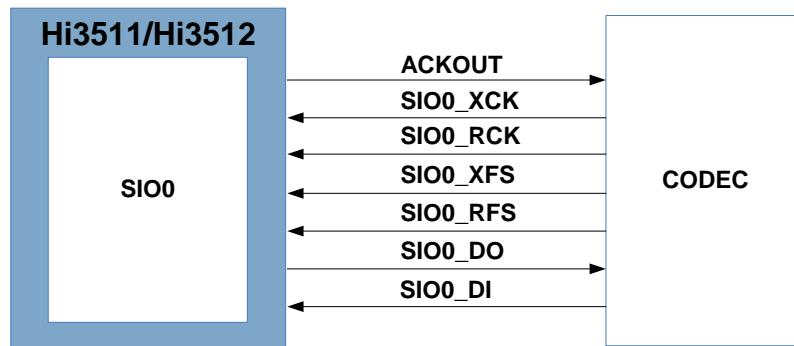
图7-2 I²S 接口主模式连接示意图二



在主模式下, 位时钟和帧同步信号由 SIO 送给 AUDIO CODEC; 在从模式下, 位流时钟和帧同步信号由 AUDIO CODEC 送给 SIO。

3. 从模式下, 如果 CODEC 能同时提供接收和发送位流时钟、帧同步 4 根管脚信号, 则连接方式如图 7-3 所示。

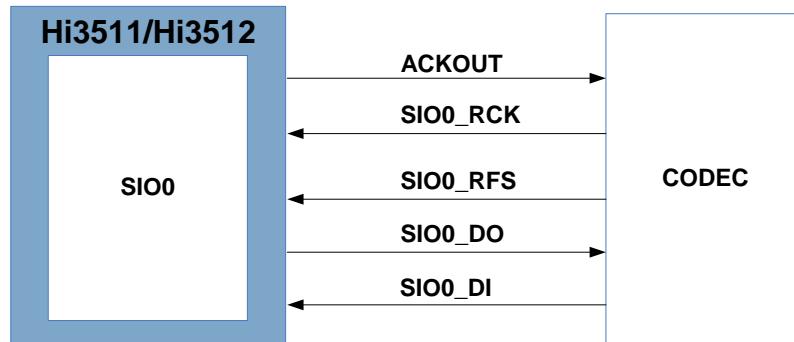
图7-3 I²S 接口从模式连接示意图一





4. 从模式下, 如果 CODEC 能同时提供位流时钟、帧同步 2 根管脚信号 (连接 SIO0_RFS 和 SIO0_RCK), 则连接方式如图 7-4 所示。

图7-4 I²S 接口从模式连接示意图二

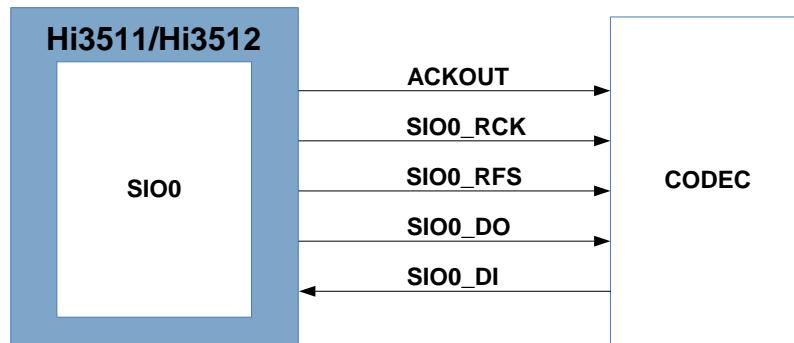


在从模式下, AUDIO CODEC 的主工作时钟可以使用芯片提供的主时钟 (ACKOUT) 信号, 也可以使用外接的晶振。

PCM 连接方式同 I²S。

由 SIO 提供时钟和同步信号时, 如果 CODEC 能同时提供位流时钟、帧同步 2 根管脚信号, PCM 接口的典型连接如图 7-5 所示。AUDIO CODEC 的工作时钟要用芯片提供的 ACKOUT 信号, 而不能用外接的晶振, 否则, 可能导致声音失真。

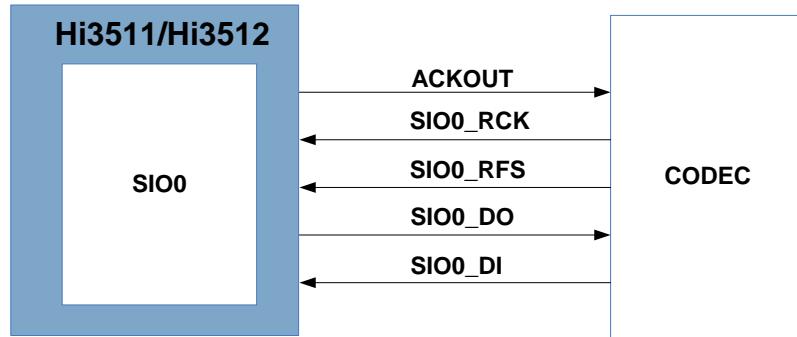
图7-5 PCM 接口连接示意图 (由 SIO 提供时钟和同步信号)



由 AUDIO CODEC 提供时钟和同步信号时, AUDIO CODEC 的主工作时钟可以使用芯片提供的主时钟 (ACKOUT) 信号, 也可以使用外接的晶振。PCM 接口的连接如图 7-6 所示。



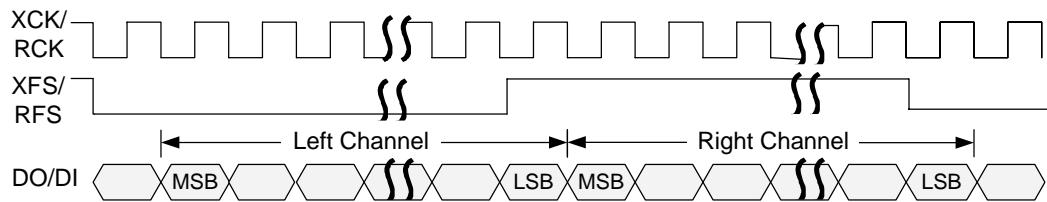
图7-6 PCM 接口连接示意图（由 CODEC 提供时钟和同步信号）



7.4.2 功能原理

I^2S 接口传输数据分为左右两个声道，根据 XFS (RFS) 信号的高低电平区分。按照协议，用 XCK/RCK 时钟的上升沿进行数据采样，MSB 在 XFS/RFS 变化的下一个时钟周期有效。总是先传送 MSB (Most Significant Bit)，后传送 LSB (Least Significant Bit)。 I^2S 接口的时序如图 7-7 所示。

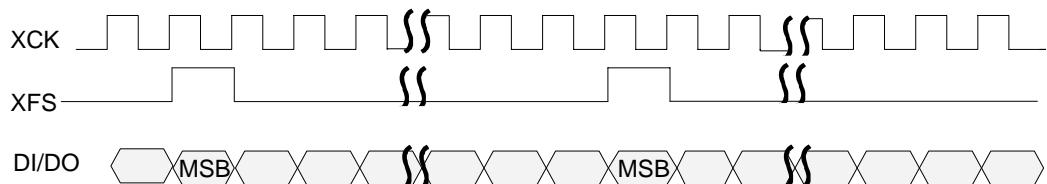
图7-7 I^2S 接口时序



PCM 接口传输的数据是单声道数据，XFS 标识数据的起始位置，先发送（接收）最高有效位 MSB，使用时钟的下降沿采样数据。标准模式时序中，MSB 数据在 XFS 高电平脉冲之后一个周期有效；自定义模式时序中，MSB 的位置是与 XFS 的高电平脉冲对齐的。SIO0 (SIO1) 支持自定义模式。

自定义模式：PCM 接口自定义模式下的时序如图 7-8 所示。

图7-8 PCM 接口自定义模式时序





7.5 工作方式

7.5.1 管脚复用配置

通过对系统寄存器 SC_PERCTRL1 进行配置实现 SIO 的管脚复用，管脚复用可参见表 7-3。

表7-3 管脚复用关系

信号名称	复用信号
SIO0XFS	GPIO6_2
SIO0XCK	GPIO6_3
SIO1DI	GPIO0_4
SIO1RFS	GPIO0_2
SIO1RCK	GPIO0_3

7.5.2 时钟门控

当不进行音频录制或者音频播放时，设置 SIO0 (SIO1) 的 I2S/PCM_CT_SET[rx_enable] 和 I2S/PCM_CT_SET[tx_enable] 为 0 后，可以通过配置 SC_PERDIS 关断 SIO 的时钟：

- 向 SC_PERDIS[sio0clkdis] 写 1，关断 SIO0 的系统时钟。
- 向 SC_PERDIS[sio1clkdis] 写 1，关断 SIO1 的系统时钟。

7.5.3 时钟配置

SIO0 和 SIO1 相互独立，但时钟配置方式相同。SIO 模块的位流时钟和同步信号的产生有 3 种方式：

- 外部输入产生，具体方法如下：
 - 配置 SC_PERCTRL4[sio0clk_sel] 或 SC_PERCTRL4[sio1clk_sel] 为 0，选择 SIO 时钟产生源为外部输入。
 - 当 SIO0 或 SIO1 工作在从模式下，需配置 SIO 内部 I2S 控制寄存器 I2S_CT_SET[3:2]=0b11 或 PCM 控制寄存器的 PCM_CT_SET[3:2]=0b11，并且根据 CODEC 是否发送接收共用决定是否需要配置系统控制器中的 SC_PERCTRL1[pinmuxctrl30] 和 SC_PERCTRL1[pinmuxctrl31] 均为 1，通过如上配置的模式即为从模式。
- 内部 CRG 分频产生，具体方法如下：

配置 SC_PERCTRL4[sio0clk_sel] 或 SC_PERCTRL4[sio1clk_sel] 为 1，选择 SIO 时钟产生源，内部 CRG 分频产生位流时钟一方面给 SIO 输送时钟，另一方面给外部 CODEC 输入时钟。可配置 SIO 分频控制寄存器 SC_PERCTRL3，控制 CRG 的分频系数，通过如上的配置即为主模式。



说明

CRG 不能产生 PCM 帧同步时钟，所以当启用 SIO 的 PCM 模式时，推荐 CODEC 工作在主模式，SIO 工作在从模式。

- SIO 内部产生，具体方法如下：

- 配置 SC_PERCTRL4[sio0clk_sel]或 SC_PERCTRL4[sio1clk_sel]为 0 来选择 SIO 时钟产生源为内部产生。
- SIO0 或 SIO1 工作在主模式下，需配置 SIO 内部 I2S 控制寄存器 I2S_CT_SET[3:2]=0b00 或 PCM 控制寄存器的 PCM_CT_SET[3:2]=0b00，内部产生同步信号和位流时钟并输出。通过如上配置的模式即为主模式（不推荐使用此模式）。

7.5.4 软复位

通过配置系统控制器 SC_PERCTRL0[sio0_srst]和 SC_PERCTRL0[sio1_srst]为 1，可实现对 SIO0 和 SIO1 的单独软复位。复位后各配置寄存器的值均复位为默认值，因此复位后需要重新对这些寄存器进行初始化配置。

7.5.5 音频播放和录制

7.5.5.1 中断或查询方式

初始化

初始化步骤如下：

- 步骤 1 设置 I2S_CT_SET[rx_enable]/PCM_CT_SET[rx_enable]和 I2S_CT_SET[tx_enable]/PCM_CT_SET[tx_enable]为 0，使 SIO 处于禁使能状态。
- 步骤 2 设置 SIO_MODE[sio_mode]，选择 I²S 或 PCM 模式；如果是 PCM 模式，设置 SIO_MODE[pcm_mode]，选择时序类型。
- 步骤 3 如果 SIO 为主模式，对时钟频率进行配置（如果是从模式，需配置 I2S_CT_SET/PCM_CT_SET[tx_clk_sel]和 I2S_CT_SET/PCM_CT_SET[tx_ws_sel]为 1）。
- 步骤 4 设置 I2S_LENGTH_CF 寄存器和 SIO_SIGNED_EXT 寄存器，设定正确的位宽。
- 步骤 5 设置 I2S_CT_SET[rx_fifo_threshold]或者 PCM_CT_SET[rx_fifo_threshold]和 I2S_CT_SET[tx_fifo_threshold]或者 PCM_CT_SET[tx_fifo_threshold]，即：设置接收 FIFO 和发送 FIFO 的水线。
- 步骤 6 如果是 I²S 模式，根据实际读写 FIFO 数据的方式，配置 SIO_I2S_POS_MERGE_EN 和 SIO_I2S_START_POS。
- 步骤 7 对外接的 AUDIO CODEC 进行设置。

----结束

音频播放

音频播放步骤如下：



- 步骤 1 清空发送 FIFO 的残留数据。方法是：把 [I2S_CT_SET\[tx_fifo_disable\]](#) 或者 [PCM_CT_SET\[tx_fifo_disable\]](#) 设置为 0，然后再设置为 1。
- 步骤 2 查询方式下，通过读取 [SIO_TX_STA](#) 判断 TX_FIFO 状态；中断方式下，则根据中断状态 [SIO_INTSTATUS\[tx_intr\]](#) 判断。当检测到发送 FIFO 中数据深度低于水线时，向发送 FIFO 写入数据，如此反复。如果数据发送全部完成，进入步骤 3。在传送完成之前，要保证 TX_FIFO 中的数据没有溢出，否则会造成声音不连续。
- 步骤 3 把 [I2S_CT_SET\[tx_enable\]](#) 或者 [PCM_CT_SET\[tx_enable\]](#) 设置为 0。
- 结束

音频录制

音频录制步骤如下：

- 步骤 1 清空接收 FIFO 的残留数据，方法是：如果接收 FIFO 不为空，CPU 就读 FIFO 数据，直到接收 FIFO 为空。
- 步骤 2 向 [I2S_CT_SET\[rx_enable\]](#) 或者 [PCM_CT_SET\[rx_enable\]](#) 写 1，启动数据接收。
- 步骤 3 查询方式下，通过读取 [SIO_RX_STA](#) 检测 RX_FIFO 状态；中断方式下，则根据相应中断状态位检测。当检测到接收 FIFO 中数据深度高于水线时，从接收 FIFO 读出数据，如此反复。如果数据接收完成，进入步骤 4。在接收完成之前，要保证 RX_FIFO 中的数据没有溢出，否则会造数据丢失。
- 步骤 4 把 [I2S_CT_SET\[rx_enable\]](#) 或者 [PCM_CT_SET\[rx_enable\]](#) 设置为 0，并把接收 FIFO 中剩余的数据全部读出。

----结束

7.5.5.2 DMA 方式

初始化

请参见“[7.5.5.1 中断或查询方式](#)”中的“初始化”。

音频播放

步骤如下：

- 步骤 1 配置 DMA 数据通道，包括数据传输源和目的地址（地址配为 FIFO 寄存器地址或者左右声道合并寄存器地址）、数据传输个数、传输类型等参数。
- 步骤 2 清除发送 FIFO 的残留数据。方法是：把 [I2S_CT_SET\[rst_n\]](#) 或 [PCM_CT_SET\[rst_n\]](#) 设置为 0，然后再设置为 1。
- 步骤 3 向发送 FIFO 写入初始数据，深度超过 FIFO 水线。（可写入全 0 的数据，代表静音。目的是：当启动播放时，因为 DMA 还没有向 FIFO 中写入数据，此时 SIO 会报发送 FIFO 下溢。如果先写入初始数据，就可以防止播放刚启动时误报 FIFO 溢出）。
- 步骤 4 设置 [I2S_CT_SET\[tx_enable\]](#) 或者 [PCM_CT_SET\[tx_enable\]](#) 为 1，启动播放。



步骤 5 通过 FIFO 水线触发 DMA 中断，判断数据是否发送完成，如果完成，则设置 [I2S_CT_SET\[tx_enable\]](#) 或者 [PCM_CT_SET\[tx_enable\]](#) 为 0。

----结束

音频录制

步骤如下：

步骤 1 配置 DMA 数据通道，包括数据传输源地址、目的地址、数据传输个数、传输类型等参数。

步骤 2 清空接收 FIFO 的残留数据，方法是：如果接收 FIFO 不为空，CPU 就读 FIFO 数据，直到接收 FIFO 为空。

步骤 3 设置 [I2S_CT_SET\[rx_enable\]](#) 或者 [PCM_CT_SET\[rx_enable\]](#) 为 1，启动数据接收。

步骤 4 如果停止音频录制，设置 [I2S_CT_SET\[rx_enable\]](#) 或者 [PCM_CT_SET\[rx_enable\]](#) 为 0。

----结束

7.6 寄存器概览

2 组 SIO 的寄存器基址如下：

- SIO0: 0x8008_0000。
- SIO1: 0x900A_0000。

表7-4 SIO 寄存器概览

偏移地址	名称	描述	页码
0x040	SIO_MODE	SIO 模式寄存器	7-12
0x044	SIO_INTSTATUS	SIO 中断状态寄存器	7-13
0x048	SIO_INTCLR	SIO 中断清除寄存器	7-14
0x04C	SIO_I2S_LEFT_XD	I ² S 左通道数据发送寄存器	7-14
0x050	SIO_I2S_RIGHT_XD	I ² S 右通道/PCM 数据发送寄存器	7-15
0x054	SIO_I2S_LEFT_RD	I ² S 左通道数据接收寄存器	7-15
0x058	SIO_I2S_RIGHT_RD	I ² S 右通道/PCM 数据接收寄存器	7-16
0x05C	I2S_CT_SET	I ² S 控制设置寄存器	7-16
0x060	I2S_CT_CLR	I ² S 控制清除寄存器	7-17
0x064	I2S_ICD	I2S 时钟分频寄存器	7-18



偏移地址	名称	描述	页码
0x068	SIO_RX_STA	SIO 接收状态寄存器	7-19
0x06C	SIO_TX_STA	SIO 发送状态寄存器	7-19
0x070	PCM_CT_SET	PCM 控制设置寄存器	7-20
0x074	PCM_CT_CLR	PCM 控制清除寄存器	7-21
0x078	I2S_LENGTH_CF	I ² S/PCM 传输数据长度配置寄存器	7-22
0x07C	SIO_I2S_START_POS	I ² S 左右通道起始位置控制寄存器	7-23
0x080	I2S_POS_FLAG	I ² S 左右声道操作当前位置状态寄存器	7-24
0x084	SIO_SIGNED_EXT	I2S 高位数据符号扩展使能寄存器	7-25
0x088	SIO_I2S_POS_MERGE_EN	I ² S 左右声道合并使能寄存器	7-25
0x090~0x09C	RESERVED	保留	-
0x0A0	SIO_I2S_DUAL_RX_C_HN	I ² S 左右通道合并后数据接收寄存器	7-26
0x0C0	SIO_I2S_DUAL_TX_C_HN	I ² S 左右通道合并后数据发送寄存器	7-26

7.7 寄存器描述

SIO_MODE

SIO_MODE 为模式寄存器。



SIO_INTSTATUS

SIO_INTSTATUS 为 SIO 的中断状态指示寄存器。

对于接收中断，当接收 FIFO 的数据深度大于 FIFO 阈值时，会一直把高电平锁存到中断状态寄存器中，一直产生中断（即使 CPU 清一次中断，但中断状态寄存器会在下一个时钟周期再次置位）。CPU 的处理步骤为：

步骤1 读中断状态寄存器 `SIO_INTSTATUS`。

步骤 2 根据中断源进行相应处理。

步骤 3 向 SIO_INTCLR 的相应位写 1，清除中断。

----结束

发送中断的产生方式与接收中断产生方式相同，因此对于发送中断，建议也采用相同的方式处理。

[1]	RO	tx_intr	发送 FIFO 低于阈值中断状态。 0: 未产生中断。 1: 已产生中断。
[0]	RO	rx_intr	接收 FIFO 高于阈值中断状态。 0: 未产生中断。 1: 已产生中断。

SIO_INTCLR

SIO_INTCLR 为中断清除寄存器，可以按位清除。

	Offset Address	Register Name	Total Reset Value
	0x048	SIO_INTCLR	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name		reserved	 
Reset	0 0		
Bits	Access	Name	Description
[31:2]	-	reserved	保留。
[1]	WO	tx_intr	发送 FIFO 低于阈值中断清除。 0: 不清除该中断。 1: 清除该中断。
[0]	WO	rx_intr	接收 FIFO 高于阈值中断清除。 0: 不清除该中断。 1: 清除该中断。

SIO_I2S_LEFT_XD

SIO_I2S_LEFT_XD 为 I²S 模式下的左声道数据发送寄存器。

向寄存器写数据时，有效数据需放在寄存器的低比特区域。例如，数据宽度为 8bit 时，bit[7:0]为有效数据，bit[31:8]为无效数据；数据宽度为 16bit 时，bit[15:0]为有效数据，bit[31:16]为无效数据。超出有效数据宽度的比特位由 SIO 模块自动置 0。

	offset Address	Register Name	Total Reset Value
	0x04C	SIO_I2S_LEFT_XD	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		



Name	tx_left_data																			
Reset	0 0																			
Bits	Access	Name	Description																	
[31:0]	WO	tx_left_data	左声道发送数据。																	

SIO_I2S_RIGHT_XD

SIO_I2S_RIGHT_XD 为 I²S 模式下的右声道数据发送寄存器。PCM 数据发送寄存器与 I²S 右声道数据发送寄存器是复用的。

向寄存器写数据时，有效数据需放在寄存器的低比特区域。

	Offset Address																				Register Name				Total Reset Value							
	0x050																				SIO_I2S_RIGHT_XD				0x0000_0000							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	tx_right_data																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	WO	tx_right_data	右声道发送数据。																													

SIO_I2S_LEFT_RD

SIO_I2S_LEFT_RD 为 I²S 左声道数据接收寄存器。

SIO 模块把接收到的有效数据放在寄存器的低比特区域。例如，数据宽度为 8bit 时，bit[7:0]为有效数据，bit[31:8]为无效数据；数据宽度为 16bit 时，bit[15:0]为有效数据，bit[31:16]为无效数据。超出有效数据宽度的比特位由 SIO 模块自动置 0。

	Offset Address																				Register Name				Total Reset Value							
	0x054																				SIO_I2S_LEFT_RD				0x0000_0000							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rx_left_data																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	rx_left_data	I2S 左声道接收数据。																													

注：I²S 模式下，在接收不使能时，右声道数据可能还没有写入 FIFO，从而导致左声道 FIFO 数据数目比右声道 FIFO 数据数目多一个。因此在 CPU 启动下一次接收前，应该把左右声道 FIFO 中数据全部读空。

SIO_I2S_RIGHT_RD

SIO_I2S_RIGHT_RD 为 I²S 右声道数据接收寄存器。PCM 数据接收寄存器与 I²S 右声道数据接收寄存器复用。

SIO 模块把接收到的有效数据放在寄存器的低比特区域。

Offset Address			Register Name			Total Reset Value																										
0x058			SIO_I2S_RIGHT_RD			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rx_right_data																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:0]	RO	rx_right_data	I ² S 右声道/PCM 接收数据。																													

注：I²S 模式下，在接收不使能时，右声道数据可能还没有写入 FIFO，从而导致左声道 FIFO 数据数目比右声道 FIFO 数据数目多一个。因此在 CPU 启动下一次接收前，应该把左右声道 FIFO 中数据全部读空。

I2S_CT_SET

I2S_CT_SET 为 I²S 控制设置寄存器。

为了能够方便的对 I²S 控制寄存器进行位操作，在 SIO 中，为 I²S 控制寄存器设置了 2 个地址：0x05C (I2S_CT_SET) 和 0x060 (I2S_CT_CLR)，其中：0x05C 为设置寄存器地址，当向 0x05C 寄存器中相应位写入 1 时，对应位被设为 1，写 0 无效；该寄存器属性为读写。

Offset Address			Register Name			Total Reset Value																										
0x05C			SIO_CT_SET			0x0000_800C																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																											reserved				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15]	RW	rst_n	I ² S/PCM 通道复位，低电平有效。 它对 I ² S/PCM 接收和发送模块（包括 FIFO）进行复位，即对 FIFO 清 0。但不会复位 CPU 接口寄存器模块。																													
[14]	-	reserved	保留。																													



[13]	RW	rx_enable	接收通道使能。 0: 禁止。 1: 使能。
[12]	RW	tx_enable	发送通道使能。 0: 禁止。 1: 使能。
[11]	RW	rx_fifo_disable	接收 FIFO 禁止。 0: 使能。 1: 禁止。
[10]	RW	tx_fifo_disable	发送 FIFO 禁止。 0: 使能。 1: 禁止。
[9:7]	RW	rx_fifo_threshold	接收 FIFO 阈值。 当 $rx_right_depth \geq (rx_fifo_threshold + 1)$ 时, 报接收中断和 DMA 请求。
[6:4]	RW	tx_fifo_threshold	发送 FIFO 阈值。 当 $tx_right_depth < (tx_fifo_threshold + 1)$ 时, 报发送中断和 DMA 请求。
[3]	RW	tx_clk_sel	发送时钟选择。 0: 选择内部产生。 1: 选择 CRG 或外部输入。
[2]	RW	tx_ws_sel	发送同步信号选择。 0: 选择内部产生。 1: 选择 CRG 或外部输入。
[1:0]	-	reserved	保留。

I2S_CT_CLR

I2S_CT_CLR 为清除寄存器。

为了能够方便的对 I2S 控制寄存器进行位操作, 在 SIO 中, 为 I²S 控制寄存器设置了 2 个地址: 0x05C (I2S_CT_SET) 和 0x060 (I2S_CT_CLR), 其中: 0x060 为清除寄存器地址, 当向 0x060 寄存器中相应位写入 1 时, 对应位被清除, 写 0 无效。该寄存器属性为只写。

Offset Address	Register Name	Total Reset Value
0x060	SIO_CT_CLR	0x0000_800C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															rst_n	reserved	rx_enable	tx_fifo_disable	tx_fifo_enable	tx_fifo_disable	tx_fifo_threshold	tx_fifo_threshold	tx_clk_sel	tx_ws_sel	reserved	reserved					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0		
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15]	WO	rst_n	I ² S/PCM 通道复位。 它对 I ² S/PCM 接收和发送模块（包括 FIFO）进行复位，即对 FIFO 清 0。但不会复位 CPU 接口寄存器模块。 0：复位。 1：不复位。																													
[14]	-	reserved	保留。																													
[13]	WO	rx_enable	接收通道使能。																													
[12]	WO	tx_enable	发送通道使能。																													
[11]	WO	rx_fifo_disable	接收 FIFO 禁止。																													
[10]	WO	tx_fifo_disable	发送 FIFO 禁止。																													
[9:7]	WO	rx_fifo_threshold	接收 FIFO 阈值。																													
[6:4]	WO	tx_fifo_threshold	发送 FIFO 阈值。																													
[3]	WO	tx_clk_sel	发送时钟选择。																													
[2]	WO	tx_ws_sel	发送同步信号选择。																													
[1:0]	-	reserved	保留。																													

I2S_ICD

I2S_ICD 为 I²S 时钟分频寄存器。

	Offset	Address	Register Name	Total Reset Value																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved				i2s_icd																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	



Bits	Access	Name	Description
[31:16]	-	reserved	保留。
[15:0]	RW	i2s_icd	设置 SIO 模块的时钟分频系数，其分频值为设置值加 1。

SIO_RX_STA

SIO_RX_STA 为 SIO 接收状态寄存器。

Offset Address	Register Name	Total Reset Value	
0x068	SIO_RX_STA	0x0000_0000	
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved	rx_left_depth rx_right_depth rx_left_fifo_over rx_right_fifo_over	
Reset	0 0		
Bits	Access	Name	Description
[31:10]	-	reserved	保留。
[9:6]	RO	rx_left_depth	左声道接收 FIFO 深度指示。
[5:2]	RO	rx_right_depth	右声道接收 FIFO 深度指示。
[1]	RO	rx_left_fifo_over	左声道接收 FIFO 上溢出指示。 0: 不上溢。 1: 上溢。
[0]	RO	rx_right_fifo_over	右声道接收 FIFO 上溢出指示。 0: 不上溢。 1: 溢出。

SIO_TX_STA

SIO_TX_STA 为 SIO 发送状态寄存器。

Offset Address	Register Name	Total Reset Value
0x06C	SIO_TX_STA	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	

PCM_CT_SET

PCM CT SET 为设置寄存器。

为了能够方便的对 PCM 控制寄存器进行位操作，在 SIO 中，为 PCM 控制寄存器设置了 2 个地址：0x070（PCM_CT_SET）和 0x074（PCM_CT_CLR），其中：0x070 为设置寄存器地址，当向 0x070 寄存器中相应位写入 1 时，对应位被设为 1，写 0 无效；该寄存器属性为读写。



[13]	RW	rx_enable	接收通道使能。 0: 禁止。 1: 使能。
[12]	RW	tx_enable	发送通道使能。 0: 禁止。 1: 使能。
[11]	RW	rx_fifo_disable	接收 FIFO 禁止。 0: 使能。 1: 禁止。
[10]	RW	tx_fifo_disable	发送 FIFO 禁止。 0: 使能。 1: 禁止。
[9:7]	R/W	rx_fifo_threshold	接收 FIFO 阈值。 当 $rx_right_depth \geq (rx_fifo_threshold + 1)$ 时, 报接收中断和 DMA 请求。
[6:4]	RW	tx_fifo_threshold	发送 FIFO 阈值。 当 $tx_right_depth < (tx_fifo_threshold + 1)$ 时, 报发送中断和 DMA 请求。
[3]	RW	tx_clk_sel	发送时钟选择。 0: 选择内部产生。 1: 选择外部输入。
[2]	RW	tx_ws_sel	发送同步信号选择。 0: 选择内部产生。 1: 选择外部输入。
[1:0]	-	reserved	保留。

PCM_CT_CLR

PCM_CT_CLR 为清除寄存器。

为了能够方便的对 PCM 控制寄存器进行位操作, 在 SIO 中, 为 PCM 控制寄存器设置了 2 个地址: 0x070 (PCM_CT_SET) 和 0x074 (PCM_CT_CLR), 其中: 0x074 为清除寄存器地址, 当向 0x074 寄存器中相应位写入 1 时, 对应位被清除, 写 0 无效。该寄存器属性为只写。

Offset Address			Register Name																Total Reset Value													
0x074			PCM_CT_CLR																0x0000_0000													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																reserved	tx_fifo_threshold	rx_fifo_threshold	tx_fifo_disable	tx_fifo_disable	tx_enable	tx_enable	edge_sel	edge_sel	tx_ws_sel	tx_ws_sel	tx_clk_sel	tx_clk_sel	reserved	reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15]	WO	reserved	保留。																													
[14]	WO	edge_sel	发送时钟沿设置。																													
[13]	WO	rx_enable	接收通道使能。																													
[12]	WO	tx_enable	发送通道使能。																													
[11]	WO	rx_fifo_disable	接收 FIFO 禁止。																													
[10]	WO	tx_fifo_disable	发送 FIFO 禁止。																													
[9:7]	WO	rx_fifo_threshold	接收 FIFO 阈值。																													
[6:4]	WO	tx_fifo_threshold	发送 FIFO 阈值。																													
[3]	WO	tx_clk_sel	发送时钟选择。																													
[2]	WO	tx_ws_sel	发送同步信号选择。																													
[1:0]	-	reserved	保留。																													

I2S_LENGTH_CF

I2S_LENGTH_CF 为 I²S/PCM 模式下的数据宽度寄存器。



Offset Address		Register Name		Total Reset Value					
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	I2S_LENGTH_CF		0x0000_0000					
Name	reserved		rx_mode	tx_mode	rx_mode	tx_mode			
Reset	0 0								
Bits		Access	Name	Description					
[31:8]		-	reserved	保留。					
[7:6]	RW	rx_mode	PCM 接收数据长度配置。 00: 16bit。 01: 20bit。 10: 24bit。 11: 32bit。						
			PCM 发送数据长度配置。 00: 16bit。 01: 20bit。 10: 24bit。 11: 32bit。						
[5:4]	RW	tx_mode	I ² S 接收数据长度配置位。 00: 16bit。 01: 20bit。 10: 24bit。 11: 32bit。						
			I ² S 发送数据长度配置位。 00: 16bit。 01: 20bit。 10: 24bit。 11: 32bit。						
[3:2]	RW	rx_mode							
[1:0]	RW	tx_mode							

SIO_I2S_START_POS

SIO_I2S_START_POS 为 I²S 左右声道起始位置配置控制寄存器。

在 I²S 模式下，左右声道数据操作地址合并使能后，控制起始访问是从左声道开始还是从右声道开始。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved		
Reset	0 0		
Bits	Access	Name	Description
[31:2]	-	reserved	保留。
[1]	RW	start_pos_write	写发送 FIFO 时： 0: 从左声道开始访问。 1: 从右声道开始访问。
[0]	RW	start_pos_read	读接收 FIFO 时： 0: 从左声道开始访问。 1: 从右声道开始访问。

I2S_POS_FLAG

I2S_POS_FLAG 为 I²S 左右声道操作当前位置状态寄存器。

在 I²S 模式下，左右声道数据操作地址合并使能后，指示下一次访问寄存器是从左声道开始还是从右声道开始。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved		
Reset	0 0		
Bits	Access	Name	Description
[31:2]	-	reserved	保留。



[1]	RO	start_pos_write	写发送 FIFO 时： 0: 下一次从左声道开始访问。 1: 下一次从右声道开始访问。
[0]	RO	start_pos_read	读接收 FIFO 时： 0: 下一次从左声道开始访问。 1: 下一次从右声道开始访问。

SIO_SIGNED_EXT

SIO_SIGNED_EXT 为高位数据符号扩展使能寄存器。该标志只对接收数据有效，对发送数据无效。PCM 模式和 I²S 模式下接收到的数据都支持符号扩展。

在接收有效数据位宽为 8/16/18/20/24bit 时，如果该标志使能，把接收到的数据转换为 32bit 数据时，把 32bit 数据的高位无效比特设置为接收数据最高有效比特对应的值，然后再写入接收 FIFO。

以 16bit 位宽为例：

```
if (data_rx[15] == 1)
    data_rx[31:16]=0xffff;
else
    data_rx[31:16]=0x0000;
```

	Offset	Address	Register Name	Total Reset Value
		0x084	SIO_SIGNED_EXT	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name			reserved	signed_ext_en
Reset	0 0			
Bits	Access	Name	Description	
[31:1]	-	reserved	保留。	
[0]	RW	signed_ext_en	高位数据符号扩展使能。 0: 不使能。 1: 使能。	

SIO_I2S_POS_MERGE_EN

SIO_I2S_POS_MERGE_EN 为 I²S 模式下对左右声道数据的操作地址合并使能。

在 I²S 模式下, 用 DMA 方式读写 SIO 的 FIFO 数据时, 因为左右声道数据地址不同, 需要 CPU 不断配置 DMA 操作的地址, 导致 CPU 效率低。为了提高 CPU 的效率, 提供左右声道数据的读写的统一地址使能控制。

I²S 模式下对左右声道数据的操作地址合并使能的情况下, 读左右声道数据统一使用 [SIO_I2S_DUAL_RX_CHN](#) 寄存器, 写左右声道数据统一使用 [SIO_I2S_DUAL_TX_CHN](#) 寄存器。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	reserved		
Reset	0 0		
Bits	Access	Name	Description
[31:1]	-	reserved	保留。
[0]	RW	merge_en	在 I ² S 模式下, 对左右声道数据的操作地址合并使能。 0: 不使能。 1: 使能。

SIO_I2S_DUAL_RX_CHN

SIO_I2S_DUAL_RX_CHN 为 I²S 左右声道操作地址合并使能以后, 读取接收数据的寄存器。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
Name	rx_data		
Reset	0 0		
Bits	Access	Name	Description
[31:0]	RO	rx_data	接收到的数据。

SIO_I2S_DUAL_TX_CHN

SIO_I2S_DUAL_TX_CHN 为 I²S 左右声道操作地址合并使能后, 写发送数据的寄存器。





8 MMC/SD/SDIO 控制器

关于本章

本章描述内容如下表所示。

标题	内容
8.1 概述	介绍 MMC/SD/SDIO 控制器的功能。
8.2 特点	介绍 MMC/SD/SDIO 控制器的特点。
8.3 信号描述	介绍 MMC 接口的信号。
8.4 功能描述	介绍 MMC/SD/SDIO 控制器的典型应用、功能原理、指令与响应和数据传输指令。
8.5 工作方式	介绍 MMC/SD/SDIO 控制器的管脚复用配置、时钟门控、软复位、时钟配置、初始化、非数据传输指令、单块或多块数据操作等。
8.6 寄存器概览	概况介绍 MMC/SD/SDIO 控制器的寄存器。
8.7 寄存器描述	详细描述 MMC/SD/SDIO 控制器的寄存器。



8.1 概述

MMC/SD/SDIO 控制器（以下简称 MMC）用来处理对 SD/MMC 存储卡的读写等操作，并可以通过 SDIO（Secure Digital Input/Output）协议实现对扩展外设（如 Blue Tooth、WiFi 等）的支持。MMC 可以控制符合以下协议的设备（并对低版本的协议向下兼容）：

- SD mem-version 2.00
- SDIO-version 1.10
- MMC-version 4.2

8.2 特点

MMC 控制器具有以下特点：

- 支持 DMA 数据传输方式。
- 包含数据发送方向与接收方向 2 个 FIFO，大小为 16×32bit。
- FIFO 阈值可配置，DMA 传输时 burst 大小可配置。
- FIFO 上溢出与下溢出中断告警，防止数据传输错误。
- 命令与数据的 CRC 生成与校验。
- 支持接口时钟频率可编程。
- 支持低功耗模式关断 MMC 控制器时钟。
- 支持低功耗模式关断接口时钟。
- 支持接口数据位宽为 1bit 和 4bit 模式下的数据传输。
- 支持 1byte~65535byte 的块数据读写操作。
- 支持 MMC 卡流数据读写方式。
- 支持接口数据位宽为 1bit 和 4bit 数据传输模式下的 SDIO 中断检测。
- 支持 SDIO suspend 和 resume 操作。
- 支持 SDIO read wait 操作。

8.3 信号描述

MMC 接口信号如表 8-1 所示。

表8-1 MMC 接口信号描述

信号名称	方向	描述	对应管脚
MMC_CCLK	O	输出接口时钟信号，与 GPIO 复用。（复用时的配置信息请参见“ 8.5.1 管脚复用配置 ”）	SDIOCK



信号名称	方向	描述	对应管脚
MMC_CMD	I/O	双向指令信号, 与 GPIO 复用。(复用时的配置信息请参见“ 8.5.1 管脚复用配置 ”)	SDIOCMD
MMC_DAT0	I/O	双向数据信号 0, 与 GPIO 复用。(复用时的配置信息请参见“ 8.5.1 管脚复用配置 ”)	SDIODAT0
MMC_DAT1	I/O	双向数据信号 1, 与 GPIO 复用。(复用时的配置信息请参见“ 8.5.1 管脚复用配置 ”)	SDIODAT1
MMC_DAT2	I/O	双向数据信号 2, 与 GPIO 复用。(复用时的配置信息请参见“ 8.5.1 管脚复用配置 ”)	SDIODAT2
MMC_DAT3	I/O	双向数据信号 3, 与 GPIO 复用。(复用时的配置信息请参见“ 8.5.1 管脚复用配置 ”)	SDIODAT3

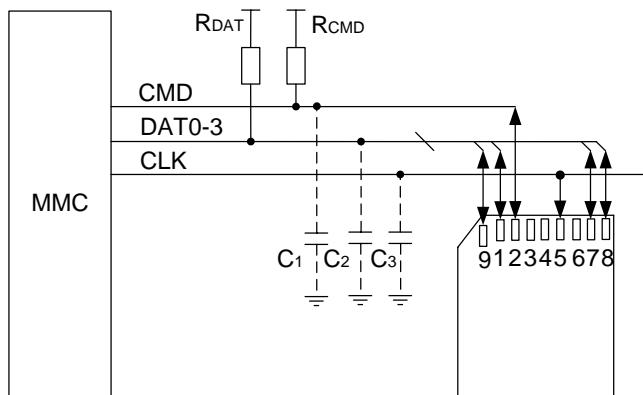
注: MMC_CCLK、MMC_CMD、MMC_DAT0~MMC_DAT3 分别对应后续说明中 CLK、CMD 及 DAT0~3 信号。

8.4 功能描述

典型应用

MMC 控制器的典型应用电路如[图 8-1](#) 所示。

图8-1 MMC 控制器典型应用电路图



说明

除[图 8-1](#) 中信号线外, 卡槽一般还提供机械写保护信号、卡检测信号, 系统可借助 GPIO 检测这 2 根信号线上的电平来完成机械写保护、热插拔时的卡检测功能。

MMC 控制器通过 1 根时钟信号线、1 根双向指令信号线和 4 根双向数据信号线与卡设备对接来完成命令与数据的交互。指令信号、数据信号均工作在上拉模式, 上拉电阻参数及各信号线负载电容限制如[表 8-2](#) 所示。

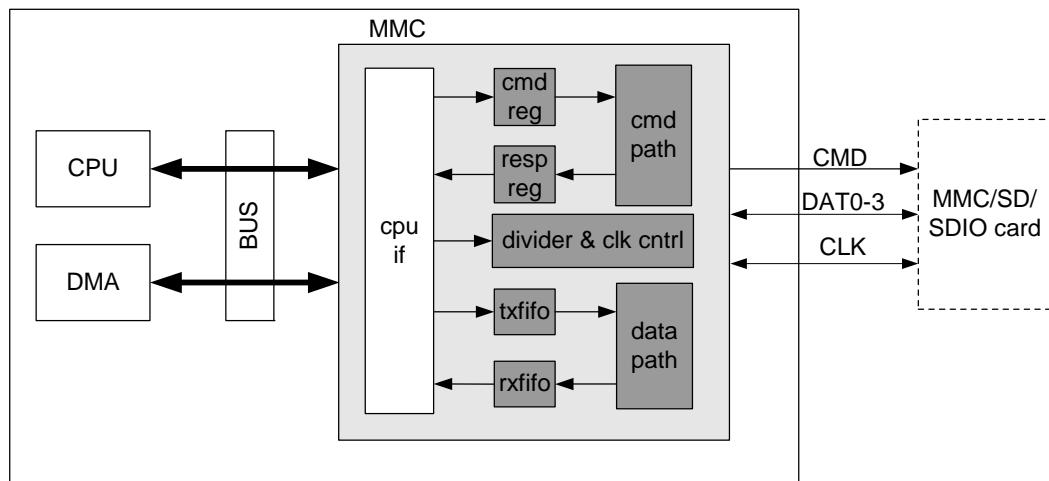
表8-2 信号线负载参数

参数	最小值	最大值	描述
R_{DAT} 、 R_{CMD}	10k Ω	100k Ω	上拉电阻。
负载电容 C_x	-	40pF	负载电容 $C_x = C_{mmchost} + C_{bus} + C_{card}$ 。每张卡最大负载电容 C_{card} 为 10pF，所以 $C_{mmchost} + C_{bus} \leq 30pF$ 。
信号线电感	-	16nH	$F_{pp} \leq 20MHz$ 。

功能原理

MMC 控制器的功能框图如图 8-2 所示。

图8-2 MMC 控制器功能框图



MMC 控制器通过内部总线与系统连接，主要由命令通道、数据通道、接口时钟控制单元构成，各单元功能如下：

- 命令通道
完成指令的发送与响应的接收。
- 数据通道
配合命令通道完成数据读写操作。
- 接口时钟控制单元
根据应用需求改变接口时钟频率，控制接口时钟的关断与开启。

指令与响应

MMC 控制器与卡设备之间的所有交互操作均是通过指令的形式来完成的，包括卡初始化序列、寄存器读写、状态查询、数据传输等。



指令根据是否有数据传输可分为数据传输指令和非数据传输指令：

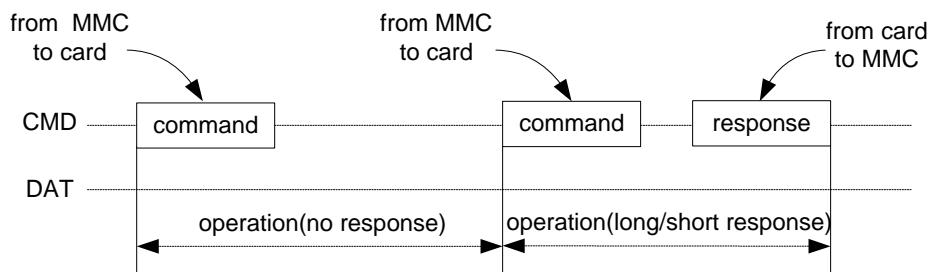
- 非数据传输指令
MMC 控制器基于指令信号线 CMD，进行串行方式指令的发送与响应的接收。
- 数据传输指令
除指令线上的交互外还伴随着数据线 DAT0~3 上的数据传输。

指令根据响应的类型又可分为：

- 无响应指令
如卡复位指令。
- 短响应指令
数据传输指令、卡状态查询等均属于这类指令。
- 长响应指令
仅用于读取卡的 CID、CSD 寄存器信息。

MMC 控制器与卡设备之间的非数据传输指令操作如图 8-3 所示。数据传输指令请参见图 8-6、图 8-7 所示。

图8-3 MMC 控制器非数据指令操作



指令（包括非数据传输指令和数据传输指令）为 48bit 的串行数据，由起始位、传输位、指令序号、指令参数、CRC 校验位和终止位组成。卡收到指令后，会根据指令类型返回 48bit 或 136bit 的响应。MMC 控制器的指令格式及响应格式如图 8-4 和图 8-5 所示。

图8-4 MMC 控制器指令格式

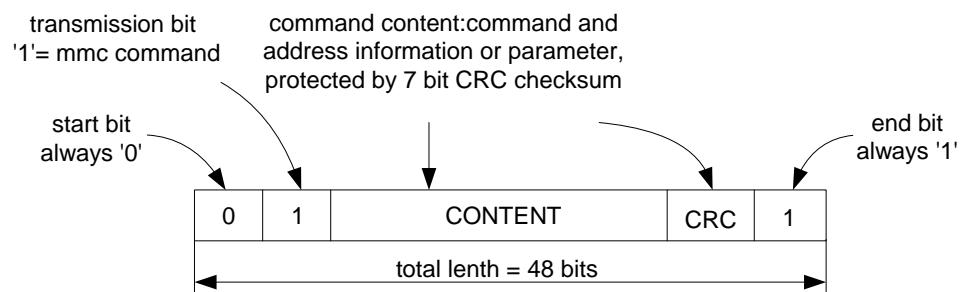
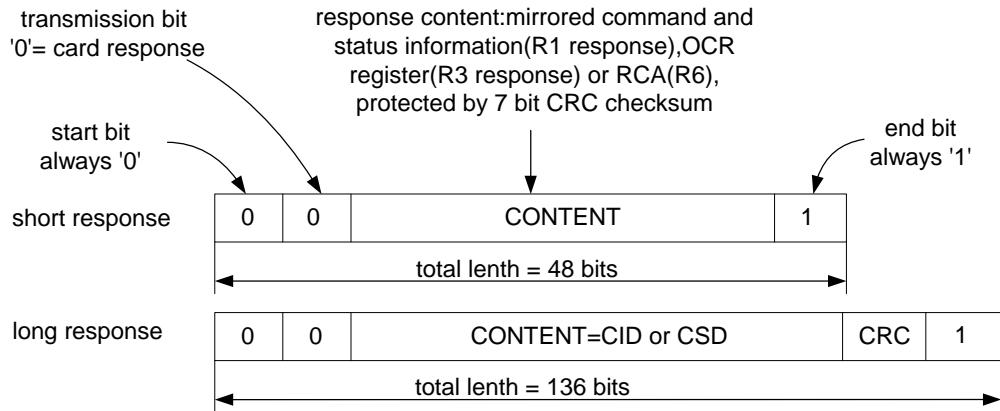




图8-5 卡设备响应格式



数据传输指令

MMC/SD/SDIO 卡支持数据读写方式如下：

- 流数据读写指令 (stream)
仅 MMC 卡支持，其只使用 1 根数据线即 DAT0 进行数据传输，无 CRC 校验。
 - 单块读写指令 (single block)
一次传输完成一个块大小的数据读写，不需要使用停止命令结束一次数据传输。
 - 多块读写指令 (multiple block)
 - predefined block count 方式
在多块读写指令前，发送块数量指令指定待传输的数据量。
 - open ended 方式
发送读写指令后，在数据传输末尾，需使用停止指令来结束一次数据传输。
两种方式的差别在于控制器通知卡结束一次传输的方式不同。SD 卡仅支持 open ended 方式，而 MMC 卡两种方式均支持。
- SDIO 设备的多块读写指令，不同于上述两种方式，在发送读写指令时，指令参数中包含待传输的数据量。

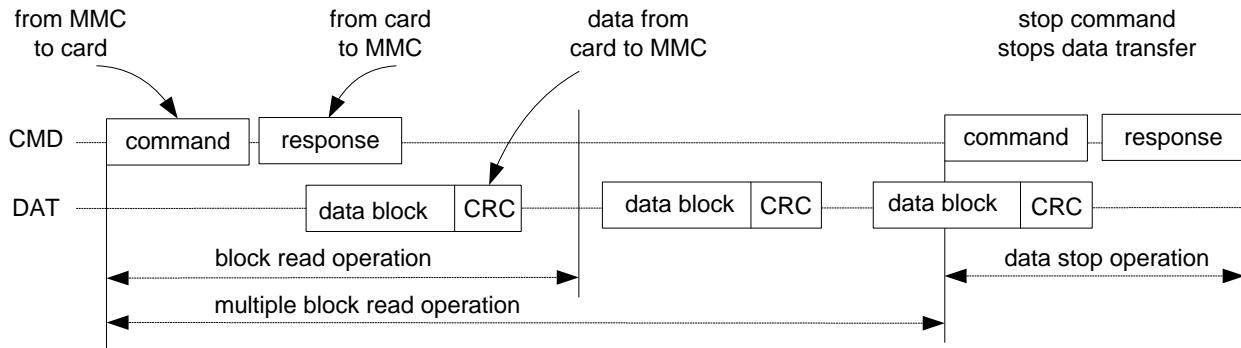
其中单块和多块读写指令为较常用的数据传输方式。通常 SD/MMC 卡数据传输的一个块大小为 512 字节，而 SDIO 设备可根据应用自定义。

说明

以块读写指令方式进行数据传输时，传输数据总量必须为块大小的整数倍。

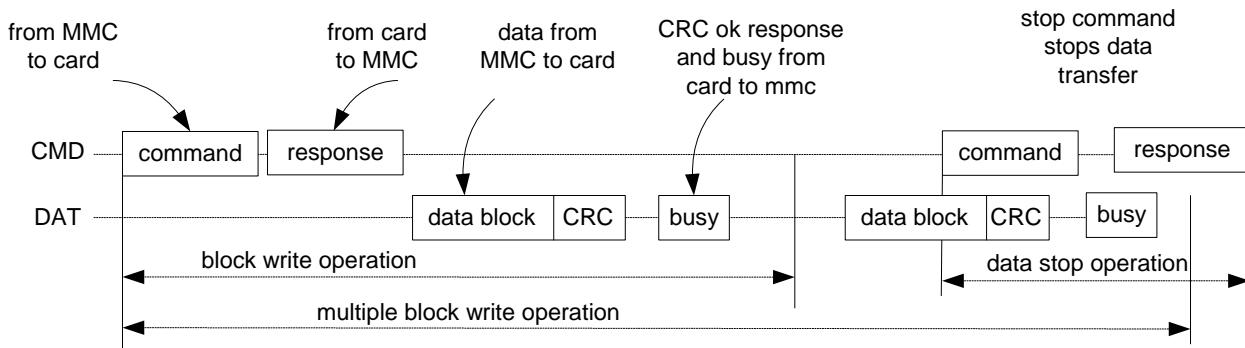
数据传输指令均为短响应指令，并伴随着数据线上的数据传输。指令、响应及数据线上的时序配合关系如图 8-6、图 8-7 所示。

图8-6 单块与多块读操作



MMC 控制器向卡发送单块或多块读指令，在接收响应的过程中，并开始接收以块为单位的数据，其中每块数据中均包含有 CRC 校验位，以保证数据传输的完整性。在单块读指令操作时，控制器在接收一块数据后完成一次数据传输；在 open ended 方式的多块读操作中，控制器在接收多块数据后，需发送一条停止指令结束本次数据传输。

图8-7 单块与多块写操作



MMC 控制器向卡发送单块或多块写指令，在接收到响应后，开始往卡发送以块为单位的数据，其中每块数据中均包含有 CRC 校验位，卡会对每块数据进行 CRC 校验，并反馈 CRC 状态以确认数据传输的正确性。在单块写指令操作时，控制器在发送一块数据后完成一次数据传输；在 open ended 方式的多块写操作中，控制器在发送多块数据后，需发送一条停止指令结束本次数据传输。写操作结束后，卡可能会因为编程 flash 而处于繁忙状态，控制器需查询 DAT0 状态，以确认卡脱离繁忙状态后才能对卡进行下一步操作。

块方式读写中控制器与卡之间可采用 1bit 或 4bit 数据线方式进行数据传输。

在进行数据传输指令之前，应分别设置控制器与卡的数据传输位宽模式（1bit 或 4bit），使其保持一致。控制器的数据位宽可通过 [MMC_CTYPE](#) 寄存器来进行设置，卡的数据位宽则通过发送相应的指令进行设置。1bit 和 4bit 模式下的数据传输格式分别如图 8-8 和图 8-9 所示。



图8-8 1bit 数据线传输模式下的块数据格式

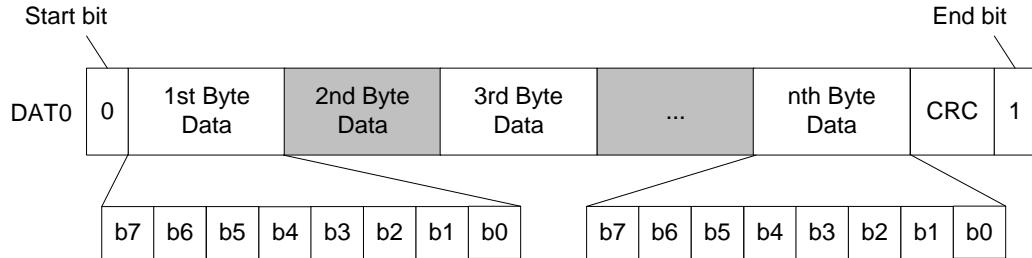
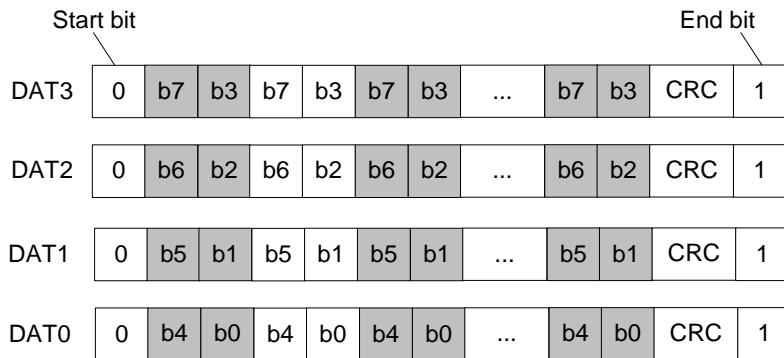


图8-9 4bit 数据线传输模式下的块数据格式



8.5 工作方式

8.5.1 管脚复用配置

MMC 控制器对外管脚与 GPIO 复用，使用 MMC 前，需要配置系统控制器使能相应管脚的 MMC 功能，详见寄存器 SC_PERCTRL1[pinmuxctrl9]配置说明。

8.5.2 时钟门控

在软件完成当前命令或数据传输且未启动新的传输的情况下，可关断 MMC 时钟，但需要确保控制器已处于空闲状态。

步骤如下：

步骤 1 读取控制器状态寄存器 [MMC_STATUS](#)。

步骤 2 若 [MMC_STATUS\[7:4\]](#)、[MMC_STATUS\[10\]](#)均为 0，则向 [MMC_CTRL](#) 写入 0，屏蔽控制器中断、DMA 请求使能等，进入步骤 3；若其中有一个为非 0，则延时等待，返回步骤 1。

步骤 3 将寄存器 [SC_PERDIS\[mmcclkdis\]](#)配置为 1，关闭控制器时钟。

步骤 4 如果需重新开启控制器工作时钟，则向 [SC_PEREN\[mmcclken\]](#)写 1。



----结束

8.5.3 软复位

在数据传输出现异常而导致控制器无法回到空闲状态时，可以配置 SC_PERCTRL0[mmc_srst]寄存器软复位控制器。可通过查询寄存器 **MMC_STATUS**[data_fsm_busy]确认控制器是否处于空闲状态。

建议在带电热插拔应用场景中在插卡后软复位控制器。

8.5.4 时钟配置

工作时钟配置

在使用 MMC 控制器前，需为其配置合适的工作时钟频率。MMC 控制器的工作时钟使用系统 PLL 的分频时钟，控制器工作时钟与 PLL 输出时钟频率关系为：

$$F_{MMCCLK} = F_{PLL} / [(2 \times mmcclk_sel) + 8]$$

其中分频因子 mmcclk_sel 为寄存器 SC_PERCTRL2[mmcclk_sel]配置值，MMCCLK 频率 F_{MMCCLK} 应低于 50MHz。

接口时钟配置

遵从不同协议版本的卡处于不同的状态时，均使用不同的时钟频率。因此 MMC 控制器内部提供一个偶数分频器以便于将工作时钟分频至合适的接口时钟。控制器工作时钟 MMCCLK 与接口时钟 MMC_CCLK 的频率关系为：

$$F_{MMC_CCLK} = F_{MMCCLK} / (2 \times clk_divider)$$

其中分频因子 clk_divider 为 **MMC_CLKDIV**[clk_divider]的配置值。

在改变卡的时钟频率之前，程序必须保证没有数据或指令正在传输。为了避免输出到多媒体卡的时钟产生毛刺，当改变卡的时钟频率的时候应该遵照以下步骤：

- 步骤 1 关闭接口时钟。将 **MMC_CLKENA** 寄存器配置为 0x0000_0000，并将 **MMC_CMD**[start_cmd]、**MMC_CMD**[update_clk_regs_only]和 **MMC_CMD**[wait_prvdata_complete]置 1，等待直到 **MMC_CMD**[start_cmd]被自动清除。
- 步骤 2 设置分频因子。根据所需要的时钟频率设置 **MMC_CLKDIV**，并将 **MMC_CMD**[start_cmd]和 **MMC_CMD**[update_clk_regs_only]置 1，等待直到 **MMC_CMD**[start_cmd]被自动清除。
- 步骤 3 重新使能接口时钟。将 **MMC_CLKENA** 寄存器配置为 0x0000_0001，并将 **MMC_CMD**[start_cmd]和 **MMC_CMD**[update_clk_regs_only]置 1，等待直到 **MMC_CMD**[start_cmd]被自动清除。

----结束



注意

- 只有当 **MMC_CMD[start_cmd]** 和 **MMC_CMD[update_clk_only]** 置 1，时钟配置寄存器的 **MMC_CLKDIV**、**MMC_CLKENA** 的值才会被载入。当载入成功以后，控制器会自动清除 **MMC_CMD[start_cmd]**。如果此时有其它指令正在执行，则会产生 HLE(Hardware Locked Error) 中断。若产生 HLE 中断，重新执行操作即可。
- 当有指令执行和数据传输时，不能变更卡的时钟参数。

8.5.5 初始化

在使用 MMC 控制器与卡进行命令和数据交互前，要进行初始化。步骤如下：

- 步骤 1 配置系统控制器，使能相应管脚的 MMC 功能。请参见“[8.5.1 管脚复用配置](#)”。
- 步骤 2 配置 MMC 控制器工作时钟频率。请参见“[8.5.4 时钟配置](#)”中的“[工作时钟配置](#)”。
- 步骤 3 当卡上电、指令和数据信号线上拉稳定后，软复位 MMC 控制器。请参见“[8.5.3 软复位](#)”。
- 步骤 4 清中断。将 **MMC_RINTSTS** 所有位置 1，清除原始中断状态位。
- 步骤 5 设置中断屏蔽寄存器。将 **MMC_INTMASK** 所有位置 1，使能各中断源。若使用 DMA 方式进行数据传输，应屏蔽接收/发送 FIFO 数据请求中断，将 **MMC_INTMASK[txdr_int_mask]**、**MMC_INTMASK[rxdr_int_mask]** 置 0。
- 步骤 6 将 **MMC_CTRL[int_enable]** 置 1，使能控制器中断功能。若使用 DMA 方式进行数据传输，应将 **MMC_CTRL[dma_enable]** 置 1，使能控制器 DMA 请求功能。
- 步骤 7 配置超时参数寄存器 **MMC_TMOOUT**。
- 步骤 8 配置 FIFO 参数寄存器 **MMC_FIFOTH**。其中包括 DMA 传输时的 burstsize 大小及接收方向与发送方向的阈值 rx_wmark、tx_wmark。

----结束

完成以上步骤后，就可以配置接口时钟，往卡发送指令了。

8.5.6 非数据传输指令

MMC 控制器在指令发送后，一旦收到任何响应（包括错误响应、有效响应和产生响应超时），控制器都会将 **MMC_RINTSTS[cmd_done]** 置位。短响应被保存到 **MMC_RESP0** 寄存器中，长响应被保存到 **MMC_RESP0**~**MMC_RESP3** 寄存器中，**MMC_RESP3[31]** 为最高有效位，**MMC_RESP0[0]** 为最低有效位。当指令发出以后，其错误是由指令响应以及 **MMC_RINTSTS** 相应的错误位来反映的。

发送非数据传输指令的步骤如下：

- 步骤 1 在 **MMC_CMDARG** 中设置相应的指令参数。
- 步骤 2 设置指令寄存器 **MMC_CMD**，其设置如表 8-3 所示。



- 步骤 3 等待指令被 MMC 控制器执行。如果指令已执行，控制器自动对 [MMC_CMD\[start_cmd\]](#) 清零。
- 步骤 4 通过查询 [MMC_RINTSTS\[hle_int_status\]](#)，检查是否产生 HLE 中断。
- 步骤 5 等待指令执行完毕。不管是收到响应或者响应时间超时，MMC 控制器都会将 [MMC_RINTSTS\[cmd_done\]](#) 置为 1。
- 步骤 6 检查是否有响应异常，有必要可读取响应值。可读取 [MMC_RINTSTS\[rto_int_status\]](#)、[MMC_RINTSTS\[rcrc_int_status\]](#)、[MMC_RINTSTS\[re_int_status\]](#) 寄存器来进行检查响应超时、响应 CRC 错误或响应错误等。

----结束



注意

只有当 [MMC_CMD\[start_cmd\]](#) 置 1、[MMC_CMD\[Update_clk_only\]](#) 置 0 时，指令相关寄存器 [MMC_BYTCNT](#)、[MMC_BLKSIZ](#)、[MMC_CMDARG](#)、[MMC_CMD](#) 的值才会被载入。当载入成功以后，控制器会自动清除 [MMC_CMD\[start_cmd\]](#)，除非有其它指令正在执行，在这种情况下，就会产生 HLE 中断。若产生 HLE 中断，重新执行操作即可。在非数据传输指令执行时，[MMC_BYTCNT](#)、[MMC_BLKSIZ](#) 值被忽略。

表8-3 非数据指令 MMC_CMD 参考配置

参数	取值	描述
Start_cmd	1	指令发送启动位。
Update_clk_regs_only	0	非时钟参数更新指令。
data_expected	0	非数据传输指令。
cmd_index	Cmd index	命令序号。
send_initialization	0	当指令为卡复位时置 1，如 CMD0。
stop_abort_cmd	0	当指令为停止数据传输时置 1，如 CMD12。
rsponse_length	0	当响应为长响应类型时置 1。
rsponse_expect	1	当指令无响应时置 0，如：CMD0、CMD4、CMD15。
Wait_prvdata_complete	1 或 0	在发送指令之前，控制器必须等待直到正在处理的数据传输指令结束。建议此位总置 1，除非该指令是为了在数据传输时查询卡状态或停止当前数据的传输。
Check_response_crc	1 或 0	控制器是否会检查响应的 CRC 校验位。



8.5.7 单块或多块读数据

读取单块或多块数据的步骤如下：

- 步骤 1 向 **MMC_CTRL[fifo_reset]** 写 1，复位 FIFO 指针，查询等待直至该位自动清 0。
- 步骤 2 在 **MMC_BYTCNT** 寄存器写入待传输数据的字节数。
- 步骤 3 向 **MMC_BLKSIZ** 寄存器写入块的大小。
- 步骤 4 向 **MMC_CMDARG** 写入读取数据的起始地址。
- 步骤 5 根据表 8-4 的参数来设置指令寄存器 **MMC_CMD**，对于 SD/MMC 卡，分别使用 CMD17/CMD18 来进行单块/多块操作；对于 SDIO 卡，使用 CMD53 来进行单块/多块的读操作。一旦 **MMC_CMD** 寄存器配置完成，MMC 控制器就开始执行指令；而当指令被送到总线上以后，就会产生 **cmd_done** 中断。
- 步骤 6 检测 **MMC_RINTSTS[rxdr_int_status]** 和 **MMC_RINTSTS[hto_int_status]**，如果其中之一为 1 或都为 1，应该从 **MMC_DATA** 寄存器读取 FIFO 中的数据，以便 MMC 控制器接收后面的数据；同时程序应检查数据错误中断，即寄存器 **MMC_RINTSTS[7]**、**MMC_RINTSTS[9]**、**MMC_RINTSTS[13]** 和 **MMC_RINTSTS[15]**，如果有需要，程序可以发送一个停止指令中止数据的传输。
- 步骤 7 当 **MMC_RINTSTS[dto_int_status]** 为 1 时，数据传输完成，从 **MMC_DATA** 寄存器中读取残留在 FIFO 中的数据。
- 步骤 8 若执行指令时已将 **MMC_CMD[send_auto_stop]** 置为 1，控制器会自动发送停止指令结束一次数据传输，请参见“[8.5.11 Auto-stop 使用配置](#)”。

----结束

表8-4 单块或多块读数据 MMC_CMD 参考配置

参数	取值	描述
默认		
start_cmd	1	指令发送启动位。
update_clk_regs_only	0	非时钟参数更新指令。
card_number	0	-
send_initialization	0	当指令为卡复位时置 1，如 CMD0。
stop_abort_cmd	0	当指令为停止数据传输时置 1，如 CMD12。
send_auto_stop	0 or 1	请参见“ 8.5.11 Auto-stop 使用配置 ”。
transfer_mode	0	块传输。
read_write	0	从卡中读取数据。
rsponse_length	0	数据指令均为短响应。
data_expected	1	数据传输指令。



参数	取值	描述
rspone_expect	1	当指令无响应时置 0, 如: CMD0、CMD4、CMD15。
cmd_index	Cmd index	命令序号。
wait_prvdata_complete	1 或 0	在发送指令之前, 主设备必须等待正在处理的数据传输指令结束, 建议此位总置 1, 除非该指令是为了查询卡状态或停止当前数据的传输。
check_response_crc	1 或 0	控制器是否会检查响应的 CRC 校验位。

8.5.8 单块与多块写数据

写入单块或多块数据的步骤如下:

- 步骤 1 向 **MMC_CTRL**[fifo_reset]写 1, 复位 FIFO 指针, 查询等待直至该位自动清 0。
- 步骤 2 向 **MMC_BYTCNT** 寄存器写入待传输数据的大小。
- 步骤 3 向 **MMC_BLKSIZ** 寄存器写入块的大小。
- 步骤 4 向 **MMC_CMDARG** 写入数据的起始地址。
- 步骤 5 将数据写入 FIFO (写 **MMC_DATA** 寄存器), 通常在最开始的时候应写满 FIFO。
- 步骤 6 根据表 8-5 的参数来设置 **MMC_CMD**, 对于 SD/MMC 卡, 分别使用 CMD24/CMD25 来进行单块/多块操作; 对于 SDIO 卡, 使用 CMD53 来进行单块/多块的写操作。
- 步骤 7 检测 **MMC_RINTSTS**[txdr_int_status]和 **MMC_RINTSTS**[hto_int_status], 如果其中之一为 1 或两者都为 1, 写寄存器 **MMC_DATA** 往 FIFO 填充数据; 同时应检测数据错误中断, 即检测 **MMC_RINTSTS**[7]、**MMC_RINTSTS**[9]、**MMC_RINTSTS**[13]和 **MMC_RINTSTS**[15], 如果有需要, 程序可以发送一个停止指令以中止数据的传输。当 **MMC_RINTSTS**[dto_int_status]为 1, 数据传输结束。
- 步骤 8 若执行指令时已将 **MMC_CMD**[send_auto_stop]置 1, 控制器会自动发送停止指令结束一次数据传输。请参见“[8.5.11 Auto-stop 使用配置](#)”。
- 步骤 9 查询并等待 **MMC_STATUS**[data_busy]由 1 变为 0。

----结束

表8-5 单块或多块写数据 MMC_CMD 参考配置

参数	取值	描述
Default		
start_cmd	1	指令发送启动位。
update_clk_regs_only	0	非时钟参数更新指令。



参数	取值	描述
card_number	0	-
send_initialization	0	当指令为卡复位时置 1, 如 CMD0。
stop_abort_cmd	0	当指令为停止数据传输时置 1, 如 CMD12。
send_auto_stop	0 or 1	请参见“ 8.5.11 Auto-stop 使用配置 ”。
transfer_mode	0	块传输。
read_write	1	往卡写入数据。
rsponse_length	0	数据指令均为短响应。
data_expected	1	数据传输指令。
rsponse_expect	1	当指令无响应时置 0, 如: CMD0, CMD4, CMD15。
cmd_index	Cmd index	-
wait_prvdata_complete	1 或 0	在发送指令之前, 主设备必须等待直到正在处理的数据传输指令结束, 建议此位总置 1, 除非该指令是为了查询卡状态或停止当前数据的传输。
check_response_crc	1 或 0	控制器是否会检查响应的 CRC 校验位。

8.5.9 流数据读写

流数据的读写方式, 除了将 [MMC_CMD\[transfer_mode\]](#)置 1 外, 其它与块数据的读写方式一致。对于流数据的传输, 通常需要使用控制器 auto-stop 功能。

8.5.10 DMA 方式数据传输

使用 DMA 方式进行数据传输前, 需将 [MMC_CMD\[dma_enable\]](#)置 1 来使能控制器 DMA 功能, 并将寄存器 [MMC_INTMASK\[txdr_int_mask\]](#)和 [MMC_INTMASK\[txdr_int_mask\]](#)清 0, 屏蔽控制器的 Receive FIFO data request 和 Transmit FIFO data request 中断。

使用 DMA 方式进行数据传输步骤如下:

步骤 1 复位 FIFO 指针。

步骤 2 配置 [MMC_BYTCNT](#)、[MMC_BLKSIZ](#)、[MMC_CMDARG](#)、[MMC_CMD](#) 寄存器启动数据传输指令。

步骤 3 分配 DMA 通道, 并配置相应通道寄存器 [DMAC_CX_SRC_ADDR](#)、[DMAC_CX_DEST_ADDR](#)、[DMAC_CX_CONTROL](#)、[DMAC_CX_CONFIG](#); 若使用



DMA 链表方式传输，还需配置通道的 DMAC_CX_LLI 寄存器，并启动该通道进行传输。

步骤 4 查询等待至 DMA 中断上报，表明数据传输结束。

步骤 5 检测控制器数据传输错误中断，数据传输完成中断。

步骤 6 执行停止指令，并查询 **MMC_STATUS**[data_busy]。

----结束

进行数据传输时，需要注意如下事项：

- 以非 DMA 方式读写卡时，只有当数据传送完成时，DTO (Data Transfer Over) 中断才会产生。而当卡的数据全部送出的时候，有可能还会有数据残留在 FIFO 中，而 Rx_wmark 中断的产生与否是由残留在 FIFO 中数据的字节数来决定的。软件判断出 DTO 中断产生后，应读出 FIFO 中所有的残留数据。
- 以 DMA 方式读写卡时，只有当所有 FIFO 中的数据都从 DMA 接口单元写入到内存中以后，DTO 中断才会产生。
- 数据传输时，数据量大小应该为 FIFO 数据宽度的整数倍。如果需要写入到卡的数据为 15 个字节，那么应该传送 16 个字节的数据到 FIFO，或当 DMA 方式使能时对 DMA 进行编程以执行 16 个字节数据的传输。程序仍然可以把 **MMC_BYTCNT** 设为 15，这时，只有 15 个字节的数据被传输到卡中。类似的，当从卡中读取 15 个比特的数据的时候，主设备将会从 FIFO 中读出 16 个字节的数据。

8.5.11 Auto-stop 使用配置

在多块读写指令操作中，需使用停止指令完成一次数据传输。停止指令的发送可以通过**非数据传输指令**的方式发送，也可以使用控制器的 auto-stop 功能发送。auto-stop 使用配置如下：

在执行块数据传输指令操作中，将 **MMC_CMD**[send_auto_stop]置 1，在所有数据传输完成后，控制器会自动发送一次停止指令，以便卡能返回相应状态；该停止指令的完成由 **MMC_RINTSTS**[auto_cmd_done]中断位来反映，其响应被保存在 **MMC_RESP1** 中。

Auto-stop 功能应用场合：

- SD 卡
 - 多块读写操作，如 CMD18 和 CMD25。
- MMC 卡
 - 流数据读写操作。
 - open-ended 方式的多块读写操作，如 CMD18 和 CMD25。
 - predefined block count 方式的多块读写操作不需要使用 auto-stop 功能，在 CMD18 和 CMD25 指令之前发送 CMD23 指令指定此次待传输的块数量。
- SDIO 设备
 - 不需要使用 auto-stop 的功能。



8.5.12 停止或中止数据传输

停止指令用于打断 MMC 控制器与卡之间的数据传输，中止指令用于打断 I/O 数据的传输（仅用于 SDIO_IOONLY 或 SDIO_COMBO）。这两种指令的用法如下：

- 停止指令

该指令可以在数据传送的任何阶段进行发送。因为该指令是为了停止数据的传输，所以需要将指令寄存器 **MMC_CMD**[5:0] 设为 CMD12、将 **MMC_CMD**[14] 设置为 1、将 **MMC_CMD**[13] 设置为 0。

- 发送中止指令

仅用于 SDIO_IOONLY 或 SDIO_COMBO。为了中止数据的传输，需要通过 CMD52 设置 SDIO 卡的寄存器 CCCR[ASx] 位。

8.5.13 Suspend 和 Resume 操作

对于 SDIO 卡（内部最多可容纳 7 个功能设备），控制器可通过 suspend 操作暂停某一功能设备的数据传输，将 SD 接口总线让给出另一个有着更高优先级的功能设备。高优先级的功能设备完成数据传输后，控制器可以通过 resume 操作恢复前一功能设备未完成的传输。

Suspend 与 Resume 操作通过设置 SDIO 卡的 CCCR 寄存器相应比特来实现。读写 CCCR 寄存器，使用指令 CMD52。

Suspend 操作步骤如下：

步骤 1 通过查询 CCCR 寄存器的 SBS 位，判断 SDIO 卡是否支持 suspend/resume 操作。

步骤 2 通过查询 CCCR 寄存器的 FSx 位和 BS 位，判断待暂停的功能设备是否正在进行数据传输。**注意：**如果 BS 位为 1，那么 FSx 位所指定的功能设备正在进行数据传输。

步骤 3 为了暂停当前数据传输，需要将 CCCR 寄存器的 BR 位置 1。

步骤 4 检测 CCCR 寄存器的 BS（Bus 状态）位和 BR（Bus 释放）位状态是否清零。BS 位在数据总线正被使用的时候保持为 1；BR 位在总线完全释放之前都保持为 1。当 BR 与 BS 位都为 0 的时候，所选功能设备的数据传输就被成功暂停。

步骤 5 如果暂停正在进行的读操作，那么在 suspend 操作成功完成以后，必须置 **MMC_CTRL**[abort_read_data] 位来复位控制器数据传输功能，复位完成后，**MMC_CTRL**[abort_read_data] 位自动清零。

步骤 6 读 **MMC_TCBCNT** 寄存器获取已传输数据字节数。

----结束

Resume 操作步骤如下：

步骤 1 检查卡是否处于非传输状态，以确认总线处于空闲状态。

步骤 2 如果卡处于 disconnect 状态，使用 CMD7 将它选中。卡的状态可以通过 CMD52/CMD53 指令获取。

步骤 3 检查待恢复的功能设备是否准备好进行数据传输。可以通过查询 CCCR 寄存器的 RFx 标志位来进行确认，如果 RF=1，那么该功能设备就已准备好进行数据传输。



- 步骤 4 为了恢复传输，使用指令 CMD52 将功能设备号写入 CCCR 寄存器的 FSx 位。发送 CMD52 指令的同时应启动控制器进入数据传输状态：即往 [MMC_BLKSIZ](#) 寄存器写入块的大小，[MMC_BYTCNT](#) 寄存器写入剩余待传输数据量，[MMC_CMDARG](#) 寄存器的配置如表 8-6 所示，[MMC_CMD](#) 寄存器的配置与块传输类似。
- 步骤 5 当 CMD52 指令成功发送以后，数据传输被恢复了。读取 DF (Resume Data Flag) 标志位，如果为 1，那么在功能被恢复的同时，数据就开始传输了；如果标志是 0，那么就已无数据需要传输。
- 步骤 6 如果 DF 标志位为 0，那么在读数据的情况下，控制器会等待一段时间后产生数据超时错误中断。

----结束

表8-6 Resume 操作 [MMC_CMDARG](#) 参考配置

MMC_CMDARG	描述	取值
bit[31]	读写标志	1
bit[30:28]	功能设备号	0，访问 CCCR
bit[27]	实时标志	1，先写后读
bit[26]	-	-
bit[25:9]	寄存器地址	0x0D
bit[8]	-	-
bit[7:0]	写数据	被恢复的功能号

8.5.14 Read wait 操作

Read wait 操作用于对 SDIO 卡暂停当前功能设备的数据传输，给任一功能设备发送指令。控制器可以根据应用需要随意决定暂停数据传输的时间长度。其步骤如下：

- 步骤 1 检查卡是否支持 read wait 操作；读取 CCCR 寄存器的 SRW 位。如果这一位为 1，那么卡的所有功能设备都支持 read wait 操作。使用 CMD52 读取这一位。
- 步骤 2 如果卡支持 read wait 等操作，将 [MMC_CTRL\[read_wait\]](#) 位置 1。
- 步骤 3 如需恢复数据传输，清零 [MMC_CTRL\[read_wait\]](#)。

----结束

8.6 寄存器概览

MMC 寄存器概览如表 8-7 所示。

表8-7 MMC 寄存器概览 (基址是 0x9002_0000)

偏移地址	名称	描述	页码
0x000	MMC_CTRL	控制寄存器	8-19
0x004	RESERVED	保留	-
0x008	MMC_CLKDIV	时钟分频因子寄存器	8-20
0x00C	RESERVED	保留	-
0x010	MMC_CLKENA	时钟使能寄存器	8-21
0x014	MMC_TMOUT	超时参数寄存器	8-21
0x018	MMC_CTYPE	接口位宽寄存器	8-22
0x01C	MMC_BLKSIZ	块大小寄存器	8-22
0x020	MMC_BYTCNT	传输数据大小寄存器	8-23
0x024	MMC_INTMASK	中断屏蔽寄存器	8-23
0x028	MMC_CMDARG	指令参数寄存器	8-25
0x02C	MMC_CMD	指令寄存器	8-26
0x030	MMC_RESP0	指令响应寄存器 0	8-28
0x034	MMC_RESP1	指令响应寄存器 1	8-28
0x038	MMC_RESP2	指令响应寄存器 2	8-29
0x03C	MMC_RESP3	指令响应寄存器 3	8-29
0x040	MMC_MINTSTS	中断状态寄存器	8-29
0x044	MMC_RINTSTS	原始中断状态寄存器	8-31
0x048	MMC_STATUS	状态寄存器	8-34
0x04C	MMC_FIFOTH	FIFO 参数寄存器	8-35
0x050~0x058	RESERVED	保留	-
0x05C	MMC_TCBCNT	接口传输计数寄存器	8-37
0x060	MMC_TBBCNT	FIFO 传输计数寄存器	8-37
0x064~0xFF	RESERVED	保留	-
0x100	MMC_DATA	数据寄存器	8-38



8.7 寄存器描述

MMC_CTRL

MMC_CTRL 为 MMC 控制寄存器，用于完成控制器的全局功能控制，包括中断全局使能控制、DMA 使能控制等。

	Offset Address	Register Name	Total Reset Value
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	MMC_CTRL	0x0000_0000
Name	reserved		
Reset	0 0		
Bits	Access	Name	Description
[31:9]	-	reserved	保留。默认为 0，不可写 1。
[8]	RW	abort_read_data	读状态机复位控制位。 0：默认保持值。 1：如果使用 suspend 操作暂停正在进行的数据读传输，在 suspend 操作完成后，软件将该位使能，使控制器从数据传输状态（等待下一个 block 数据）恢复到 IDLE 状态。当控制器回到 IDLE 状态后，该比特会自动清 0。
[7]	RW	send_irq_response	0：默认保持值。 1：发送自动 IRQ 响应。 该比特在发送响应后自动清 0。 为了等待 MMC 产生中断，控制器发送 CMD40 并等待来自 MMC 中断响应。同时，如果控制器希望不再停留在中断等待状态，可以使能该比特，送出 CMD40 响应并回到 IDLE 状态。
[6]	RW	read_wait	0：清除读等待。 1：使能读等待。 发送读等待到 SDIO 设备。

[5]	RW	dma_enable	DMA 传输模式使能位。 0: 禁止。 1: 使能。 即使 DMA 模式已经使能, CPU 仍然可以对 FIFO 进行读写, 但在实际操作中应该避免。如果 DMA 和 CPU 同时对 FIFO 进行读写, 控制器无法仲裁其优先级。
[4]	RW	int_enable	全局中断使能位。 0: 禁止。 1: 使能。 只有该比特有效且有中断源被使能的情况下, 中断输出才有效。
[3]	-	reserved	保留。
[2]	RW	dma_reset	DMA 接口功能复位控制。 0: 不复位。 1: 复位。 写 1 使能复位, 该比特在完成复位后自动清 0。
[1]	RW	fifo_reset	FIFO 复位控制位。 0: 不复位 FIFO 读写指针。 1: 复位 FIFO 读写指针。 写 1 使能复位, 该比特在完成复位后自动清 0。
[0]	-	reserved	保留。默认为 0, 不可写 1。

MMC_CLKDIV

MMC_CLKDIV 为接口时钟分频因子寄存器, 用于控制接口时钟频率。控制器工作时钟 MMCCCLK 与接口时钟 MMC_CCLK 的频率关系为:

$F_{MMC_CCLK} = F_{MMCCCLK}/2 \times \text{clk_divider}$ 。只有当 [MMC_CMD\[start_cmd\]](#) 和 [MMC_CMD\[Update_clk_only\]](#) 置 1, 该寄存器的值才会被载入。

	Offset	Address	Register Name	Total Reset Value
Bit	0x008			MMC_CLKDIV
Name	reserved			clk_divider
Reset	0	0	0	0
Bits	Access	Name	Description	
[31:8]	-	reserved	保留。	



[7:0]	RW	clk_divider	接口时钟分频系数，偶数分频。 如 0x0 为无分频、0x1 为 2 分频、0xFF 为 510 分频。
-------	----	-------------	--

MMC_CLKENA

MMC_CLKENA 为 MMC 接口时钟使能寄存器。只有当 [MMC_CMD\[start_cmd\]](#) 和 [MMC_CMD\[Update_clk_only\]](#) 置 1，该寄存器的值才会被载入。

	Offset	Address	Register Name	Total Reset Value																												
Bit	0x010																															
Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:17]	-	reserved	保留。																													
[16]	RW	cclk_low_power	卡的低功耗控制。 0: 非低功耗模式。 1: 低功耗模式。当卡处于 IDLE 状态，控制器自动关闭接口时钟。该功能一般只使用在 MMC、SD 存储卡，对于 SDIO 设备，为了能检测到 SDIO 中断，不可关闭接口时钟。																													
[15:1]	-	reserved	保留。																													
[0]	RW	cclk_enable	卡的时钟使能控制。 0: 时钟关闭。 1: 时钟使能。																													

MMC_TMOUT

MMC_TMOUT 为超时参数寄存器。用于数据读操作、命令响应中超时参数配置。



Offset Address			Register Name			Total Reset Value																										
0x014			MMC_TMOUT			0xFFFF_FF40																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	data_timeout															response_timeout																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:8]	RW	data_timeout	数据读操作超时参数，该值也做 data starvation 中断超时参数。单位为接口时钟周期，建议设置为 0xFF_FFFF。																													
[7:0]	RW	response_timeout	响应超时参数。单位为接口时钟周期，建议值为 0xFF。																													

MMC_CTYPE

MMC_CTYPE 为接口位宽配置寄存器，用于配置控制器工作在 1bit 数据位宽或 4bit 数据位宽。控制器与卡的位宽应该保持一致。

Offset Address			Register Name			Total Reset Value																										
0x018			MMC_CTYPE			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															reserved	reserved														card_width	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description																													
[31:17]	-	reserved	保留。																													
[16]	-	reserved	保留，不可写 1。																													
[15:1]	-	reserved	保留。																													
[0]	RW	card_width	配置卡接口的总线宽度。 0: 1bit 模式。 1: 4bit 模式。																													

MMC_BLKSIZ

MMC_BLKSIZ 为块大小寄存器，SD/MMC 卡一般为 512 字节，SDIO 设备可根据应用需求自定义。



Offset Address			Register Name	Total Reset Value																												
0x01C			MMC_BLKSIZ	0x0000_0200																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																block_size															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留。																													
[15:0]	RW	block_size	块大小配置, 如配置为 0x0200, 表示块大小为 512 字节。																													

MMC_BYTCNT

MMC_BYTCNT 为传输数据大小寄存器。若传输数据大小等于块大小, 为单块数据传输; 若传输数据大小等于块大小的整数倍, 为多块数据传输。传输数据大小必须为块大小的整数倍。

Offset Address			Register Name	Total Reset Value																												
0x020			MMC_BYTCNT	0x0000_0200																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	byte_count																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:0]	RW	byte_count	待传输数据字节数, 如配置为 0x0200, 表示待传输数据为 512 字节。块传输方式时应该设置为 block size 的整数倍。																													

MMC_INTMASK

MMC_INTMASK 为中断屏蔽寄存器, 用于屏蔽 [MMC_RINTSTS](#) 对应位中断请求。



Offset Address		Register Name		Total Reset Value															
0x024		MMC_INTMASK		0x0000_0000															
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																	reserved	
Name		reserved																	
Reset	0 0																		
Bits	Access	Name	Description																
[31:17]	-	reserved	保留。																
[16]	RW	sdio_int_mask	SDIO 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[15]	RW	ebe_int_mask	EBE (End-bit error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[14]	RW	acd_int_mask	ACD (Auto Command Done) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[13]	RW	sbe_int_mask	SBE (Start-bit Error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[12]	RW	hle_int_mask	HLE (Hardware Locked Write Error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[11]	RW	frun_int_mask	FRUN (FIFO underrun/overrun error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[10]	RW	hto_int_mask	HTO (Data starvation-by-host timeout) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																
[9]	RW	drto_int_mask	DRTO (Data Read Timeout) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。																



[8]	RW	rto_int_mask	RTO (Response Timeout) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[7]	RW	dcrc_int_mask	DCRC (Data CRC error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[6]	RW	rcrc_int_mask	RCRC (Response CRC error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[5]	RW	rxdr_int_mask	RXDR (Receive FIFO data request) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[4]	RW	txdr_int_mask	TXDR (Transmit FIFO data request) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[3]	RW	dto_int_mask	DTO (Data Transfer Over) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[2]	RW	cd_int_mask	CD (Command done) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[1]	RW	re_int_mask	RE (Response error) 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[0]	-	reserved	保留。

MMC_CMDARG

MMC_CMDARG 为指令参数寄存器。寄存器 **MMC_CMD**[5:0]指定指令序号所对应的指令参数。如 cmd17 单块数据读操作, **MMC_CMD**[5:0]=17, 指令参数寄存器应配置为卡内空间地址。各指令序号对应指令参数请参考 SD/MMC/SDIO 协议。



Offset Address			Register Name	Total Reset Value																												
0x028			MMC_CMDARG	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	cmd_arg																															
Reset	0 0																															
Bits	Access	Name	Description																													
31:0	RW	cmd_arg	指令参数配置。																													

MMC_CMD

MMC_CMD 为指令寄存器，用于指定指令特征，如序号、响应类型、是否进行数据传输、数据传输方向、传输模式等。

Offset Address			Register Name	Total Reset Value																												
0x02C			MMC_CMD	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	start_cmd reserved update_clk_reg_only card_number response_expect response_length check_response_crc data_transfer_expected data_transfer_expected read/write cmd_index																															
Reset	0 0																															
Bits	Access	Name	Description																													
[31]	RW	start_cmd	指令执行或加载接口时钟参数启动位。 与 update_clk_reg_only 位配合使用，在 update_clk_reg_only 为 0 时将该位置 1，用于启动指令执行；在 update_clk_reg_only 为 1 时将该位置 1，用于加载接口时钟参数。 当指令被执行或时钟参数已载入，该比特自动清 0。当该比特为 1 时，CPU 不允许修改时钟和指令相关的寄存器。如果修改，hardware lock error 中断就会产生。																													
[30:22]	-	reserved	保留。不可写 1。																													
[21]	RW	update_clk_reg_only	0: 正常指令顺序。 1: 不发送命令，重新载入接口时钟控制寄存器 (MMC_CLKDIV, MMC_CLKENA) 的值。在不需要发指令给卡的前提下，用来调整接口时钟的频率和控制接口时钟的开关。 在正常命令顺序中，即该位置为 0，下面的寄存器值会被控制																													



			器载入：MMC_CMD, MMC_CMDARG, MMC_TMOOUT, MMC_CTYPE, MMC_BLKSIZ, MMC_BYTCNT。控制器将新的寄存器值使用到新的指令中去。 如果该比特设为 1，指令不会传到卡中，不会产生 Command Done 中断。
[20:16]	RW	card_number	正在使用的卡的序号，应设置为 0。
[15]	RW	send_initialization	0：在发送该指令前不要送出初始序列。 1：在发送该指令前送出初始序列（80 个时钟周期）。 在上电以后，卡需要 80 个时钟周期进行初始化；因此需要在往卡发送第一条指令时需设置该位。
[14]	RW	stop_abort_cmd	数据读写状态机复位控制位。 0：无影响。 1：在数据传输过程中发送停止指令异常中止本次传输时，将该位置 1 用于将控制器从数据传输状态恢复到 IDLE 状态。
[13]	RW	wait_prvdata_complete	0：立即发送指令（即使前一个数据传输还没完成）。 1：等到前一个数据传输完成后才开始发送指令。 在指令发送时将该位配置为 0 可用来在数据传输时读取卡状态或中止本次传输。
[12]	RW	send_auto_stop	0：数据传完以后不会发停止指令。 1：数据传完以后发送停止指令。 当被使能以后，控制器在每次数据传输完以后就会自动送出停止指令。 predefined block count 方式、cmd53 等不要停止指令。 open-ended 传输方式需要停止指令。 在非数据传输时，该位被忽略。
[11]	RW	transfer_mode	0：用于块读写指令。 1：用于流数据读写指令。 在非数据传输时，该位被忽略。
[10]	RW	read/write	0：从卡读取数据。 1：往卡写数据。 在非数据传输时，该位被忽略。
[9]	RW	data_transfer_expected	0：非数据指令。 1：数据指令。
[8]	RW	check_response_crc	0：不检查指令响应 CRC。 1：检查指令响应 CRC。 一些指令回复没有返回有效的 CRC。为了禁止控制器对 CRC 进行检查，软件需要针对这些指令禁止该功能。



[7]	RW	response_length	0: 短响应指令。 1: 长响应指令。
[6]	RW	response_expect	0: 无响应指令。 1: 有响应指令。
[5:0]	RW	cmd_index	指令序号。

MMC_RESP0

MMC_RESP0 为指令响应寄存器 0。

	Offset Address																									Register Name	Total Reset Value					
	0x030																								MMC_RESP0	0x0000_0000						
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	response0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:0]	RO	response 0	48bit 短响应的 bit[39:8]或 136 比特长响应的 bit[31:0]。																													

MMC_RESP1

MMC_RESP1 为指令响应寄存器 1。

	Offset Address																									Register Name	Total Reset Value					
	0x034																								MMC_RESP1	0x0000_0000						
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	response1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:0]	RO	response1	指令长响应 bit[63:32]或控制器 auto-stop 指令的响应。 当控制器发出 auto-stop 指令, 响应的 bit[39:8]就会被保存在该寄存器。上一个指令的响应仍然会被保存在 MMC_RESP0 寄存器内。Auto-stop 只供数据传输使用, 而且回复的类型总是短响应。																													



MMC_RESP2

MMC_RESP2 为指令响应寄存器 2。

	Offset Address 0x038																																Register Name MMC_RESP2	Total Reset Value 0x0000_0000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	response2																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description																												[31:0]	RO	response2	指令长响应的 bit[95:64]。

MMC_RESP3

MMC_RESP3 为指令响应寄存器 3。

	Offset Address 0x03C																																Register Name MMC_RESP3	Total Reset Value 0x0000_0000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	response3																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description																											[31:0]	RO	response3	指令长响应 bit[127:96]。	

MMC_MINTSTS

MMC_MINTSTS 为屏蔽后的中断状态寄存器。MMC_MINTSTS = MMC_RINTSTS & MMC_INTMASK。只有当 [MMC_RINTSTS](#) 和 [MMC_INTMASK](#) 对应位均为 1，且 [MMC_CTRL\[int_enable\]](#) 为 1 时，中断才会上报。



		Offset Address	Register Name	Total Reset Value																											
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	MMC_MINTSTS	0x0000_0000																												
Name	reserved																														
Reset	0 0																														
Bits	Access	Name	Description																												
[31:17]	-	reserved	保留。																												
[16]	RO	sdio_int	屏蔽后的 SDIO 中断。 0: 未产生中断。 1: 已产生中断。																												
[15]	RO	ebe_int	屏蔽后的 EBE (End-bit error (read)/Write no CRC) 中断。 0: 未产生中断。 1: 已产生中断。																												
[14]	RO	acd_int	屏蔽后的 ACD (Auto Command Done) 中断。 0: 未产生中断。 1: 已产生中断。																												
[13]	RO	sbe_int	屏蔽后的 SBE (Start-bit Error) 中断。 0: 未产生中断。 1: 已产生中断。																												
[12]	RO	hle_int	屏蔽后的 HLE (Hardware Locked Write Error) 中断。 0: 未产生中断。 1: 已产生中断。																												
[11]	RO	frun_int	屏蔽后的 FRUN (FIFO underrun/overrun error) 中断。 0: 未产生中断。 1: 已产生中断。																												
[10]	RO	hto_int	屏蔽后的 HTO (Data starvation-by-host timeout) 中断。 0: 未产生中断。 1: 已产生中断。																												
[9]	RO	drto_int	屏蔽后的 DRTO (Data Read Timeout) 中断。 0: 未产生中断。 1: 已产生中断。																												



[8]	RO	rto_int	屏蔽后 RTO (Response Timeout) 中断。 0: 未产生中断。 1: 已产生中断。
[7]	RO	dcrc_int	屏蔽后的 DCRC (Data CRC error) 中断。 0: 未产生中断。 1: 已产生中断。
[6]	RO	rcrc_int	屏蔽后的 RCRC (Response CRC error) 中断。 0: 未产生中断。 1: 已产生中断。
[5]	RO	rxdr_int	屏蔽后的 RXDR (Receive FIFO data request) 中断。 0: 未产生中断。 1: 已产生中断。
[4]	RO	txdr_int	屏蔽后的 TXDR (Transmit FIFO data request) 中断。 0: 未产生中断。 1: 已产生中断。
[3]	RO	dto_int	屏蔽后的 DTO (Data Transfer Over) 中断。 0: 未产生中断。 1: 已产生中断。
[2]	RO	cd_int	屏蔽后的 CD (Command done) 中断。 0: 未产生中断。 1: 已产生中断。
[1]	RO	re_int	屏蔽后的 RE (Response error) 中断。 0: 未产生中断。 1: 已产生中断。
[0]	-	reserved	保留。

MMC_RINTSTS

MMC_RINTSTS 为原始中断状态寄存器。对应位写 1 清 0，写 0 无效，即写 1 表示清中断。





			(MMC_CCLK) 会被停止。当时钟被停止后, data-starvation 计数器就会启动。如果计数器数满溢出, 而此时系统仍然没有往空的 FIFO 写数据或从满的 FIFO 读数据, 该中断产生。这时候, 需要系统对 FIFO 的进行读写操作, 输出时钟才会重新启动。
[9]	RW	drto_int_status	屏蔽前的 DRTO (Data Read Timeout) 中断。 0: 未产生中断。 1: 接收数据超时中断。
[8]	RW	rto_int_status	屏蔽前 RTO (Response Timeout) 中断。 0: 未产生中断。 1: 指令响应超时中断 (未收到响应)。
[7]	RW	dcrc_int_status	屏蔽前的 DCRC (Data CRC error) 中断。 0: 未产生中断。 1: 接收数据 CRC 校验错误中断。
[6]	RW	rcrc_int_status	屏蔽前的 RCRC (Response CRC error) 中断。 0: 未产生中断。 1: 指令响应 CRC 校验错误中断。
[5]	RW	rxdr_int_status	屏蔽前的 RXDR (Receive FIFO data request) 中断。 0: 未产生中断。 1: 当从卡读数据时, FIFO 中数据多于 FIFO 读阈值 rx_wmark 时产生中断。使用 DMA 进行数据传输时, 应屏蔽该中断。
[4]	RW	txdr_int_status	屏蔽前的 TXDR (Transmit FIFO data request) 中断。 0: 未产生中断。 1: 当往卡写数据时, FIFO 中数据少于或等于 FIFO 写阈值 tx_wmark 时产生中断。使用 DMA 进行数据传输时, 应屏蔽该中断。
[3]	RW	dto_int_status	屏蔽后的 DTO (Data Transfer Over) 中断。 0: 未产生中断。 1: 数据传输完毕中断。即使出现 Start Bit Error、CRC error 或 Read Data Timeout, 该中断仍然产生。
[2]	RW	cd_int_status	屏蔽前的 CD (Command done) 中断。 0: 未产生中断。 1: 命令执行完毕并收到响应产生中断。即使出现 response error、response CRC error 或 response timeout, 该中断仍然产生。



[1]	RW	re_int_status	屏蔽前的 RE (Response error) 中断。 0: 未产生中断。 1: 收到的指令响应有错误时该中断产生。
[0]	-	reserved	保留。

MMC_STATUS

MMC_STATUS 为 MMC 状态寄存器，反映控制器工作状态。

		Offset Address	Register Name	Total Reset Value			
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	0x048	MMC_STATUS	0x0000_0000			
Name	dma_req	fifo_count	resp_index	cmd_fsm_state	fifo_rx_watermark		
Reset	0 0						
Bits	Access	Name	Description				
[31]	RO	dma_req	DMA 请求状态位。 0: 控制器未请求 DMA 传输。 1: 控制器正请求 DMA 传输。				
[30]	RO	dma_ack	DMA 确认状态位。 0: DMAC 未清除 MMC 控制器请求。 1: DMAC 清除 MMC 控制器请求。				
[29:17]	RO	fifo_count	FIFO 已有数据数量，以 word 为单位。				
[16:11]	RO	resp_index	上一个指令响应的序号。包括 auto-stop 指令的响应。				
[10]	RO	data_fsm_busy	数据发送/接收状态机的状态。 0: 数据发送/接收状态机处于闲置状态。 1: 数据发送/接收状态机处于繁忙状态。				
[9]	RO	data_busy	0: 卡处于闲置状态。 1: 执行写或擦除操作后卡处于繁忙状态。 该状态位直接反映卡数据线 dat[0]信号的取反。执行写或擦除等操作后需由软件查询该位，直至该位由 1 变为 0 后，才能对卡进行下一步的操作。				
[8]	RO	data_3_status	该位直接反映卡数据线 data[3]信号状态。				
[7:4]	RO	cmd_fsm_state	控制器命令状态机状态。				



			0000: Idle。 0001: Send init sequence。 0010: Tx cmd start bit。 0011: Tx cmd tx bit。 0100: Tx cmd index +arg。 0101: Tx cmd crc7。 0110: Tx cmd end bit。 0111: Rx resp start bit。 1000: Rx resp IRQ response。 1001: Rx resp tx bit。 1010: Rx resp cmd idx。 1011: Rx resp data。 1100: Rx resp crc7。 1101: Rx resp end bit。 1110: Cmd path wait NCC。 1111: Wait; CMD-to-response turnaround。
[3]	RO	fifo_full	FIFO 满状态标志位。 0: FIFO 不满。 1: FIFO 满。
[2]	RO	fifo_empty	FIFO 空状态标志位。 0: FIFO 非空。 1: FIFO 空。
[1]	RO	fifo_tx_watermark	FIFO 内数据个数小于写阈值 tx_wmark。
[0]	RO	fifo_rx_watermark	FIFO 内数据大于或等于读阈值 rx_wmark。

MMC_FIFOTH

MMC_FIFOTH 为 MMC FIFO 参数寄存器，建议配置值为 0x2007_0008。用 DMA 进行数据传输时，应将 DMA 与控制器的 burst 设置为同样大小，当数据正以 DMA 模式传输时不要改变该寄存器值。



Offset Address			Register Name	Total Reset Value																												
Bit	0x04C MMC_FIFOTH			0x0000_0000																												
Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31]	-	reseved	保留。																													
[30:28]	RW	burst_size	DMA 一次 burst 传输数据量, 以 word 为单位。应该被设置为跟 DMAC 相同的大小。 000: 1。 001: 4。 010: 8。 其它: 保留。 允许使用的 burst_size 和 tx_wmark 组合为: Burst_size = 1, Tx_wmark = 1~15。 Burst_size = 4, Tx_wmark = 4。 Burst_size = 4, Tx_wmark = 4。 Burst_size = 4, Tx_wmark = 12。 Burst_size = 8, Tx_wmark = 8。 Burst_size = 8, Tx_wmark = 4。 允许使用的 burst_size 和 rx_wmark 组合为: Burst_size = 1, Rx_wmark = 0~14。 Burst_size = 4, Rx_wmark = 3。 Burst_size = 4, Rx_wmark = 7。 Burst_size = 4, Rx_wmark = 11。 Burst_size = 8, Rx_wmark = 7。 Burst_size = 8, Rx_wmark = 11。																													
[27:16]	RW	rx_wmark	读数据时的 FIFO 阈值。当 FIFO 已有数据个数大于该值时, 产生 DMA 请求; 若中断使能, 则产生中断请求。 非 DMA 模式下 receive FIFO data request (RXDR) 中断会被使能并产生中断请求。在数据传输末尾, 如果 FIFO 计数没有大于该值, 不会产生中断。在 Data Transfer Done 中断产生后来由软件完成剩余数据的读取。 DMA 模式下, 在数据传输末尾, 当剩余的数据比阈值低, DMA 仍会在 Data Transfer Done 中断产生之前通过 burst 方式把数据读取。																													



			限制: Tx_wmark \leq FIFO_DEPTH - 2 建议值: (FIFO_DEPTH/2) - 1; 即多于(FIFO_DEPTH / 2) - 1)时发出请求。
[15:12]	-	resevered	保留。
[11:0]	RW	tx_wmark	发送数据时的 FIFO 阈值。当 FIFO 已有数据个数小于或等于该值时, 产生 DMA 请求; 若中断使能, 则产生中断请求。 非 DMA 模式下 transmit FIFO data request (TXDR) 中断会被使能并产生中断请求。在数据传输末尾, 中断产生后由软件完成剩余字节的传送。 DMA 模式下, 在数据传输末尾, 若剩余的数据小于 burst size, DMA 仍通过 burst 方式传送完成剩余数据的传输。 限制: Tx_wmark \geq 1 建议值: FIFO_DEPTH/2, 即小于或等于 FIFO_DEPTH/2 时发出请求。

MMC_TCBCNT

MMC_TCBCNT 为接口传输计数寄存器, 用于统计完成一次数据传输指令后接口传输的数据字节数。在数据传输过程中, 该寄存器返回值为 0; 数据传输结束后该寄存器反映控制器与卡之间数据传输字节数。该寄存器在启动一次新的数据传输指令时清 0。

	Offset Address			Register Name	Total Reset Value																											
	0x05c			MMC_TCBCNT	0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	trans_card_byte_coun																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31: 0]	RO	trans_card_byte_co	控制器与卡之间数据传输字节数。																													

MMC_TBBCNT

MMC_TBBCNT 为 FIFO 传输计数寄存器。用于实时统计执行数据传输指令时 CPU/DMA 与控制器 FIFO 之间传输的数据字节数, 在数据传输过程中动态变化。该寄存器在启动一次新的数据传输指令时清 0。



MMC_DATA

MMC_DATA 为数据寄存器，为 FIFO 入口地址。在读写 FIFO 时，应先读取 **MMC_STATUS[fifo_count]** 得到 FIFO 剩余空间，以此确定读写的数据量，以免造成 FIFO 溢出。



9 PCI

关于本章

本章描述内容如下表所示。

标题	内容
9.1 概述	介绍 PCI 模块的功能。
9.2 特点	介绍 PCI 模块的特点。
9.3 信号描述	介绍 PCI 模块的接口信号。
9.4 功能描述	介绍 PCI 模块的典型应用和功能原理。
9.5 工作方式	介绍 PCI 模块的工作方式。
9.6 寄存器概览	概况介绍 PCI 模块的寄存器。
9.7 寄存器描述	详细描述 PCI 模块的寄存器。



9.1 概述

PCI (Peripheral Component Interconnect) 总线是一种通用的本地总线 (Local Bus)。PCI 接口以其灵活的可扩展性用于实现产品的不同应用形态和应用场合, 实现多设备级联和扩展符合 PCI/miniPCI 接口的外设, 如 SATA、WiFi、PCI-to-PCI Bridge 等。

Hi3511/Hi3512 的 PCI 接口用来实现片外 PCI 总线到片内 AHB (Advanced High-performance Bus) 总线的转换, 符合 PCI2.3 总线协议, 并兼容 miniPCI 接口协议。Hi3511/Hi3512 的 PCI 接口有两种应用模式, 可分别实现 PCI Host 和 PCI Device 功能:

- Host 模式

当配置为 Host 模式时, 可用作整个 PCI 总线的主控设备, 对整个 PCI 总线进行配置管理和仲裁。

- Device 模式

当配置为 Device 模式时, 可用于实现 PCI Device 功能, 通过 PCI 总线和 PCI Host, 以及 PCI 总线上其它 PCI 设备进行通信。

9.2 特点

Hi3511/Hi3512 的 PCI 接口有以下特点:

- 支持 PCI2.3 协议。
- 总线位宽 32bit, 总线频率最高可达 66MHz。
- 支持 PCI Host 和 PCI Device 功能:

PCI Host 模式下

- 支持 PCI_INTA、PCI_INTB 中断。
- 支持 memory read/write, configuration read/write 命令, 配置访问时可支持 Type0 和 Type1 配置命令。
- 内建 PCI Bus Arbiter, 且最多支持 5 个 PCI 设备的总线仲裁。

PCI Device 模式下:

- 支持 PCI_INTA 中断。
- 支持 memory read/write, I/O read/write, configuration read/write 命令。
- 支持 PCI 侧到 AHB 侧的地址翻译。
- 支持用户可编程的 doorbell 中断。
- 支持 PCI 总线奇偶校验。
- 支持对 memory 的 prefetchable 访问和 non-prefetchable 访问。



9.3 信号描述

PCI 接口信号如表 9-1 所示。

表9-1 PCI 接口信号描述

信号名称	方向	描述	对应管脚
PCI_CLK	I/O	PCI 总线时钟信号。 • PCI Host 模式下，PCI 时钟信号可由芯片内部产生，PCI_CLK 为输出信号，将片内产生的 PCI 时钟输出给总线上的其它设备使用；在 PCI Host 模式下 PCI 时钟也可选择为由外部时钟源输入，片内不产生 PCI 时钟，PCI_CLK 为输入信号。 • PCI Device 模式下，PCI_CLK 则固定为输入信号。	PCICLK
PCI_RST	I/O	PCI 总线复位信号。 • PCI Host 模式下，PCI_RST 由片内产生，并输出到总线上供其它设备使用，PCI_RST 固定为输出信号。 • PCI Device 模式下，PCI_RST 固定为输入信号。	PCIRSTN
PCI_AD[31:0]	I/O	PCI 总线地址/数据信号。	PCIAD0~PCIAD31
PCI_CBE[3:0]	I/O	PCI 总线命令/字节使能信号。	PCICBE0~PCICBE3
PCI_FRAME	I/O	PCI 总线 frame 信号。	PCIFRAMEN
PCI_IRDY	I/O	PCI 总线 initiator ready 信号。	PCIIRDYN
PCI_TRDY	I/O	PCI 总线 target ready 信号。	PCITRDYN
PCI_DEVSEL	I/O	PCI 总线 device select 信号。	PCIDEVSELN
PCI_STOP	I/O	PCI 总线 stop 信号。	PCISTOPN
PCI_IDSEL	I	PCI 总线 initial device select 信号。 • PCI Device 模式下为 PCI_IDSEL。 • PCI Host 模式下为 PCI_INTB。	PCIIDSEL
PCI_INTB	I	PCI 总线 INTB 信号	PCIIDSEL
PCI_PAR	I/O	PCI 总线 parity 信号。	PCIPAR
PCI_SERR	I/O	PCI 总线 system error 信号。	PCISERRN
PCI_PERR	I/O	PCI 总线 parity error 信号。	PCIPERRN

信号名称	方向	描述	对应管脚
PCI_INTA	I/O	PCI 总线 INTA 信号。 • PCI Host 模式下, PCI_INTA 为输入信号。 • PCI Device 模式下, PCI_INTA 为输出信号。	PCIINTAN
PCI_REQ[4:0]	I	PCI 总线 bus request 信号。 • PCI Host 模式下, PCI_REQ[4:0] 为输入信号。 • PCI Device 模式下, PCI_REQ[0] 被复用作 PCI 的总线仲裁信号 PCI_GNT; PCI_REQ[4] 被复用作 GPIO1_7。 说明 当 Hi3511/Hi3512 为 PCI Device 模式时, 不会使用 PCI_REQ[3:1], 此时需要对管脚进行上拉处理, 将其固定接为高电平。	PCIREQ0N~ PCIREQ4N
PCI_GNT[4:0]	O	PCI 总线 bus grant 信号。 • PCI Host 模式下, PCI_GNT[4:0] 为输出信号。 • PCI Device 模式下, PCI_GNT[0] 被复用作 PCI 的总线申请信号 PCI_REQ; PCI_GNT[4] 被复用作 GPIO2_0。	PCIGRANT0N~ PCIGRANT4N

注: PCI 接口信号中 PCI_FRAME、PCI_IRDY、PCI_TRDY、PCI_STOP、PCI_DEVSEL、PCI_SERR、PCI_PERR、PCI_INTA 和 PCI_INTB 在实际应用需做上拉处理。

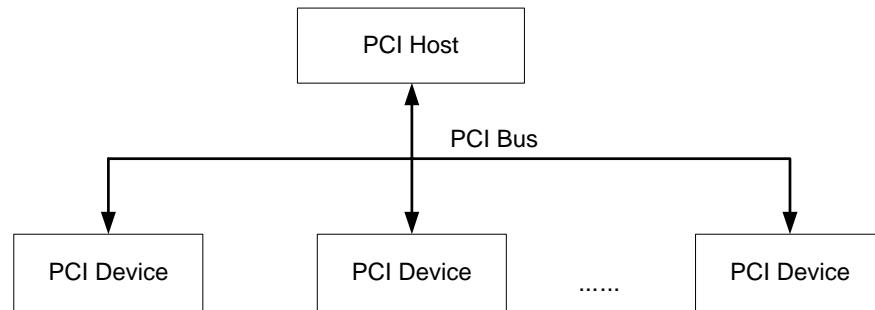
9.4 功能描述

典型应用

PCI 总线典型应用模式如图 9-1 所示。



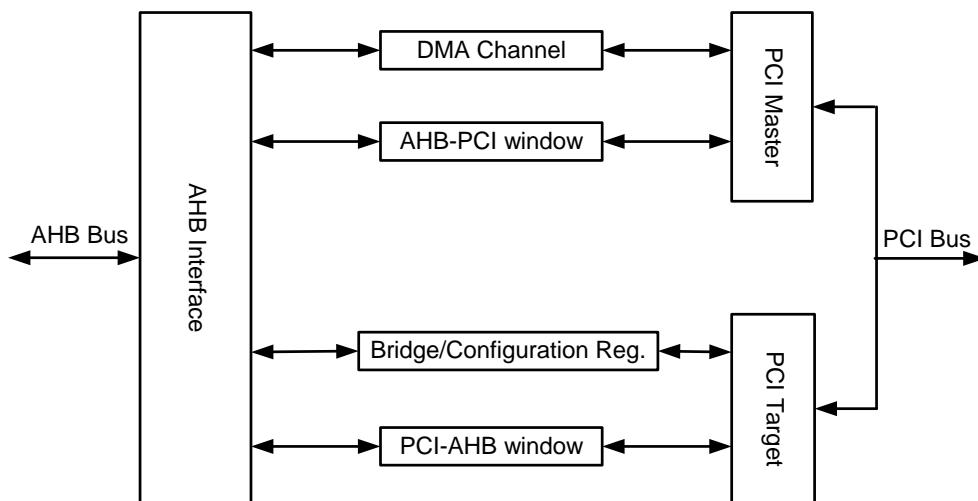
图9-1 PCI 总线典型应用



PCI 总线是一种并行的本地总线，具有总线带宽高，扩展性强的特点。因此，基于 PCI 接口可以非常灵活地实现芯片间的级联扩展以及各种其它外设的扩展应用。

Hi3511/Hi3512 PCI 模块架构如图 9-2 所示。

图9-2 Hi3511/Hi3512 PCI 模块架构示意图



Hi3511/Hi3512 PCI 模块内建 DMAC，DMA 通道可用于以 DMA 方式进行数据的搬运。除此之外，ARM core 也可以通过 AHB-PCI window 访问 PCI 总线上的其它 PCI 设备。PCI 总线上其它 PCI 设备通过 PCI-AHB window 来访问 Hi3511/Hi3512。

功能原理

PCI 总线的主要信号包括地址/数据信号，命令/字节使能信号，以及接口控制信号等。基于不同的应用模式，PCI 总线操作主要有以下几种：memory 操作、I/O 操作和配置操作。各种操作的时序图如图 9-3~图 9-5 所示。



图9-3 PCI 总线 memory (I/O) 读操作时序图

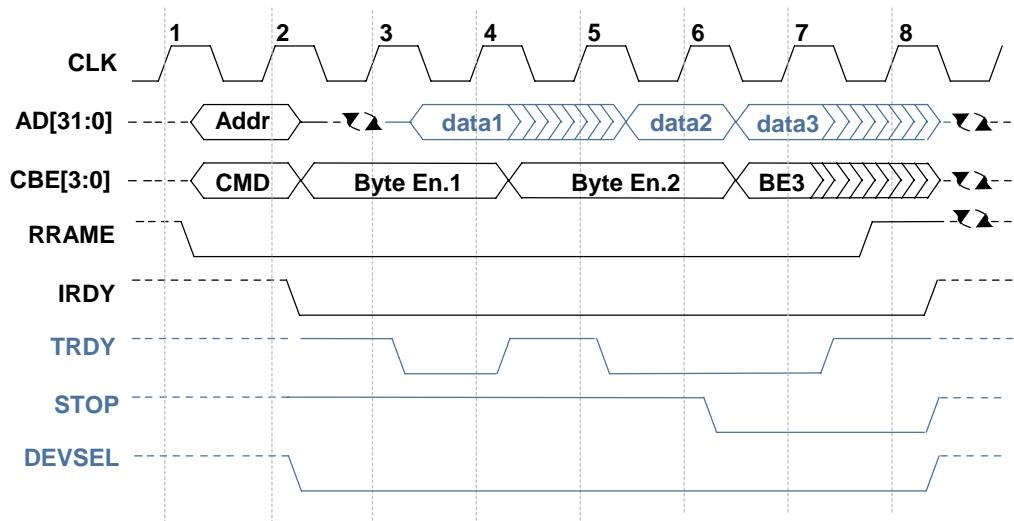


图9-4 PCI 总线 memory (I/O) 写操作时序图

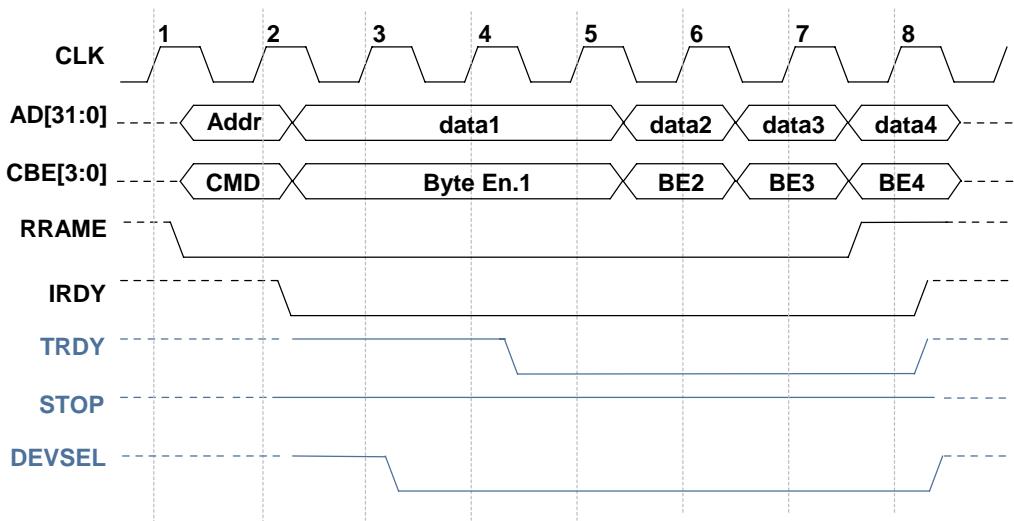
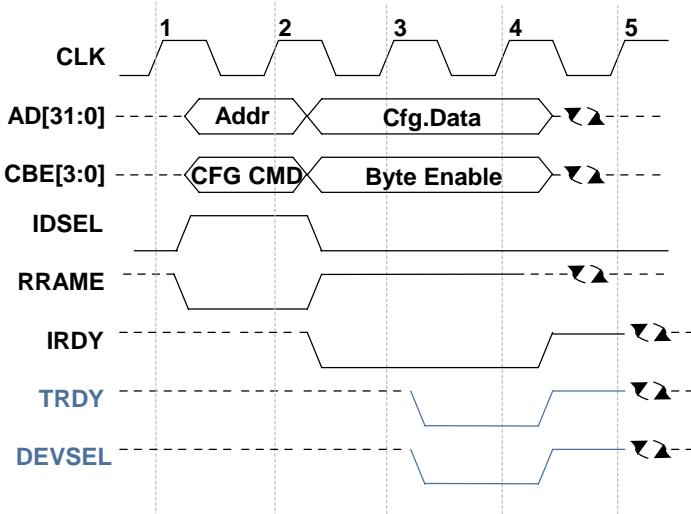




图9-5 PCI 总线配置访问时序图



9.5 工作方式

9.5.1 管脚复用配置

PCI 接口管脚复用关系如下:

- 当 Hi3511/Hi3512 为 PCI Device 模式时, 管脚复用关系如下:
 - PCIIDSEL 复用为 PCI_IDSEL。
 - PCIREQ0N 复用为 PCI 总线仲裁信号 PCIGRANT_SLAVE。
 - PCIGRANT0N 复用为 PCI 总线申请信号 PCIREQ_SLAVE。
 - PCIREQ4N 复用为 GPIO1_7。
 - PCIGRANT4N 复用为 GPIO2_0。

PCI 接口管脚复用的配置是通过配置相应的系统控制寄存器来完成的, 详见 SC_PERCTRL1[13]配置说明。

- 当 Hi3511/Hi3512 为 PCI Host 模式时, PCIIDSEL 管脚为 PCI_INTB 功能。



说明

- 当 Hi3511/Hi3512 为 PCI Device 模式时, PCIIDSEL、PCIREQ0N 和 PCIGRANT0N 的复用是自动的, 不需要配置系统控制寄存器; 而 PCIREQ4N 和 PCIGRANT4N 与 GPIO 之间的复用则需要配置系统控制寄存器 SC_PERCTRL1[13]来实现。
- 当 Hi3511/Hi3512 为 PCI Host 模式时, PCIIDSEL 的复用是自动的, 不需要配置系统控制寄存器。

9.5.2 时钟门控

在当前 PCI 总线处于空闲状态时, 可通过配置系统控制寄存器来关断 PCI 时钟:

- 当 Hi3511/Hi3512 为 PCI Host 模式时, PCI 时钟可由如下两种方式产生:



- 内部产生：PCI 时钟是由 Hi3511/Hi3512 内部时钟模块产生，除了 Hi3511/Hi3512 PCI 模块使用之外还从芯片管脚输出供其它 PCI 设备使用。此时如果 Hi3511/Hi3512 关断 PCI 时钟，将只关断自己 PCI 模块的时钟，不会影响送往片外的时钟。
- 外部产生：Hi3511/Hi3512 的时钟是由外部时钟源输入，Hi3511/Hi3512 将只关断 PCI 模块的时钟。
- 当 Hi3511/Hi3512 为 PCI Device 模式时，PCI 时钟固定为 PCI 总线时钟输入，此时 Hi3511/Hi3512 将只关断自己 PCI 模块的时钟，不会影响到其它 PCI 设备。

关于 Hi3511/Hi3512 PCI 时钟门控的配置，详见 SC_PEREN[9]和 SC_PERDIS[9]配置说明。

9.5.3 时钟配置



说明

系统控制寄存器 SC_PERCTRL4[20]只有在 PCI Host 模式下才可配置。

通过配置系统控制寄存器 SC_PERCTRL4[20]，可控制芯片 PCI 时钟是由片内产生还是由外部时钟源输入：

- 当 SC_PERCTRL4[20]为 1 时，PCICLK 为输出，此时芯片内部产生 PCI 时钟通过 PCICLK 管脚送出到片外供整个 PCI 总线使用。
- 当 SC_PERCTRL4[20]为 0 时，PCICLK 为输入，此时整个 PCI 总线的时钟都由外部时钟源提供。

在 PCI Device 模式下，SC_PERCTRL4[20]固定配置为 0，即固定为外部输入 PCI 时钟。

当 Hi3511/Hi3512 为 PCI Host 模式且 PCI 时钟是由 Hi3511/Hi3512 片内产生时，可通过配置系统控制寄存器 SC_PERCTRL2[23:20]来选择所产生 PCI 时钟的频率。具体的配置请参见 SC_PERCTRL2[23:20]的配置说明。

9.5.4 软复位

通过配置系统控制器 SC_PERCTRL0[8]为 1，可实现对整个 PCI 模块的单独软复位。特别需要注意的是，当作为 PCI Host 模式时，此时对 PCI Host 的软复位会通过PCIRST 管脚将整个 PCI 总线上的所有设备都复位。因此，在对 PCI Host 进行软复位后，要对所有 PCI Device 重新进行初始化。

当作为 PCI Device 模式时，配置系统控制器 SC_PERCTRL0[8]为 1 只是对自己的 PCI 模块进行复位，不会对 PCI 总线的其它 PCI 设备产生影响。但是在软复位结束后，同样需要 PCI Host 对复位过的 PCI Device 进行初始化。

对 Hi3511/Hi3512 PCI 模块的一次软复位操作包括配置软复位和撤消软复位。当对系统控制器 SC_PERCTRL0[8]写入 1 后，PCI 模块进入复位状态，在不小于 4ms 的时间间隔后，对系统控制器 SC_PERCTRL0[8]写入 0，撤销软复位。

9.5.5 工作模式配置

通过配置系统控制器 SC_PERCTRL4[17]，可实现将 Hi3511/Hi3512 配置为 PCI Host 模式或者 PCI Device 模式：



- 当 SC_PERCTRL4[17]为 1 时, Hi3511/Hi3512 为 PCI Host 模式。
- 当 SC_PERCTRL4[17]为 0 时, Hi3511/Hi3512 为 PCI Device 模式。

9.5.6 通过 window 进行数据传输



注意

只有当 Hi3511/Hi3512 作为 PCI Host 模式时才能通过 AHB-PCI window 对 PCI 总线上的其它设备进行访问。

通过 window 进行数据传输时, 可相应选择访问 prefetchable 空间还是 non-prefetchable 空间。

Hi3511/Hi3512 可通过 AHB-PCI window 实现 ARM core 对 PCI 总线上其它 PCI 设备的访问, PCI 总线上的其它 PCI 设备也通过 PCI-AHB window 实现对 Hi3511/Hi3512 的访问。

Hi3511/Hi3512 通过 AHB-PCI window 访问 PCI 设备

Hi3511/Hi3512 通过 AHB-PCI window 访问 PCI 设备的步骤如下:

步骤 1 Hi3511/Hi3512 作为 PCI Host, 初始化 PCI 总线上其它 PCI 设备。

步骤 2 ARM core 发起 AHB 总线操作, 此时的 AHB 地址就是所要访问的 PCI 设备的目标地址。即 AHB-PCI window 两侧的地址是透传的, 不需要地址翻译。

----结束

Hi3511/Hi3512 作为 PCI Master, 支持发出 Memory Write、I/O Write、I/O Read 和 Memory Read Multiple 命令。

PCI 设备通过 PCI-AHB window 访问 Hi3511/Hi3512

PCI 设备通过 PCI-AHB window 访问 Hi3511/Hi3512 的步骤如下:

步骤 1 PCI Host 设备初始化整个 PCI 总线。

步骤 2 PCI 设备发起对 Hi3511/Hi3512 的访问。当 Hi3511/Hi3512 为 PCI Host 模式时, 此时的 PCI 地址就是所要访问的 Hi3511/Hi3512 的目标地址, 直接透传到 AHB 侧, 不需要地址翻译。当 Hi3511/Hi3512 为 PCI Device 模式, PCI 地址需要通过地址翻译才能到达 AHB 侧。

----结束

Hi3511 作为 PCI Target 时, 支持如下 PCI 访问命令: Memory Write、Memory Read、Configuration Read with Type0/Type1、Configuration Writer with Type0/Type1 和 Memory Read Multiple。



9.5.7 通过 DMA 通道进行数据传输

Hi3511/Hi3512 PCI 模块内建 DMAC，可直接由 PCI 接口发起 DMA 操作，此时不需要 ARM core 的干预，可获得更好的系统性能。进行数据传输时存在以下两种情况：

- 当 Hi3511/Hi3512 为 PCI Host 模式时，通过 DMA 通道进行数据传输不会引起 PCI 总线中断。
- 当 Hi3511/Hi3512 为 PCI Device 模式时，此时通过 DMA 通道进行数据传输会涉及到 PCI 总线中断。

当 Hi3511/Hi3512 工作在 PCI HOST 模式时，对 PCI 总线的初始化是通过配置访问来实现的。需要注意的是，此时的配置访问都是通过 DMA 方式来完成。

Hi3511/Hi3512 PCI 接口的 DMA 操作命令如表 9-2 所示。

表9-2 PCI 接口信号描述

操作	命令	描述
Read DMA	0010	I/O Read
	0110	Memory Read
	1010	Configuration Read
	1100	Memory Read Multiple
Write DMA	0011	I/O Write
	0111	Memory Write
	1011	Configuration Write

注：表 9-2 中未列出的命令均做保留处理。

当 Hi3511/Hi3512 为 PCI Host 模式时，通过 DMA 通道进行数据传输的步骤如下：

- 步骤 1 配置 CPU_IMASK 寄存器，使能 DMA Read/Write 中断。
- 步骤 2 将数据传输的目标地址写入 [RDMA_PCI_ADDR/WDMA_PCI_ADDR](#)。
- 步骤 3 将数据传输的源地址写入 [RDMA_AHB_ADDR/WDMA_AHB_ADDR](#)。
- 步骤 4 将 DMA 控制命令写入 [RDMA_CONTROL/WDMA_CONTROL](#)，其中：

1. [RDMA_CONTROL/WDMA_CONTROL\[31:8\]](#) 为 transfer size
2. [RDMA_CONTROL/WDMA_CONTROL\[7:4\]](#) 为 command
3. [RDMA_CONTROL/WDMA_CONTROL\[1\]](#) 为 stop bit
4. [RDMA_CONTROL/WDMA_CONTROL\[0\]](#) 为 start bit

----结束

当 Hi3511/Hi3512 为 PCI Device 模式时，通过 DMA 通道进行数据传输的步骤如下：



配置 CPU_IMASK 寄存器，打开 DMA Read/Write 中断使能。

步骤 5 将数据传输的目标地址写入 [RDMA_PCI_ADDR/WDMA_PCI_ADDR](#)。

步骤 6 将数据传输的源地址写入 [RDMA_AHB_ADDR/WDMA_AHB_ADDR](#)。

步骤 7 将 DMA 控制命令写入 [RDMA_CONTROL/WDMA_CONTROL](#)，其中：

1. [RDMA_CONTROL/WDMA_CONTROL\[31:8\]](#) 为 transfer size
2. [RDMA_CONTROL/WDMA_CONTROL\[7:4\]](#) 为 command
3. [RDMA_CONTROL/WDMA_CONTROL\[3\]](#) 为 pci_interrupt bit
4. [RDMA_CONTROL/WDMA_CONTROL\[1\]](#) 为 stop bit
5. [RDMA_CONTROL/WDMA_CONTROL\[0\]](#) 为 start bit

----结束



说明

- 无论 Hi3511/Hi3512 为 PCI Host 模式还是 PCI Device 模式，当 DMA 操作结束后不需要以中断方式通知 CPU 时，此时可不执行步骤 1，或者配置 CPU_IMASK 寄存器将对应的中断屏蔽位关闭。
- 当 Hi3511/Hi3512 为 PCI Host 模式时，[RDMA_CONTROL/WDMA_CONTROL\[3\]](#) 固定为 0b0。

9.6 寄存器概览

Hi3511/Hi3512 PCI 接口实现了一个标准的符合 PCI V2.3 规范的 PCI 配置空间。此外，在 AHB 侧也实现了一些其它的配置寄存器，以方便 CPU 对 PCI 模块的各种工作方式的配置。

AHB 侧寄存器

AHB 侧寄存器概览如表 9-3 所示。

表9-3 AHB 侧寄存器概览（基址是 0xB000_0000）

偏移地址	名称	描述	页码
0x000	WDMA_PCI_ADDR	DMA 写操作目的地址寄存器	9-14
0x004	WDMA_AHB_ADDR	DMA 写操作源地址寄存器	9-14
0x008	WDMA_CONTROL	DMA 写操作 transfer size 和控制命令寄存器	9-15
0x00C ~ 0x01C	RESERVED	保留	-
0x020	RDMA_PCI_ADDR	DMA 读操作目的地址寄存器	9-16
0x024	RDMA_AHB_ADDR	DMA 读操作源地址寄存器	9-16

偏移地址	名称	描述	页码
0x028	RDMA_CONTROL	DMA 读操作 transfer size 和控制命令寄存器	9-16
0x02C ~ 0x03C	RESERVED	保留	-
0x040	CPU_IMASK	中断屏蔽寄存器	9-17
0x044	CPU_ISTATUS	中断状态寄存器	9-20
0x048	CPU_ICMD	中断命令寄存器	9-22
0x04C	CPU_VERSION	设备版本寄存器	9-22
0x050 ~ 0x06C	RESERVED	保留	-
0x070	PCIAHB_ADDR_NP	PCI-AHB 窗口非预取范围地址控制寄存器	9-23
0x074	PCIAHB_ADDR_PF	PCI-AHB 窗口预取范围地址控制寄存器	9-23
0x078	PCIAHB_TIMER	PCI-AHB 读操作超时寄存器	9-24
0x07C	AHBPCI_TIMER	AHB-PCI 读操作超时寄存器	9-24
0x080	PCI_CONTROL	PCI 控制寄存器	9-25
0x084	PCI_DV	PCI Vendor 和 Vendor ID 寄存器	9-26
0x088	PCI_SUB	PCI Subsystem 设备和 Subsystem Vendor ID 寄存器	9-26
0x08C	PCI_CREV	PCI Class Code 和 Revision ID 寄存器	9-26
0x090	PCI_BROKEN	PCI 仲裁 Master 死锁状态寄存器	9-27
0x094	PCIAHB_SIZE_NP	PCI-AHB 窗口非预取空间范围寄存器	9-28
0x098	PCIAHB_SIZE_PF	PCI-AHB 窗口预取空间范围寄存器	9-28
0x09C ~ 0x3FC	RESERVED	保留	-



PCI 配置空间头标区寄存器

表9-4 PCI 配置空间寄存器概览

偏移地址	名称	描述	页码
0x000	Vendor ID Device ID	供应商识别字段和设备识别字段寄存器	9-28
0x004	Command and Status	设备命令和状态寄存器	9-29
0x008	Revision ID and Class Code	版本识别字段和分类代码字段寄存器	9-30
0x00C	Cacheline Size Master Latency Timer Head Typer	Cache 行容量、PCI Master Latency Time、头标类型寄存器	9-30
0x010	Base Address 0 (BAR0)	基地址寄存器 0 (保留)	-
0x014	Base Address 1 (BAR1)	基地址寄存器 1 (保留)	-
0x018	Base Address 2 (BAR2)	基地址寄存器 2 (保留)	-
0x01C	Base Address 3 (BAR3)	基地址寄存器 3	9-31
0x020	Base Address 4 (BAR4)	基地址寄存器 4	9-31
0x024	Base Address 5 (BAR5)	基地址寄存器 5	9-32
0x028	Cardbus CIS 指针	Cardbus CIS 指针寄存器	9-32
0x02C	Subsystem Vendor ID Subsystem ID	子系统厂商 ID 和子系统 ID 寄存器	9-33
0x030	RESERVED	保留	-
0x034	Capabilities Pointer	能力指针寄存器	9-33
0x038	RESERVED	保留	-
0x03C	Interrupt Line Interrupt Pin Min_Gnt Max_Lat	中断寄存器	9-34



偏移地址	名称	描述	页码
0x040 ~ 0x07C	RESERVED	保留	-
0x080	PCI_IMASK	中断屏蔽寄存器 (PCI Device Only)	9-34
0x084	PCI_ISTATUS	中断状态寄存器 (PCI Device Only)	9-35
0x088	PCI_ICMD	中断命令寄存器 (PCI Device Only)	9-36
0x08C	PCI_VERSION	PCI 模块版本寄存器	9-37
0x090 ~ 0x0FC	RESERVED	保留	-

注: [PCI_IMASK](#)、[PCI_ISTATUS](#)、[PCI_ICMD](#) 寄存器只有当 Hi3511/Hi3512 为 PCI Device 模式时才会使用。

9.7 寄存器描述

9.7.1 AHB 侧寄存器描述

WDMA_PCI_ADDR

WDMA_PCI_ADDR 为 DMA 写操作目的地址寄存器。

Offset Address		Register Name																Total Reset Value																								
0x00		WDMA_PCI_ADDR																0x0000_0000																								
Bit	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0																																		
Name																																										
Reset																																										
Bits	Access	Name		Description																																						
[31:0]	RW	wdma pci addr		DMA 写操作目的地址。																																						

WDMA_AHB_ADDR

WDMA_AHB_ADDR 为 DMA 写操作源地址寄存器。



Offset Address								Register Name								Total Reset Value																
0x04				WDMA_AHB_ADDR								0x0000_0000																				
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	wdma ahb addr																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name			Description																											
[31:0]	RW	wdma ahb addr			DMA 写操作源地址。																											

WDMA CONTROL

WDMA_CONTROL 为 DMA 写操作 transfer size 和控制命令寄存器。

Offset Address								Register Name								Total Reset Value																										
0x08								WDMA_CONTROL								0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
Name	write DMA transfer size																dma command				pci interrupt enable				stop dma		start dma															
Reset	0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0																	
Bits	Access	Name		Description																																						
[31:8]	RW	write dmatransfer size		DMA 写操作 transfer size, 以 byte 为单位, 最大 16MB。																																						
[7:4]	RW	dma command		DMA 写操作命令。具体描述请参考 表 9-2 。																																						
[3]	RW	pci interrupt enable		PCI 中断使能。 0: PCI 中断不使能。 1: PCI 中断使能。 说明 在发起 DMA 操作时若将此位置为 1, 在 DMA 操作结束后将会产生 PCI 中断通过 PCI_INTA 上报给 PCI Host。因此, 只有当 Hi3511/Hi3512 为 PCI Device 模式时才会使用这一比特, 当 Hi3511/Hi3512 为 PCI Host 时此位固定为 0。																																						
[2]	-	reserved		保留。																																						



[1]	RW	stop dma	DMA 写操作中止命令。 0: DMA 操作继续。 1: DMA 操作中止。
[0]	RW	start dma	DMA 写操作开始命令。 0: DMA 操作已经结束。 1: 开始 DMA 操作。

RDMA_PCI_ADDR

RDMA_PCI_ADDR 为 DMA 读操作目的地址寄存器。

Offset Address																Register Name								Total Reset Value								
0x20																RDMA_PCI_ADDR								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rdma pci addr																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name			Description																											
[31:0]	RW	rdma pci addr			DMA 读操作目的地址。																											

RDMA_AHB_ADDR

RDMA_AHB_ADDR 为 DMA 读操作源地址寄存器。

Offset Address																Register Name								Total Reset Value								
0x24																RDMA_AHB_ADDR								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rdma ahb addr																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name			Description																											
[31:0]	RW	rdma ahb addr			DMA 读操作源地址。																											

RDMA_CONTROL

RDMA_CONTROL 为 DMA 读操作 transfer size 和控制命令寄存器。



Offset Address												Register Name												Total Reset Value																		
0x28												RDMA_CONTROL												0x0000_0000																		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	start dma	stop dma	reserved	pci int.	dma command					
Name	read DMA transfer size																																									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name	Description																																							
[31:8]	RW	read dma transfer size	DMA 读操作 transfer size, 以 byte 为单位, 最大 16MB。																																							
[7:4]	RW	dma command	DMA 读操作命令。具体描述请参考表 9-2。																																							
[3]	RW	pci int	PCI 中断使能。 0: PCI 中断不使能。 1: PCI 中断使能。 说明 在发起 DMA 操作时若将此位置为 1, 在 DMA 操作结束后将会产生 PCI 中断通过 PCI_INTA 上报给 PCI Host。因此, 只有当 Hi3511/Hi3512 为 PCI Device 模式时才会使用这一比特, 当 Hi3511/Hi3512 为 PCI Host 时此位固定为 0。																																							
[2:1]	-	reserved	保留。																																							
[1]	RW	stop dma	DMA 读操作中止命令。 0: DMA 操作继续。 1: DMA 操作中止。																																							
[0]	RW	sart dma	DMA 读操作开始命令。 0: DMA 操作已经结束。 1: 开始 DMA 操作。																																							

CPU_IMASK

CPU_IMASK 为中断屏蔽寄存器。





[10]	RW	imask_rdma_parity	DMA 读操作 parity 中断屏蔽。 0: 屏蔽 DMA 读操作 parity 中断。 1: 打开 DMA 读操作 parity 中断。
[9]	RW	imask_rdma_abort	DMA 读操作 abort 中断屏蔽。 0: 屏蔽 DMA 读操作 abort 中断。 1: 打开 DMA 读操作 abort 中断。
[8]	RW	imask_rdma_end	DMA 读操作 end 中断屏蔽。 0: 屏蔽 DMA 读操作 end 中断。 1: 打开 DMA 读操作 end 中断。
[7]	-	reserved	保留。
[6]	RW	imask_pciahb_post	PCI-AHB 窗口 post 中断屏蔽。 0: 屏蔽 AHB-PCI 窗口 post 中断。 1: 打开 AHB-PCI 窗口 post 中断。
[5]	RW	imask_pciahb_fetch	PCI-AHB 窗口 fetch 中断屏蔽。 0: 屏蔽 AHB-PCI 窗口 fetch 中断。 1: 打开 AHB-PCI 窗口 fetch 中断。
[4]	RW	imask_pciahb_discard	PCI-AHB 窗口 discard 中断屏蔽。 0: 屏蔽 AHB-PCI 窗口 discard 中断。 1: 打开 AHB-PCI 窗口 discard 中断。
[3]	RW	imask_wdma_ahb_error	DMA 写操作 ahb_error 中断屏蔽。 0: 屏蔽 DMA 写操作 ahb_error 中断。 1: 打开 DMA 写操作 ahb_error 中断。
[2]	RW	imask_wdma_parity	DMA 写操作 parity 中断屏蔽。 0: 屏蔽 DMA 写操作 parity 中断。 1: 打开 DMA 写操作 parity 中断。
[1]	RW	imask_wdma_abort	DMA 读操作 abort 中断屏蔽。 0: 屏蔽 DMA 读操作 abort 中断。 1: 打开 DMA 读操作 abort 中断。
[0]	RW	imask_wdma_end	DMA 写操作 end 中断屏蔽。 0: 屏蔽 DMA 写操作 end 中断。 1: 打开 DMA 写操作 end 中断。



CPU_ISTATUS

CPU_ISTATUS 为中断状态寄存器。此寄存器中所有的中断状态位为 1 时表明已经发生此中断，中断状态位为 0 时表明没有发生此中断；当向对应的比特位写入 1 时将清除此中断。



[12]	W1C	istatus_ahbpci_discard	AHB-PCI 窗口 discard 中断状态。 0: 无中断; 1: 有中断。
[11]	W1C	istatus_rdma_ahb_error	DMA 读操作 ahb_error 中断状态。 0: 无中断; 1: 有中断。
[10]	W1C	istatus_rdma_parity	DMA 读操作 parity 中断状态。 0: 无中断; 1: 有中断。
[9]	W1C	istatus_rdma_abort	DMA 读操作 abort 中断状态。 0: 无中断; 1: 有中断。
[8]	W1C	istatus_rdma_end	DMA 读操作 end 中断状态。 0: 无中断; 1: 有中断。
[7]	-	reserved	保留。
[6]	W1C	istatus_pciahb_post	PCI-AHB 窗口 post 中断状态。 0: 无中断; 1: 有中断。
[5]	W1C	istatus_pciahb_fetch	PCI-AHB 窗口 fetch 中断状态。 0: 无中断; 1: 有中断。
[4]	W1C	istatus_pciahb_discard	PCI-AHB 窗口 discard 中断状态。 0: 无中断; 1: 有中断。
[3]	W1C	istatus_wdma_ahb_error	DMA 写操作 ahb_error 中断状态。 0: 无中断; 1: 有中断。
[2]	W1C	istatus_wdma_parity	DMA 写操作 parity 中断状态。 0: 无中断; 1: 有中断。
[1]	W1C	istatus_wdma_abort	DMA 写操作 abort 中断状态。 0: 无中断; 1: 有中断。



[0]	W1C	istatus_wdma_end	DMA 写操作 end 中断状态。 0: 无中断; 1: 有中断。
-----	-----	------------------	---

CPU_ICMD

CPU_ICMD 为中断命令寄存器。

Offset Address								Register Name								Total Reset Value																
0x48								CPU_ICMD								0x0000_0000																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved				cpu_doorbell				reserved																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access		Name				Description																									
[31:24]	-		reserved				保留。																									
[23:20]	WO		cpu_doorbell				CPU doorbell 中断命令。CPU doorbell 中断是提供给用户自定义的中断资源，用户通过配置 CPU_ICMD[23:20]的任何一 bit 或者几 bit 可发出通过 PCI_INTA 的中断到达 PCI Host 设备。CPU doorbell 中断仅用于 Hi3511/Hi3512 为 PCI Device 模式下，且此中断命令高电平有效。																									
[19:0]	-		reserved				保留。																									

CPU_VERSION

CPU_VERSION 为设备版本寄存器。



说明

- PCI Device 模式时, 复位值为 0x0000_1380。
 - PCI Host 模式时, 复位值为 0x0000_2380。



[15:12]	RO	device type	设备类型。 0x1: PCI Device。 0x2: PCI Host。
[11:0]	RO	device version	设备版本。

PCIAHB_ADDR_NP

PCIAHB_ADDR_NP 为 PCI-AHB 窗口非预取范围地址控制寄存器。

	Offset Address																Register Name								Total Reset Value							
	0x70																PCIAHB_ADDR_NP								0xB000_0001							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	pciahb_addr_np																								reserved				window_en			
Reset	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		
Bits	Access	Name	Description																													
[31:8]	RW	pciahb_addr_np	PCI-AHB 窗口非预取地址 AHB 侧基地址。																													
[7:1]	-	reserved	保留。																													
[0]	RW	window_en	PCI-AHB 窗口非预取使能。 0: PCI-AHB 窗口非预取不使能。 1: PCI-AHB 窗口非预取使能。																													



说明

当 Hi3511/Hi3512 为 PCI Host 模式时, PCIAHB_ADDR_NP[pciahb_addr_np]和 PCIAHB_ADDR_PF[pciahb_addr_pf]的值和 PCI 配置空间中 [Base Address 3](#) 和 [Base Address 4](#) 的值相同, 此时 PCI Device 通过 PCI 总线访问 Hi3511/Hi3512 时就是依据这两个寄存器的值来发出目标地址的。

PCIAHB_ADDR_PF

PCIAHB_ADDR_PF 为 PCI-AHB 窗口预取范围地址控制寄存器。

Offset Address												Register Name												Total Reset Value											
0x74												PCIAHB_ADDR_PF												0xB000_0001											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	pciahb_addr_pf												reserved												window_en										
Reset	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1						
Bits	Access	Name		Description																															
[31:8]	RW	pciahb_addr_pf		PCI-AHB window 预取窗口地址 AHB 侧基地址。																															
[7:1]	-	reserved		保留。																															
[0]	RW	window_en		PCI-AHB 窗口预取使能。 0: PCI-AHB 窗口预取不使能。 1: PCI-AHB 窗口预取使能。																															

PCIAHB_TIMER

PCIAHB_TIMER 为 PCI-AHB 读操作超时寄存器。

Offset Address												Register Name												Total Reset Value											
0x78												PCIAHB_TIMER												0x0000_0100											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												pciahb_discard																						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0					
Bits	Access	Name		Description																															
[31:12]	-	reserved		保留。																															
[11:0]	RW	pciahb_discard		PCI-AHB window 读操作超时计数值。																															

AHBPCI_TIMER

AHBPCI_TIMER 为 AHB-PCI 读操作超时寄存器。



Offset Address																Register Name								Total Reset Value								
Bit	0x7C															AHBPCI_TIMER								0x0000_0100								
Name	reserved															ahbpci_discard																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
Bits	Access	Name		Description																												
[31:12]	-	reserved		保留。																												
[11:0]	RW	ahbpci_discard		AHB-PCI window 读操作超时计数值。																												

PCI_CONTROL

PCI_CONTROL 为 PCI 控制寄存器。



说明

- PCI Device 模式时, 复位值为 0x0000_0001。
- PCI Host 模式时, 复位值为 0x0000_0007。

Offset Address																Register Name								Total Reset Value								
Bit	0x80															PCI_CONTROL								-								
Name	reserved															master_en								memory_en								
Reset	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	? ? ? ?	
Bits	Access	Name		Description																												
[31:3]	-	reserved		保留。																												
[2]	RW	master_en		PCI Master 使能。 0: PCI Master 不使能。 1: PCI Master 使能。																												
[1]	RW	memory_en		Memory 空间访问使能。 0: Memory 访问不使能。 1: Memory 访问使能。																												
[0]	RO	pci_ready		PCI 准备好。 此位在 PCI 复位的过程被清 0, 然后再过 2^{25} 个 PCI 时钟周期置位。CPU 必须等到这个比特被置位才能开始对 PCI 总线进行配置访问。																												



说明

- 当 Hi3511/Hi3512 为 PCI Host 模式时, PCI_CONTROL[master_en]、PCI_CONTROL[memory_en]和 PCI_CONTROL[pci_ready]在 PCI 复位结束后自动置 1。
 - 当 Hi3511/Hi3512 为 PCI Device 模式时, PCI_CONTROL[master_en]和 PCI_CONTROL[memory_en]分别与 PCI 配置空间中命令状态寄存器 COMMAND_STATUS[1] 和 COMMAND_STATUS[0]的作用相同, 此时 PCI_CONTROL[master_en]和 PCI_CONTROL[memory_en]的配置是由 PCI Host 来完成。

PCI_DV

PCI DV 为 PCI Vendor 和 Vendor ID 寄存器。

Offset Address								Register Name								Total Reset Value																		
0x84				PCI_DV								0x3511_19E5																						
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	device ID												vendor ID																					
Reset	0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1
Bits	Access	Name			Description																													
[31:16]	RO	device ID			设备 ID。																													
[16:15]	RO	vendor ID			厂商 ID。																													

PCI_SUB

PCI_SUB 为 PCI Subsystem 设备和 Subsystem Vendor ID 寄存器。

Offset Address								Register Name								Total Reset Value																
0x88								PCI_SUB								0x3511_19E5																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	subsystem ID												subsystem vendor ID																			
Reset	0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	0	0	1	0	
Bits	Access	Name		Description																												
[31:16]	RO	subsystem ID		子系统 ID。																												
[16:15]	RO	subsystem vendor ID		子系统生产商 ID。																												

PCI CREV

PCI CREV 为 PCI 设备分类类别码和 Revision ID 寄存器。



Offset Address								Register Name								Total Reset Value																
0x8C				PCI_CREV								0x0400_0010																				
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	class_code																revision															
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		
Bits	Access	Name			Description																											
[31:8]	RO	class_code			设备分类类别码。																											
[7:0]	RO	revision			Revision ID。																											

PCI_BROKEN

PCI_BROKEN 为 PCI 仲裁 Master 死锁状态寄存器。可通过对相应的比特位写入 1 清除 PCI Master broken 标志。

Offset Address								Register Name								Total Reset Value																							
0x90								PCI_BROKEN								0x0000_0000																							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved																broken_status																						
Reset	0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0																		
Bits	Access	Name			Description																																		
[31:8]	-	reserved			保留。																																		
[7]	-	broken_status			PCI Master broken 标志，该 bit 位对应于 PCI Host 本身的 PCI Master 的仲裁状态。 0: PCI Master OK。 1: PCI Master broken。																																		
[6]	W1C	broken_status			PCI Master broken 标志，该 bit 位对应于 PCI_REQ[0]和 PCI_GNT[0]所连接 PCI Device 的 Master 的仲裁状态。 0: PCI Master OK。 1: PCI Master broken。																																		
[5:4]	-	reserved			保留。																																		
[3:0]	W1C	broken_status			PCI Master broken 标志，PCI_BROKEN[3:0]分别对应于 bridge_req[7:0]和 bridge_gnt[7:0]中每一对点对点信号所连接 PCI Device 的 Master 的仲裁状态。 0: PCI Master OK。 1: PCI Master broken。																																		



注：PCI 总线上的 PCI Master 在申请到 PCI 总线后 16 个 PCI 时钟周期内未发出任何总线操作，此时 PCI Host 会把 PCI_BROKEN 寄存器中该 PCI Master 对应比特置 1，此后，PCI Master 如果再申请 PCI 总线，PCI Host 的仲裁器将屏蔽该 Master 的申请，直到 CPU 将其对应的 broken status 标志清除为止。

PCIAHB_SIZ_NP

PCIAHB SIZ NP 为 PCI-AHB 窗口非预取空间范围寄存器。

PCIAHB_SIZ_PF

PCIAHB SIZ PF 为 PCI-AHB 窗口预取空间范围寄存器。

Offset Address								Register Name								Total Reset Value																
0x98								PCIAHB_SIZ_PF								0xFF00_0000																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	pciahb_siz_pf																reserved															
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access		Name				Description																									
[31:8]	RW		pciahb_siz_pf				PCI-AHB 窗口预取空间范围。																									
[7:0]	-		reserved				保留。																									

9.7.2 PCI 配置空间寄存器描述

Vendor ID and Device ID

Vendor ID and Device ID 为供应商识别字段和设备识别字段寄存器。



Command and Status

Command and Status 为设备命令寄存器和状态寄存器。



[19]	RO	int_status	中断状态。
[18:11]	-	reserved	保留。
[10]	RW	int_en	中断使能。
[9]	RO	fast_btob	高速背对背传输允许。
[8]	RW	serr_en	SERR 使能。
[7]	-	reserved	保留。
[6]	RW	perr_resp	奇偶错误响应。
[5]	RO	VGA_en	VGA 调色板控制。
[4]	RO	mem_w_i_en	存储器写并无效使能。
[3]	RO	special_cyc	特殊周期控制。
[2]	RW	master_en	总线主设备使能。
[1]	RW	memory_en	存储器空间使能。
[0]	RW	I/O_en	I/O 空间使能。

Revision ID and Class Code

Revision ID and Class Code 为版本识别字段和分类代码字段寄存器。

Offset Address												Register Name												Total Reset Value												
0x08												Revision ID&Class code												0x0400_0010												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Name	class code																								revision ID											
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0				
Bits	Access	Name		Description																																
[31:8]	RO	class code		设备分类类别码。																																
[7:0]	RO	revision		revision 版本号。																																

Cacheline Size and Master Latency Timer

Cacheline Size and Master Latency Timer 为 Cache 行容量和 PCI Master Latency Time 寄存器。



Offset Address																Register Name								Total Reset Value								
0x0C																Cacheline Size and Latency Timer								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	bist								head type								master latency timer				reserved	cacheline size										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:24]	-	bist		保留。																												
[23:16]	RO	head type		头标类型。																												
[15:8]	RW	master latency timer		PCI master 延时寄存器，其中最低两位为只读位，值为 0b00，表明 latency timer 的时钟颗粒度为 8 个 PCI 时钟周期。																												
[7:0]	RO	cacheline size		cacheline 大小。 说明 Hi3511/Hi3512 PCI 接口并未实现对 Memory Cacheline 的操作，因此，此处的 cacheline size 固定为 0x00。																												

Base Address 3

Base Address 3 为基地址寄存器 3。

Offset Address																Register Name								Total Reset Value								
0x1C																Base Address 3								0x0000_0000								
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	base_address3																reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name		Description																												
[31:8]	RW	base_address3		基地址寄存器 3。																												
[7:0]	-	reserved		保留。																												

Base Address 4

Base Address 4 为基地址寄存器 4。

	Offset Address																Register Name				Total Reset Value														
	0x20																Base Address 4				0x0000_0008														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	base address 4																reserved				prefet_flag				reserved										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0					
Bits	Access	Name		Description																															
[31:0]	RW	base address 4		基地址寄存器 4。																															
[7:4]	-	reserved		保留。																															
[3]	RO	prefet_flag		预取空间标志。																															
[2:0]	-	reserved		保留。																															

Base Address 5

Base Address 5 为基地址寄存器 5。

	Offset Address																Register Name				Total Reset Value														
	0x24																Base Address 5				0x0000_0000														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	base address 5																prefet_flag				reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		
Bits	Access	Name		Description																															
[31:4]	RW	base address 5		基地址寄存器 5。																															
[3]	RO	prefet_flag		预取空间标志。																															
[2:0]	-	reserved		保留。																															

Cardbus CIS Pointer

Cardbus CIS Pointer 为 Cardbus CIS 指针寄存器。



Offset Address																Register Name	Total Reset Value															
0x28																Cardbus CIS Pointer	0x0000_0000															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	cardbus_cis_point																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	cardbus_cis_point	Description																												
[31:0]	RO	cardbus_cis_point		Cardbus CIS 指针寄存器。																												

Subsystem Vendor ID and Subsystem ID

Subsystem Vendor ID and Subsystem ID 为子系统厂商 ID 和子系统供应商 ID 寄存器。

Offset Address																Register Name	Total Reset Value															
0x2C																Subsystem Vendor ID and Subsystem ID	0x3511_19E5															
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	subsystem ID																subsystem vendor ID															
Reset	0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1
Bits	Access	Name	subsystem ID	Description																												
[31:16]	RO	subsystem ID		子系统 ID。																												
[16:15]	RO	subsystem vendor ID	subsystem vendor ID	子系统供应商 ID。																												

Capabilities Pointer

Capabilities Pointer 为能力指针寄存器。



Hi3511/Hi3512 PCI 接口部分并未实现 PCI 的扩展能力，此寄存器默认值为 0x0000_0078。



Offset Address			Register Name																Total Reset Value													
0x34			Capabilities Pointer																0x0000_0078													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																capabilities pointer															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
Bits	Access	Name		Description																												
[31:16]	-	reserved		保留。																												
[15:0]	RO	capabilities pointer		能力指针寄存器。																												

Interrupt Line Interrupt Pin MIN_GNT MAX_LAT

Interrupt Line Interrupt Pin MIN_GNT MAX_LAT 为中断寄存器。

Offset Address			Register Name																Total Reset Value													
0x3C			Interrupt Line & Interrupt Pin & MIN_GNT & MAX_LAT																0x0000_0100													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	max_lat																min_lat				interrupt pin				interrupt line							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name		Description																												
[31:24]	RO	max_lat		MAX latency 寄存器。																												
[23:16]	RO	min_gnt		MIN grant 寄存器。																												
[15:8]	RO	interrupt in		中断引脚寄存器。																												
[7:0]	RW	interrupt line		中断线寄存器。																												

PCI_IMASK

PCI_IMASK 为中断屏蔽寄存器 (PCI Device Only)。



说明

PCI_IMASK、PCI_ISTATUS、PCI_ICMD 这三个寄存器只有当 Hi3511/Hi3512 为 PCI Device 模式时才会使用。此时如果 Hi3511/Hi3512 触发了 PCI 中断，那么此时中断就会通过 PCI_INTA 发送到 PCI Host 设备。



Offset Address										Register Name										Total Reset Value														
0x80										PCI_IMASK										0x0000_0000														
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved										imask_cpu_doorbell										reserved										rdma_end	reserved		wdma_end
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																														
[31:24]	-	reserved		保留。																														
[23:20]	RW	imask_cpu_doorbell		CPU Doorbell 中断屏蔽。 0: 屏蔽 CPU Doorbell 中断。 1: 打开 CPU Doorbell 中断。																														
[19:9]	-	reserved		保留。																														
[8]	RW	imask_rdma_end		DMA 读操作结束中断屏蔽。 0: 屏蔽 DMA 读操作结束中断。 1: 打开 DMA 读操作结束中断。																														
[7:1]	-	reserved		保留。																														
[0]	RW	imask_wdma_end		DMA 写操作结束中断屏蔽。 0: 屏蔽 DMA 写操作结束中断。 1: 打开 DMA 写操作结束中断。																														

PCI_ISTATUS

PCI_ISTATUS 为中断状态寄存器 (PCI Device Only)。此寄存器中所有的中断状态位为 1 时表明已经发生此中断，中断状态位为 0 时表明没有发生此中断；当将此位写入 1 时将清除此中断。



Offset Address								Register Name								Total Reset Value																
0x84								PCI_ISTATUS								0x0000_0000																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved								istatus_cpu_doorbell								reserved								rdma_end		reserved		wdma_end			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description																												
[31:24]	-	reserved		保留。																												
[23:20]	W1C	cpu_doorbell		CPU Doorbell 中断状态。当用户配置了 CPU_ICMD[23:20]的 CPU_dooebell 中断后，此时 Hi3511/Hi3512 会通过 PCI_INTA 发出 doorbell 中断，同时所配置的中断状态会反映在 PCI_ISTATUS[23:20]的 4bit 上。 0: 无中断； 1: 有中断。																												
[19:9]	-	reserved		保留。																												
[8]	W1C	rdma_end		DMA 读操作结束中断状态。 0: 无中断； 1: 有中断。																												
[7:1]	-	reserved		保留																												
[0]	W1C	wdma_end		DMA 写操作结束中断状态。 0: 无中断； 1: 有中断。																												

PCI_ICMD

PCI_ICMD 为中断命令寄存器 (PCI Device Only)。



Offset Address												Register Name												Total Reset Value											
0x88												PCI_ICMD												0x0000_0000											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												icmd_cpu_doorbell	reserved																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
Bits	Access	Name			Description																														
[31:20]	-	reserved			保留。																														
[19:16]	RW	icmd_pci_doorbell			PCI Doorbell Interrupt 命令。PCI doorbell 中断是提供给用户自定义的可通过 PCI 总线发起的中断资源。此中断由 PCI Host 设备通过配置 PCI_ICMD[19:16]的任何 bit 位来发起中断，当配置了上述 bit 位之后，Hi3511/Hi3512 会产生上报 ARM CPU 的中断。PCI Doorbell Interrupt 命令为高电平有效。 PCI Doorbell Interrupt 仅用于 Hi3511/Hi3512 作为 PCI Device 模式时。																														
[15:0]	-	reserved			保留。																														

PCI_VERSION

PCI_VERSION 为 PCI 模块版本寄存器。



说明

- PCI Device 模式时，复位值为 0x0000_0138。
- PCI Host 模式时，复位值为 0x0000_0238。

Offset Address												Register Name												Total Reset Value											
0x4C												CPU_VERSION												-											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												device type	device version																					
Reset	?	?	?	?	?	?	?	?	?	?	?	?		?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?					
Bits	Access	Name			Description																														
[31:16]	-	reserved			保留。																														
[15:12]	RO	device type			设备类型。																														
[11:0]	RO	device version			设备版本。																														



10 USB

关于本章

本章描述内容如下表所示。

标题	内容
10.1 USB 1.1 HOST	介绍 USB 1.1 HOST 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。
10.2 USB 2.0 OTG	介绍 USB 2.0 OTG 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。



10.1 USB 1.1 HOST

10.1.1 概述

Hi3511/Hi3512 提供标准的 USB 1.1 主机控制器，完全支持 OHCI (Open Host Controller Interface) 和 USB 1.1 协议，控制器的大部分硬件逻辑主要除完成对传输的控制和处理，对数据包的解析和打包，以及对 USB 传输信号的编码和解码外，还为驱动程序提供中断向量等接口。

10.1.2 特点

- 完全兼容 USB Spec.1.1。
- 支持与低速、全速和高速设备相连。
- 完全符合 OHCI (Open Host Controller Interface) Rev1.0 规范。
- roothub 支持 1 个下行端口。
- 低功耗的解决方案。
- 可支持 Control Transfer, Interrupt Transfer, Bulk Transfer, Isochronous Transfer 4 种基本数据传送类型。
- 可以通过连接 USB Hub 连接最多 127 个 Device。

10.1.3 信号描述

USB 1.1 HOST 接口信号如表 10-1 所示。

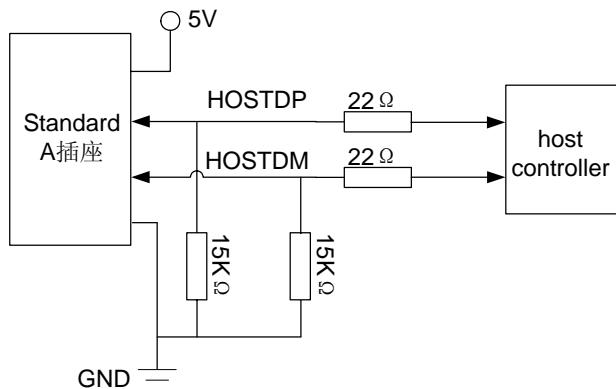
表10-1 USB 1.1 HOST 接口信号描述

信号名称	方向	描述	对应管脚
USBHOST_DM	I/O	USB 1.1 HOST 端口差分数据总线 D-	USBDM
USBHOST_DP	I/O	USB 1.1 HOST 端口差分数据总线 D+	USBDP

USB 1.1 HOST 参考设计图如图 10-1 所示。



图10-1 USB 1.1 HOST 参考设计图



在 DP、DM 数据线上需要匹配电阻。

作为主机，DP、DM 需接 $15K\Omega$ 大小的下拉电阻，如图 10-1 所示。

10.1.4 功能描述

应用框图

USB 1.1 HOST 按照 OHCI 协议进行传输的步骤如下：

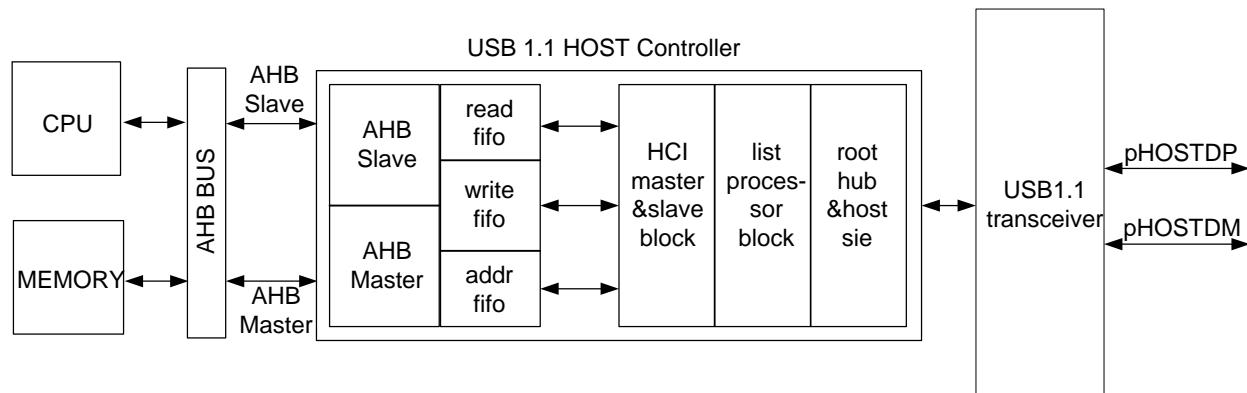
步骤 1 软件建立传输链表，配置 host 进入 OPERATIONAL 状态并使能端口。

步骤 2 host 硬件取链表节点并解析，根据节点的指示进行相应的 USB 传输。

----结束

USB 1.1 HOST 应用框图如图 10-2 所示。

图10-2 USB 1.1 HOST 应用框图





工作原理

USB 1.1 HOST 支持以下 4 种标准的传输方式：

- **控制传输**

主要用于 USB 主机与 USB 设备端点 0 之间的传输，某些特定型号的 USB 设备的控制传输可能用到其它的端点。控制传输是双向传输，数据量通常较小，可以传输 8 字节、16 字节、32 字节和 64 字节的数据，依赖于设备和传输速度。

- **批量传输**

主要用于没有带宽和间隔时间要求的情况下发送和接收大量的数据，这种类型的设备适合于传输非常慢和大量被延迟的传输，可以等到所有其它类型的数据传送完成之后再传送和接收数据。它的特点是以错误检测和重传的方式保证 USB 主机与设备的数据被无差错地发送。

- **同步传输**

主要用于时间严格并具有较强容错性的流数据传输，或者用于要求恒定的数据传输率的即时应用中。同步传输提供了确定的带宽和间隔时间。

- **中断传输**

中断传输类型用于支持那些不需要经常发送或者接收数据、但是服务周期有限的设备。中断管道是一个流管道，因此一般都是单向的。端点描述符能识别出给定中断管道的通信流是流入还是流出主机。

10.1.5 工作方式

时钟门控

如果需开启控制器工作时钟，则向 SC_PEREN[usbclken]写 1。如果关闭工作时钟，则向 SC_PERDIS[usbclkdis]写 1。

时钟配置

在使用 USB 1.1 控制器前，需为其配置合适的工作时钟频率。USB 控制器的工作时钟使用系统 PLL 的分频时钟，PLL 输出时钟与控制器工作时钟频率关系为：

$$F_{USB48} = F_{PLL} / n$$

$$F_{USB12} = F_{PLL} / (4n)$$

其中：分频因子 n 为 11，由于必须是较为精确的 12MHz、48MHz 时钟，所以对于有些情况下 PLL 分频得到的时钟不能满足要求时则 USB 1.1 控制器不可用。

12MHz 时钟提供控制器内部 USB 侧的操作时钟，48MHz 时钟是提供接收数据时的恢复时钟。

USB 1.1 HOST 具体的时钟配置可参见系统控制器的相关描述。

软复位

USB 1.1 HOST 默认一直处于复位状态，要启动 USB 1.1 HOST 进入工作状态，必须配置系统控制器的 SC_PERCTRL0[host_srst]为 1 撤消复位。



10.1.6 寄存器概览

USB 1.1 HOST 寄存器概览如表 10-2 所示。

表10-2 USB 1.1 HOST 寄存器概览（基址是 0xA000_0000）

偏移地址	名称	描述	页码
0x000	HcRevision	控制器版本号寄存器	10-6
0x004	HcControl	控制器配置寄存器	10-6
0x008	HcCommandStatus	控制器命令状态寄存器	10-7
0x00C	HcInterruptStatus	控制器中断状态寄存器	10-8
0x010	HcInterruptEnable	控制器中断使能寄存器	10-9
0x014	HcInterruptDisable	控制器中断禁止寄存器	10-10
0x018	HcHCCA	HCCA (Host Controller Communication Area) 首地址	10-11
0x01c	HcPeriodCurrentED	当前实时或中断传输 ED 的物理地址	10-12
0x020	HcControlHeadED	ED 控制链的物理首地址	10-12
0x024	HcControlCurrentED	当前控制 ED 物理地址	10-13
0x028	HcBulkHeadED	ED 批量链的物理首地址	10-13
0x02c	HcBulkCurrentED	当前批量 ED 物理地址	10-14
0x030	HcDoneHead	传输结束 TD 链的首地址	10-14
0x034	HcFmInterval	毫秒帧的时间间隔	10-15
0x038	HcFmRemaining	毫秒帧的剩余时间	10-15
0x03c	HcFmNumber	传输的毫秒帧数目	10-16
0x040	HcPeriodicStart	开始处理周期 ED 链的时刻	10-16
0x044	HcLSThreshold	与 HcFmRemaining 一起指示是否在 EOF (End Of Frame) 之前一个最大 8 字节的 LS 包	10-17
0x048	HcRhDescriptorA	根集线器描述符寄存器 A	10-17
0x04c	HcRhDescriptorB	根集线器描述符寄存器 B	10-18
0x050	HcRhStatus	根集线器状态寄存器	10-19
0x054	HcRhPortStatus[1]	根集线器的端口 0 状态寄存器	10-20



10.1.7 寄存器描述



寄存器描述的表格中 HC 表示主机控制器硬件逻辑；HCD 表示主机控制器驱动，即软件驱动。

HcRevision

HcRevision 为控制器版本号寄存器。

Offset Address			Register Name			Total Reset Value																										
0x000			HcRevision			0x0000_0010																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															rev																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		
Bits	Access	Name	Description																													
[31:8]	-	reserved	保留。																													
[7:0]	HC:R HCD:R	rev	版本号。																													

HcControl

HcControl 为控制器配置寄存器。

Offset Address			Register Name			Total Reset Value																										
0x004			HcControl			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															rwe rwo ir hcfs bje cle ie ap cbsr																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		
Bits	Access	Name	Description																													
[31:11]	-	reserved	保留。																													
[10]	HC:R HCD:RW	rwe	控制器对远程唤醒允许使能。 0：不使能。 1：使能。																													
[9]	HC:RW HCD:RW	rwc	控制器对远程唤醒信号处理支持使能。 0：不使能。 1：使能。																													



[8]	HC:R HCD:RW	ir	0: 中断交给处理主机总线中断的机制处理。 1: 中断交给系统中断管理进程处理。
[7:6]	HC:RW HCD:RW	hdfs	控制器当前状态。 00: USBRESET。 01: USBRESUME。 10: USBOPERATIONAL。 11: USBSUSPEND。
[5]	HC:R HCD:RW	ble	控制器处理批量类型的 ED 链表允许使能。 0: 不使能。 1: 使能。
[4]	HC:R HCD:RW	cle	控制器处理控制类型的 ED 链表允许使能。 0: 不使能。 1: 使能。
[3]	HC:R HCD:RW	ie	控制器处理实时传输类型的 ED 链表允许使能。 0: 不使能。 1: 使能。
[2]	HC:R HCD:RW	ple	控制器处理周期性 ED 链表允许使能。 0: 不使能。 1: 使能。
[1:0]	HC:R HCD:RW	cbsr	控制器驱动程序对控制传输和批量传输的次数进行设置。 00: 1:1。 01: 2:1。 10: 3:1。 11: 4:1。

HcCommandStatus

HcCommandStatus 为控制器命令状态寄存器。



HcInterruptStatus

HcInterruptStatus 为控制器的中断状态寄存器。



[29:7]	HC:RW HCD:RW	reserved	保留。
[6]	HC:RW HCD:RW	rhsc	根集线器的状态有变化。
[5]	HC:RW HCD:RW	fno	HcFmNumber bit[15]从 1 变成 0 或从 0 变成 1, 表示帧号溢出。
[4]	HC:RW HCD:RW	ue	控制器检测到与 USB 无关的系统错误。
[3]	HC:RW HCD:RW	rd	控制器接收到总线设备发出的恢复信号。
[2]	HC:RW HCD:RW	sf	控制器每发出一个 sof 包产生此中断。
[1]	HC:RW HCD:RW	wdh	请求驱动程序处理 TD 的处理完成队列。当控制器将 HcDoneHead 中信息写入到 HccaDoneHead 中产生此中断。
[0]	HC:RW HCD:RW	so	控制器在一帧的时间内不能完全完成对周期性链表的数据传输处理。

HcInterrupEnable

HcInterrupEnable 为控制器中断使能寄存器。

	Offset	Address	Register Name	Total Reset Value
	0x010		HcInterrupEnable	0x0000_0000
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
Name	mie	oc	reserved	rhsc fno ue rd sf wdh so
Reset	0 0			
Bits	Access	Name	Description	
[31]	HC:R HCD:RW	mie	中断使能。 0: 控制器忽略。 1: 允许中断。	
[30]	HC:R HCD:RW	oc	驱动程序发出了改变控制器 Ownership 的请求操作。 0: 控制器忽略。 1: 允许中断。	
[29:7]	HC:R HCD:RW	reserved	保留。	



[6]	HC:R HCD:RW	rhsc	根集线器的状态有变化。 0: 控制器忽略。 1: 允许中断。
[5]	HC:R HCD:RW	fno	HcFmNumber bit[15]从 1 变成 0 或从 0 变成 1, 表示帧号溢出。 0: 控制器忽略。 1: 允许中断。
[4]	HC:R HCD:RW	ue	控制器检测到与 USB 无关的系统错误。 0: 控制器忽略。 1: 允许中断。
[3]	HC:R HCD:RW	rd	控制器接收到总线设备发出的恢复信号。 0: 控制器忽略。 1: 允许中断。
[2]	HC:R HCD:RW	sf	控制器每发出一个 sof 包产生此中断。 0: 控制器忽略。 1: 允许中断。
[1]	HC:R HCD:RW	wdh	请求驱动程序处理 TD 的处理完成队列。当控制器将 HcDoneHead 中信息写入到 HccaDoneHead 中产生此中断。 0: 控制器忽略。 1: 允许中断。
[0]	HC:R HCD:RW	so	控制器在一帧的时间内不能完全完成对周期性链表的数据传输处理。 0: 控制器忽略。 1: 允许中断。

HcInterruptDisable

HcInterruptDisable 为控制器中断禁止寄存器。

Offset Address		Register Name		Total Reset Value	
0x014		HcInterruptDisable		0x0000_0000	
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	reserved			
Name	mie oc			rhsc fno ue rd sf upw so	
Reset	0 0				
Bits	Access	Name	Description		



[31]	HC:R HCD:RW	mie	中断使能。 0: 控制器忽略。 1: 禁止中断。
[30]	HC:R HCD:RW	oc	驱动程序发出了改变控制器 Ownership 的请求操作。 0: 控制器忽略。 1: 禁止中断。
[29:7]	HC:R HCD:RW	reserved	保留。
[6]	HC:R HCD:RW	rhsc	根集线器的状态有变化。 0: 控制器忽略。 1: 禁止中断。
[5]	HC:R HCD:RW	fno	HcFmNumber bit[15]从 1 变成 0 或从 0 变成 1, 表示帧号溢出。 0: 控制器忽略。 1: 禁止中断。
[4]	HC:R HCD:RW	ue	控制器检测到与 USB 无关的系统错误。 0: 控制器忽略。 1: 禁止中断。
[3]	HC:R HCD:RW	rd	控制器接收到总线设备发出的恢复信号。 0: 控制器忽略。 1: 禁止中断。
[2]	HC:R HCD:RW	sf	控制器每发出一个 sof 包产生此中断。 0: 控制器忽略。 1: 禁止中断。
[1]	HC:R HCD:RW	wdh	请求驱动程序处理 TD 的处理完成队列。当控制器将 HcDoneHead 中信息写入到 HccaDoneHead 中产生此中断。 0: 控制器忽略。 1: 禁止中断。
[0]	HC:R HCD:RW	so	控制器在一帧的时间内不能完全完成对周期性链表的数据传输处理。 0: 控制器忽略。 1: 禁止中断。

HcHCCA

HcHCCA 为 HCCA 区域首地址寄存器。

Offset Address			Register Name			Total Reset Value																										
0x018			HcHCCA			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	hcca																								zero							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:8]	HC:R HCD:RW	hcca	HCCA 的高 24 位地址。																													
[7:0]	-	zero	固定为 0。																													

HcPeriodCurrentED

HcPeriodCurrentED 为当前实时或中断传输 ED 的物理地址。

Offset Address			Register Name			Total Reset Value																										
0x01C			HcPeriodCurrentED			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	pced																									zero						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:4]	HC:RW HCD:R	pced	当前周期性 ED 链表的首地址。																													
[3:0]	-	zero	固定为 0。																													

HcControlHeadED

HcControlHeadED 为 ED 控制链的物理首地址。



Offset Address			Register Name			Total Reset Value																										
0x020			HcControlHeadED			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ched																											zero				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:4]	HC:R HCD:RW	ched	控制 ED 链表的首地址。																													
[3:0]	-	zero	固定为 0。																													

HcControlCurrentED

HcControlCurrentED 为当前控制 ED 物理地址。

Offset Address			Register Name			Total Reset Value																										
0x024			HcControlCurrentED			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	cced																											zero				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:4]	HC:RW HCD:RW	cced	当前处理的控制 ED 的地址。																													
[3:0]	-	zero	固定为 0。																													

HcBulkHeadED

HcBulkHeadED 为 ED 批量链的物理首地址。



HcBulkCurrentED

HcBulkCurrentED 为当前批量 ED 物理地址。

HcDoneHead

HcDoneHead 为传输结束 TD 链的首地址。



HcFmInterval

HcFmInterval 为毫秒帧的时间间隔。

HcFmRemaining

HcFmRemaining 为毫秒帧的剩余时间。



Offset Address			Register Name			Total Reset Value																										
0x038			HcFmRemaining			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	fr	reserved																		fr												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31]	HC:RW HCD:R	frt	用来同步 FR 和 FI 域，驱动程序将 fit 中的值读出来设置到此域中。																													
[30:14]	-	reserved	保留。																													
[13:0]	HC:RW HCD:R	fr	当前帧空余的位数。 当减成 0 时，控制器会从 FrameInterval 中读取初值初始化此域。																													

HcFmNumber

HcFmNumber 为传输的毫秒帧数目。

Offset Address			Register Name			Total Reset Value																										
0x03c			HcFmNumber			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																		fn													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:16]	-	reserved	保留																													
[15:0]	HC:RW HCD:R	fn	总线当前的帧号的低 16 位。 每改变一次，控制器将它更新到 HCCA 中供驱动程序使用。																													

HcPeriodicStart

HcPeriodicStart 为开始处理周期 ED 链的时刻。



Offset Address			Register Name			Total Reset Value																										
0x040			HcPeriodicStart			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															ps																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																													
[31:14]	-	reserved	保留。																													
[13:0]	HC:R HCD:RW	ps	一帧中周期性传输的起始位置。 由驱动程序在控制器初始化时设置，一般设为 FrameInterval 的 10%。																													

HcLSThreshold

HcLSThreshold 为 HcLSThreshold 寄存器。

Offset Address			Register Name			Total Reset Value																										
0x044			HcLSThreshold			0x0000_0628																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved															lst																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0
Bits	Access	Name	Description																													
[31:12]	-	reserved	保留。																													
[11:0]	HC:R HCD:R	lst	低速传输 8 个字节数据包的事务处理的总长度。 当进行低速传输时，控制器比较 FmRemaining 和此域，如果 FmRemaining 大，则可以进行低速传输。																													

HcRhDescriptorA

HcRhDescriptorA 为根集线器描述符寄存器 A。

		Offset Address	Register Name	Total Reset Value														
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			0x0200_1202														
Name	potpgt	reserved	nocp	ocpm	dt	nps	psm	fr										
Reset	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0																	
Bits	Access	Name	Description															
[31:24]	HC:R HCD:RW	potpgt	对端口加电后，驱动程序在访问之前等待的时间。															
[23:13]	-	reserved	保留。															
[12]	HC:R HCD:RW	nocp	是否支持过载保护。 0: 支持。 1: 不支持。															
[11]	HC:R HCD:RW	ocpm	过载保护模式。 0: 所有端口的状态一起汇报。 1: 每个端口都能汇报过载情况。															
[10]	HC:R HCD:R	dt	设备类型。 指明集线器是否是一个复合设备，根集线器不允许是复合设备，所以此位为 0。															
[9]	HC:R HCD:RW	nps	端口是否有电源开关。 0: 有电源开关，可以控制端口供电。 1: 没有电源开关，端口一直有电。															
[8]	HC:R HCD:RW	psm	端口电源的开关模式，只有当 NPS 为 0 时有效。 0: 所有端口一起加电。 1: 端口可以各自加电。															
[7:0]	HC:R HCD:R	fr	当前帧空余的位数，当减成 0 时，控制器会从 FrameInterval 中读取初值初始化此域。															

HcRhDescriptorB

HcRhDescriptorB 为根集线器描述符寄存器 B。



Offset Address			Register Name			Total Reset Value																										
0x04c			HcRhDescriptorB			0x0000_0000																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ppcm															dr																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:16]	HC:R HCD:RW	ppcm	端口的电源开关是否被全局控制。 0: 可以。 1: 不可以。																													
[15:0]	HC:R HCD:RW	dr	连在端口上的设备是否可以断开。 0: 可以。 1: 不可以。																													

HcRhStatus

HcRhStatus 为根集线器状态寄存器。

Offset Address			Register Name			Total Reset Value																													
0x050			HcRhStatus			0x0000_0000																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved															ocic	lpsc	dfwe	reserved															oci	lps
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
Bits	Access	Name	Description																																
[31]	HC:RW HCD:RW	crwe	清远程唤醒。 写 1 清 0, 写 0 无效。																																
[30:18]	-	reserved	保留。																																
[17]	HC:RW HCD:RW	ocic	过流指示改变。 当该寄存器的 oci 位发生变化硬件对该位置 1。驱动写 1 清 0。																																
[16]	HC:RW HCD:RW	lpsc	读时： 本地电源状态改变。 根集线器不支持本地电源状态特性，因此该位总是读出为 0。 写时：																																

			设置全局供电模式。 在全局供电模式，对该位写一打开所有端口的电源。在单个端口供电模式，只能在 port power control mask 位没有设置的端口上设置 PortPowerStatus。写 0 无效。
[15]	HC:RW HCD:RW	drwe	读时： 设备远程唤醒使能状态。 该位使能 ConnectStatusChange 位作为 resume 事件，引起 USBSUSPEND 到 USBRESUME 状态的变化并产生一个 resume-detected 的中断。 0: ConnectStatusChange 不是一个远程唤醒事件。 1: ConnectStatusChange 是一个远程唤醒事件。 写时： 设置远程唤醒使能。 0: 不使能。 1: 使能。
[14:2]	-	reserved	保留。
[1]	HC:RW HCD:RW	oci	过流情况。 当全局执行记录时这一位记录过流情况。置 1 表明有一个过流情况存在，清 0 表明所有电源操作正常。如果每个端口的过流保护启动执行则该位始终为 0。
[0]	HC:RW HCD:RW	lps	读时： 本地电源状态。 根集线器不支持本地电源状态特性，故该位始终读出为 0。 写时： 清全局供电模式。 在全局供电模式下，对该位写 1 关闭所有端口电源。在单个端口供电模式下，只能在 port power control mask 位没有设置的端口上设置 PortPowerStatus。写 0 无效。

HcRhPortStatus[1]

HcRhPortStatus[1]为根集线器的端口 0 状态寄存器。





[9]	HC:RW HCD:RW	lsda	<p>读时： 低速设备连接。 这一位表明连接在这个端口上的设备的速度。置 1 表明连接的是一个低速设备，清 0 表示该端口连接的是一个全速设备，该位只在 ccs 为 1 时有效。 0: 连接的一个全速设备。 1: 连接的一个低速设备。</p> <p>写时： 清端口电源。 驱动对该位写 1 则清 0PortPowerStatus 位，写 0 无效。</p>
[8]	HC:RW HCD:RW	pps	<p>读时： 端口电源状态。 无论执行何种电源切换，该位反映端口电源状态。如果检测到过流情况就对该位清 0。 0: 端口电源关闭。 1: 端口电源打开。</p> <p>写时： 设置端口电源。 驱动对该位写 1 设置 PortPowerStatus 位。写 0 无效。 注：如果不支持电源切换则该位始终读出为 1。</p>
[7:5]	-	reserved	保留。
[4]	HC:RW HCD:RW	prs	<p>读时： 端口复位状态。 对 SetPortReset 写操作就设置这一位，说明端口复位信号已经声明。当复位结束，清除该位，同时 PortResetStatusChange 置 1。当 CurrentConnectStatus 清除时该位不可设置。 0: 端口复位信号不起作用。 1: 端口复位信号起作用。</p> <p>写时： 设置端口复位。 驱动对该位写 1 发起端口复位信号。写 0 无效。</p>
[3]	HC:RW HCD:RW	poci	<p>读时： 过流情况。 这一位只在当根集线器配置为每个端口都能记录过流时才有效。如果不支持每个端口的过流记录这一位就设置为 0。如果清除该位，该端口的所有电源情况都正常，如果置 1 则表明该端口存在过流情况。 0: 无过流情况。</p>



			1: 检测到过流情况。 写时： 清除 suspend 状态。 驱动写 1 初始化发起一个 resume 操作，写 0 无效。只有在 PortSuspendStatus 置位的情况下可以初始化一个 resume。
[2]	HC:RW HCD:RW	pss	读时： 端口挂起状态。 该位指示端口挂起或接收唤醒序列。当写 SetSuspendState 和当 PortSuspendStatusChange 在唤醒序列最后置位时清 0 SetSuspendState 时对该位置 1。当 CurrentConnectStatus 清 0 时不能置位该位。当 PortResetStatusChange 在端口复位最后置 1 或控制器处于 USBRESUME 状态时该位清 0。 0: 端口没有挂起。 1: 端口挂起。 读时： 设置端口挂起。 驱动通过对 pss 置 1 设置 PortSuspendStatus 位，写 0 无效。 如果 CurrentConnectStatus 清 0，对该位置 1 就不能设置 PortSuspendStatus 位，而是对设置 CurrentConnectStatus。这表明驱动正在试图挂起一个未连接的端口。
[1]	HC:RW HCD:RW	pes	读时： 端口使能状态。 指示端口是否使能。当检测到过流、断连事件、下电或操作总线错误的情况下根集线器会清除该位。 0: 端口禁止。 1: 端口使能。 写时： 设置端口使能。 驱动将 PortEnableStatus 写为 1，写 0 无效。如果 CurrentConnectStatus 清 0，则写 1 不能置位 PortEnableStatus，而是置位 ConnectStatusChange，这表明驱动试图使能一个未连接的端口。
[0]	HC:RW HCD:RW	ccs	读时： 目前连接状态。 0: 无设备连接。 1: 有设备连接。 写时： 清端口使能。 驱动程序对该位写一清除 PortEnableStatus 位，写 0 无作用。



10.2 USB 2.0 OTG

10.2.1 概述

USB 2.0 OTG (On The Go) 接口支持 USB 2.0 协议，同时支持 USB 2.0 OTG 补充协议。

USB 2.0 OTG 接口作为设备接口时，支持以下传输速率：

- 高速 (480Mbit/s)
- 全速 (12Mbit/s)

USB 2.0 OTG 接口作为主机接口时，支持与以下速率类型的设备相连：

- 高速 (480Mbit/s)
- 全速 (12Mbit/s)
- 低速 (1.5Mbit/s)

USB 2.0 OTG 接口包括 USB 2.0 OTG 控制器与 PHY，严格来讲，属于一种功能增强的 USB 设备，能够作为 USB 主机完成 USB 主机到 USB 设备的连接，它降低了对原来以 PC 系统为核心的 USB 拓扑结构的依赖性。

在没有数据传输的时候，USB 2.0 OTG 可以进入挂起状态，关闭 USB 2.0 OTG PHY 的时钟，可使功耗大大降低。

10.2.2 特点

USB 2.0 OTG 的主要特点如下：

- 支持配置为主机模式或外设模式。
- 支持 USB 2.0 协议，同时支持 USB 2.0 的 OTG 补充协议。
- 作主机时支持与高速、全速或低速的设备连接。
- 支持 SRP 会话请求协议和 HNP 主机协商协议。
- 支持挂起与复位。
- 支持唤醒与远程唤醒。
- 支持控制、中断、批量与同步传输。
- 支持高带宽的同步传输和中断传输。
- 支持 2 个发送端点和 3 个接收端点（除端点 0）。
- 支持软件控制连接和断开。
- 支持 CPU 数据传输方式。
- 支持内置 DMA 数据传输方式。
- 支持内置 DMA 模式 0 与 DMA 模式 1（仅用于批量传输）传输方式。
- 支持自动组包与解包功能（仅用于批量传输）。



10.2.3 信号描述

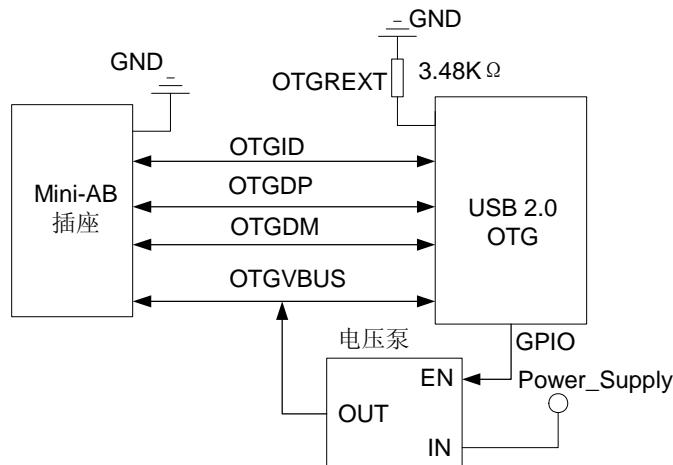
USB 2.0 OTG 接口信号如表 10-3 所示。

表10-3 USB 2.0 OTG 接口信号描述

信号名称	方向	描述	对应管脚
OTG_OTGID	I	USB 2.0 OTG ID 信号输入管脚	OTGID
OTG_DM	I/O	USB 2.0 OTG 端口差分数据总线 D-	OTGDM
OTG_DP	I/O	USB 2.0 OTG 端口差分数据总线 D+	OTGDP
OTG_VBUS	I/O	USB 2.0 OTG VBUS 电源	OTGVBUS
OTG_REXT	I/O	USB 2.0 OTG 外部电阻连接 (外接参考电阻值为 3.48K Ω±1%)	OTGREXT

USB 2.0 OTG 参考设计图如图 10-3 所示。

图10-3 USB 2.0 OTG 参考设计图



注：USB 2.0 OTG 不提供内置电压泵，需要外接电压泵为 VBUS 供电。

USB 2.0 OTG 既可以作主机，也可以作设备：

- 当作主机时，对接 USB 设备应该通过 Mini-A 插头接入图 10-3 中的 Mini-AB 插座，此时 OTGID 线接地，当 USB 2.0 OTG 检测到 OTGID 为低时，进入“A”设备状态，同时通过 GPIO 信号控制电压泵给 VBUS 供电。
- 当作设备时，对接 USB 主机必须通过 Mini-B 插头接入图 10-3 中的 Mini-AB 插座，此时 OTGID 线悬空，同时对接的 USB 主机给 OTGVBUS 供电，USB 2.0 OTG 将进入设备状态。



USB 2.0 OTG PHY 必须通过 OTGREXT 来设置偏流和提供参考电压，外接电阻必须为 $3.48K\Omega \pm 1\%$ ，保证压降在 $1.1V \sim 1.3V$ 。



注意

OTGDP、OTGDM 板级走线尽可能短、直，宽度大约 7.5mil，值应该固定；走线长度差距应小于 150mils，并且平行，间距固定在 7.5mils 左右；走线尽可能同层，远离时钟信号。否则将影响 USB 2.0 OTG 信号质量。

10.2.4 功能描述

应用框图

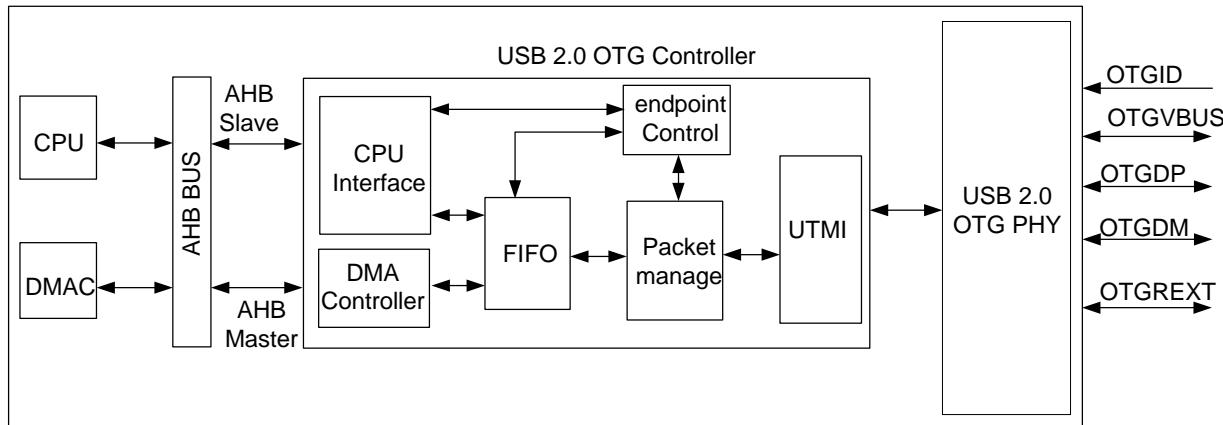
USB 2.0 OTG 支持 CPI 传输方式、内置 DMA 传输方式。

当接收数据时，来自 PHY 侧 USB 总线的数据经过解码后放入当前使用端点的 FIFO 中，当 FIFO 满 1 个或 2 个包后，通过 CPU

当发送数据时，USB 2.0 OTG 控制器通过 CPU 方式或 DMA 方式把数据从源存储器搬到 FIFO 中，当 FIFO 满 1 个包后，把数据从 FIFO 中取出，同时经过编码发送给 USB 总线。

USB 2.0 OTG 应用框图如图 10-4 所示。

图10-4 USB 2.0 OTG 应用框图



工作原理

USB 2.0 OTG 支持以下 4 种标准的传输方式：

- 控制传输

主要用于 USB 主机与 USB 设备端点 0 之间的传输，某些特定型号的 USB 设备的控制传输可能用到其它的端点。控制传输是双向传输，数据量通常较小，可以传



输 8 字节、16 字节、32 字节和 64 字节的数据，依赖于设备和传输速度。端点 0 的 FIFO 为 64 字节，支持这种类型的数据传输。

- 批量传输

主要用于没有带宽和间隔时间要求的情况下发送和接收大量的数据，这种类型的设备适合于传输非常慢和大量被延迟的传输，可以等到所有其它类型的数据传送完成之后再发送和接收数据。它的特点是以错误检测和重传的方式保证 USB 主机与设备的数据被无差错地发送。根据工作速率，支持高速模式 512 字节，全速模式最大 64 字节 payload 大小的数据传输。端点 1 的发送和接收 FIFO 分别为 1024 字节，端点 2 的发送和接收 FIFO 分别为 512 字节，端点 3 的接收 FIFO 为 64 字节，支持这种类型的数据传输。

- 同步传输

主要用于时间严格并具有较强容错性的流数据传输，或者用于要求恒定的数据传输率的即时应用中。同步传输提供了确定的带宽和间隔时间。支持高速模式最大 1024 字节 payload 的数据传输，全速模式最大 1023 字节的数据传输。端点 1 支持这种类型的数据传输。

- 中断传输

中断传输类型用于支持那些不需要经常发送或者接收数据、但是服务周期有限的设备。中断管道是一个流管道，因此一般都是单向的。端点描述符能识别出给定中断管道的通信流是流入还是流出主机。高速端点允许的最大数据有效负载高达 1024 字节。全速端点允许的中断数据最大有效负载小于等于 64 字节。低速设备的最大数据有效负载小于等于 8 字节。

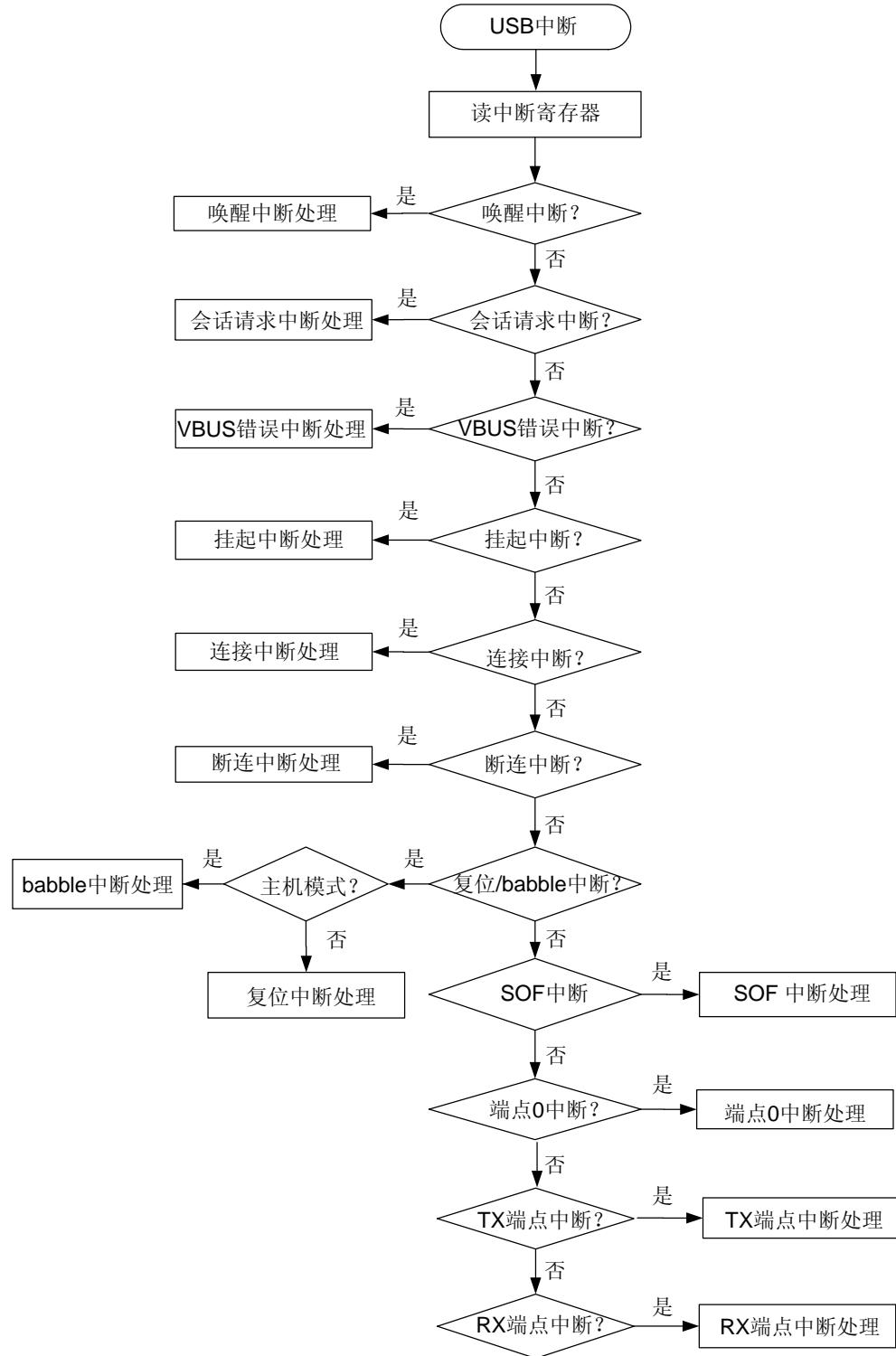
中断处理

USB 2.0 OTG 中断 CPU 时，CPU 需要读 [OTG_INTRUSB](#)、[OTG_INTRTX](#) 和 [OTG_INTRRX](#) 来确定哪个端点产生了中断，并进行相应的中断处理。如果多个端点产生了中断，应该最先处理端点 0 的中断。

具体处理流程如图 10-5 所示。如果采用内置 DMA 工作方式，还要查询内置 DMA 寄存器，判断是否产生内置 DMA 中断。



图10-5 USB 2.0 OTG 中断处理流程图





10.2.5 工作方式

时钟配置

USB 2.0 OTG 时钟配置如表 10-4 所示，具体请参见系统控制器的相关描述。

表10-4 USB 2.0 OTG 时钟配置相关寄存器列表

寄存器	功能
SC_PEREN[otgclken]	OTG 模块时钟使能控制。
SC_PERDIS[otgclkdis]	OTG 模块时钟禁止控制。
SC_PERCLKEN[otgclken]	OTG 模块时钟状态。
SC_PERCTRL2[otg24clk_sel]	USB OTG PHY 24MHz 时钟源选择。 0：选择管脚输入的 24MHz 时钟。 1：选择内部 PLL 分频产生 24MHz 时钟。

软复位

USB 2.0 OTG 默认一直处于复位状态，要启动 USB 2.0 OTG 进入工作状态，必须配置系统控制器的 SC_PERCTRL0[otg_srst] 为 1 撤消复位。

在进行复位时，一般要求控制器与 PHY 一起复位，但也可只对 PHY 进行单独复位。如果要对控制器进行复位（控制器寄存器复位为初始值），必须同时也对 PHY 进行复位，同时必须有驱动软件对 USB 2.0 OTG 进行重新初始化才能使它进行正常工作。本位设计成 OTG PHY 和控制器的复位使用一个寄存器控制，即 SC_PERCTRL0[21]寄存器。

低功耗配置

在系统中，可以使用通过以下方法使 USB 接口模块进入低功耗模式：当暂时不使用 USB 接口时，USB 接口可以进入挂起状态。如果需要重新使用 USB 接口，可以进行唤醒操作。唤醒操作完成之后，就可以正常使用 USB 接口了。具体步骤请参见“[10.2.5 工作方式](#)”中“[挂起与唤醒](#)”。

VBUS 供电方式

当 USB 2.0 OTG 检测到 OTGID 信号为低电平，即进入 A 设备状态（OTG_DEVCTL[b-device] 为 0），此时，USB 2.0 OTG 必须通过控制电压泵给 VBUS 提供 5V 电压。控制管脚为某个空闲的 GPIO，控制信号电平取决于选用的电压泵。

软连接/断开

软连接/断开是在 USB 2.0 OTG 作设备时，通过控制 OTG_POWER[soft conn]来进行：

- 当 OTG_POWER[soft conn] 为 1 时，OTG D+/D-进入正常工作模式。



- 当 OTG_POWER[soft conn] 为 0 时, USB D+/D- 进入非驱动模式, USB D+/D- 相当于断开。

寄存器复位值保持在断开模式, 在驱动软件对 USB 2.0 OTG 进行初始化之后, 枚举之前设置 OTG_POWER[soft conn] 为 1, 使 USB 2.0 OTG 进入正常工作模式。

工作模式配置

USB 2.0 OTG 支持 2 种工作模式: 外设模式和主机模式。

- 当 OTG_DEVCTL[host mode] 位为 0 时, OTG 工作在外设模式。IN 事务的数据通过 TxFIFOs 进行处理; OUT 事务的数据通过 RxFIFOs 进行处理。
- 当 OTG_DEVCTL[host mode] 位为 1 时, OTG 工作在主机模式, 可以支持连接高速、全速或者低速的外设, 但不支持与集线器 (Hub) 相连。支持控制、批量、同步、中断这 4 种传输方式。

当 OTG_DEVCTL[b-device] 位为 1, 同时 OTG_DEVCTL[vbus[1]] 与 OTG_DEVCTL[vbus[0]] 为 1 时, 表示有 USB 主机插入, 此时 USB 2.0 OTG 进入外设模式。由 USB 主机对其进行复位、枚举。

进入主机模式的步骤如下:

- 步骤 1 配置 OTG_DEVCTL[session] 位为 1, 当读到 OTG_DEVCTL[b-device] 位为 0 时, 表示有 USB 设备插入, 此时 USB 2.0 OTG 进入“A”设备状态。
- 步骤 2 配置 VBUS 供电 (请参见 VBUS 供电方式) 给 VBUS 提供电压。
- 步骤 3 对设备进行复位、枚举。

----结束

挂起与唤醒

主机模式下的挂起与唤醒操作如下:

- 步骤 1 进入挂起模式。通过置 OTG_POWER[suspend mode] 位为 1, 可使主机进入挂起模式, 主机停止当前事务, 处于挂起模式。如果同时置 OTG_POWER[enable suspendm] 位为 1, PHY 将进入低功耗模式, 同时关闭控制器 PHY 时钟域时钟。
- 步骤 2 发送唤醒信号。当需要 USB 2.0 OTG 离开挂起模式时, 需要清除 OTG_POWER[suspend mode] 位, 同时置 OTG_POWER[resume] 位为 1, 此时在 USB 总线上生成唤醒信号, 20ms 后 CPU 清除 OTG_POWER[resume] 位, USB 2.0 OTG 离开挂起模式。
- 步骤 3 响应远程唤醒。处于挂起模式的主机, 当检测到 USB 总线上有唤醒信号时, PHY 将退出低功耗模式, 打开控制器 PHY 时钟域时钟; 接着, USB 2.0 OTG 将退出挂起模式, 同时置 OTG_POWER[resume] 位为 1 来生成唤醒信号, 与 USB 总线的唤醒信号进行叠加。如果唤醒中断被使能, 将产生唤醒中断。

----结束

设备模式下的挂起与唤醒操作如下:



步骤 1 进入挂起模式。当 USB 2.0 OTG 作为设备时，它将检测 USB 总线的活动，当总线 3ms 没有活动时，就进入挂起模式。如果挂起中断被使能，将产生挂起中断。

步骤 2 响应唤醒信号。检测到 USB 总线有唤醒信号时，USB 2.0 OTG 将自动退出挂起模式。如果唤醒中断被使能，将产生唤醒中断。

步骤 3 发起远程唤醒。处于挂起模式的 USB 2.0 OTG 想发起远程唤醒，必须置 OTG_POWER[resume]位为 1，并且保持大约 10ms（最小 2ms，最大 15ms）后清除。

----结束

复位设置与总线复位

在主机模式下，置 OTG_POWER[reset]位为 1，在 USB 总线上将生成复位信号（如果设置 OTG_POWER[hs enab]位为 1，它将协商进入高速操作），CPU 应该保持此位至少 20ms，以便使目标设备正确复位。目标设备复位后，CPU 应该清除此位，开始帧计数和事务调度。

在设备模式下，当 USB 2.0 OTG 检测到 USB 总线上有复位信号时，需要执行下列步骤：

步骤 1 生成复位中断。

步骤 2 设置 OTG_FADDR 为 0x00。

步骤 3 设置 OTG_INDEX 为 0x0。

步骤 4 清空所有端点 FIFO。

步骤 5 清除所有控制/状态寄存器。

步骤 6 使能所有端点中断。

----结束

如果设置 OTG_POWER[hs enab]位为 1，它将协商进入高速操作（是否进入高速操作可读 OTG_POWER[hs mode]位）。当驱动软件接收到复位中断后，应该关闭所有管道，等待总线枚举。

会话请求

为了降低功耗，USB 2.0 OTG 协议允许总线电源 VBUS 在需要的时候拉高，在总线不使用时关闭（VBUS 总是由“A”设备提供的）。USB 2.0 OTG 是通过采样 OTGID 信号来决定是作“A”设备还是“B”设备，当一个 Mini-A 接口插入时，OTGID 信号变低（表明 USB 2.0 OTG 要作“A”设备）；当一个 Mini-B 接口插入时，OTGID 信号变高（表明 USB 2.0 OTG 要作“B”设备）。

要采样 OTGID 信号，CPU 应该将 OTG_DEVCTL[session]置 1 来使能 OTGID 采样。采样完后读 OTG_DEVCTL[b-device]位可知 OTG 是“A”设备还是“B”设备。

当 USB 2.0 OTG 为“A”设备时，与设备会话的步骤如下：

步骤 1 USB 2.0 OTG 进入主机模式，打开 VBUS，使电压上升到 VBUS VALID (OTG_DEVCTL[vbus[1:0]]=11)。



- 步骤 2 当发现有设备连接时，产生一个连接中断，这时可从 OTG_DEVCTL[fsdev]与 OTG_DEVCTL[lsdev]位读到连接的设备是高速/全速或低速设备。
- 步骤 3 USB 2.0 OTG 对设备进行复位。
- 步骤 4 如果 OTG_DEVCTL[fsdev]与 OTG_POWER[hs enab]被置 1，USB 2.0 OTG 在复位时将检测总线上信号，检测是否有来自设备的高速握手信号。
- 步骤 5 如果检测到来自设备的高速握手信号，USB 2.0 OTG 将响应高速握手信号，进入高速主机模式。
- 步骤 6 会话结束后，USB 2.0 OTG 清除 OTG_DEVCTL[session]位。

----结束

当 USB 2.0 OTG 为“B”设备时，与主机会话的步骤如下：

- 步骤 1 发出 SRP 请求。释放 VBUS 上的电到 OTG_DEVCTL[vbus[1:0]]=00，在总线状态保持 SE0 2ms 后，从数据线发脉冲，之后再从 VBUS 发脉冲。
- 步骤 2 作“A”设备的 USB 2.0 OTG 检测到脉冲时，它将产生会话请求中断（如果中断使能有效），“A”设备将通过 CPU 置 OTG_DEVCTL[session]位为 1 来启动会话。
- 步骤 3 会话结束后，OTG_DEVCTL[session]位被清除。

----结束

主机协商

当 USB 2.0 OTG 工作于“A”设备模式 (OTGID=0, (OTG_DEVCTL[b_device]) =0) 时，会在事务开始时自动进入主机模式；当 USB 2.0 OTG 工作于“B”设备模式 (OTGID=1, OTG_DEVCTL[b_device]=1) 时，会在事务开始时自动进入外设模式。

当 USB 2.0 OTG 作为外设时，CPU 可以通过设置 OTG_DEVCTL[host req]为 1 来请求变为主机（这个寄存器的位可以在发起一个会话请求（设置 OTG_DEVCTL[Session]）或者在会话开始后的任何时间设置）。当 USB 2.0 OTG 进入挂起模式时（总线上没有活动 3ms），如果此时 OTG_DEVCTL[host req]位为 1，它将进入主机模式，通过 PHY D+ 的上拉电阻断开来进行主机协商。这使得对接的“A”设备通过连接 PHY D+ 的上拉电阻进入设备模式。

当 USB 2.0 OTG 作为主机时，如果检测到外设连接信号，它将生成一个连接中断（如果中断使能有效），接着，置 OTG_POWER[reset]位为 1 来复位“A”设备，20ms 后清除该位，枚举“A”设备。当工作于“B”设备模式的 USB 2.0 OTG 使用完总线后，CPU 应该设置 OTG_POWER[suspend mode]位为 1，使它进入挂起模式，“A”设备检测到后终止会话，变回主机模式。如果此时中断使能有效，将产生一个断开中断。

批量传输 DMA 传输方式配置

USB 2.0 OTG 的 DMA 传输方式分为以下两种：

- DMA 模式 0

该模式下，DMA 控制器只可以配置寄存器一次传输一个包，传输完后中断一次 CPU。该模式可用于控制、中断、批量、或同步传输。



- DMA 模式 1

该模式下，DMA 控制器可以配置寄存器一次完成一次事务（一次事务包括很多个包），完成一次事务后才中断一次 CPU，效率高于模式 0。该模式只能用于批量传输。

USB 2.0 OTG 内置 DMA 模式 0 发送数据步骤如下：

步骤 1 IDLE 状态。

步骤 2 通过配置 OTG_INTRTXE 使能对应端点的中断，设置 OTG_TXCSR[dmareqenab]为 0。

步骤 3 根据需要配置 OTG_INDMA_ADDR、OTG_INDMA_COUNT，设置 OTG_INDMA_CNTL[enable dma]、OTG_INDMA_CNTL[direction]、OTG_INDMA_CNTL[interrupt enable]为 1，设置 OTG_INDMA_CNTL[dma mode]、OTG_INDMA_CNTL[burst mode]为 0。

步骤 4 内置 DMA 控制器请求总线。

步骤 5 若 AHB_HGRANT 为高，进入步骤 6。

步骤 6 内置 DMA 控制器从地址中读数据写入 FIFO。

步骤 7 若 DMA 中断 OTG_INDMA_INTR[N]为 1，进入步骤 8。

步骤 8 将 OTG_TXCSR[txpktrdy]置 1。

步骤 9 继续处理批量 IN 事务。

----结束

USB 2.0 OTG 内置 DMA 模式 0 接收数据步骤如下：

步骤 1 IDLE 状态。

步骤 2 设置 OTG_INTRRXE[n]为 1，设置 OTG_RXCSR[dmareqenab]为 0。

步骤 3 若有 CPU 中断（OTG_INTRRXE[n]为 1），进入步骤 4。

步骤 4 根据需要设置 OTG_INDMA_ADDR、OTG_INDMA_COUNT，设置 OTG_INDMA_CNTL[enable dma]、OTG_INDMA_CNTL[interrupt enable]为 1，设置 OTG_INDMA_CNTL[direction]、OTG_INDMA_CNTL[dma mode]为 0，根据需要设置 OTG_INDMA_CNTL[burst mode]。

步骤 5 内置 DMA 控制器请求总线。

步骤 6 若 AHB_HGRANT 为高，进入步骤 7。

步骤 7 内置 DMA 控制器从 FIFO 中读取数据写入内部地址中。

步骤 8 若 DMA 中断 OTG_INDMA_INTR[N]为 1，进入步骤 9。

步骤 9 清除 OTG_RXCSR[rxpkrtdy]。

步骤 10 IDLE 状态。

----结束



USB 2.0 OTG 内置 DMA 模式 1 发送数据步骤如下：

- 步骤 1 IDLE 状态。
- 步骤 2 设置 OTG_INTRTXE[n]、OTG_RXCSR[autoset]、OTG_RXCSR[dmareqenab]、OTG_RXCSR[dmareqmode]为 1。
- 步骤 3 设置 OTG_INDMA_ADDR 为发送包的地址、OTG_INDMA_COUNT 为发送包的大小，设置 OTG_INDMA_CNTL[enable dma]、OTG_INDMA_CNTL[direction]、OTG_INDMA_CNTL[interrupt enable]、OTG_INDMA_CNTL[dma mode]为 1，OTG_INDMA_CNTL[burst mode]为所需值。
- 步骤 4 若 DMA 控制器请求线 DMA_REQ[n-1]为高，进入步骤 5。
- 步骤 5 内置 DMA 控制器请求总线。
- 步骤 6 若 AHB_HGRANT 为高，进入步骤 7。
- 步骤 7 内置 DMA 控制器从内部地址中读数据写入 FIFO，OTG_INDMA_COUNT 递减。
- 步骤 8 若 OTG_INDMA_COUNT 不为 0，返回步骤 4；若为 0，进入步骤 9。
- 步骤 9 产生 DMA 中断，OTG_INDMA_INTR[N]为 1。
- 步骤 10 设置 OTG_RXCSR[txpktrdy]为 1。
- 步骤 11 发送最后一个包。
- 步骤 12 IDLE 状态。

----结束

USB 2.0 OTG 内置 DMA 模式 1 接收数据步骤如下：

- 步骤 1 IDLE 状态。
- 步骤 2 设置 OTG_INTRRXE[n]为 0，设置 OTG_RXCSR[autoclear]、OTG_RXCSR[dmareqenab]、OTG_RXCSR[dmareqmode]为 1，如果是主机置 OTG_RXCSR[autoreq]为 1。
- 步骤 3 设置 OTG_INDMA_ADDR 为从 FIFO 搬入数据的地址、OTG_INDMA_COUNT 内部地址接收数据总数，设置 OTG_INDMA_CNTL[enable dma]、OTG_INDMA_CNTL[dma mode]、OTG_INDMA_CNTL[interrupt enable]为 1，设置 OTG_INDMA_CNTL[direction]为 0，根据需要设置 OTG_INDMA_CNTL[burst mode]。
- 步骤 4 若 OTG_INDMA_COUNT 为 0，进入步骤 5，否则进入步骤 8。
- 步骤 5 DMA 产生中断，OTG_INDMA_INTR[N]为 1。
- 步骤 6 清除 OTG_RXCSR[rxpktrdy]。
- 步骤 7 IDLE 状态。
- 步骤 8 若 DMA_REQ[n-1]为高，进入步骤 9。
- 步骤 9 内置 DMA 控制器请求总线。
- 步骤 10 若 AHB_HGRANT 为高，进入步骤 11。



步骤 11 内置 DMA 控制器从 FIFO 中读取数据写入内部地址中，递减 OTG_INDMA_COUNT。

步骤 12 若 OTG_INDMA_INTR[N] 为 1，进入步骤 8。

步骤 13 清除 OTG_RXCSR[rxpktrdy]。

步骤 14 若全部数据接收完成，则结束，否则进入步骤 4。

----结束

10.2.6 寄存器概览

USB 2.0 OTG 寄存器的基地址是 0x8009_0000，如图 10-6 所示，其中从偏移地址 0x100~0x13F 为四个端点（包括端点 0）的控制状态寄存器，端点映射控制寄存器反映的是 OTG_INDEX 寄存器值所对应端点的控制状态寄存器值。除了端点控制状态寄存器外还有通用寄存器（表 10-5）、映射控制状态寄存器（外设模式请参见表 10-6、主机模式请参见表 10-7）、控制与配置寄存器（表 10-8）和内置 DMA 寄存器（表 10-9）。

USB 2.0 OTG 寄存器值都可以通过 AHB slave 来配置或读取。所有的寄存器都可以按字节或半字进行读写。

图10-6 USB 2.0 OTG 寄存器地址分配图

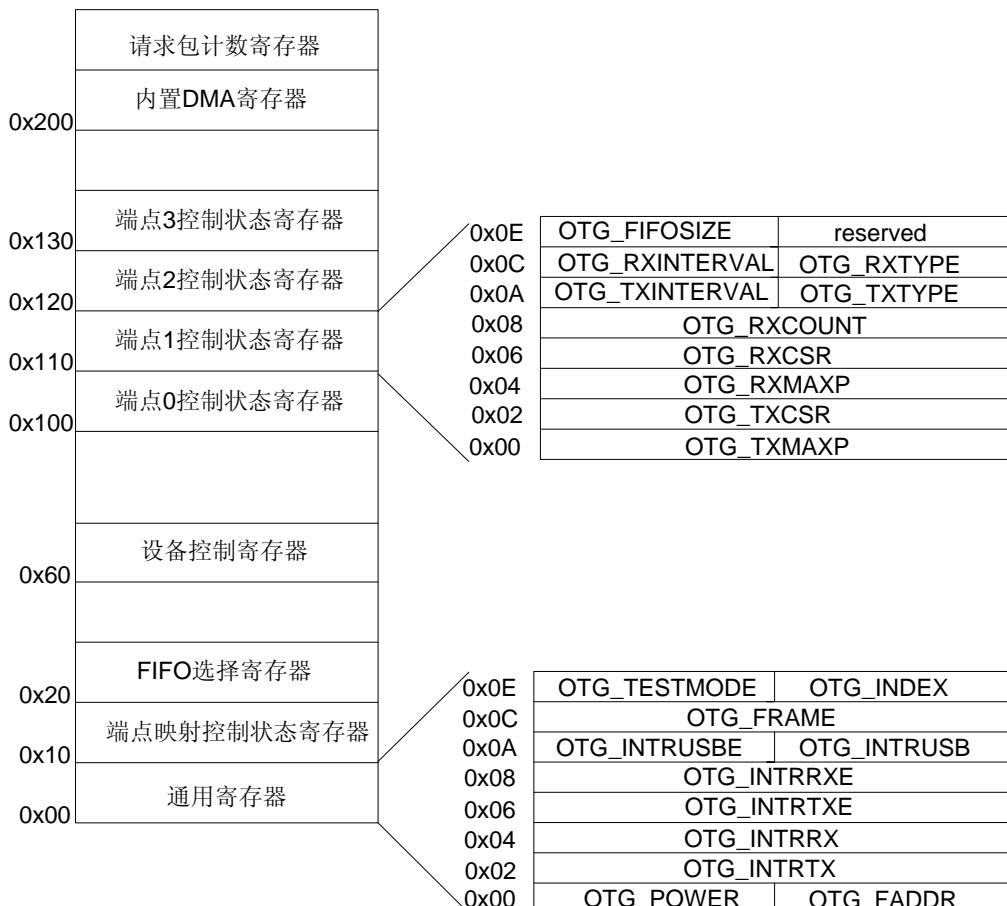




表10-5 USB 2.0 OTG 通用寄存器概览（基址是 0x8009_0000）

偏移地址	名称	描述	页码
0x000	OTG_FADDR	功能地址寄存器	10-38
0x001	OTG_POWER	供电管理寄存器	10-39
0x002~0x003	OTG_INTRTX	发送端点中断寄存器	10-41
0x004~0x005	OTG_INTRRX	接收端点中断寄存器	10-41
0x006~0x007	OTG_INTRTXE	发送端点中断使能寄存器	10-42
0x008~0x009	OTG_INTRRXE	接收端点中断使能寄存器	10-43
0x0A	OTG_INTRUSB	USB 中断寄存器	10-44
0x0B	OTG_INTRUSBE	USB 中断使能寄存器	10-45
0x00C~0x00D	OTG_FRAME	帧号寄存器	10-47
0x00E	OTG_INDEX	端点选择寄存器	10-47
0x00F	OTG_TESTMODE	USB 2.0 测试模式使能寄存器	10-47

表10-6 USB 2.0 OTG 端点映射控制状态寄存器（设备模式）

偏移地址	名称	描述	页码
0x010~0x011	OTG_TXMAXP	外设 Tx 端点最大包尺寸寄存器	10-49
0x012~0x013	OTG_CSR0	端点 0 控制状态寄存器	10-50
	OTG_TXCSR	外设 Tx 端点控制状态寄存器	10-53
0x014~0x015	OTG_RXMAXP	外设 Rx 端点最大包尺寸寄存器	10-58
0x016~0x017	OTG_RXCSR	外设 Rx 端点控制状态寄存器	10-59
0x018~0x019	OTG_COUNT0	端点 0 FIFO 接收数据字节数寄存器	10-65
	OTG_RXCOUNT	外设 Rx 端点 FIFO 接收数据字节数寄存器	10-66
0x01A~0x01B	RESERVED	保留	-
0x01C~0x01E	RESERVED	保留	-
0x01F	OTG_CONFIGDATA	配置信息寄存器 (INDEX=0)	10-69
	OTG_FIFOSIZE	FIFO 尺寸寄存器	10-70



表10-7 USB 2.0 OTG 端点映射控制状态寄存器（主机模式）

偏移地址	名称	描述	页码
0x10	OTG_TXMAXP	主机 Tx 端点的最大包尺寸	10-49
0x12	OTG_CSR0	0 端点的控制状态寄存器	10-50
	OTG_TXCSR	主机 Tx 端点的控制状态寄存器	10-53
0x14	OTG_RXMAXP	主机 Rx 端点的最大包尺寸寄存器	10-58
0x16	OTG_RXCSR	主机 Rx 端点的控制状态寄存器	10-59
0x18	OTG_COUNT0	端点 0 的 FIFO 接收数据的字节数寄存器	10-65
	OTG_RXCOUNT	主机 Rx 端点 FIFO 接收数据的字节数寄存器	10-66
0x1A	OTG_TXTYPE	主机 Tx 端点设置事务传输协议和外围设备端点号寄存器	10-66
0x1B	OTG_NAKLIMIT0	端点 0 的 NAK 回应时间限制寄存器	10-67
	OTG_TXINTERVAL	主机 Tx 端点中断传输或者 NAK 回应时限和批量传输的时间间隔寄存器	10-67
0x1C	OTG_RXTYPE	主机 Rx 端点设置事务传输协议和外围设备端点号	10-68
0x1D	OTG_RXINTERVAL	主机 Rx 端点中断传输或者 NAK 回应时限和批量传输的时间间隔寄存器	10-69
0x1E	RESERVED	保留	-
0x1F	OTG_CONFIGDATA	配置信息寄存器	10-69
	OTG_FIFOSIZE	FIFO 尺寸寄存器	10-70
0x020~0x05F	OTG_FIFOX	FIFO 选择寄存器	10-71

表10-8 USB 2.0 OTG 控制与配置寄存器

偏移地址	名称	描述	页码
0x060	OTG_DEVCTL	OTG 设备控制寄存器	10-71
0x061~0x06F	RESERVED	保留	-

表10-9 USB 2.0 OTG 内置 DMA 寄存器

偏移地址	名称	描述	页码
0x200	OTG_INDMA_INTR	内置 DMA 中断寄存器	10-72
0x204	OTG_INDMA_CNT_L(1)	内置 DMA 通道 1 控制寄存器	10-73
0x208	OTG_INDMA_ADD_R(1)	内置 DMA 通道 1 AHB 存储器地址寄存器	10-74
0x20C	OTG_INDMA_COUNT(1)	内置 DMA 通道 1 字节计数寄存器	10-75
0x210	RESERVED	保留	-
0x214	OTG_INDMA_CNT_L(2)	内置 DMA 通道 2 控制寄存器	10-73
0x218	OTG_INDMA_ADD_R(2)	内置 DMA 通道 2 AHB 存储器地址寄存器	10-74
0x21C	OTG_INDMA_COUNT(2)	内置 DMA 通道 2 字节计数寄存器	10-75

表10-10 USB 2.0 OTG 接收包计数寄存器

偏移地址	名称	描述	页码
0x300+2×n	OTG_RQPKTCOUNT	请求包计数寄存器	10-75

10.2.7 寄存器描述

OTG_FADDR

OTG_FADDR 为功能地址寄存器，用来记录事务处理时的 7 位对接 USB 设备地址。

- 当 USB 2.0 OTG 工作在设备状态 (OTG_DEVCTL[host mode]=0) 时，这个寄存器被 Set_Address 命令写入，然后对后来的令牌包进行译码。
- 当 USB 2.0 OTG 工作在主机状态 (OTG_DEVCTL[host mode]=1) 时，这个寄存器的值通过 Set_Address 命令来设置，在枚举设备时作为对接 USB 设备的地址。



Offset Address								Register Name		Total Reset Value	
0x000								OTG_FADDR		0x00	
Bit	7	6	5	4	3	2	1	0			
Name	reserved	func addr									
Reset	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description								
[7]	-	reserved	保留。								
[6:0]	RW	func addr	设备地址。								

OTG_POWER

OTG_POWER 为供电管理寄存器，用来控制挂起和复位信号，以及进行一些其它的基本控制操作。

Offset Address								Register Name		Total Reset Value	
0x001								OTG_POWER		0x20	
Bit	7	6	5	4	3	2	1	0			
Name	iso update	soft conn	hs enab	hs mode	reset	resume	suspend mode	enable suspendm			
Reset	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description								
[7]	RW	iso update	同步更新。 0: 无效。 1: OTG 作设备时，在送包之前，置 txpktrdy 后，等待 SOF (Start Of Frame) 令牌。如果在 SOF 令牌前收到一个 IN 令牌，将返回一个 0 字节的数据包。 注: 只用于同步传输。								
[6]	RW	soft conn	软连接。 0: OTG 作设备时，断开 D+/D-线。 1: OTG 作设备时，连接上 D+/D-线。 注: 只用于设备模式。								



[5]	RW	hs enab	高速使能。 0: 作设备时, 只工作在全速状态。 1: 作设备时, 被 Hub 复位后, OTG 会进入高速状态。
[4]	RO	hs mode	高速模式。 0: OTG 工作在全速状态。 1: OTG 主机或设备工作在高速状态。作设备时在 USB 复位完成后有效; 作主机在 OTG_POWER[reset]位清除后有效。
[3]	RW	reset	复位。 作设备时: 0: USB 总线没有复位信号。 1: USB 总线有复位信号。 作主机时: 0: 不向 USB 总线发复位信号。 1: 向 USB 总线发复位信号。 注: 这位在主机模式时可读写, 在设备模式时只读。
[2]	RW	resume	唤醒。 0: 不向 USB 总线送 resume 信号。 1: 向 USB 总线送 resume 信号。CPU 应该在 10ms (不超过 15ms) 后清除该位。读时表明 USB 总线有 resume 信号。 注: 当主机处于 suspend 模式下时, 检测到 USB 总线上有 resume 信号时, 将自动把此位置为 1。
[1]	RW	suspend mode	挂起模式。 0: 主机没有进入 suspend 状态; 设备没有从 USB 总线检测到 suspend 信号。 1: 主机模式下表示主机进入 suspend 状态; 设备模式下读到 USB 总线有 suspend 信号时, 自动置此位为 1, 并进入 suspend 状态。
[0]	RW	enable suspendm	挂起模式使能。 0: 不关闭 PHY 时钟。 1: 在 OTG_POWER[suspend mode]位有效时, 关闭 PHY 时钟。



OTG_INTRTX

OTG_INTRTX 为发送端点中断寄存器，用来表明发送端点中断。

Bit	Offset Address 0x002, 0x003 Register Name OTG_INTRTX Total Reset Value 0x0000																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved														ep2 tx	ep1 tx	ep0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description														
[15:3]	-	reserved	保留。														
[2]	RC	ep2 tx	发送端点 2 中断状态。 0: 未产生中断。 1: 已产生中断。														
[1]	RC	ep1 tx	发送端点 1 中断状态。 0: 未产生中断。 1: 已产生中断。														
[0]	RC	ep0	端点 0 中断状态。 0: 未产生中断。 1: 已产生中断。														

OTG_INTRRX

OTG_INTRRX 为接收端点中断寄存器，用来表明接收端点中断。



说明

保留位读出的值为 0。所有中断都采用读清除的形式。



Offset Address												Register Name				Total Reset Value			
0x004, 0x005												OTG_INTRRX				0x0000			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												ep3 rx	ep2 rx	ep1 rx	reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description																
[15:4]	-	reserved	保留。																
[3]	RC	ep3 rx	接收端点 3 中断状态。 0: 未产生中断。 1: 已产生中断。																
[2]	RC	ep2 rx	接收端点 2 中断状态。 0: 未产生中断。 1: 已产生中断。																
[1]	RC	ep1 rx	接收端点 1 中断状态。 0: 未产生中断。 1: 已产生中断。																
[0]	-	reserved	保留。																

OTG_INTRTXE

OTG_INTRTXE 为发送端点中断使能寄存器，用来使能 OTG_INTRTX 中断。

Offset Address												Register Name				Total Reset Value			
0x006												OTG_INTRTXE				0x0007			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												ep2 tx	ep1 tx	ep0				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1			
Bits	Access	Name	Description																
[15:3]	-	reserved	保留。																



[2]	RW	ep2 tx	Tx 端点 2 中断使能。 0: 禁止。 1: 使能。
[1]	RW	ep1 tx	Tx 端点 1 中断使能。 0: 禁止。 1: 使能。
[0]	RW	ep0	端点 0 中断使能。 0: 禁止。 1: 使能。

OTG_INTRRXE

OTG_INTRRXE 为接收端点中断使能寄存器，用来使能 OTG_INTRRX 中断。

Bit	Offset Address 0x008															Register Name OTG_INTRRXE				Total Reset Value 0x000E			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved															ep3 rx	ep2 rx	ep1 rx	reserved				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0							
Bits	Access	Name	Description																				
[15:4]	-	reserved	保留。																				
[3]	RW	ep3 rx	Rx 端点 3 中断使能。 0: 禁止。 1: 使能。																				
[2]	RW	ep2 rx	Rx 端点 2 中断使能。 0: 禁止。 1: 使能。																				
[1]	RW	ep1 rx	Rx 端点 1 中断使能。 0: 禁止。 1: 使能。																				
[0]	-	reserved	保留。																				



OTG_INTRUSB

OTG_INTRUSB 为 USB 中断寄存器，用来表明 USB 总线中断。所有中断都采用读清除的形式。

Offset Address								Register Name	Total Reset Value	
Bit	7	6	5	4	3	2	1	0	OTG_INTRUSB	0x00
Name	vbus error	see reg	discon	conn	sof	reset/babble	resume	suspend		
Reset	0	0	0	0	0	0	0	0		
Bits Access Name Description										
[7]	RC	vbus error	VBUS 错误中断状态。 0: 未产生中断。 1: 已产生中断。 注: 在会话过程中，当总线电位下降到 VBUS 有效门槛电压以下时，产生中断；只在“A”设备模式下有效。							
[6]	RC	sess req	会话请求中断状态。 0: 未产生中断。 1: 已产生中断。 注: 当检测到会话请求信号时，产生中断；只在“A”设备模式下有效。							
[5]	RC	discon	断连中断状态。 在主机模式下，当检测到设备断开时，产生中断；在设备模式下，会话结束时，产生中断。 0: 未产生中断。 1: 已产生中断。							
[4]	RC	conn	连接中断状态。 在主机模式下，当检测到设备连接时，产生中断。 0: 未产生中断。 1: 已产生中断。 注: 只在作主机时有效。							



[3]	RC	sof	SOF 中断状态。 0: 未产生中断。 1: 已产生中断。 注: 当一新帧开始时, 产生中断。
[2]	RC	reset	复位中断状态。 0: 未产生中断。 1: 已产生中断。 注: 在外设模式, 当检测到 USB 总线上 reset 信号时, 产生中断。
		babble	Babble 中断状态。 0: 未产生中断。 1: 已产生中断。 注: 在主机模式, 当检测到 babble 信号时, 产生中断。
[1]	RC	resume	唤醒中断状态。 0: 未产生中断。 1: 已产生中断。 注: 在总线挂起状态, 如果检测到 resume 信号, 则产生中断。
[0]	RC	suspend	挂起中断状态。 0: 未产生中断。 1: 已产生中断。 注: 设备模式, 当检测到 USB 总线有 suspend 信号时, 则产生中断; 只在设备模式下有效。

OTG_INTRUSBE

OTG_INTRUSBE 为 USB 中断使能寄存器, 用来使能 OTG_INTRUSBE 中断。



Offset Address								Register Name	Total Reset Value		
Bit	0x0B								OTG_INTRUSBE	0x06	
Name	7	6	5	4	3	2	1	0	reset / babbble	resume	suspend
Reset	0	0	0	0	0	1	1	0			
Bits	Access	Name	Description								
[7]	RW	vbus error	VBUS 错误中断使能。 0: 禁止。 1: 使能。								
[6]	RW	sess req	会话请求中断使能。 0: 禁止。 1: 使能。								
[5]	RW	discon	软断开中断使能。 0: 禁止。 1: 使能。								
[4]	RW	conn	软连接中断使能。 0: 禁止。 1: 使能。								
[3]	R/W	sof	SOF 中断使能。 0: 禁止。 1: 使能。								
[2]	RW	reset babble	复位/Babble 中断使能。 0: 禁止。 1: 使能。								
[1]	RW	resume	唤醒中断使能。 0: 禁止。 1: 使能。								
[0]	RW	suspend	挂起中断使能。 0: 禁止。 1: 使能。								



OTG_FRAME

OTG_FRAME 为帧号寄存器，用来保存最后一个帧号。

Bit	Offset Address 0x0C														Register Name OTG_FRAME	Total Reset Value 0x0000	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		
Name	reserved							frame number									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name		Description													
[15:11]	-	reserved		保留。													
[10:0]	RO	frame number		帧号。													

OTG_INDEX

OTG_INDEX 为端点选择寄存器，所有的端点状态与控制寄存器都与此寄存器对应。比如 OTG_INDEX 寄存器选择端点 1，那么偏移地址为 0x10~0x19 的一系列寄存器的值将反映该端点情况。每个 Tx 端点和 Rx 端点有它们自己的控制寄存器 0x100~0x1FF。

Bit	Offset Address 0x0E			Register Name OTG_INDEX	Total Reset Value 0x0
	3	2	1		
Name	selected endpoint				
Reset	0	0	0	0	0
Bits	Access	Name		Description	
[3:0]	RW	selected endpoint		端点选择，有效值范围 0~3。 注：在访问端点控制寄存器之前，端点号要写入 Index 寄存器中。	

OTG_TESTMODE

OTG_TESTMODE 为使能 USB 2.0 测试模式寄存器，用来将 OTG 转入 USB2.0 协议高速状态下的四种测试模式，以响应 SET FEATURE 命令，在一般操作中不使用。



说明

每次只能配置寄存器的第 0~6 位中的一位。



Offset Address								Register Name				Total Reset Value							
0x0F								OTG_TESTMODE				0x00							
Bit	7	6	5	4	3	2	1	0											
Name	force_host	fifo_access	force_fs	force_hs	test_packet	test_k	test_j	test_se0_nak											
Reset	0	0	0	0	0	0	0	0											
Bits	Access	Name		Description															
[7]	RW	force_host		强制进入主机模式。 0: 不进入主机模式。 1: 同时 OTG_DEVCTL[session] 位置为 1 时, OTG 不管是否连接设备, 都将进入主机模式。 注: 主机模式工作的速率由 OTG_TESTMODE[force_hs] 位与 OTG_TESTMODE[force_fs]共同决定。 00: OTG 工作在低速模式。 01: OTG 工作在全速模式。 10: OTG 工作在高速模式。 11: 保留。															
[6]	WO	fifo_access		FIFO 存取。 0: 不传送包。 1: 把包从端点 0 的 Tx FIFO 传送到端点 0 的 Rx FIFO, 传送完后自清 0。															
[5]	RW	force_fs		强制进入全速模式。 0: 不进入全速模式。 1: 与 OTG_TESTMODE[force_host]关联; 或者接收到 USB 总线复位信号后, 进入全速模式。															
[4]	RW	force_hs		强制进入高速模式。 0: 不进入高速模式。 1: 与 OTG_TESTMODE[force_host]关联, 或者接收到 USB 总线复位信号后, 进入高速模式。															



[3]	RW	test_packet	<p>发测试包。</p> <p>0: 不进入高速 test_packet 测试模式。</p> <p>1: 进入高速 test_packet 测试模式。</p> <p>注: 在该模式下, OTG 在总线上发送一个 53 字节的符合 USB2.0 协议的测试包 (请参考 USB2.0 协议)。测试包有固定的模式, 并且在发送前必须把测试包装入端点 0 的 FIFO 中。</p>
[2]	RW	test_k	<p>发 K 信号。</p> <p>0: 不进入高速 test_k 模式。</p> <p>1: 进入高速 test_k 模式。在该模式下, OTG 向总线上发送连续的 K。</p>
[1]	RW	test_j	<p>发 J 信号。</p> <p>0: 不进入高速 test_j 模式。</p> <p>1: 进入高速 test_j 模式。在该模式下, OTG 向总线上发送连续的 J。</p>
[0]	RW	test_se0_nak	<p>发 NAK 信号。</p> <p>0: 不进入高速 test_se0_nak 模式。</p> <p>1: 进入高速 test_se0_nak 模式。在该模式下, OTG 保持发送 NAK 来响应任何有效的 IN 令牌。</p>

OTG_TXMAXP

OTG_TXMAXP 为外设 Tx 端点的最大包尺寸寄存器, 定义了 Tx 端点在一次操作中能传输的最大数据量。每个 Tx 端点 (除了端点 0) 都对应一个 TXMAXP 寄存器。

	Offset Address										Register Name		Total Reset Value			
	0x10										OTG_TXMAXP		0x0000			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	m-1										maximum payload/transaction					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name			Description											
[15:11]	RW	m-1			<p>乘积因子。</p> <p>指示 FIFO 中可以填充的最大包负荷的个数</p> <p>注: $m \times \text{maximum payload}$ 不能超过 Tx 端点 FIFO。</p>											
[10:0]	RW	maximum			定义了在信号传输时的最大有效传输。最大											



	payload/transaction	可以达到 1024 字节, 同时 USB 规范约束了批量、中断、同步传输时包的大小。
--	---------------------	--

OTG_CSR0

OTG_CSR0 为端点 0 控制状态寄存器, 提供端点 0 的控制和状态位。

外设模式下端点 0 控制状态寄存器如下:

	Offset Address 0x12										Register Name OTG_CSR0								Total Reset Value 0x0000
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved										flushfifo	serviced	rxpktrdy	sendstall	sendstall	dataend	sentstall	txpktrdy	rxpktrdy
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																
[15:9]	-	reserved	保留。																
[8]	WO	flushfifo	清空 FIFO。 0: 不清空端点 0 FIFO 即将要发送或接收的包。 1: 清空端点 0 FIFO 即将要发送或接收的包, 同时 FIFO 指针将复位, txpktrdy/rxpktrdy 位被清除, 清除完包后该位自清 0。 注 1: 只有当 txpktrdy/rxpktrdy 置 1 时, 该位才能使用, 否则将造成数据破坏。 注 2: 只能置 1, 写 0 无效。																
[7]	WO	serviced	setupend 位控制。 0: 不清除 setupend 位。 1: 清除 setupend 位, 清除后自清 0。 注: 只能置 1, 写 0 无效。																
[6]	WO	rxpktrdy	rxpktrdy 位控制。 0: 不清除 rxpktrdy 位。 1: 清除 rxpktrdy 位, 清除后自清 0。 注: 只能置 1, 写 0 无效。																



[5]	WO	sendstall	送 STALL 握手。 0: 不送 STALL 握手来终止当前事务。 1: 送 STALL 握手来终止当前事务, 后自清0。 注 1: 在置 sendstall 为 1 前, 应该清除在使用的 FIFO。 注 2: 只能置 1, 写 0 无效。
[4]	RO	setupend	SETUP 令牌结束。 0: CPU 置 servicedsetupend 为 1 时, 此位清0。 1: 在 dataend 为 1 之前, 一次控制事务结束; 同时生成一个中断, 并且清空 FIFO。
[3]	WO	dataend	数据结束。 0: 自动清 0。 1: 以下情况 CPU 置该位为 1。 最后一个数据包置 TxPktRdy 后。 最后一个数据包从 FIFO 中搬走清除 RxPktRdy 时。 发送一个 0 长度的数据包置 TxPktRdy 时。 注: 只能置 1, 写 0 无效。
[2]	RO	sentstall	送完 STALL 握手。 0: 为 1 时, CPU 应该清除该位。 1: 当一个 STALL 握手包被发送后, 该位被置为 1。 注: 只能写 0, 写 1 无效。
[1]	RW	txpktrdy	准备发送数据包。 0: 不准备不发送数据包。 1: 当 FIFO 中装满一个数据包时, 则 CPU 设置该位为 1 来发送数据包, 发送完后自动清0, 同时产生一个 Tx 中断 (如果 Tx 中断使能有效)。 注: 只能置 1, 写 0 无效。
[0]	RO	rxpktrdy	准备接收数据包。 0: CPU 设置 ServicedRxPktRdy 位来清除该位。 1: 当 FIFO 接收满一个数据包时, 把数据包从 FIFO 中搬走, 同时产生一个中断。



主机模式下端点 0 控制状态寄存器如下：

Offset Address								Register Name					Total Reset Value					
Bit	0x12				OTG_CSR0				0x0000									
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved				dis ping	reserved		flushfifo	nak timeout	statuspkt	reqpkt	error	setupkt	rxstall	txpktrdy	rxpktrdy		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name		Description														
[15:12]	-	reserved		保留。														
[11]	RW	dis ping		PING 令牌无效。 0: OTG 在高速控制传输的数据与状态阶段发 PING 令牌。 1: OTG 在高速控制传输的数据与状态阶段不发 PING 令牌。														
[10:9]	-	reserved		保留。														
[8]	WO	flushfifo		清空 FIFO。 0: 清空 FIFO 后该位自清 0。 1: 清空端点 0 FIFO 即将要发送或接收的包，同时 FIFO 指针将复位，txpktrdy/rxpktrdy 位被清除；清空 FIFO 后该位自清 0。 注 1: 只有当 txpktrdy/rxpktrdy 置 1 时，该位才能使用，否则将造成数据破坏。 注 2: 只能置 1，写 0 无效。														
[7]	RW	nak timeout		NAK 超时。 0: NAK 没有超时。 1: 表明端点 0 收到 NAK 响应超过 OTG_NAKLIMIT0 寄存器定义的时间。 注: CPU 清除该位以使端点继续工作；可读，可清 0。														
[6]	RW	statuspkt		状态包。 0: 不执行状态阶段事务。 1: 当同时 txpktrdy 或 reqpkt 为 1 时，执行一个状态阶段事务。 注: 置该位为 1 保证了 data toggle 被置为 1，这样就使 DATA1 包用于状态阶段事务。														



[5]	RW	reqpkt	请求包。 0: 不请求 IN 事务。 1: 请求一个 IN 事务。 注: 当 rxpktrdy 位为 1 时, 自动清除该位。
[4]	RW	error	错误。 0: 执行一次事务, 在尝试三次之前设备有响应。 1: 执行一次事务, 尝试了三次设备都没有响应。当这一位为 1 时, 将产生一个中断。 CPU 应该清除该位。 注: 只读, 可清 0。
[3]	RW	setuppkt	SETUP 令牌包。 0: 在事务中不发送 SETUP 令牌包取代 OUT 令牌包。 1: 当同时 txpktrdy 位为 1 时, 在事务中发送 SETUP 令牌包取代 OUT 令牌包。 注: 只能置 1, 写 0 无效。
[2]	RW	rxstall	接收 STALL 握手。 0: 没有接收到 STALL 握手。 1: 接收到 STALL 握手。 注: CPU 应该清除。可读, 可清 0。
[1]	RW	txpktrdy	准备发送数据。 0: 不准备发送数据。 1: 当 FIFO 装满一个数据包后, CPU 置该位为 1。当包发送完后, 该位自动清 0。当该位被清 0 时, 产生一个 Tx 中断 (如果 Tx 中断使能有效)。 注: 只能置 1, 写 0 无效。
[0]	RO	rxpktrdy	准备接收数据。 0: 不准备接收数据包。 1: 当 FIFO 接收满一个数据包时, 把数据包从 FIFO 中搬走。同时产生一个中断 (如果 Rx 中断使能有效), 当包从 FIFO 中被读走后该位被自动清 0。

OTG_TXCSR

OTG_TXCSR 为外设 Tx 端点的控制状态寄存器。



外设模式下外设 Tx 端点的控制状态寄存器如下：

Offset Address 0x12								Register Name OTG_TXCSR					Total Reset Value 0x0000				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	autoset	iso	mode	dmareqenab	frcdatatog	dmareqmode	reserved		incomptx	clrdatatog	sendstall	sendstall	flushfifo	underrun	fifoempty	txpktrdy	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name		Description													
[15]	RO	autoset		自动设置 txpktrdy。 0: 当最大包尺寸的数据装入 Tx FIFO 时, OTG_TXCSR[txpktrdy]不能自动置为 1。 1: 当最大包尺寸的数据装入 Tx FIFO 时, OTG_TXCSR[txpktrdy]将自动置为 1。 注 1: 如果一个小于最大包尺寸的包载入则需要手动置位 OTG_TXCSR[txpktrdy]。 注 2: 不要在高带宽的同步端点置该位为 1。													
[14]	RO	iso		同步传输。 0: Tx 端点用于批量和中断传输。 1: Tx 端点进行同步传输。 注: 此位只在外设模式下有效, 主机模式下总是返回 0。													
[13]	RO	mode		模式使能。 0: 使能 Rx。 1: 使能 Tx。													
[12]	RO	dmareqenab		DMA 请求使能。 0: 不使能 Tx 端点的 DMA 请求信号。 1: 使能 Tx 端点的 DMA 请求信号。													
[11]	RO	frcdatatog		切换 data toggle。 0: 不切换 data toggle。 1: 无论是否收到 ACK 握手, 都强制端点切换 data toggle, 并且清空 FIFO 中数据包。用于与同步端点交流速率的 Tx 中断端点。													
[10]	RO	dmareqmode		DMA 请求模式。 0: 选择 DMA 模式 0。 1: 选择 DMA 模式 1。													



[9:8]	-	reserved	保留。
[7]	RO	incomptx	<p>正在进行 Tx。</p> <p>0: 完成所有的 IN 令牌取包。</p> <p>1: 表明在传输中已经将一个大包分解成 2 或 3 个包, 但是没有足够的 IN 令牌来取走所有的包。</p> <p>注: 当端点处于高带宽同步传输或中断传输模式时, 此位有效, 否则, 该位总返回 0; 可写 0, 置 1 无效。</p>
[6]	RO	clrdatatog	<p>清除 data toggle。</p> <p>0: 不复位端点 data toggle 为 0。</p> <p>1: 复位端点 data toggle 为 0。</p> <p>注: 可置 1, 写 0 无效。</p>
[5]	RO	sentstall	<p>送完 STALL 握手。</p> <p>0: 没有发送 STALL 握手信号。</p> <p>1: 发送了一个 STALL 握手信号。CPU 清 0 此位。</p> <p>注: 可写 0, 置 1 无效。</p>
[4]	RO	sendstall	<p>送 STALL 握手。</p> <p>0: 终止 STALL 握手。</p> <p>1: 当有 IN 令牌时, 发送 STALL 握手。</p> <p>注 1: 在此位置 1 前, FIFO 应该被清空。</p> <p>注 2: 端点处于同步传输模式下此位无效。</p>
[3]	RO	flushfifo	<p>清空 FIFO。</p> <p>0: 不清除 Tx FIFO 中当前包。</p> <p>1: 清除 Tx FIFO 中当前包。同时 FIFO 指针复位, OTG_TXCSR[txpktrdy]位清 0, 并生成一个中断; 清空 FIFO 后自清 0。</p> <p>注: OTG_TXCSR[txpktrdy]位为 1 时才能对此位置 1, 否则将破坏数据; 另外如果 FIFO 是双包缓冲, 则需要置 1 两次才能清空 FIFO。</p>
[2]	RO	underrun	<p>正在运行。</p> <p>0: 没有接收到 IN 令牌。</p> <p>1: 接收到 IN 令牌但 OTG_TXCSR[txpktrdy]还未置 1。CPU 应该清 0 此位。</p> <p>注: 可写 0, 置 1 无效。</p>



[1]	RO	fifoempty	Tx 端点 FIFO 非空标志位。 0: Tx FIFO 为空。 1: Tx FIFO 中至少有一个包。 注: 可写 0。
[0]	RO	txpktrdy	准备发送数据。 0: 不发送 FIFO 中的数据包。 1: 发送 FIFO 中的数据包。当数据包发送完成后自动清 0。同时产生一个中断（如果中断使能有效）。 注: 可置 1, 写 0 无效。

主机模式下外设 Tx 端点的控制状态寄存器如下:

Offset Address 0x12												OTG_TXCSR					Total Reset Value 0x0000
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	autoset	reserved	mode	dmareqenab	fifodataog	dmareqmode	reserved	reserved	nak timeout	clidataog	rxfifo	rxstall	reserved	flushfifo	error	fifoempty	txpktrdy
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description														
[15]	RW	autoset	自动置 txpktrdy 位。 0: 当最大包尺寸的数据装入 Tx FIFO 时, OTG_TXCSR[txpktrdy]不能自动置为 1。 1: 当最大包尺寸的数据装入 Tx FIFO 时, OTG_TXCSR[txpktrdy]将自动置为 1。 注 1: 如果一个小于最大包尺寸的包载入则需要手动置位 OTG_TXCSR[txpktrdy]。 注 2: 不要在高带宽的同步端点置该位为 1。														
[14]	-	reserved	保留。														
[13]	RW	mode	模式使能。 0: 使能 Rx。 1: 使能 Tx。														
[12]	RW	dmareqenab	DMA 请求使能。 0: 不使能 Tx 端点的 DMA 请求信号。 1: 使能 Tx 端点的 DMA 请求信号。														



[11]	RW	frcdatatog	切换 data toggle。 0: 不切换 data toggle。 1: 无论是否收到 ACK 握手, 都强制端点切换 data toggle, 并且清空 FIFO 中数据包。这被与同步端点通信速率反馈的 Tx 中断端点所用。
[10]	RW	dmareqmode	DMA 请求模式。 0: DMA 请求模式 0。 1: DMA 请求模式 1。
[9:8]	-	reserved	保留。
[7]	RW	nak timeout	当 bulk 传输时表示 NAK 超时中断。 0: 没有接收到 NAK 握手, 或 Tx 端点接收 NAK 握手没有超过 OTG_TXINTERVAL 寄存器限定的时间。 1: Tx 端点接收 NAK 握手超过了 OTG_TXINTERVAL 寄存器限定的时间。 CPU 必须清除这位允许端点继续工作。 注: 只用于 Bulk 端点。可读, 可写 0。
		incomptx	当高带宽同步传输时表示正在发送包。 0: 没送包, 或收到了响应。 1: 送出包后还没有收到设备的响应。 注: 只用于高带宽中断端点。
[6]	RW	clrdatatog	清除 data toggle。 0: 不复位点 data toggle 为 0。 1: 复位端点 data toggle 为 0。 注: 可读, 可置 1。
[5]	RW	rxstall	收到 STALL 握手。 0: 没接收到 STALL 握手信号。 1: 接收到 STALL 握手信号。此时停止 DMA 请求, 清空 FIFO 且 TxPktRdy 被清 0。 CPU 需要清 0 此位。
[4]	-	reserved	保留。

[3]	RW	flushfifo	<p>清空 FIFO。</p> <p>0: 不清除 Tx FIFO 中当前包。</p> <p>1: 清除 Tx FIFO 中当前包。同时 FIFO 指针复位, OTG_TXCSR[txpktrdy]位清 0, 并生成一个中断; 清空 FIFO 后自清 0。</p> <p>注: OTG_TXCSR[txpktrdy]位为 1 时才能对此位置 1, 否则将破坏数据; 另外如果 FIFO 是双包缓冲, 则需要置 1 两次才能清空 FIFO。</p>
[2]	RW	error	<p>错误。</p> <p>0: 在重复发送三次或更少次包后收到了握手包。</p> <p>1: 重复发送三次包都没有收到握手包。同时产生中断, OTG_TXCSR[txpktrdy]位被清除, FIFO 被清空。CPU 应该清 0 此位。</p> <p>注: 仅当 Tx 端点工作在批量或中断模式下时此位有效。在同步模式下始终为 0。</p>
[1]	RW	fifonempty	<p>FIFO 非空。</p> <p>0: 在 Tx FIFO 中没有包。</p> <p>1: 在 Tx FIFO 中有包。</p>
[0]	RW	txpktrdy	<p>准备发送数据。</p> <p>0: 不发送数据包。</p> <p>1: 发送数据包。</p> <p>注: 通常在一个数据包装入 FIFO 之后对此位置 1 来发送数据。当数据包发送完后该位自动清 0。同时在清 0 时产生一个中断 (如果中断使能有效)。</p>

OTG_RXMAXP

OTG_RXMAXP 为外设 Rx 端点的最大包尺寸寄存器, 定义了一次信号操作中, 选中 Rx 端点的最大数据传输量, 除端点 0 外, 每个 Rx 端点都有。



Offset Address												Register Name			Total Reset Value								
Bit	0x14											OTG_RXMAXP			0x0000								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
Name	m-1											maximum payload / transaction											
Bits	Access	Name	Description																				
[15:11]	RW	m-1	乘积因子。 指示 FIFO 中可以接收的最大包负载的个数。 注：m × maximum payload 不能超过 Rx 端点 FIFO。																				
[10:0]	RW	maximum payload / transaction	定义了在信号传输时的最大有效传输。最大可以到 1024 字节，但是同时被 USB 规范约束批量、中断、同步传输时包的大小。																				

OTG_RXCSR

OTG_RXCSR 为外设 Rx 端点的控制状态寄存器。

外设模式下外设 Rx 端点的控制状态寄存器如下：



Offset Address 0x16												Register Name OTG_RXCSR					Total Reset Value 0x0000				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Name	autoclear	iso	dmareqenab	disnyet	dmareqmode	reserved	incomprx	clidataq	sentstall	sendstall	flushfifo	dataerror	overrun	fiotfull	rxpktrdy						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
Bits	Access	Name	Description																		
[15]	RW	autoclear	自动清除 rxpktrdy。 0: 当 RXMAXP 字节的包从 Rx FIFO 中搬走时, OTG_RXCSR[rxpktrdy]不自动清 0, 需要手动清 0。 1: 当 RXMAXP 字节的包从 Rx FIFO 中搬走时, OTG_RXCSR[rxpktrdy]将自动清 0。 注 1: 如果一个小于最大包尺寸的包从 FIFO 中搬走时, 则需要将 OTG_RXCSR[rxpktrdy] 手动清 0。 注 2: 不要在高带宽的同步端点中使用。																		
[14]	RW	iso	同步传输使能。 0: 使能批量和中断传输。 1: 使能 Rx 端点进行同步传输																		
[13]	RW	dmareqenab	DMA 请求使能。 0: 不使能 Rx 端点的 DMA 请求信号。 1: 使能 Rx 端点的 DMA 请求信号。																		
[12]	RW	disnyet	同步传输时表示无效 NYET 握手。 0: NYET 握手发送有效。 1: 使 NYET 握手发送无效。此时, 所有正确接收到的 Rx 包都返回 ACK 握手, 包括在 FIFO 满时。 注: 此位只在高速模式下有效。																		
	RO	pid error	Bulk 和中断传输时: 表示 PID (Packet ID) 错误。 0: 在接收到的包中没有 PID 错误。 1: 在接收到的包中有 PID 错误。 注: 只用于同步传输。																		



[11]	RW	dmareqmode	DMA 请求模式。 0: 选择 DMA 请求模式 0。 1: 选择 DMA 请求模式 1。 注: 此位不能与 OTG_RXCSR[rxpktrdy]在同周期清 0。
[10:9]	-	reserved	保留。
[8]	RW	incomprx	正在进行数据接收。 0: Rx FIFO 中的包被接收完全。 1: Rx FIFO 中的包没有被接收完全。当 OTG_RXCSR[rxpktrdy]清 0 时该位也被清 0。 注: 只用于高带宽的同步与中断传输, 除了高带宽传输模式下, 该位都返回 0。
[7]	RW	clrdatatog	清除 Data Toggle。 0: 不复位端点 Data Toggle 为 0。 1: 复位端点 Data Toggle 为 0。
[6]	RW	sentstall	送完 STALL 握手。 0: 没有发送 STALL 握手。 1: 发送一个 STALL 握手。CPU 应该清除此位。
[5]	RW	sendstall	送 STALL 握手。 0: 不发送 STALL 握手。 1: 发送一个 STALL 握手信号。 注 1: 在置 1 此位前, 必须清空 FIFO; 注 2: 同步传输模式下无效。
[4]	RW	flushfifo	清空 FIFO。 0: 不清空的 FIFO。 1: 清空一个要从 Rx FIFO 端点读走的包。同时 FIFO 指针复位, OTG_RXCSR[rxpktrdy]位清 0。 注 1: 只有在 OTG_RXCSR[rxpktrdy]为 1 时这一位才能置 1, 否则可能破坏数据。 注 2: 如果 FIFO 为双缓冲型, 需要执行两次置 1 操作才能清空 FIFO。



[3]	RO	dataerror	<p>数据包错误。</p> <p>0: 没有 CRC 或位填充错误。</p> <p>1: 在 OTG_RXCSR[rxpktardy] 为 1 时, 表明数据包有 CRC 或位填充错误。</p> <p>OTG_RXCSR[rxpktardy] 清 0 该位也被清 0。</p> <p>注: 只在同步模式下有效, 在批量模式下总是返回 0。</p>
[2]	RW	overrun	<p>不能接收数据。</p> <p>0: OUT 包能装进 Rx FIFO。</p> <p>1: 一个 OUT 包不能装进 Rx FIFO 中。CPU 应该清除此位。</p> <p>注: 仅在同步模式下有效, 批量模式下总返回 0。</p>
[1]	RO	fifofull	<p>FIFO 满。</p> <p>0: Rx FIFO 还能装更多的包。</p> <p>1: Rx FIFO 已满。</p>
[0]	RW	rxpktrdy	<p>准备接收数据。</p> <p>0: 没有接收到一个数据包或包已从 Rx FIFO 中搬走。</p> <p>1: 接收到一个数据包。当包从 Rx FIFO 被搬走时, CPU 要清 0 此位。当该位被置 1 时产生一个中断。</p> <p>注: 可读, 可写 0。</p>

主机模式下外设 Rx 端点的控制状态寄存器如下:



Bit	Offset Address 0x16				Register Name OTG_RXCSR				Total Reset Value 0x0000							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	autoclear	autoreq	dmareqenab	pid error	dmareqmode	reserved	incomprx	clrdatatog	rxstall	reqpkt	flushfifo	flushfifo	dataerror/nak timeout	fitofull	rxpktrdy	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name		Description												
[15]	RW	autoclear		自动清除 rxpktrdy 位。 0: 当 RXMAXP 字节的包从 Rx FIFO 中搬走时, OTG_RXCSR[rxpktrdy]不自动清 0, 需要手动清 0。 1: 当 RXMAXP 字节的包从 Rx FIFO 中搬走时, OTG_RXCSR[rxpktrdy]将自动清 0。 注 1: 如果一个小于最大包尺寸的包从 FIFO 中搬走时, 则需要将 OTG_RXCSR[rxpktrdy] 手动清 0; 注 2: 不要在高带宽的同步端点中使用。												
[14]	RW	autoreq		自动置 reqpkt 位。 0: rxpktrdy 位清 0 时, reqpkt 位不自动置 1。 1: rxpktrdy 位清 0 时, reqpkt 位自动置 1。												
[13]	RW	dmareqenab		DMA 请求使能。 0: 不使能 Rx 端点 DMA 请求。 1: 使能 Rx 端点 DMA 请求。												
[12]	RO	pid error		PID 错误。 0: 在接收到的包中没有 PID 错误。 1: 在接收到的包中有 PID 错误。 注: 只用于同步传输, 批量与中断传输此位忽略。												
[11]	RW	dmareqmode		DMA 请求模式。 0: 选择 DMA 请求模式 0。 1: 选择 DMA 请求模式 1。												
[10:9]	-	reserved		保留。												



[8]	RW	incomprx	<p>正在接收数据。</p> <p>0: Rx FIFO 中的包被接收完全。</p> <p>1: Rx FIFO 中的包没有被接收完全。当 OTG_RXCSR[rxpktardy] 清 0 时该位也被清 0。</p> <p>注: 只用于高带宽的同步与中断传输, 除了高带宽传输模式下, 该位都返回 0。</p>
[7]	RW	clrdatatog	<p>清除 Data Toggle。</p> <p>0: 不复位端点 Data Toggle 为 0。</p> <p>1: 复位端点 Data Toggle 为 0。</p>
[6]	RW	rxstall	<p>收到 STALL 握手。</p> <p>0: 没有收到 STALL 握手信号。</p> <p>1: 收到一个 STALL 握手信号。同时产生一个中断, CPU 应该清 0 此位。</p>
[5]	RW	reqpkt	<p>请求包。</p> <p>0: 不请求 IN 事务。</p> <p>1 : 请求一个 IN 事务。 OTG_RXCSR[rxpktardy] 置位时该位清 0。</p>
[4]	RW	flushfifo	<p>清空 FIFO。</p> <p>0: 不清除 FIFO 中的包。</p> <p>1: 清除下一个要从 Rx FIFO 端点读走的包。同时 FIFO 指针复位, OTG_RXCSR[rxpktardy] 位清 0。</p> <p>注 1: 只有在 OTG_RXCSR[rxpktardy] 为 1 时此位才能置 1, 否则可能破坏数据;</p> <p>注 2: 如果 FIFO 是双缓冲的, 需要置 1 两次才能清空 FIFO。</p>
[3]	RO	dataerror	<p>数据包错误。</p> <p>0: 没有 CRC 或位填充错误。</p> <p>1: 在 OTG_RXCSR[rxpktardy] 为 1 时, 表明数据包有 CRC 或位填充错误。 OTG_RXCSR[rxpktardy] 清 0 该位也被清 0。</p> <p>注: 只在同步模式下有效。</p>



	RW	nak timeout	NAK 超时。 0: 没有接收到 NAK 握手, 或 Rx 端点接收 NAK 握手没有超过 OTG_TXINTERVAL 寄存器限定的时间。 1: Rx 端点接收 NAK 握手超过了 OTG_TXINTERVAL 寄存器限定的时间。此时 Rx 端点停止工作, CPU 必须清除这位来允许端点继续工作。 注: 只用于 Bulk 端点。可读, 可写 0。
[2]	RW	error	错误。 0: 经过等于或小于三次的尝试接收数据, 结果接收到数据。 1: 三次尝试接收数据, 但均没有收到数据。同时产生中断, CPU 应该清 0 此位。 注: 在批量或中断模式下此位有效。在同步模式下总返回 0。
[1]	RO	fifofull	FIFO 满。 0: Rx FIFO 还能装更多的包。 1: Rx FIFO 已满。
[0]	RW	rxpktrdy	准备接收数据。 0: 没有接收到一个数据包或包已从 Rx FIFO 中搬走。 1: 接收到一个数据包。当包从 Rx FIFO 被搬走时, CPU 要清 0 此位。当该位被置 1 时产生一个中断。 注: 可读, 可写 0。

OTG_COUNT0

OTG_COUNT0 为端点 0 FIFO 接收数据的字节数寄存器, 记录端点 0 FIFO 接收到的数据字节数。只有当 OTG_CSR0[rxpktrdy]被设置时, 返回的值才是有效的。

Offset Address								Register Name								Total Reset Value															
0x018								OTG_COUNT0								0x00															
Bit	7	6	5	4	3	2	1	0	endpoint0 rx count																						
Name	reserved																														
Reset	0	0	0	0	0	0	0	0																							
Bits	Access	Name	Description																												
[7]	-	reserved	保留。																												
[6:0]	RO	endpoint0 rx count	记录端点 0 FIFO 接收到的数据字节数。																												

OTG_RXCOUNT

OTG_RXCOUNT 为 Rx 端点 FIFO 字节数寄存器，用来记录当前从 Rx FIFO 中读走的包的字节数。如果是多个批量包，则这个数值是多个包合并后的数值。只有当 OTG_RXCSR[RxPktRdy] 为 1 时，返回的值才是有效的。

Offset Address																Register Name								Total Reset Value							
0x018																OTG_RXCOUNT								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	endpoint rx count														
Name	reserved																														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															
Bits	Access	Name	Description																												
[15:13]	-	reserved	保留。																												
[12:0]	RO	endpoint rx count	记录接收到的数据字节数。																												

OTG_TXTYPE

OTG_TXTYPE 为主机 Tx 端点设置事务传输协议和外围设备端点号寄存器，低 4 位用来存储端点号，高 2 位存储对应的传输协议。



Offset Address								Register Name	Total Reset Value
0x01A								OTG_TXTYPE	0x00
Bit	7	6	5	4	3	2	1	0	
Name	reserved			protocol			target endpoint number		
Reset	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description						
[7:6]	-	reserved	保留。						
[5:4]	RW	protocol	CPU 设置该位以选择 Tx 端点的传输协议。 00: Illegal。 01: Isochronous。 10: Bulk。 11: Interrupt。						
[3:0]	RW	target endpoint number	主机端使用这个从枚举的过程中从设备描述符获得的发送端点号来设置该位。						

OTG_NAKLIMIT0

OTG_NAKLIMIT0 为端点 0 的 NAK 回应时间限制寄存器，用来设置多少帧（微帧）后端点 0 停止等待 NAK 响应包。

Offset Address								Register Name	Total Reset Value
0x01B								OTG_NAKLIMIT0	0x00
Bit	7	6	5	4	3	2	1	0	
Name	reserved			endpoint nak limit (m)					
Reset	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description						
[7:5]	-	reserved	保留。						
[4:0]	RW	endpoint nak limit (m)	等待 NAK 包的超时时间定义 ($2^{(m-1)}$)，表示值的有效范围是 2~16。						

OTG_TXINTERVAL

OTG_TXINTERVAL 为主机 Tx 端点中断传输或者 NAK 回应时限和批量传输的时间间隔寄存器，每个 Tx 端点都有一个 TxInterval。



Bit	Offset Address 0x01B Register Name OTG_TXINTERVAL Total Reset Value 0x00							
	7	6	5	4	3	2	1	0
Name	tx polling interval/nak limit (m)							
Reset	0	0	0	0	0	0	0	0
Bits	Access	Name	Description					
[7:0]	RW	tx polling interval/nak limit (m)	在中断和同步传输中, 定义当前选中的 Tx 端点查询间隔时间。 在批量传输中, 寄存器设置经过一定时间, 端点就认为接收 NAK 响应超时。					

不同情况下查询间隔定义如表 10-11 所示。

表10-11 查询间隔定义

传输类型	速度	有效值 (m)	查询间隔
中断	低速或全速	1~255	查询间隔是 m 帧。
	高速	1~6	查询间隔是 2^{m-1} 微帧。
同步	全速或高速	1~16	查询间隔是 2^{m-1} 帧/微帧。
批量	全速或高速	2~16	NAK 超时是 2^{m-1} 帧/微帧。 注: m 为 0 或 1 时, NAK 超时功能无效。

OTG_RXTYPE

OTG_RXTYPE 为主机 Tx 端点设置事务传输协议和外围设备端点号寄存器, 低 4 位用来存储端点号, 高 2 位用来存储对应的传输协议。

Bit	Offset Address 0x01C Register Name OTG_RXTYPE Total Reset Value 0x00							
	7	6	5	4	3	2	1	0
Name	reserved		protocol		target endpoint number			
Reset	0	0	0	0	0	0	0	0
Bits	Access	Name	Description					
[7:6]	-	reserved	保留。					



[5:4]	RW	protocol	CPU 设置该位以选择 Rx 端点的传输协议。 00: 非法。 01: 同步。 10: 批量。 11: 中断。
[3:0]	RW	target endpoint number	主机端使用这个从枚举的过程中从设备描述符获得的接收端点号来设置该位。

OTG_RXINTERVAL

OTG_RXINTERVAL 为主机 Rx 端点中断与同步传输查询间隔寄存器或主机 Rx 端点批量传输 NAK 响应时限寄存器，不同情况下的 m 值定义请参见表 10-11。

Offset Address								Register Name	Total Reset Value	
Bit	0x01D								OTG_RXINTERVAL	0x00
Name	rx polling interval/nak limit (m)									
Reset	0 0 0 0 0 0 0 0									
Bits	Access	Name			Description					
[7:0]	RW	rx interval/nak limit (m)	polling		在中断和同步传输中，定义当前选中的 Rx 端点查询间隔时间。 在批量传输中，寄存器设置经过一定时间，端点就认为接收 NAK 响应超时。					

OTG_CONFIGDATA

OTG_CONFIGDATA 为配置信息寄存器，返回配置信息，读时 INDEX 寄存器值设置为 0。

Offset Address								Register Name	Total Reset Value	
Bit	0x01F								OTG_CONFIGDATA	0xDA
Name	mprxe	mptxe	bigendian	hbrxe	hbtqe	dyninfo sizing	softcone	utmi datawidth		
Reset	1	1	0	1	1	0	1	0		



Bits	Access	Name	Description
[7]	RO	mprxe	自动合并批量包。 0: 不自动合并批量包。 1: 自动合并批量包。
[6]	RO	mptxe	自动分解批量包。 0: 不自动分解批量包。 1: 自动分解批量包。
[5]	RO	bigendian	对齐方式。 0: little endian。 1: big endian。
[4]	RO	hbrxe	高带宽 Rx 同步端点。 0: 不支持高带宽 Rx 同步端点。 1: 支持高带宽 Rx 同步端点。
[3]	RO	hbtqe	高带宽 Tx 同步端点。 0: 不支持高带宽 Tx 同步端点。 1: 支持高带宽 Tx 同步端点。
[2]	RO	dynfifo sizing	动态 FIFO。 0: 不使用动态 FIFO。 1: 使用动态 FIFO。
[1]	RO	softcone	软连接。 0: 关闭软连接/断开模式。 1: 启用软连接/断开模式。
[0]	RO	utmi datawidth	UTMI+ data width。 0: 8 位 UTMI+数据接口。 1: 16 位 UTMI+数据接口。

OTG_FIFOSIZE

OTG_FIFOSIZE 为 FIFO 尺寸寄存器，返回被选择的 Tx/Rx 端点 FIFOs 的尺寸。如果端点没有被配置，显示 0。

Offset Address		Register Name		Total Reset Value				
0x01F		OTG_FIFOSIZE		0x00				
Bit	7	6	5	4	3	2	1	0
Name	rx fifo size					tx fifo size		



Reset	0	0	0	0	0	0	0	0
Bits	Access	Name		Description				
[7:4]	RO	rx fifo size		选中的 Rx 端点 FIFO 的尺寸。				
[3:0]	RO	tx fifo size		选中的 Tx 端点 FIFO 的尺寸。				

OTG_FIFOX

OTG_FIFOX 为端点 0~3 的 FIFO 选择寄存器，为 CPU 提供了 4 个地址，分别可以访问四个端点对应的 FIFO。写该地址时，将数据载入相应的端点的 Tx FIFO 中；读该地址时，将相应端点的 Rx FIFO 中数据读出。

地址范围是 0x20~0x2C（端点 0 对应 0x20，端点 1 对应 0x24，端点 2 对应 0x28，端点 3 对应 0x2C）。



说明

可以按字节、半字或字存取 FIFO，但是在一个包的传输中只能采用一种存取形式。

OTG_DEVCTL

OTG_DEVCTL 为 USB 2.0 OTG 设备控制寄存器，用来选择控制器是工作在外设模式还是工作在主机模式，并且控制和监测 USB 总线电压。

	Offset Address								Register Name	Total Reset Value	
	0x060								OTG_DEVCTL	0x80	
Bit	7	6	5	4	3	2	1	0			
Name	b-device	fsdev	lsdev	vbus[1:0]	host mode	host req	session				
Reset	1	0	0	0	0	0	0	0			
Bits	Access	Name		Description							
[7]	RO	b-device		“B” 设备标志位。 0: “A” 设备。 1: “B” 设备。 注: 只在会话过程中有效。							
[6]	RO	fsdev		全速设备标准位。 0: 没有高速或全速设备连接。 1: 有高速或全速设备连接。 注: 只在主机模式下有效。							



[5]	RO	lsdev	低速设备标志位。 0: 没有低速设备连接。 1: 有低速设备连接。 注: 只在主机模式有效。
[4:3]	RO	vbus[1:0]	VBUS 电压状态。 00: 在会话结束门槛电压之下。 01: 在会话结束门槛电压之上, 在会话有效门槛电压之下。 10: 在会话有效门槛电压之上, 在 VBUS 有效门槛电压之下。 11: 在 VBUS 有效门槛电压之上。
[2]	RO	host mode	主机模式。 0: 没有工作在主机模式。 1: 工作在主机模式。
[1]	RW	host req	主机请求。 0: 不发送主机协商请求。 1: 当进入挂起模式时, 发送一个主机协商请求。协商结束后清 0。 注: 只有在 B 设备模式下有效。
[0]	RW	session	会话。 0: 结束会话。 1: 启动会话。 注 1: 作 A 设备时, 通过 CPU 控制启动或结束会话; 作 B 设备时, 通过 CPU 置 1 来进行会话请求协议; 当 OTG 在挂起模式下时, 此位可能被 CPU 清 0 来执行软断开。 注 2: 不要在非挂起状态下清 0 此位, 否则导致不可预想的错误。

OTG_INDMA_INTR

OTG_INDMA_INTR 为 USB 2.0 OTG 内置 DMA 中断寄存器。



Bits	Access	Name	Description
[15:2]	-	reserved	保留。
[1:0]	RO	Intr[15:0]	DMA 中断。 intr[0]: 通道 1 中断。 intr[1]: 通道 2 中断。

OTG_INDMA_CNTL

OTG_INDMA_CNTL 为 USB 2.0 OTG 内置 DMA 控制寄存器。

Offset Address											Register Name				Total Reset Value			
0x204 (通道 1), 0x214 (通道 2)											OTG_INDMA_CNTL				0x0000			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved		burst mode		bus error		reserved		endpoint number		interrupt enable		dma mode		direction		enable dma	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description															
[15:11]	-	reserved	保留。															
[10:9]	RW	burst mode	突发模式： 00: 没有定义长度的突发模式。 01: INCR4 或者没有定义长度的突发模式。 10: INCR8, INCR4 或者没有定义长度的突发模式。 11: INCR16, INCR8, INCR4 或者没有定义长度的突发模式。															
[8]	RO	bus error	总线错误标准位。 0: 无总线错误。 1: 有总线错误。 注: 如果有总线错误, 将终止 DMA 传输。															
[7:6]	-	reserved	保留。															



[5:4]	RW	endpoint number	端点号。 00: 端点 0 不使用 DMA, 此位保留。 01: 端点 1。 10: 端点 2。 11: 保留。
[3]	RW	interrupt enable	中断使能位。 0: 禁止。 1: 使能。
[2]	RW	dma mode	DMA 模式。 0: DMA 模式 0。 1: DMA 模式 1。 注: 在 DMA 模式 0 下, DMA 每次搬运一个包都要中断一次 CPU; 在 DMA 模式 1 下, DMA 要把一次批量传输的所有包都搬运完后才中断 CPU。
[1]	RW	direction	DMA 方向。 0: DMA 写, 用于 Rx 端点。 1: DMA 读, 由于 Tx 端点。
[0]	RW	enable dma	DMA 使能。 0: 禁止。 1: 使能。

OTG_INDMA_ADDR

OTG_INDMA_ADDR 为 USB 2.0 OTG 内置 DMA 地址寄存器。

Offset Address		Register Name		Total Reset Value		
0x208 (通道 1), 0x218 (通道 2)		OTG_INDMA_ADDR		0x0000_0000		
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0					
Name	addr[31:0]					
Reset	0 0					
Bits	Access	Name	Description			
[31:0]	RW	addr[31:0]	DMA 通道 AHB 存储器地址值。			

OTG_INDMA_COUNT

OTG INDMA COUNT 为 USB 2.0 OTG 内置 DMA 字节计数寄存器。

OTG_RQPKTCOUNT

OTG_RQPKTCOUNT 为请求包计数寄存器。接收端点 1~3 都有一个请求包计数寄存器。



其中 n 表示端点号。



11 其他外设接口

关于本章

本章描述内容如下表所示。

标题	内容
11.1 I ² C 控制器	介绍 I ² C 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。
11.2 通用异步收发器	介绍 UART 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。
11.3 同步串口	介绍 SSP 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。
11.4 红外接口	介绍 IR 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。
11.5 通用目的输入输出接口	介绍 GPIO 模块的功能、特点、接口信号、工作方式、寄存器概览和寄存器描述。



11.1 I²C 控制器

11.1.1 概述

I²C (The Inter-Integrated Circuit) 控制器实现标准 I²C 主设备和从设备功能，兼容 Philips I²C 总线协议，可完成对 I²C 总线上的从设备的数据发送和接收，也可作为从设备对主设备的发送数据和接受数据请求做出相应的响应。。主要用于音视频 A/D、D/A 等外部 I²C 器件的控制以及多个 Hi3511/Hi3512 芯片的级联。

11.1.2 特性

I²C 控制器具备以下特性：

- 支持标准 I²C 总线协议
- 支持 7bit 和 10bit 从设备地址
- 支持可编程时钟，可实现通讯速率控制
- 支持 DMA (Directory Memory Access) 接口
- 支持中断和查询两种工作方式

11.1.3 信号描述

表11-1 I²C 接口信号描述

信号名称	方向	描述	对应管脚
I2C_SDA	I/O	I ² C 双向数据信号。与 GPIO 复用（复用时的配置信息请参见“ 管脚复用配置 ”）。	SDA
I2C_SCL	O	I ² C 输出时钟信号。与 GPIO 复用（复用时的配置信息请参见“ 管脚复用配置 ”）。	SDL

I²C 接口信号 I2C_SDA 和 I2C_SCL 使用 OD 门连接方式以符合 I²C 规范。

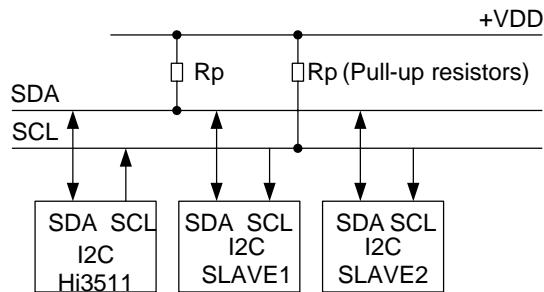
11.1.4 功能描述

典型应用

I²C 总线典型应用电路如[图 1-1](#) 所示。



图11-1 I²C 典型应用电路原理图



I²C 总线是一种双线、双向串行总线，它提供了一种简单有效的数据传输方式，可以很大的简化设备间的连接，非常适合多设备间的短距离少量数据传输。I²C 总线的灵活性使得系统开发和设备扩展容易实现。

功能原理

一次典型的 I²C 数据传输主要包括起始信号、从地址发送、数据传送、结束信号等部分。其时序如图 11-2 所示，数据帧格式如图 11-3 所示。

图11-2 I²C 数据传输时序图

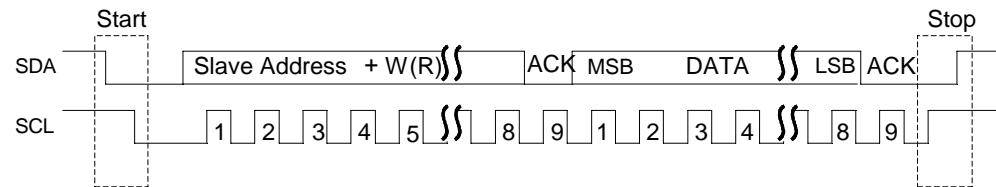
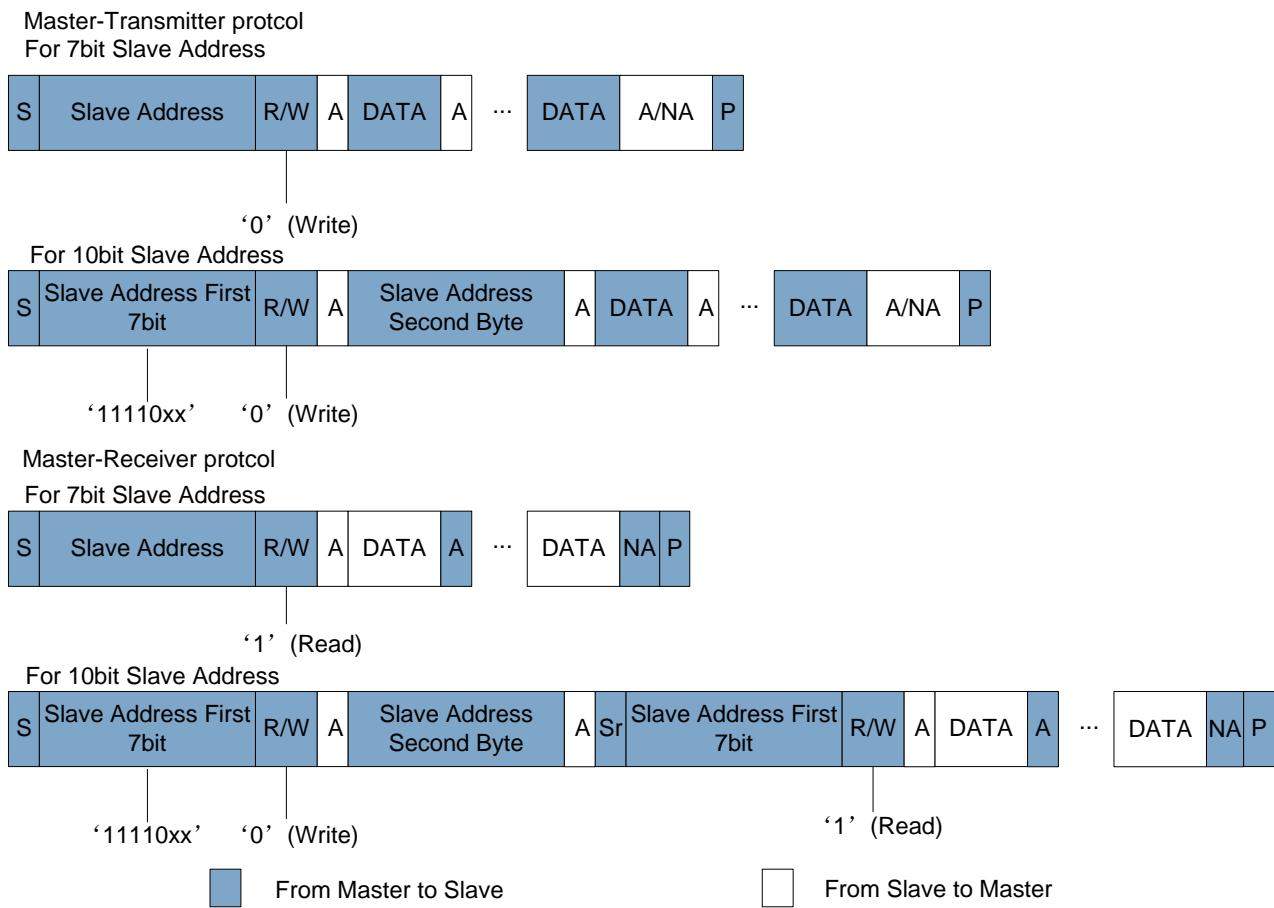


图11-3 I²C 数据传输帧格式图



A: Acknowledge(SDA low)

NA: No Acknowledge(SDA high)

S: Start Condition

Sr: repeated Start Condition

P: Stop Condition

起始信号（Start）是 I²C 标准协议规定，由总线上的主设备发出，用来唤醒所有从设备并指示数据传送开始的特殊信号。

从地址发送（Slave Address + W/R）是指主设备在发出起始信号后所发出的第一个数据包，包括 7 个从地址位（如果是 10bit 从地址，则需发送两个数据包）和 1 个读/写数据位。各从设备依靠该地址信息来确认是否需要传送或接收数据。当从地址被成功接收后，相应从设备会在第 9 个时钟周期（SCL）将 SDA（Serial Data）拉低，从而反馈给主设备一个应答信号（ACK），然后就可以开始进行后续的数据传输。

数据传送 (DATA) 是指主设备在成功接收到从地址响应信号后按照读、写命令进行相应数据接收或发送。在数据传输过程中 SDA 只能在 SCL 为低电平时变化，而在 SCL 高电平时保持；同时接收设备在每接收到 1 个字节后需要给发送设备发出应答信号 (ACK)。如果发送设备没有成功接收到应答信号，就会终止数据传输或者重新开始发送。



结束信号（Stop）是主设备在当前数据传输完成并且没有新的数据传输需要发起时，所发出的 I²C 标准协议规定的表示数据传输结束的特殊信号。当主设备发出结束信号后从设备必须释放总线。

11.1.5 工作方式

11.1.5.1 管脚复用配置

I²C 管脚与 GPIO 复用，使用 I²C 前，需要配置系统控制器使能相应管脚的 I²C 功能，详见 SC_PERCTRL1[14]配置说明。

11.1.5.2 时钟配置

I²C 主设备模式下需要通过配置 I2C_SS_SCL_HCNT、I2C_SS_SCL_LCNT、I2C_FS_SCL_HCNT、I2C_FS_SCL_LCNT 等寄存器可以设置标准模式和快速模式下 I²C 总线 SCL 信号的高低电平宽度，即相对于 I²C 工作时钟的周期数，其配置值的计算公式分别为：* HCNT = T_{scl_h} × F_{I2C} - 8 和 * LCNT = T_{scl_l} × F_{I2C} - 1



说明

- T_{scl_h} 为 SCL 高电平宽度。
- T_{scl_l} 为 SCL 低电平宽度，单位 μ s。
- F_{I2C} 为 I²C 工作时钟，单位 MHz。
- I²C 从设备模式下无需配置这 4 个寄存器。

I2C_SS_SCL_HCNT、I2C_SS_SCL_LCNT、I2C_FS_SCL_HCNT 和 I2C_FS_SCL_LCNT 典型配置值如表 1-2~表 1-5 所示。

表11-2 I2C_SS_SCL_HCNT 典型配置值

I ² C 总线速率 (kbit/s)	I ² C 工作时钟 (MHz)	SCL 高电平宽度 (μ s)	I2C_SS_SCL_HCNT (个)
100	135	4	532

表11-3 I2C_SS_SCL_LCNT 典型配置值

I ² C 总线速率 (kbit/s)	I ² C 工作时钟 (MHz)	SCL 低电平宽度 (μ s)	I2C_SS_SCL_LCNT (个)
100	135	4.7	634

表11-4 I2C_FS_SCL_HCNT 典型配置值

I ² C 总线速率 (kbit/s)	I ² C 工作时钟 (MHz)	SCL 高电平宽度 (μ s)	I2C_FS_SCL_HCNT (个)
400	135	0.6	73

表11-5 I2C_FS_SCL_LCNT 典型配置值

I ² C 总线速率 (kbit/s)	I ² C 工作时钟 (MHz)	SCL 低电平宽度 (μs)	I2C_FS_SCL_LCNT (个)
400	135	1.3	174

11.1.5.3 软复位

通过配置系统控制器 SC_PERCTRL0[25]为 1，可实现对 I²C 控制器的单独软复位。复位后各配置寄存器的值均为默认值，因此复位后需要重新对这些寄存器进行初始化配置。

11.1.5.4 中断或查询方式下的数据传输

初始化

主设备模式初始化步骤如下：

- 步骤 1 设定超时标志，检测 I2C_STATUS[0]（判断 I²C 控制器是否空闲），如果是 1，则延时等待，继续检测，直到 I2C_STATUS[0]变为 0 或者超时，然后向 I2C_ENABLE[0]写 0，使 I²C 处于关闭状态。
- 步骤 2 写相应的配置值到 I2C_CON，配置主模式以及传输速率模式等参数。
- 步骤 3 将对接器件的从设备地址写入 I2C_TAR[9:0]。
- 步骤 4 配置 I2C_SS_SCL_HCNT、I2C_SS_SCL_LCNT 等寄存器，设定相应的 I²C 总线时钟周期。
- 步骤 5 配置 I2C_RX_TL、I2C_TX_TL，设定相应的发送及接收 FIFO 水线值。
- 步骤 6 如果驱动程序采用中断方式则需设定 I2C_INTR_MASK，使能相应中断信号；采用查询方式时应禁止产生相应中断信号。
- 步骤 7 向 I2C_ENABLE[0]写 1，使能 I²C，完成初始化配置。

----结束

从设备模式初始化步骤如下：

- 步骤 1 设定超时标志，检测 I2C_STATUS[0]（判断 I²C 控制器是否空闲），如果是 1，则延时等待，继续检测，直到 I2C_STATUS[0]变为 0 或者超时，然后向 I2C_ENABLE[0]写 0，使 I²C 处于关闭状态。
- 步骤 2 写相应的配置值到 I2C_CON，配置从模式以及传输模式等参数。
- 步骤 3 将作为从设备模式时响应的从地址写入 I2C_SAR[9:0]。
- 步骤 4 配置 I2C_RX_TL、I2C_TX_TL，设定相应的发送及接收 FIFO 水线值。
- 步骤 5 如果驱动程序采用中断方式则需设定 I2C_INTR_MASK，使能相应中断信号；采用查询方式时应禁止产生相应中断信号。



步骤 6 向 **I2C_ENABLE[0]** 写 1，使能 I²C，完成初始化配置。

----结束

数据发送

主设备模式数据发送步骤如下：

步骤 1 将发送数据写入 **I2C_DATA_CMD**，启动数据发送。

步骤 2 查询方式下，进行连续数据发送时通过读取 **I2C_STATUS** 和 **I2C_TXFLR** 检测 TX_FIFO 状态；中断方式下，则根据相应中断状态位检测。在数据传送完成之前，既要保证 TX_FIFO 中的数据没有溢出（否则会造成数据丢失），同时也要保证 TX_FIFO 中始终有数据（否则 I²C 会认为该次传输已全部完成而发出结束信号）。

步骤 3 通过检测 **I2C_STATUS[2]** 是否为 1，判断 I²C 是否完成全部数据发送。

----结束

从设备模式数据发送步骤如下：

步骤 1 I²C 总线上的其它主设备发起数据接收操作，并且该操作的地址与 **I2C_SAR** 寄存器中的值相匹配，I²C 使能 **I2C_RAW_INTR_STAT** 寄存器的 **r_rd_req** 位，在中断模式下使能中断。

步骤 2 查询方式下，软件检测到 **I2C_RAW_INTR_STAT** 寄存器 **r_rd_req** 位有效，将需要发送的数据写入 **I2C_DATA_CMD** 寄存器；中断方式下，则根据相应中断状态位检测，发现是 **r_rd_req** 中断，即将需要发送的数据写入 **I2C_DATA_CMD** 寄存器。

步骤 3 通过检测 **I2C_STATUS[2]** 是否为 1，判断 I²C 是否完成数据发送。

----结束

数据接收

主设备模式数据接收步骤如下：

步骤 1 将读命令（0x100）写入 **I2C_DATA_CMD**，启动数据接收。

步骤 2 进行连续数据接收时，首先要发送次数与接收数据数目相同的读命令（0x100）到 **I2C_DATA_CMD**（例如要接收 3 个数据，则需发送 3 次读命令到 **I2C_DATA_CMD** 寄存器）。在数据接收完成之前，既要保证 TX_FIFO 中的数据（即读命令 0x100）没有溢出，又保证 TX_FIFO 非空（否则 I²C 会认为没有新的数据需要接收而结束），另外在连续数据接收过程中还需要检测 RX_FIFO 的状态（如果采用中断方式，则根据相应中断状态位检测），避免 RX_FIFO 溢出。

步骤 3 通过检测 **I2C_STATUS[0]** 是否为 0，判断 I²C 是否完成全部数据接收。

----结束

从设备模式数据接收步骤如下：

步骤 1 I²C 总线上的其它主设备发起数据接收操作，并且该操作的地址与 I2C_SAR 寄存器中的值相匹配。

步骤 2 I²C 接收数据，将其存放在接收 FIFO 中。在连续数据接收过程中需要检测 RX_FIFO 的状态（如果采用中断方式，则根据相应中断状态位检测），避免 RX_FIFO 溢出。

----结束

11.1.5.5 DMA 方式下的数据传输

初始化

初始化步骤如下：

步骤 1 设定超时标志，检测 I2C_STATUS[0]（判断 I²C 控制器是否空闲），如果是 1，则延时等待，继续检测，直到 I2C_STATUS[0] 变为 0 或者超时，然后向 I2C_ENABLE[0] 写 0，使 I²C 处于关闭状态；

步骤 2 写相应的配置值到 I2C_CON，配置主模式以及传输速率模式等参数；

步骤 3 主设备模式下将对接器件的从设备地址写入 I2C_TAR[9:0]；

步骤 4 配置 I2C_SS_SCL_HCNT、I2C_SS_SCL_LCNT 等寄存器，设定相应的 I²C 总线时钟周期；

步骤 5 配置 I2C_DMA_TDLR、I2C_DMA_RDLR，设定 DMA 发送及接收 FIFO 水线值；

步骤 6 向 I2C_ENABLE[0] 写 1，使能 I²C，完成初始化配置。

----结束

数据发送

数据发送步骤如下：

步骤 1 配置 DMA 数据通道，包括数据传输源和目的地址、数据传输个数、传输类型等参数。

步骤 2 配置 I2C_DMA_CR 为 0x2，使能 I²C 的 DMA 发送功能。

步骤 3 通过 DMA 中断状态查询，判断数据是否发送完成，如果完成则关闭 I²C 的 DMA 发送功能。

----结束

数据接收

数据接收包括数据发送和数据接收两步，首先将读命令发送给 I²C 控制器，然后从接收 FIFO 中读取接收数据，步骤如下：

步骤 1 配置 DMA 数据通道，包括数据传输源（传输源内容包括读命令 0x100 等）和目的地址、数据接收区地址、数据传输个数、传输类型等参数。



步骤 2 配置 `I2C_DMA_CR` 为 0x3，使能 I²C 的 DMA 发送和接收功能（因数据接收时要先发送 0x100 到 `I2C_DATA_CMD` 寄存器，所以要使能 DMA 发送功能）。

步骤 3 通过 DMA 中断状态查询，判断数据是否接收完成，如果完成则关闭 I²C 的 DMA 接收功能。

----结束

11.1.6 寄存器概览

表11-6 I²C 寄存器概览（基址是 0x101F_6000）

偏移地址	名称	描述	页码
0x000	I2C_CON	I ² C 控制寄存器	11-10
0x004	I2C_TAR	I ² C 访问从设备地址寄存器	11-11
0x008	I2C_SAR	I ² C 从设备模式下自身地址寄存器	11-12
0x010	I2C_DATA_CMD	I ² C 数据通道寄存器	11-12
0x014	I2C_SS_SCL_HCNT	标准速度下的 SCL 时钟高电平时间配置寄存器	11-13
0x018	I2C_SS_SCL_LCNT	标准速度下的 SCL 时钟低电平时间配置寄存器	11-13
0x01C	I2C_FS_SCL_HCNT	快速速度下的 SCL 时钟高电平时间配置寄存器	11-14
0x020	I2C_FS_SCL_LCNT	快速速度下的 SCL 时钟低电平时间配置寄存器	11-14
0x02C	I2C_INTR_STAT	中断状态寄存器	11-15
0x030	I2C_INTR_MASK	中断屏蔽寄存器	11-16
0x034	I2C_RAW_INTR_STAT	原始中断状态寄存器	11-18
0x038	I2C_RX_TL	RX_FIFO 的水线配置寄存器	11-19
0x03C	I2C_TX_TL	TX_FIFO 的水线配置寄存器	11-20
0x040	I2C_CLR_INTR	组合及独立中断清除寄存器	11-20
0x044	I2C_CLR_RX_UNDER	RX_UNDER 中断清除寄存器	11-21
0x048	I2C_CLR_RX_OVER	RX_OVER 中断清除寄存器	11-21
0x04C	I2C_CLR_TX_OVER	TX_OVER 中断清除寄存器	11-22
0x054	I2C_CLR_TX_ABRT	ABRT 中断清除寄存器	11-22
0x05C	I2C_CLR_ACTIVITY	ACTIVITY 状态寄存器	11-23

偏移地址	名称	描述	页码
0x060	I2C_CLR_STOP_DET	STOP_DET 中断清除寄存器	11-23
0x064	I2C_CLR_START_DET	START_DET 中断清除寄存器	11-24
0x068	I2C_CLR_GEN_CALL	GEN_CALL 中断清除寄存器	11-24
0x06C	I2C_ENABLE	I ² C 工作模式使能寄存器	11-25
0x070	I2C_STATUS	I ² C 状态寄存器	11-25
0x074	I2C_TXFLR	TX_FIFO 中的数据个数指示寄存器	11-26
0x078	I2C_RXFLR	RX_FIFO 中的数据个数指示寄存器	11-27
0x080	I2C_TX_ABRT_SOUR CE	TX_ABRT 的源头中断寄存器	11-27
0x088	I2C_DMA_CR	I ² C DMA 通道开启控制寄存器	11-29
0x08C	I2C_DMA_TDLR	TX_FIFO 进行 DMA 操作时的水线 配置寄存器	11-29
0x090	I2C_DMA_RDLR	RX_FIFO 进行 DMA 操作时的水线 配置寄存器	11-30

11.1.7 寄存器描述

I2C_CON

I2C_CON 为 I²C 控制寄存器，只在 I²C 被禁止（即 I2C_ENABLE[0] 为 0）时才可配置。

Offset Address				Register Name								Total Reset Value				
0x0				I2C_CON								0x0075				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved								reserved	i2c_10bitaddr_master	reserved	reserved	speed	master_mode		
Reset	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1
Bits	Access	Name		Description												
[15:7]	-	reserved		保留。												
[6]	RW	reserved		保留，此位只能设置为 1。												



[5]	RW	i2c_restart_en	主设备产生“重新开始”条件使能位。 0: 禁止, 此时无法实现“重新开始”条件的功能。 1: 使能。
[4]	RW	i2c_10bitaddr_master	发送 7 位地址/10 位地址选择。 0: 7 位地址。 1: 10 位地址。
[3]	-	reserved	保留。
[2:1]	RW	speed	I ² C 操作速度的选择。 00: 非法, 但写 00 会被认为配置为快速模式。 01: 标准速度。 10: 快速速度。 11: 保留。 如配置时写入 11, 会被认为是写入 10。
[0]	RW	master_mode	主设备功能使能。 0: 禁止。 1: 使能。

I2C_TAR

I2C_TAR 为 I²C 访问从设备地址寄存器, 只在 I²C 被禁止 (即 [I2C_ENABLE](#) 寄存器设置为 0) 的时候才可配置。

Offset Address								Register Name								Total Reset Value							
0x4								I2C_TAR								0x009C							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved				special	gc_or_start	i2c_tar																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0							
Bits	Access	Name			Description																		
[15:12]	-	reserved			保留。																		
[11]	RW	special			general call 和 start byte 功能使能。 0: 禁止。 1: 使能。																		

[10]	RW	gc_or_start	如果 bit[11]为 1, 决定执行功能是 general call 还是 start byte。 0: general call。 1: start byte。
[9:0]	RW	i2c_tar	I ² C 作为主设备时要访问的从设备的地址。

I2C_SAR

I2C_SAR 为 I²C 作为从设备时自身的地址寄存器, 只在 I²C 被禁止 (即 [I2C_ENABLE](#) 寄存器设置为 0) 时才可配置。

Offset Address				Register Name								Total Reset Value				
0x008				I2C_SAR								0x0055				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved														i2c_sar	
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	
Bits	Access	Name			Description											
[15:10]	-	reserved			保留。											
[9:0]	RW	i2c_sar			I ² C 作为从设备时自身的地址。											

I2C_DATA_CMD

I2C_DATA_CMD 为 I²C 数据通道寄存器。

Offset Address				Register Name								Total Reset Value				
0x010				I2C_DATA_CMD								0x0000				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved							cmd	dat							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name			Description											
[15:9]	-	reserved			保留。											



[8]	RW	cmd	读/写控制位。 0: 写操作, 表示 I ² C 控制器将要向 I ² C 总线发送数据, 此时低 8 位 (DAT) 是 I ² C 要向 I ² C 总线发送的数据。 1: 读操作, 表示 I ² C 控制器将要从 I ² C 总线读回数据。
[7:0]	RW	dat	将要在 I ² C 总线上发送/接收的数据。 读这 8 位会读出在 I ² C 总线上接收的数据。 写这 8 位会把写入的数据发送到 I ² C 总线上去。

I2C_SS_SCL_HCNT

I2C_SS_SCL_HCNT 为标准速度下的 SCL 时钟高电平时间配置寄存器, 只在 I²C 被禁止 (即 I2C_ENABLE 寄存器设置为 0) 的时候才可配置。

Offset Address				Register Name								Total Reset Value				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	i2c_ss_scl_hcnt															
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	
Bits	Access	Name			Description											
[15:0]	RW	i2c_ss_scl_hcnt			标准速度下的 SCL 时钟高电平时间, 计算公式请参见“ 时钟配置 ”。 配置的最小值为 6, 写入小于 6 的值时会被认为是 6。											

I2C_SS_SCL_LCNT

I2C_SS_SCL_LCNT 为标准速度下的 SCL 时钟低电平时间配置寄存器, 只在 I²C 被禁止 (即 I2C_ENABLE 寄存器设置为 0) 的时候才可配置。

Offset Address								Register Name								Total Reset Value																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	I2C_SS_SCL_LCNT								0x008F							
Name																	i2c_ss_scl_lcnt															
Reset	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1																
Bits	Access	Name								Description																						
[15:0]	RW	i2c_ss_scl_lcnt								标准速度下的 SCL 时钟低电平时间，计算公式请参见“ 时钟配置 ”。 配置的最小值为 8，写入小于 8 的值时会被认为是 8。																						

I2C_FS_SCL_HCNT

I2C_FS_SCL_HCNT 为快速速度下的 SCL 时钟高电平时间配置寄存器，只在 I²C 被禁止（即 I2C_ENABLE 寄存器设置为 0）的时候才可配置。

Offset Address								Register Name								Total Reset Value								
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0x0013							
Name																	i2c_fs_scl_hcnt							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	1	1
Bits	Access	Name								Description														
[15:0]	RW	i2c_fs_scl_hcnt								快速速度下的 SCL 时钟高电平时间，计算公式请参见“ 时钟配置 ”。 配置的最小值为 6，写入小于 6 的值时会被认为是 6。														

I2C_FS_SCL_LCNT

I2C_FS_SCL_LCNT 为快速速度下的 SCL 时钟低电平时间配置寄存器，只在 I²C 被禁止（即 I2C_ENABLE 寄存器设置为 0）的时候才可配置。



Offset Address								Register Name								Total Reset Value							
0x020								I2C_FS_SCL_LCNT								0x0028							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	i2c_fs_scl_lcnt																						
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0							
Bits	Access	Name			Description																		
[15:0]	RW	i2c_fs_scl_lcnt			快速速度下的 SCL 时钟低电平时间，计算公式请参见“ 时钟配置 ”。 配置的最小值为 8，写入小于 8 的值时会被认为是 8。																		

I2C_INTR_STAT

I2C_INTR_STAT 为中断状态寄存器。

Offset Address								Register Name								Total Reset Value							
0x02C								I2C_INTR_STAT								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved				gen_call	start_det	stop_det	activity	rx_done	tx_abrt	rd_req	tx_empty	tx_over	rx_full	rx_over	rx_under							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:12]	-	reserved			保留。																		
[11]	RO	gen_call			一次 general call 请求被接收状态。 0: 未被接收。 1: 已被接收。 I ² C 将接收到的数据放在 RX Buffer 中。																		
[10]	RO	start_det			指示在 I ² C 总线上是否发生了开始条件。 0: 未发生开始条件。 1: 已发生开始条件。																		
[9]	RO	stop_det			指示在 I ² C 总线上是否发生了停止条件。 0: 未发生停止条件。 1: 已发生停止条件。																		

[8]	RO	activity	指示 I ² C 的 ACTIVITY 状态。 0: 空闲。 1: 忙。
[7]	RO	rx_done	当 I ² C 作为从 Slave 时, 指示数据接收是否完成。 0: 未完成。 1: 已完成。
[6]	RO	tx_abrt	有多种情况可以触发此位。详细描述请参见“ I2C_TX_ABRT_SOURCE ”。
[5]	RO	rd_req	当 I ² C 作为从 Slave 时, 指示是否有 Master 设备发起读数据请求。 0: 无请求。 1: 有请求。
[4]	RO	tx_empty	指示 TX_FIFO 中数据是否到达或低于水线值。 0: 大于水线。 1: 等于或低于水线。
[3]	RO	tx_over	TX_FIFO 溢出标志。 0: 未溢出。 1: 溢出。
[2]	RO	rx_full	指示 RX_FIFO 中数据是否到达或大于水线值。 0: 小于水线。 1: 等于或大于水线。
[1]	RO	rx_over	RX_FIFO 溢出标志。 0: 未溢出。 1: 溢出。
[0]	RO	rx_under	0: 默认值。 1: RX_FIFO 为空时, CPU 读 I2C_DATA_CMD 。

I2C_INTR_MASK

I2C_INTR_MASK 为中断屏蔽寄存器。



Bit	Offset Address 0x030				Register Name I2C_INTR_MASK								Total Reset Value 0x08FF			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved				m_gen_call	m_stop_det	m_start_det	m_activity	m_rx_done	m_tx_abrt	m_rd_req	m_tx_empty	m_rx_over	m_tx_full	m_rx_under	
Reset	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1
Bits	Access	Name			Description											
[15:12]	-	reserved			保留。											
[11]	RW	m_gen_call			GEN_CALL 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[10]	RW	m_start_det			START_DET 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[9]	RW	m_stop_det			STOP_DET 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[8]	RW	m_activity			ACTIVITY 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[7]	RW	m_rx_done			RX_DONE 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[6]	RW	m_tx_abrt			TX_ABRT 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[5]	RW	m_rd_req			RD_REQ 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											
[4]	RW	m_tx_empty			TX_EMPTY 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。											

[3]	RW	m_tx_over	TX_OVER 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[2]	RW	m_rx_full	RX_FULL 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[1]	RW	m_rx_over	RX_OVER 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[0]	RW	m_rx_under	RX_UNDER 中断屏蔽。 0: 屏蔽该中断。 1: 不屏蔽该中断。

I2C_RAW_INTR_STAT

I2C_RAW_INTR_STAT 为原始中断状态寄存器。

Offset Address				Register Name								Total Reset Value				
0x034				I2C_RAW_INTR_STAT								0x0000				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved				r_gen_call	r_start_det	r_stop_det	r_activity	r_rx_done	r_tx_abrt	r_rd_req	r_tx_empty	r_tx_over	r_rx_full	r_rx_over	r_rx_under
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name			Description											
[15:12]	-	reserved			保留。											
[11]	RO	r_gen_call			GEN_CALL 原始中断状态。 0: 未产生中断。 1: 已产生中断。											
[10]	RO	r_start_det			START_DET 原始中断状态。 0: 未产生中断。 1: 已产生中断。											
[9]	RO	r_stop_det			STOP_DET 原始中断状态。 0: 未产生中断。 1: 已产生中断。											



[8]	RO	r_activity	ACTIVITY 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[7]	RO	r_rx_done	RX_DONE 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[6]	RO	r_tx_abrt	TX_ABRT 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[5]	RO	r_rd_req	RD_REQ 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[4]	RO	r_tx_empty	TX_EMPTY 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[3]	RO	r_tx_over	TX_OVER 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[2]	RO	r_rx_full	RX_FULL 原始中断状态。 0: 未产生中断。 1: 发生中断。
[1]	RO	r_rx_over	RX_OVER 原始中断状态。 0: 未产生中断。 1: 已产生中断。
[0]	RO	r_rx_under	RX_UNDER 原始中断状态。 0: 未产生中断。 1: 已产生中断。

I2C_RX_TL

I2C_RX_TL 为 RX_FIFO 的水线配置寄存器。

Offset Address								Register Name								Total Reset Value							
Bit	0x038				I2C_RX_TL				0x0003														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1						
Name	reserved												rx_tl										
Bits	Access	Name				Description																	
[15:8]	-	reserved				保留。																	
[7:0]	RW	rx_tl				RX_FIFO 的水线值, 实际值等于配置值加 1。当配置值超过 FIFO 深度 (8) 时被认为是 8。																	

I2C_TX_TL

I2C_TX_TL 为 TX_FIFO 的水线配置寄存器。

Offset Address								Register Name								Total Reset Value							
Bit	0x03C				I2C_TX_TL				0x0003														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1						
Name	reserved												tx_tl										
Bits	Access	Name				Description																	
[15:8]	RW	reserved				保留。																	
[7:0]	RW	tx_tl				TX_FIFO 的水线值。当配置值超过 FIFO 深度 (8) 时被认为是 8。																	

I2C_CLR_INTR

I2C_CLR_INTR 为组合及独立中断清除寄存器。



Offset Address								Register Name								Total Reset Value							
0x040								I2C_CLR_INTR								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved															clr_intr							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:1]	-	reserved			保留。																		
[0]	RO	clr_intr			读此寄存器，清所有组合中断和独立中断、 以及 I2C_TX_ABRT_SOURCE 寄存器。 注：I2C_TX_ABRT_SOURCE[9]及其引发的 组合中断无法被清除。																		

I2C_CLR_RX_UNDER

I2C_CLR_RX_UNDER 为 RX_UNDER 中断清除寄存器。

Offset Address								Register Name								Total Reset Value							
0x044								I2C_CLR_RX_UNDER								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved															clr_rx_under							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:1]	-	reserved			保留。																		
[0]	RO	clr_rx_under			读此寄存器，清 RX_UNDER 中断。																		

I2C_CLR_RX_OVER

I2C_CLR_RX_OVER 为 RX_OVER 中断清除寄存器。

Offset Address								Register Name								Total Reset Value																				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	I2C_CLR_RX_OVER								0x0000											
Name	reserved															clr_rx_over																				
Reset	0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0																			
Bits	Access	Name			Description																															
[15:1]	-	reserved			保留。																															
[0]	RO	clr_rx_over			读此寄存器，清 RX_OVER 中断。																															

I2C_CLR_RX_OVER

I2C_CLR_RX_OVER 为 TX_OVER 中断清除寄存器。

Offset Address								Register Name								Total Reset Value								0x0000												
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	I2C_CLR_RX_OVER								0x0000											
Name	reserved															clr_tx_over																				
Reset	0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0				0 0 0 0																			
Bits	Access	Name			Description																															
[15:1]	-	reserved			保留。																															
[0]	RO	clr_tx_over			读此寄存器，清 TX_OVER 中断。																															

I2C_CLR_TX_ABRT

I2C_CLR_TX_ABRT 为 ABRT 中断清除寄存器。



Offset Address								Register Name								Total Reset Value							
0x054								I2C_CLR_TX_ABRT								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved															clr_tx_abrt							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:1]	-	reserved			保留。																		
[0]	RO	clr_tx_abrt			读此寄存器，清 TX_ABRT 中断，以及 I2C_TX_ABRT_SOURCE 寄存器。																		

I2C_CLR_ACTIVITY

I2C_CLR_ACTIVITY 为 ACTIVITY 状态寄存器。

Offset Address								Register Name								Total Reset Value							
0x05C								I2C_CLR_ACTIVITY								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved															clr_activity							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:1]	-	reserved			保留。																		
[0]	RO	CLR_ACTIVITY			读此寄存器可获得 ACTIVITY 中断状态，硬件自动清 0。																		

I2C_CLR_STOP_DET

I2C_CLR_STOP_DET 为 STOP_DET 中断清除寄存器。

Offset Address								Register Name								Total Reset Value							
Bit	0x060				I2C_CLR_STOP_DET				0x0000														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Name	reserved														clr_stop_det								
Bits	Access	Name				Description																	
[15:1]	-	reserved				保留。																	
[0]	RO	clr_stop_det				读此寄存器，清 STOP_DET 中断。																	

I2C_CLR_START_DET

I2C_CLR_START_DET 为 START_DET 中断清除寄存器。

Offset Address								Register Name								Total Reset Value							
Bit	0x064				I2C_CLR_START_DET				0x0000														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Name	reserved														clr_start_det								
Bits	Access	Name				Description																	
[15:1]	-	reserved				保留。																	
[0]	RO	clr_start_det				读此寄存器，清 START_DET 中断。																	

I2C_CLR_GEN_CALL

I2C_CLR_GEN_CALL 为 GEN_CALL 中断清除寄存器。



Offset Address				Register Name					Total Reset Value					
Bit	0x068				I2C_CLR_GEN_CALL					0x0000				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Name	reserved													clr_gen_call
Bits	Access	Name			Description									
[15:1]	-	reserved			保留。									
[0]	RO	clr_gen_call			读此寄存器，清 GEN_CALL 中断。									

I2C_ENABLE

I2C_ENABLE 为 I²C 控制器使能寄存器，用于关闭或使能 I²C 控制器。

当 I²C 控制器处于数据传输状态时 (I2C_STATUS 的 ACTIVITY 位为 1) 控制器可以被关闭，但需要注意：

- 如果 I²C 控制器处于发送状态时关闭，则控制器在完成当前字节发送后停止继续传输，同时删除 TX_FIFO 的数据。
- 如果 I²C 控制器处于接收状态时关闭，则控制器在接收完当前字节后不响应这次传输，即发出 NACK。

Offset Address				Register Name					Total Reset Value						
Bit	0x06C				I2C_ENABLE					0x0000					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0		
Name	reserved													enable	
Bits	Access	Name			Description										
[15:1]	-	reserved			保留。										
[0]	RW	enable			I ² C 的使能寄存器。 0: 禁止。 1: 使能。										

I2C_STATUS

I2C_STATUS 为 I²C 状态寄存器。

Offset Address								Register Name								Total Reset Value							
0x070								I2C_STATUS								0x0006							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved														rff	rfne	tfe	tnf	activity				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0							
Bits	Access	Name		Description																			
[15:5]	-	reserved		保留。																			
[4]	RO	rff		指示 RX_FIFO 是否已满。 0: RX_FIFO 未满。 1: RX_FIFO 已满。																			
[3]	RO	rfne		指示 RX_FIFO 是否已空。 0: RX_FIFO 已空。 1: RX_FIFO 未空。																			
[2]	RO	tfe		指示 TX_FIFO 是否已空。 0: TX_FIFO 未空。 1: TX_FIFO 已空。																			
[1]	RO	tnf		指示 TX_FIFO 是否已满。 0: TX_FIFO 已满。 1: TX_FIFO 未满。																			
[0]	RO	activity		I ² C 总线状态。 0: 空闲。 1: 忙。																			

I2C_TXFLR

I2C_TXFLR 为 TX_FIFO 中的数据个数指示寄存器。



Offset Address								Register Name								Total Reset Value							
0x074								I2C_TXFLR								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved												txflr										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name				Description																	
[15:4]	-	reserved				保留。																	
[3:0]	RO	txflr				指示 TX_FIFO 中的数据个数。																	

I2C_RXFLR

I2C_RXFLR 为 RX_FIFO 中的数据个数指示寄存器。

Offset Address								Register Name								Total Reset Value							
0x078								I2C_RXFLR								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved												rxflr										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name				Description																	
[15:4]	-	reserved				保留。																	
[3:0]	RO	rxflr				指示 RX_FIFO 中的数据个数。																	

I2C_TX_ABRT_SOURCE

I2C_TX_ABRT_SOURCE 为 TX_ABRT 的源头中断寄存器。



Offset Address				Register Name				Total Reset Value								
Bit	0x080				I2C_TX_ABRT_SOURCE				0x0000							
Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name			Description											
[15:12]	-	reserved			保留。											
[11]	RO	arb_master_dis			0: 复位值。 1: Master 功能禁用的情况下, 尝试发起 Master 操作。											
[10]	RO	abrt_10b_rd_norstt			0: 复位值。 1: 不支持 restart 功能时, 作为主设备时对 10 位地址的从设备发出了读命令。											
[9]	RO	abrt_sbyte_norstt			0: 复位值。 1: 不支持 restart 功能时, 作为主设备时尝试发送 start byte。											
[8]	RO	abrt_hs_norstt			0: 复位值。 1: 不支持 restart 功能时, 作为主设备时尝试高速操作。											
[7]	RO	abrt_sbyte_ackdet			0: 复位值。 1: 作为主设备时发出 start byte 而被响应。											
[6]	RO	abrt_hs_ackdet			0: 复位值。 1: 作为主设备要进行高速传输时, 高速主机码被响应。											
[5]	RO	abrt_gcall_read			0: 复位值。 1: 作为主设备时发出 general call, 而 CPU 向 I ² C 发出读命令。											
[4]	RO	abrt_gcall_noack			0: 复位值。 1: 作为主设备时发出 general call, 但没被响应。											



[3]	RO	abrt_txdata_noack	0: 复位值。 1: 作为主设备发送器, 发送的地址被从设备响应, 而发送的数据没被响应。
[2]	RO	abrt_10addr2_noack	0: 复位值。 1: 作为主设备时, 发送的 10 位地址的第 2 字节没被响应。
[1]	RO	abrt_10addr1_noack	0: 复位值。 1: 作为主设备时, 发送的 10 位地址的第 1 字节没被响应。
[0]	RO	abrt_7b_addr_noack	0: 复位值。 1: 作为主设备时, 发送的 7 位地址没被响应。

I2C_DMA_CR

I2C_DMA_CR 为 I²C DMA 通道开启控制寄存器。

Offset Address				Register Name								Total Reset Value				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved														tdmae	rdmae
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	Access	Name	Description													
[15:2]	-	reserved	保留。													
[1]	RW	tdmae	是否打开 TX_FIFO 的 DMA 通道。 0: 不打开。 1: 打开。													
[0]	RW	rdmae	是否打开 RX_FIFO 的 DMA 通道。 0: 不打开。 1: 打开。													

I2C_DMA_TDRL

I2C_DMA_TDRL 为 TX_FIFO 进行 DMA 操作时的水线配置寄存器。



Offset Address								Register Name								Total Reset Value							
0x08C								I2C_DMA_TDLR								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved												dmatdl										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:3]	-	reserved			保留。																		
[2:0]	RW	dmatdl			TX_FIFO DMA 操作时的水线值。																		

I2C_DMA_RDLR

I2C_DMA_RDLR 为 RX_FIFO 进行 DMA 操作时的水线配置寄存器。

Offset Address								Register Name								Total Reset Value							
0x090								I2C_DMA_RDLR								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved												dmardl										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
Bits	Access	Name			Description																		
[15:3]	-	reserved			保留。																		
[2:0]	RW	dmardl			RX_FIFO DMA 操作时的水线值, 实际值等于配置值加 1。																		

11.2 通用异步收发器

11.2.1 概述

通用异步收发器 UART (Universal Asynchronous Receiver Transmitter) 是一个异步串行的通信接口, 主要功能是将来自外围设备的数据进行串并转换之后传入内部总线, 以及将数据进行并串转换之后输出到外部设备。UART 的主要功能是和外部芯片的 UART 的对接, 从而实现两芯片间的通信。

Hi3511/Hi3512 提供了以下 3 个 UART 单元:

- **UART0**
主要用于调试。



- **UART1**
主要用于接 RS-485 总线和控制云台。
- **UART2**
主要用于扩展接口，如外部 MCU (Micro Controller Unit)。

11.2.2 特点

UART 模块有以下特点：

- 支持 $16 \times 8\text{bit}$ 的发送 FIFO 和 $16 \times 12\text{bit}$ 的接收 FIFO。
- 支持数据位和停止位位宽可编程。数据位可通过编程设定为 5 位、6 位、7 位或 8 位；停止位可通过编程设定为 1 位或 2 位。
- 支持奇、偶校验方式或者无校验。
- 支持传输速率可编程。
- 支持接收 FIFO 中断、发送 FIFO 中断、接收超时中断、错误中断。
- 支持初始中断状态查询和屏蔽后中断状态查询。
- 支持通过编程禁止 UART 模块或者 UART 发送/接收功能以降低功耗。
- 支持关断 UART 时钟以节省功耗。
- **UART0** 支持 DMA 操作，**UART1** 和 **UART2** 不支持。

11.2.3 信号描述

UART0 接口信号如表 11-7 所示。

表11-7 UART0 接口信号描述

信号名称	方向	描述	对应管脚
UART0_RXD	I	UART0 接收数据信号。	URXD0
UART0_TXD	O	UART0 发送数据信号。	UTXD0

UART1 接口信号如表 11-8 所示。

表11-8 UART1 接口信号描述

信号名称	方向	描述	对应管脚
UART1_RXD	I	UART1 接收数据信号。与 GPIO 复用。（复用时的配置信息请参见“ 11.2.5.1 管脚复用配置 ”）	URXD1
UART1_TXD	O	UART1 发送数据信号。与 GPIO 复用。（复用时的配置信息请参见“ 11.2.5.1 管脚复用配置 ”）	UTXD1

信号名称	方向	描述	对应管脚
UART1_RTS	O	UART1 请求发送信号。与 GPIO 复用。（复用时的配置信息请参见“ 11.2.5.1 管脚复用配置 ”）	URTSN1
UART1_CTS	I	UART1 清除发送信号。与 GPIO 复用。（复用时的配置信息请参见“ 11.2.5.1 管脚复用配置 ”）	UCTSN1

UART2 接口信号如表 11-9 所示。

表11-9 UART2 接口信号描述

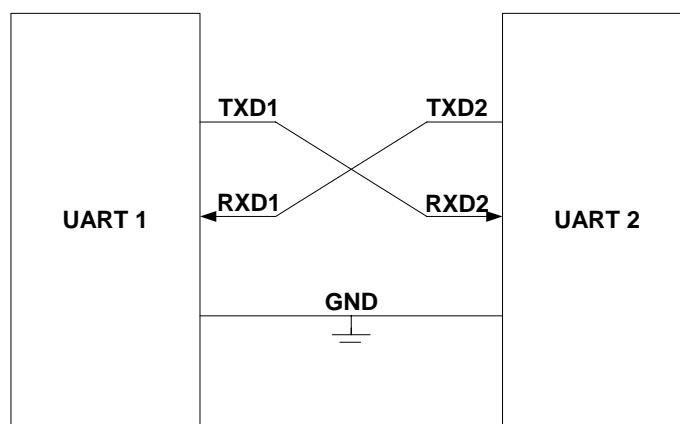
信号名称	方向	描述	对应管脚
UART2_RXD	I	UART2 接收数据信号。与 VI 和 GPIO 接口复用。（复用时的配置信息请参见“ 11.2.5.1 管脚复用配置 ”）	VI2HS
UART2_TXD	O	UART2 发送数据信号。与 VI 和 GPIO 接口复用。（复用时的配置信息请参见“ 11.2.5.1 管脚复用配置 ”）	VI2VS

11.2.4 功能描述

应用框图

UART 的典型应用框图如图 11-4 和图 11-5 所示。

图11-4 UART 的典型应用框图一





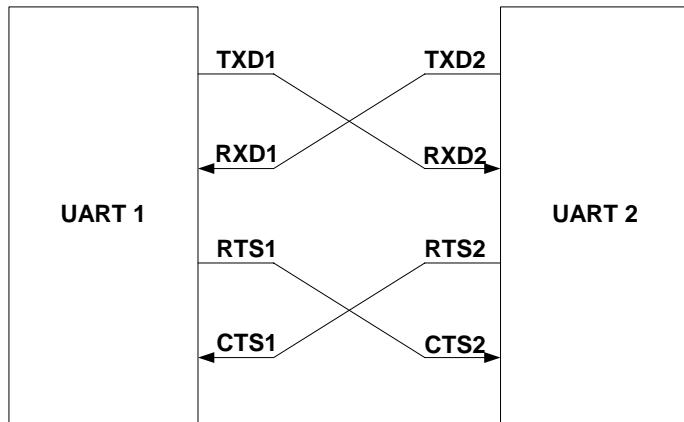
UART 是一种异步双向串行总线，它提供了一种简单有效的数据传输方式，只需要两根数据线互相对接。UART0 和 UART2 提供了这种方式。



注意

在配置成 RTS 流控时，不能通过配置 [UART_CR](#) 寄存器来控制 RTS 的输出值。

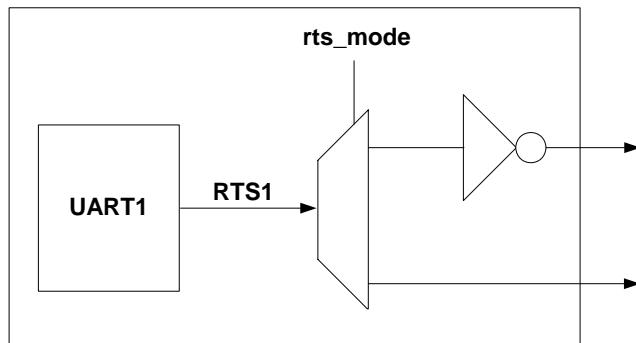
图11-5 UART 的典型应用框图二



如果对接芯片需要 RTS 或者 CTS 流控来控制数据流，也可以通过对接 RTS 和 CTS 管脚达到握手以此控制收发。UART1 对接 RS-485 总线芯片时需要连接 RTS 和 CTS 信号。

RTS 流控信号输出模式如图 11-6 所示。

图11-6 RTS 流控信号输出模式框图



在 RTS 流控下，不能通过 [UART_CR](#) 寄存器来配置 RTS 的输出值，但是可以通过配置系统控制器中的 [SC_PERCTRL4\[rts_mode\]](#) 来控制 RTS 输出的电平模式。

功能原理

UART 的一次帧传输主要包括起始信号、数据、校验位和结束信号，如图 11-7 所示。数据帧从某一 UART 的 TXD 端输出，进入另一个 UART 的 RXD 端输入。

图11-7 UART 帧格式



始信号、数据、校验位和结束信号的含义如下：

- 起始信号(start bit)

一个数据帧开始的标志，UART 协议规定 TXD 信号出现一个低电平就表示一个数据帧的开始。在 UART 不传输数据时，应该保持高电平。

- 数据信号(data bit)

数据位宽可以根据不同的应用要求进行调整，可以配置成 5、6、7 或 8 比特数据位宽。

- 校验位(parity bit)

校验位是 1 比特纠错信号，UART 的校验位有奇校验、偶校验和固定校验位，同时支持校验位的使能和禁止，详细描述请见 [UART_LCR_H](#) 寄存器。

- 结束信号(stop bit)

结束信号即数据帧的停止位，支持 1 比特和 2 比特停止位两种配置。数据帧的结束信号就是把 TXD 拉成高电平。

11.2.5 工作方式

11.2.5.1 管脚复用配置

UART1 管脚与 GPIO 复用，使用 UART1 前，可通过配置寄存器 [SC_PERCTRL1\[pinmuxctrl16\]](#) 来实现。

UART2 管脚与 VI 和 GPIO 复用，使用 UART2 前，可通过配置寄存器 [SC_PERCTRL1\[pinmuxctrl8-7\]](#) 来实现。

11.2.5.2 时钟门控

在软件完成当前数据传输且未启动新的数据传输的情况下可关断 UART 时钟，但需要确保硬件已处于空闲状态（即 [UARTFR\[3\]](#) 为 0 时）。关断 UART 时钟的步骤如下：

步骤 1 读 [UART_FR](#)。



步骤 2 若 **UART_FR[3]** 为 0，则写 **UART_ENABLE=0**，进入步骤 3；若 **UART_FR[3]** 为 1，则延时等待，返回步骤 1。

步骤 3 配置系统控制器 **SC_PERDIS[uartclkdis]=1**，关闭 UART 时钟。

----结束

11.2.5.3 时钟配置

通过配置寄存器 **UART_IBRD** 和 **UART_FBRD** 可以设置 UART 工作的波特率，波特率计算公式为：

当前波特率=UART 参考时钟频率/ (16×分频系数)

分频系数有整数和小数两部分组成，分别对应寄存器 **UART_IBRD** 和 **UART_FBRD**。

例如内部总线时钟为 60MHz，如果配置 **UART_IBRD** 为 0x1E，**UART_FBRD** 为 0x00，按照波特率计算公式：则表示当前的波特率为 $60/(16 \times 30) = 0.125\text{Mbit/s}$ 。

UART 波特率配置的典型值为：9,600bit/s、14,400bit/s、19,200bit/s、38,400bit/s、57,600bit/s、76,800bit/s、115,200bit/s、230,400bit/s、460,800bit/s。

分频系数值的计算以及分频系数寄存器的配置举例如下：

如果要求波特率为 230400，并且 **UARTCLK** 为 100MHz，那么分频系数为 $(100 \times 10^6) / (16 \times 230400) = 27.1267$ ，因此 **IBRD**（整数部分）为 27，**FBRD**（小数部分）为 0.1267。

计算 6bit **UART_FBRD** 寄存器中的数值：根据 $m = \text{integer}(FBRD \times 2^n + 0.5)$ ($n = \text{UART_FBRD}$ 寄存器的宽度)，计算出 $m = \text{integer}(0.1267 \times 2^6 + 0.5) = 8$ ，在 **UART_IBRD** 寄存器中配置 16'h001b，**UART_FBRD** 寄存器中配置 6'h08。

当分频系数小数部分配置成 8 时，波特率除数的实际数值为 $27+8/64=27.125$ ，产生的波特率为 $(100 \times 10^6) / (16 \times 27.125) = 230414.75$ ，误差率为 $(230414.75 - 230400) / 230400 \times 100 = 0.006\%$ 。

使用 6bit **UART_FBRD** 寄存器最大的误差率为 $1/64 \times 100 = 1.56\%$ ，当 $m=1$ 时会出现，误差率累计超过 64 个时钟周期。

11.2.5.4 软复位

通过配置系统控制器可实现对 UART 控制器的单独软复位。

- 配置系统控制器 **SC_PERCTRL0[4]** 为 1，可实现对 **UART0** 控制器的单独软复位。
- 配置系统控制器 **SC_PERCTRL0[5]** 为 1，可实现对 **UART1** 控制器的单独软复位。
- 配置系统控制器 **SC_PERCTRL0[24]** 为 1，可实现对 **UART2** 控制器的单独软复位。

复位后各配置寄存器的值均复位为默认值，因此复位后需要重新对这些寄存器进行初始化配置。



11.2.5.5 中断或查询方式下的数据传输

初始化

初始化步骤如下：

- 步骤 1 向 `UART_CR[0]`写 0，使 UART 处于禁止状态。
- 步骤 2 写相应的配置值到 `UART_IBRD`、`UART_FBRD` 寄存器，配置传输速率。
- 步骤 3 配置 `UART_CR`、`UART_LCR_H`，设定相应的 UART 工作模式。
- 步骤 4 配置 `UART_IFLS` 设定相应的发送及接收 FIFO 阈值。
- 步骤 5 如果驱动程序采用中断方式则需设定 `UART_IMSC`，使能相应中断信号；采用查询方式时应禁止产生相应中断信号。
- 步骤 6 向 `UART_CR[0]`写 1，使能 UART，完成初始化配置。

----结束

数据发送

数据发送步骤如下：

- 步骤 1 将发送数据写入 `UART_DR`，启动数据发送。
- 步骤 2 查询方式下，进行连续数据发送时通过读取 `UART_FR[5]`检测 `TX_FIFO` 状态，根据 `TX_FIFO` 的状态决定是否向 `TX_FIFO` 中发送数据；中断方式下，则根据相应中断状态位检测；决定是否向 `TX_FIFO` 中发送数据。
- 步骤 3 通过检测 `UART_FR[7]`是否为 1，判断 UART 是否完成全部数据发送。

----结束

数据接收

查询方式下，进行数据接收时通过读取 `UART_FR[4]`检测 `RX_FIFO` 状态，根据 `RX_FIFO` 的状态决定是否读取 `RX_FIFO` 中的数据；中断方式下，则根据相应中断状态位检测决定是否读取 `RX_FIFO` 中的数据。

11.2.5.6 DMA 方式下的数据传输



注意

UART0 支持 DMA 传输方式，而 UART1 和 UART2 不支持 DMA 传输方式。

初始化

初始化步骤如下：



- 步骤 1 向 [UART_CR\[0\]](#)写 0, 使 UART 处于禁止状态。
- 步骤 2 写相应的配置值到 [UART_IBRD](#)、[UART_FBRD](#) 寄存器, 配置传输速率。
- 步骤 3 配置 [UART_CR](#)、[UART_LCR_H](#), 设定相应的 UART 工作模式。
- 步骤 4 配置 [UART_IFLS](#) 设定相应的发送及接收 FIFO 阈值。
- 步骤 5 如果驱动程序采用中断方式则需设定 [UART_IMSC](#), 使能相应中断信号; 采用查询方式时应禁止产生相应中断信号。
- 步骤 6 向 [UART_CR\[0\]](#)写 1, 使能 UART, 完成初始化配置。
- 结束

数据发送

数据发送步骤如下:

- 步骤 1 配置 DMA 数据通道, 包括数据传输源和目的地址、数据传输个数、传输类型等参数。具体配置时请参见“3.5 直接存储器存取控制器”的相关描述。
- 步骤 2 配置 [UART_DMACR](#) 为 0x2, 使能 UART 的 DMA 发送功能。
- 步骤 3 通过 DMA 中断上报, 判断数据是否发送完成, 如果完成则关闭 UART 的 DMA 发送功能。
- 结束

数据接收

数据接收步骤如下:

- 步骤 1 配置 DMA 数据通道, 包括数据传输源和目的地址、数据接收区地址、数据传输个数、传输类型等参数。
- 步骤 2 配置 [UART_DMACR](#) 为 0x1, 使能 UART 的 DMA 接收功能。
- 步骤 3 通过 DMA 状态查询, 判断数据是否接收完成, 如果完成则关闭 UART 的 DMA 接收功能。
- 结束

11.2.6 寄存器概览

Hi3511/Hi3512 提供 3 个 UART 单元, UART0、UART1 和 UART2 的基址分别如下:

- UART0 寄存器基址是 0x101F_1000。
- UART1 寄存器基址是 0x101F_2000。
- UART2 寄存器基址是 0x101F_3000。

UART 寄存器概览如[表 11-10](#) 所示。



表11-10 UART 寄存器概览

偏移地址	名称	描述	页码
0x000	UART_DR	数据寄存器	11-38
0x004	UART_RSR	接收状态寄存器/错误清除寄存器	11-39
0x008~0x014	RESERVED	保留	-
0x018	UART_FR	标志寄存器	11-40
0x1C~0x020	RESERVED	保留	-
0x024	UART_IBRD	整数波特率寄存器	11-41
0x028	UART_FBRD	小数波特率寄存器	11-42
0x02C	UART_LCR_H	线控寄存器	11-43
0x030	UART_CR	控制寄存器	11-44
0x034	UART_IFLS	中断 FIFO 阈值选择寄存器	11-46
0x038	UART_IMSC	中断屏蔽寄存器	11-47
0x03C	UART_RIS	原始中断状态寄存器	11-48
0x040	UART_MIS	屏蔽后中断状态寄存器	11-49
0x044	UART_ICR	中断清除寄存器	11-50
0x048	UART_DMACR	DMA 控制寄存器	11-51

11.2.7 寄存器描述

UART_DR

UART_DR 为 UART 数据寄存器，存放接收数据和发送的数据，同时可以从该寄存器中读出接收状态。

Offset Address																Register Name			
0x000																UART_DR			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved					oe	be	pe	fe	data									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name				Description													
[15:12]	-	reserved				保留。													



[11]	RO	oe	溢出错误。 0: 无溢出错误。 1: 溢出错误, 接收 FIFO 满且接收了一个数据。
[10]	RO	be	break 错误。 0: 无 break 错误。 1: 检测到 break 错误, 即接收数据的输入保持低的时间比一个全字传输 (包括 start、data、parity、stop bit) 还要长。
[9]	RO	pe	校验错误。 0: 无校验错误。 1: 接收数据的校验错误。
[8]	RO	fe	帧错误。 0: 无帧错误。 1: 接收到的数据的帧错误 (错误的停止位)。
[7:0]	RW	data	接收数据和发送数据。

UART_RSR

UART_RSR 为接收状态寄存器/错误清除寄存器。

- 寄存器读时作为接收状态寄存器。
- 寄存器写时作为错误清除寄存器。

接收状态也可以从 [UART_DR](#) 中读出。从 [UART_DR](#) 中读出的 break、frame、parity 的状态信息要比从 [UART_RSR](#) 读出的信息优先级高 (即 [UART_DR](#) 中的状态变化比 [UART_RSR](#) 更快)。

对 [UART_RSR](#) 寄存器的任何写操作都会对 [UART_RSR](#) 寄存器进行复位。

	Offset Address								Register Name								Total Reset Value	
	0x004								UART_RSR								0x00	
Bit	7	6	5	4	3	2	1	0										
Name	reserved					oe	be	pe	fe									
Reset	0	0	0	0	0	0	0	0										
Bits	Access	Name	Description															
[7:4]	-	reserved	保留。															

[3]	RW	oe	溢出错误。 0: 无溢出错误。 1: 溢出错误。 当 FIFO 满时, FIFO 中的内容保持有效, 因为不会有下一个数据写到 FIFO 中, 只是移位寄存器会溢出。CPU 必须立刻读数据以腾空 FIFO。
[2]	RW	be	Break 错误。 0: 无 break 错误。 1: break 错误。 Break 的条件: 接收数据的输入保持低的时间比一个全字传输 (定义了 start、data、parity、stop bit) 还要长。
[1]	RW	pe	校验错误。 0: 无校验错误。 1: 接收数据的校验错误。 FIFO 模式下, 该错误与 FIFO 顶部的数据相关联。
[0]	RW	fe	帧错误。 0: 无帧错误。 1: 接收到的数据的停止位错误 (有效的停止位为 1)。

UART_FR

UART_FR 为 UART 标志寄存器。

	Offset Address								Register Name								Total Reset Value							
	0x018								UART_FR								0x12							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Name	reserved								txfe	rxff	txff	rxfe	busy	reserved										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
Bits	Access	Name	Description																					
[15:8]	-	reserved	保留。																					



[7]	RO	txfe	该位的含义由 UART_LCR_H [fen]的状态决定。 如果 UART_LCR_H [fen]为 0, 则当发送 holding register 空时该位置 1。 如果 UART_LCR_H [fen]为 1, 则当发送 FIFO 为空时该位置 1。
[6]	RO	rxff	该位的含义由 UART_LCR_H [FEN]的状态决定。 如果 UART_LCR_H [fen]为 0, 则当接收 holding register 满时该位置 1。 如果 UART_LCR_H [fen]为 1, 则当接收 FIFO 为满时该位置 1。
[5]	RO	txff	该位的含义由 UART_LCR_H [FEN]的状态决定。 如果 UART_LCR_H [fen]为 0, 则当发送 holding register 满时该位置 1。 如果 UART_LCR_H [fen]为 1, 当发送 FIFO 为满时该位置 1。
[4]	RO	rxfe	该位的含义由 UART_LCR_H [FEN]的状态决定。 如果 UART_LCR_H [fen]为 0, 则当接收 holding register 空时该 bit 置 1。 如果 UART_LCR_H [fen]为 1, 则当接收 FIFO 为空时该位就置 1。
[3]	RO	busy	UART 忙闲状态位。 0: UART 空闲或者完成发送数据。 1: UART 正忙于发送数据。 该位一旦置位, 该状态一直保持到整个字节(包括所有的停止位)完全从移位寄存器中发送出去。 一旦发送 FIFO 非空该位就置位, 不管 UART 使能与否。
[2:0]	-	reserved	保留。

UART_IBRD

UART_IBRD 为整数波特率寄存器。

Offset Address											Register Name				Total Reset Value				
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0x0000		
Name												baud divint							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																
[15:0]	RW	baud divint	整数波特率分频值。复位时全部清 0。																

UART_FBRD

UART_FBRD 为小数波特率寄存器。



注意

- 整数波特率寄存器和小数波特率寄存器的值必须等到当前数据发送和接收完毕才能更新。
- 最小的分频值为 1，最大的分频值为 $65535 (2^{16}-1)$ 。即 [UART_IBRD](#) = 0 是无效的，而此时 [UART_FBRD](#) 将被忽略。同样，如果 [UART_IBRD](#) = 65535 (0xFFFF)，[UART_FBRD](#) 就只能是 0，如果比 0 大，则会导致发送和接收的失败
- 假设 [UART_FBRD](#)=0x1E、[UART_IBRD](#)=0x01，这就表示分频系数的整数部分为 30，小数部分为 0.015625，整个分频系数为 30.015625。
- [UART](#) 的波特率=内部总线频率/(16×分频系数)= 内部总线频率 / (16 × 30.015625)。

Offset Address								Register Name				Total Reset Value									
Bit	7	6	5	4	3	2	1	0	0x00				0x00								
Name	reserved				baud divfrac																
Reset	0	0	0	0	0	0	0	0													
Bits	Access	Name	Description																		
[7:6]	-	reserved	保留。																		
[5:0]	RW	band divfrac	小数波特率分频值。复位时全部清 0。																		



UART_LCR_H

UART_LCR_H 为传输模式控制寄存器, [UART_LCR_H](#)、[UART_IBRD](#)、[UART_FBRD](#) 组成一个 30bit 宽的寄存器。如果更新 [UART_IBRD](#) 和 [UART_FBRD](#) 的内容, 必须同时更新 [UART_LCR_H](#)。

Bit	Offset Address															Register Name		Total Reset Value				
	0x02C															UART_LCR_H		0x0000				
Name	reserved															sps	wlen	fen	stp2	eps	pen	brk
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																			
[15:8]	-	reserved	保留。																			
[7]	RW	sps	校验选择。 当本寄存器的 bit[1]、bit[2]、bit[7]被置位时, 校验位就会作为 0 发送和检测。 当本寄存器的 bit[1]、bit[7]被置位, bit[2]为 0 时, 校验位就会作为 1 发送和检测。 当 bit[1]、bit[2]、bit[7]都清 0, 那么 stick parity 禁止。																			
[6:5]	RW	wlen	指示发送和接收一个帧里数据比特的数目。 00: 5bit。 01: 6bit。 10: 7bit。 11: 8bit。																			
[4]	RW	fen	发送和接收 FIFO 使能控制。 0: 发送和接收 FIFO 禁止。 1: 发送和接收 FIFO 使能。																			
[3]	RW	stp2	发送帧尾 2bit 停止位判断。 0: 发送的帧尾没有 2bit 停止位。 1: 发送的帧尾有 2bit 停止位。 接收逻辑在接收时不检查 2bit 的停止位。																			

[2]	RW	eps	发送和接收过程中的奇偶校验选择。 0: 在发送和接收过程中生成奇校验或检查奇校验。 1: 在发送和接收过程中生成偶校验或检查偶校验。 当 UART_LCR_H[fen] 为 0 时, 该位不起作用。
[1]	RW	pen	校验选择位。 0: 不作校验。 1: 发送方向产生校验, 接收方向作校验检查。
[0]	RW	brk	发送 break。 0: 无效。 1: 在完成当前数据的发送后, UTXD 连续输出低电平。 注: 要正确的执行 break 命令, 软件将该位置 1 的时间必须超过 2 个完整帧; 在正常使用中, 该位必须清 0。

UART_CR

UART_CR 为 UART 控制寄存器。

配置 [UART_CR](#) 遵循以下步骤:

步骤 1 向 [UART_CR\[0\]](#) 写 0, 禁止 UART。

步骤 2 等待当前数据发送或接收结束。

步骤 3 将 [UART_LCR_H\[fen\]](#) 清 0。

步骤 4 配置 [UART_CR](#)。

步骤 5 向 [UART_CR\[0\]](#) 写 1, 使能 UART。

----结束



Bit	Offset Address 0x030															Register Name UART_CR	Total Reset Value 0x0300	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1			
Name	ctsen	rtsen	reserved	rts	dtr	rxe	txe	lbe	reserved						uarten			
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0			
Bits	Access	Name							Description									
[15]	RW	ctsen							CTS 硬件流控使能。 0: 不使能 CTS 硬件流控。 1: 使能 CTS 硬件流控, 只有当 nUARTCTS 信号有效时才发送数据。									
[14]	RW	rtsen							RTS 硬件流控使能。 0: 不使能 RTS 硬件流控。 1: 使能 RTS 硬件流控, 只有当接收 FIFO 有空间时才请求接收数据。									
[13:12]	-	reserved							保留。									
[11]	RW	rts							请求发送。 该 bit 为 UART modem 状态输出信号 nUARTRTS 的取反。 0: 输出信号不变。 1: 即该 bit 配置为 1, 则输出信号为 0。									
[10]	RW	dtr							数据发送准备。 该 bit 为 UART modem 状态输出信号 nUARTDTR 的取反。 0: 输出信号不变。 1: 即该 bit 配置为 1, 则输出信号为 0。									
[9]	RW	rxe							UART 接收使能。 0: 禁止。 1: 使能。 在接收的过程中如果 UART 被禁止, 那么当前数据的接收就会在正常停止之前结束。									

[8]	RW	txe	UART 发送使能。 0: 禁止。 1: 使能。 在发送的过程中如果 UART 被禁止, 那么当前数据的发送就会在正常停止之前结束。
[7]	RW	lbe	环回使能。 0: 禁止。 1: UARTTXD 输出环回到 UARTRXD。
[6:1]	-	reserved	保留。
[0]	RW	uarten	UART 使能。 0: 禁止。 1: 使能。 如果在发送和接收过程中将 UART 禁止, 则会在正常停止之前结束当前数据的传送。

UART_IFLS

UART_IFLS 为中断 FIFO 阈值选择寄存器, 用于设置 FIFO 的中断 (UART_TXINTR 或 UART_RXINTR) 触发线。

Offset Address												Register Name				Total Reset Value			
0x034												UART_IFLS				0x0012			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												rxiflsel		txiflsel				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0		
Bits	Access	Name	Description																
[15:6]	-	reserved	保留。																
[5:3]	RW	rxiflsel	接收中断 FIFO 的阈值选择, 接收中断的触发点如下。 000: 接收 FIFO \geq 1/8full。 001: 接收 FIFO \geq 1/4full。 010: 接收 FIFO \geq 1/2full。 011: 接收 FIFO \geq 3/4full。 100: 接收 FIFO \geq 7/8full。 101~111: 保留。																



[2:0]	RW	txiflsel	发送中断 FIFO 的阈值选择，发送中断的触发点如下。 000: 发送 FIFO $\leq 1/8$ full。 001: 发送 FIFO $\leq 1/4$ full。 011: 发送 FIFO $\leq 3/4$ full。 010: 发送 FIFO $\leq 1/2$ full。 100: 发送 FIFO $\leq 7/8$ full。 101~111: 保留。
-------	----	----------	--

UART_IMSC

UART_IMSC 为中断屏蔽寄存器，用于屏蔽中断。

[5]	RW	txim	发送中断的屏蔽状态。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[4]	RW	rxim	接收中断的屏蔽状态。 0: 屏蔽该中断。 1: 不屏蔽该中断。
[3:0]	-	reserved	保留。

UART_RIS

UART_RIS 为原始中断状态寄存器，其内容不受中断屏蔽寄存器的影响。

	Offset Address 0x03C												Register Name UART_RIS				Total Reset Value 0x0000
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved						oeris	beris	peris	geris	rtris	txris	reserved				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name		Description													
[15:11]	-	reserved		保留。													
[10]	RO	oeris		原始的溢出错误中断状态。 0: 未产生中断。 1: 已产生中断。													
[9]	RO	beris		原始的 break 错误中断状态。 0: 未产生中断。 1: 已产生中断。													
[8]	RO	peris		原始的校验中断状态。 0: 未产生中断。 1: 已产生中断。													
[7]	RO	feris		原始的错误中断状态。 0: 未产生中断。 1: 已产生中断。													
[6]	RO	rtris		原始的接收超时中断状态。 0: 未产生中断。 1: 已产生中断。													



[5]	RO	txris	原始的发送中断状态。 0: 未产生中断。 1: 已产生中断。
[4]	RO	rxris	原始的接收中断状态。 0: 未产生中断。 1: 已产生中断。
[3:0]	-	reserved	保留。

UART_MIS

UART_MIS 为屏蔽后中断状态寄存器，其内容为原始中断状态和中断屏蔽进行“与”操作后的结果。

Offset Address															Register Name	Total Reset Value	
0x040															UART_MIS	0x0000	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name	reserved						oemis	bemis	permis	gemis	ttmis	txmis	rxmis	reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name			Description												
[15:11]	-	reserved			保留。												
[10]	RO	oemis			屏蔽后的溢出错误中断状态。 0: 未产生中断。 1: 已产生中断。												
[9]	RO	bemis			屏蔽后的 break 错误中断状态。 0: 未产生中断。 1: 已产生中断。												
[8]	RO	permis			屏蔽后的校验中断状态。 0: 未产生中断。 1: 已产生中断。												
[7]	RO	femis			屏蔽后的错误中断状态。 0: 未产生中断。 1: 已产生中断。												



[6]	RO	rtmis	屏蔽后的接收超时中断状态。 0: 未产生中断。 1: 已产生中断。
[5]	RO	txmis	屏蔽后的发送中断状态。 0: 未产生中断。 1: 已产生中断。
[4]	RO	rxmis	屏蔽后的接收中断状态。 0: 未产生中断。 1: 已产生中断。
[3:0]	-	reserved	保留。

UART_ICR

UART_ICR 为中断清除寄存器，写 1 时相应的中断被清除，写 0 则不起作用。



[6]	WO	rtic	清除接收超时中断。 0: 无效。 1: 清除中断。
[5]	WO	txic	清除发送中断。 0: 无效。 1: 清除中断。
[4]	WO	rxic	清除接收中断。 0: 无效。 1: 清除中断。
[3:0]	-	reserved	保留。

UART_DMCR

UART_DMCR 为 DMA 控制寄存器，用于配置发送 FIFO 和接收 FIFO 的 DMA 使能。

Bit	Offset Address 0x048															Register Name UART_DMCR				Total Reset Value 0x0000			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
Name	reserved															dmaonerr	tdmae	rxdmae					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
Bits	Access	Name	Description																				
[15:3]	-	reserved	保留。																				
[2]	RW	dmaonerr	UART 错误中断 (UARTEINTR) 出现时的接收通道 DMA 使能控制。 0: 当 UART 错误中断 (UARTEINTR) 有效时, 接收通道 DMA 的请求输出 (UARTRXDMASREQ 或 UARRTXDMABREQ) 有效。 1: 当 UART 错误中断 (UARTEINTR) 有效时, 接收通道 DMA 的请求输出 (UARTRXDMASREQ 或 UARRTXDMABREQ) 无效。																				

[1]	RW	txdmae	发送 FIFO 的 DMA 使能控制。 0: 禁止。 1: 使能。
[0]	RW	rxdmae	接收 FIFO 的 DMA 使能控制。 0: 禁止。 1: 使能。

11.3 同步串口

11.3.1 概述

SSP (Synchronous Serial Port) 控制器，可以作为一个 master 或 slave 与外部的设备来进行同步串行通信。支持以下接口协议：

- A Motorola SPI (Synchronous Peripheral Interface) -compatible interface
- A Texas Instruments synchronous serial interface
- A National Semiconductor Microwire interface

SSP 主要应用于外接触摸屏、SD 卡、WiFi 等。

11.3.2 特点

SSP 模块有以下特点：

- 支持主操作，最多支持 2 个 slave。
- 支持从操作。
- 支持接口时钟频率可编程。
- 收/发分开的 16bit 宽、深度为 8 的 FIFO (发送 FIFO 和接收 FIFO 各一个)。
- 支持三种帧格式：SPI、Microwire、TI synchronous serial。
- 数据帧大小可编程：4bit~16bit。
- 内部提供环回测试模式。
- 支持 DMA 操作。

11.3.3 信号描述

SSP 接口信号如表 11-11 所示。

表11-11 SSP 接口信号描述

信号名称	方向	描述	对应管脚
SSP_SPICK	I/O	SSP 时钟输出，只与 GPIO 复用。（复用时的配置信息请参见“ 管脚复用配置 ”）	SPICK



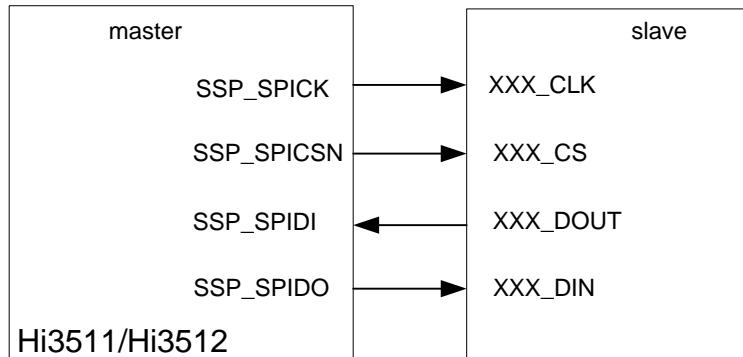
信号名称	方向	描述	对应管脚
SSP_SPIDI	I	SSP 的数据输入, 只与 GPIO 复用。(复用时的配置信息请参见“ 管脚复用配置 ”)	SPIDI
SSP_SPIDO	O	SSP 的数据输出, 只与 GPIO 复用。(复用时的配置信息请参见“ 管脚复用配置 ”)	SPIDO
SSP_SPICSN	I/O	当 SSP 配置为 SPI 和 Microwire 帧格式时, 该信号作片选信号, 当 SSP 配置为 TI 帧格式时, 该信号做帧同步信号使用。(复用时的配置信息请参见“ 管脚复用配置 ”)	SPICSN0

11.3.4 功能描述

11.3.4.1 应用框图

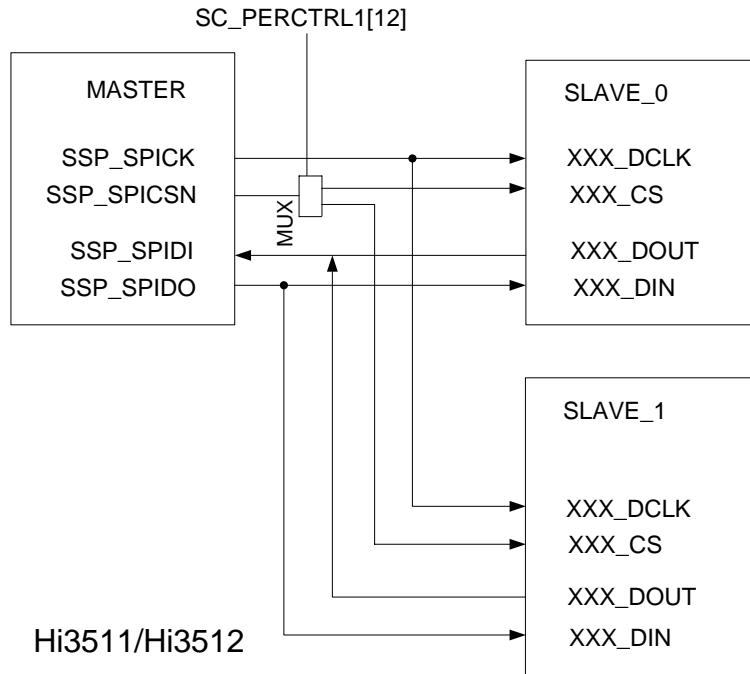
SSP 接单 slave 时的应用框图如图 11-8 所示。SSP 接单 slave 时, 使用的是 SSP 默认的片选管脚 SSP_SPICSN0。

图11-8 当 SSP 接单 slave 时的应用



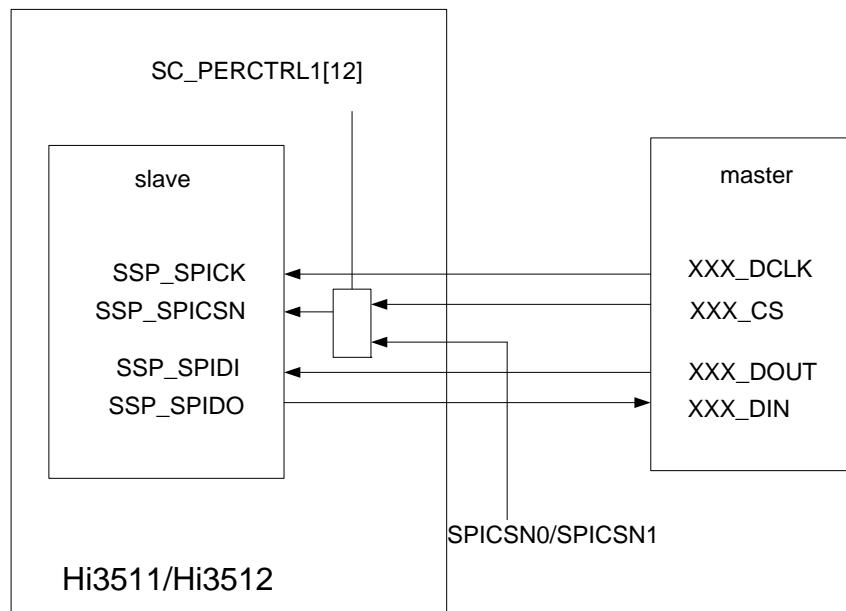
SSP 接双 slave 时的应用框图如图 11-9 所示。当 SSP 外接 2 个 slave 器件时, SSP 自带的片选信号已不能满足要求, 此时通过系统控制器 SC_PERCTRL1[12]配置, 来选择将片选信号送给 SPICSN0 和 SPICSN1 其中一个管脚。两个管脚同时只能有一个有效, SC_PERCTRL1[12]配置为 0 时, SPICSN0 有效; SC_PERCTRL1[12]配置为 1 时, SPICSN1 有效。

图11-9 当 SSP 接双 slave 时的应用



当 SSP 做为 slave 时的应用框图如图 11-10 所示。外部的 Master 设备将选择 SPICSN0 和 SPICSN1 中的一个片选管脚做为输入连接，注意此时确认系统控制器 SC_PERCTRL1[12]配置，外部的 Master 设备的片选信号才能正确输入 SSP 接口。推荐使用 SSP 的默认片选接口 SPICSN0。

图11-10 当 SSP 作 slave 时的应用





11.3.4.2 功能原理



说明

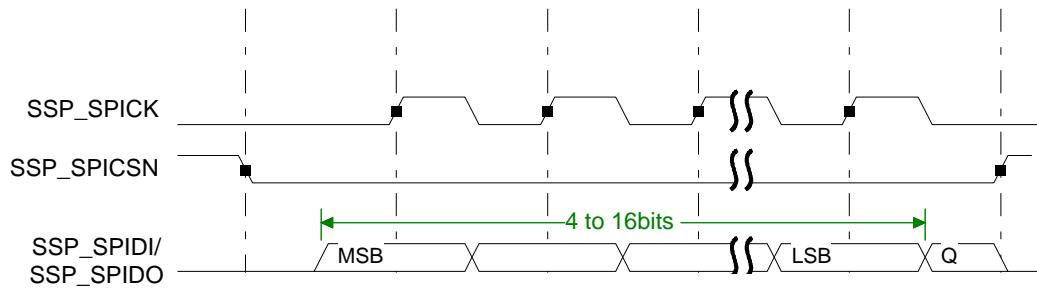
图 11-11 ~ 图 11-22 中, 以下缩略语或字母意义不变:

- MSB : Most Significant Bit
- LSB : Least Significant Bit
- Q : An undefined signal

Motorola SPI 帧格式 (spo=0、sph=0)

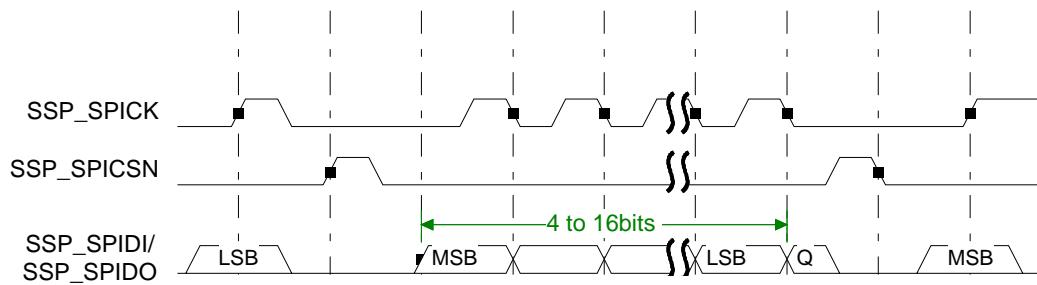
spo=0、sph=0 时 Motorola SPI 单帧帧格式如图 11-11 所示。

图11-11 Motorola SPI 单帧帧格式 (spo=0、sph=0)



spo=0、sph=0 时 Motorola SPI 连续帧帧格式如图 11-12 所示。

图11-12 Motorola SPI 连续帧帧格式 (spo=0、sph=0)



在该模式下, 当 SSP 处于空闲状态时:

- SSP_SPICK 信号设置为低电平。
- SSP_SPICSN 信号设置为高电平。
- 发送数据线 SSP_SPIDO 强制为低电平。

当 SSP 处于使能状态, 而且发送 FIFO 内部存在有效数据时, 设置 SSP_SPICSN 信号为低表示开始传输数据。这样就会使能 slave 的数据放在 master 的输入 SSP_SPIDI 线上。半个 SSP_SPICK 时钟周期之后, 有效的 master 数据传输到 SSP_SPIDO 管脚。此时 master 和 slave 数据都已经有效, SSP_SPICK 主时钟管脚会在接下来的半个



SSP_SPICK 时钟周期之后变化为高电平。数据就会在 SSP_SPICK 时钟的上升沿被捕获，在时钟的下降沿被传送。

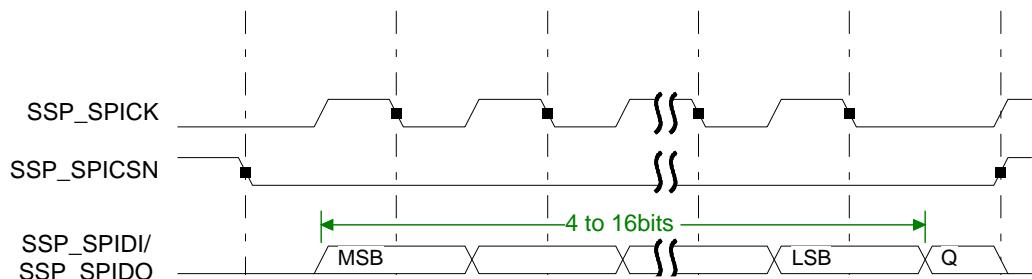
如果传输单个 word，当捕捉到传输的最后 1 个 bit，SSP_SPICSN 会在接下来的 1 个 SSP_SPICK 时钟之后恢复为高电平。

如果是连续的传输，SSP_SPICSN 信号在 2 个 word 传输之间必须将 SSP_SPICK 时钟拉高一个时钟周期。这是因为 sph 为 0 时，slave 选择管脚会固定其内部串行设备寄存器的数据，使它不会变化。因此在连续传输时，主设备必须在每 2 个 word 传输之间将 SSP_SPICSN 信号拉高。连续传输结束时，SSP_SPICSN 会在捕获到最后 1 个 bit 之后的 1 个 SSP_SPICK 时钟周期之后恢复为高电平。

Motorola SPI 帧格式 (spo=0、sph=1)

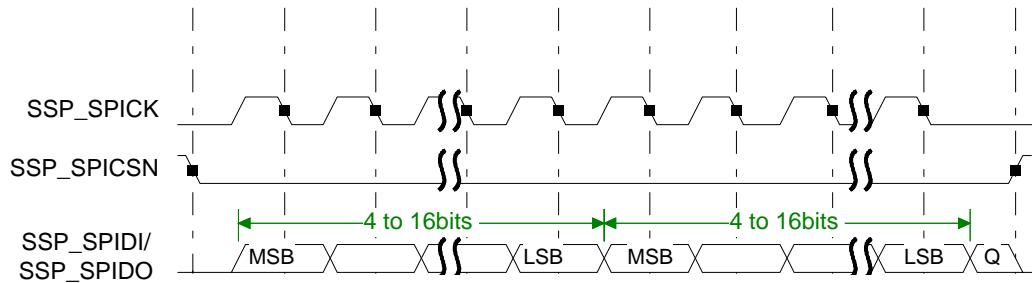
spo=0、sph=1 时 Motorola SPI 单帧帧格式如图 11-13 所示。

图11-13 Motorola SPI 单帧帧格式 (spo=0、sph=1)



spo=0、sph=1 时 Motorola SPI 连续帧帧格式如图 11-14 所示。

图11-14 Motorola SPI 连续帧帧格式 (spo=0、sph=1)



在该模式下，当 SSP 处于空闲状态时：

- SSP_SPICK 信号设置为低
- SSP_SPICSN 设置为高
- 发送数据线 SSP_SPIDO 强制为低

当 SSP 为使能状态，而且在发送 FIFO 内部有有效数据时，设置 SSP_SPICSN 信号为低表示开始传输数据。半个 SSP_SPICK 时钟周期之后，master 和 slave 的有效数据就



分别在各自的传输线上有效。同时，SSP_SPICK 从一个上升沿开始有效。数据就会在 SSP_SPICK 时钟的上升沿被捕获，在时钟的下降沿被传送。

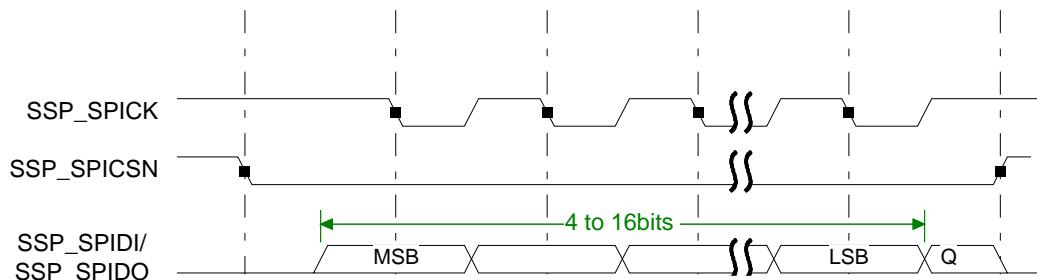
如果传输单个 word，当捕捉到传输的最后 1 个 bit，SSP_SPICSN 会在接下来的 1 个 SSP_SPICK 时钟之后恢复为高电平。

当连续传输时，在传输数据 word 之间 SSP_SPICSN 保持为低。连续传输结束时，SSP_SPICSN 会在最后 1 个 bit 捕获之后的 1 个 SSP_SPICK 时钟之后恢复为高电平。

Motorola SPI 帧格式 (spo=1、sph=0)

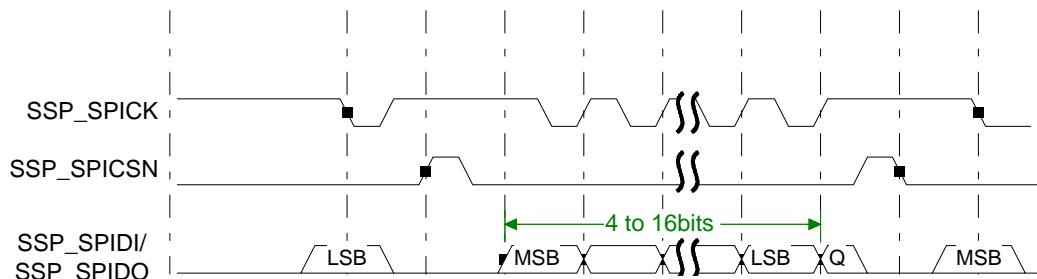
spo=1、sph=0 时 Motorola SPI 单帧帧格式如图 11-15 所示。

图11-15 Motorola SPI 单帧帧格式 (spo=1、sph=0)



spo=1、sph=0 时 Motorola SPI 连续帧帧格式如图 11-16 所示。

图11-16 Motorola SPI 连续帧帧格式 (spo=1、sph=0)



在该配置下，当 SSP 处于空闲状态时：

- SSP_SPICK 信号设置为高电平。
- SSP_SPICSN 信号设置为高电平。
- 发送数据线 SSP_SPIDO 强制为低电平。

当 SSP 为使能状态，而且在发送 FIFO 内部有有效数据时，设置 SSP_SPICSN 信号为低表示开始传输数据，此时 slave 的数据就会立刻发送到 master 的接收数据线 SSP_SPIDI 上。半个 SSP_SPICK 周期之后，master 的有效数据就传送到 SSP_SPIDO 线上。再过半个 SSP_SPICK 时钟周期之后，SSP_SPICK master pin 设置为低。这意味着数据是在 SSP_SPICK 时钟的下降沿被捕获，在 SSP_SPICK 时钟的上升沿被传送。

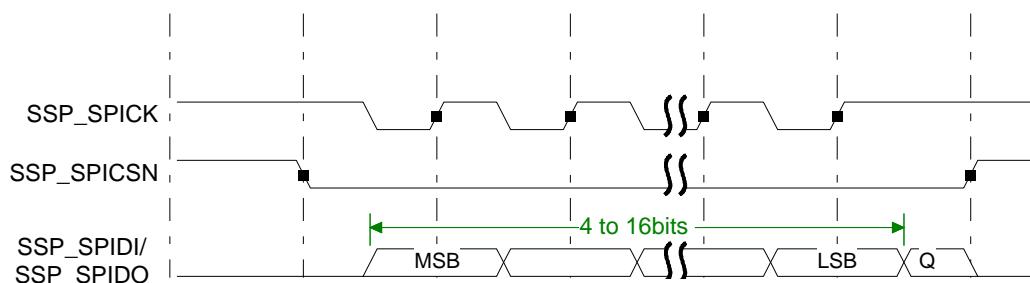
如果传输单个 word, 当捕捉到传输的最后 1 个 bit, SSP_SPICSN 会在接下来的 1 个 SSP_SPICK 时钟之后恢复为高电平。

如果是连续的传输, SSP_SPICSN 信号在 2 个 word 传输之间必须拉高。这是因为, 当 sph 为 0 时, slave 选择管脚固定其内部串行设备寄存器的数据, 使它不会变化。SSP_SPICSN 就会在捕获到最后 1 个 bit 之后的 1 个 SSP_SPICK 时钟周期之后恢复为高电平。

Motorola SPI 帧格式 (spo=1、sph=1)

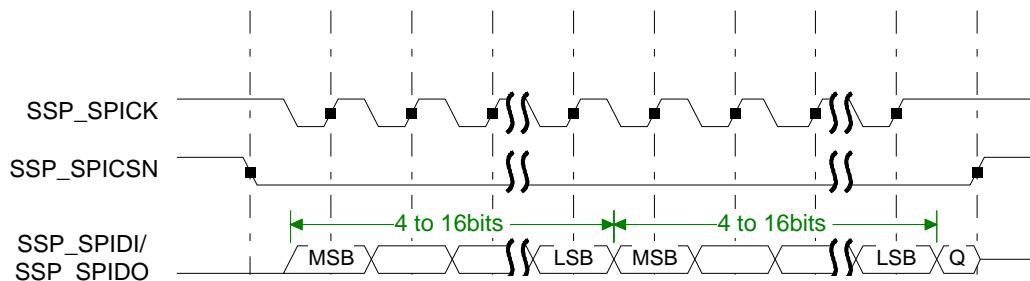
spo=1、sph=1 时 Motorola SPI 单帧帧格式如图 11-17 所示。

图11-17 Motorola SPI 单帧帧格式 (spo=1、sph=1)



spo=1、sph=1 时 Motorola SPI 连续帧帧格式如图 11-18 所示。

图11-18 Motorola SPI 连续帧帧格式 (spo=1、sph=1)



在该模式下, 当 SSP 处于空闲状态时:

- SSP_SPICK 信号设置为高电平。
- SSP_SPICSN 信号设置为高电平。
- 发送数据线 SSP_SPIDO 强制为低电平。

当 SSP 为使能状态, 而且在发送 FIFO 内部有有效数据时, 设置 SSP_SPICSN master 信号为低表示开始传输数据。半个 SSP_SPICK 时钟之后, master 和 slave 数据都会在各自的传输线上有效。同时, 时钟 SSP_SPICK 从一个下降沿开始有效。数据是在 SSP_SPICK 时钟的下降沿被捕获, 在时钟的上升沿被传送。

当传输单个 word 时, SSP_SPICSN 会在传输的最后 1 个 bit 捕获之后的一个 SSP_SPICK 时钟周期之后恢复为高电平。

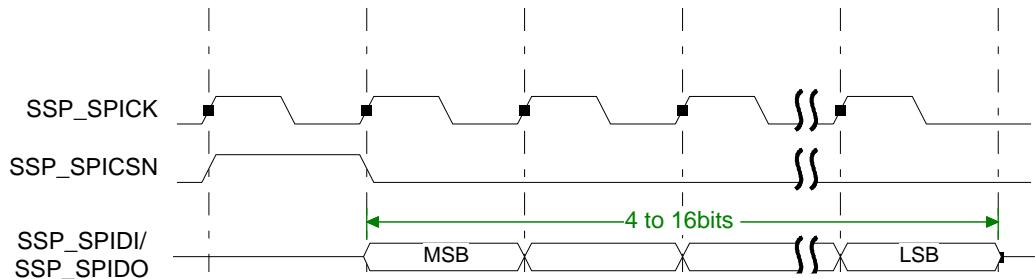


如果是连续传输，SSP_SPICSN 信号始终保持为低，直到传输结束时，SSP_SPICSN 就会在捕获到最后一 bit 之后的一个 SSP_SPICK 时钟周期之后恢复到空闲为高的状态。对于连续的传输来说，信号 SSP_SPICSN 在传输的过程中一直保持为低，结束的方式与单个传输的方式相同。

TI 同步串行帧格式

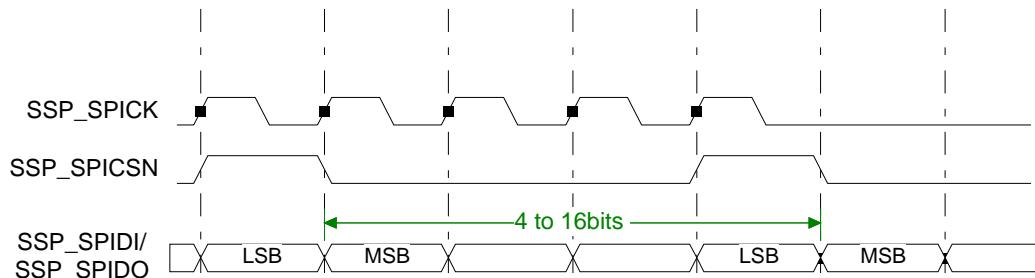
TI 同步串行单帧帧格式如图 11-19 所示。

图11-19 TI 同步串行单帧帧格式



TI 同步串行连续帧帧格式如图 11-20 所示。

图11-20 TI 同步串行连续帧帧格式



在该模式下，当 SSP 处于空闲状态时：

- SSP_SPICK 为低电平。
- SSP_SPICSN 为低电平。
- 传输数据线 SSP_SPIDO 保持为高阻。

一旦发送 FIFO 有数据，SSP_SPICSN 就会产生一个 SSP_SPICK 时钟周期的高电平脉冲，将被发送的数据就会从发送 FIFO 传送到发送逻辑串行移位寄存器。在 SSP_SPICK 时钟的下一个上升沿，4bit~16bit 数据帧的 MSB 就会从 SSP_SPIDO 移位输出。同样，从外部串行 slave 设备接收数据的 MSB 会从 SSP_SPIDI 管脚移位输入。

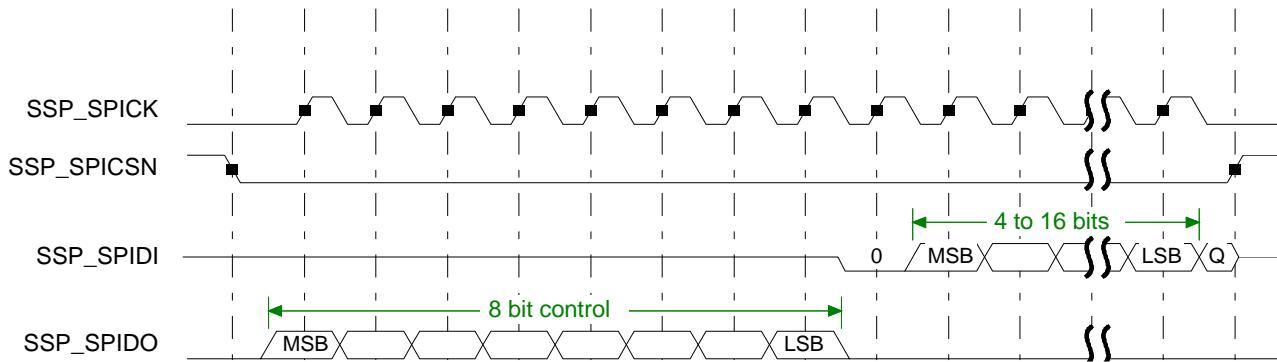
SSP 和片外串行设备在 SSP_SPICK 时钟的下降沿将数据存入串行移位寄存器。接收串行寄存器在接收到 LSB 之后的第一个 SSP_SPICK 时钟上升沿将数据送给接收 FIFO。



National Semiconductor Microwire 帧格式

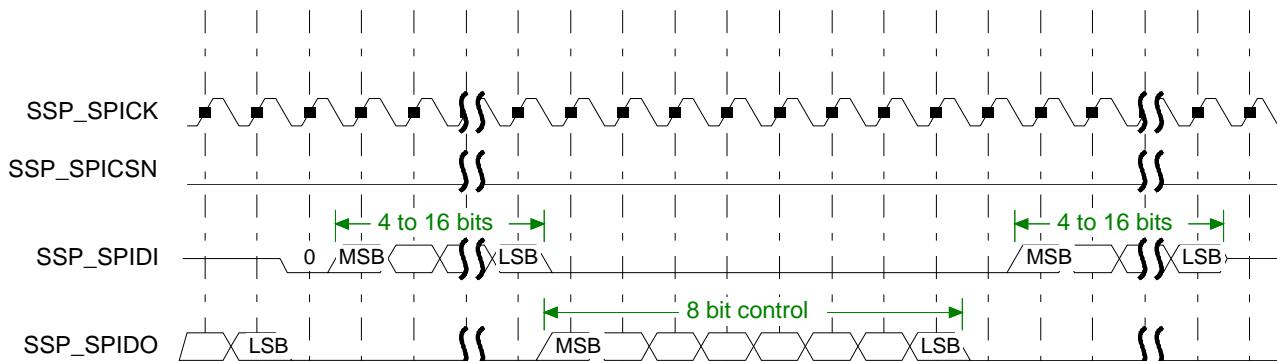
National Semiconductor Microwire 单帧帧格式如图 11-21 所示。

图11-21 National Semiconductor Microwire 单帧帧格式



National Semiconductor Microwire 连续帧帧格式如图 11-22 所示。

图11-22 National Semiconductor Microwire 连续帧帧格式



Microwire 的格式与 SPI 的格式非常相近，使用 master-slave 信息的传输技术，只不过 SPI 是全双工通信，而 Microwire 半双工通信。在 SSP 向外部芯片发送串行数据的时候，都要先加 8bit 控制字。在这个过程中，SSP 没有接收到任何数据。传输完毕之后，片外芯片对接收到的数据进行解码，在与 8bit 控制信息间隔一个时钟周期之后，slave 开始响应所需求的数据。返回的数据长度为 4bit~16bit，使得整个帧的长度为 13bit~25bit。

在该模式下，当 SSP 处于空闲状态时：

- SSP_SPICK 信号设置为低电平。
- SSP_SPICSN 设置为高电平。
- 发送数据线 SSP_SPIDO 强制为低电平。

向发送 FIFO 内部写进一个控制字节开始一次传送。SSP_SPICSN 的下降沿引发数据的传输，发送 FIFO 的数据被发送到串行移位寄存器，8bit 控制帧的 MSB 被发送到发送



管脚 **SSP_SPIDO**。在帧的传送过程中，**SSP_SPICSN** 保持为低。**SSP_SPIDI** 在这个传送过程中保持为高阻。

片外的串行从设备在 **SSP_SPICK** 时钟的每一个上升沿将数据锁存到串行移位寄存器中。当从设备锁存完最后 1 个 bit 的数据之后，在接下来的 1 个时钟周期的等待时间里，对接收到的数据开始解码，然后从设备反馈给 SSP 所要求的数据。每 1 个 bit 都是在 **SSP_SPICK** 时钟的下降沿写到 **SSP_SPIDI** 的。对单个数据传送来说，在帧的结尾，**SSP_SPICSN** 在最后 1 个 bit 写到接收串行寄存器之后的 1 个时钟周期后拉高，这样就使接收到的数据传送到接收 FIFO。

对于连续的传送来说，数据传送的开始和结束都和单个数据的传送方式相同。在这个传送过程中，信号 **SSP_SPICSN** 时一直保持为低的，传送的数据也是连续的。下一帧的控制字直接和上一帧的 LSB 相邻。当帧的 LSB 锁存到 SSP 之后，接收到的每一个数值都是在 **SSP_SPICK** 时钟的下降沿取自接收移位寄存器。

11.3.5 工作方式

11.3.5.1 管脚复用配置

SSP 管脚与其他管脚复用，使用 SSP 前，需要配置系统控制器使能相应管脚的 SSP 功能，请参见 **SC_PERCTRL1[12]**、**SC_PERCTRL1[10]**、**SC_PERCTRL1[1:0]** 配置说明。

11.3.5.2 时钟门控

在软件完成当前数据传输且未启动新的数据传输的情况下可关断 SSP 时钟，但需要确保硬件已处于空闲状态（即 **SSP_SR[bsy]** 为 0 时）。

步骤如下：

步骤 1 读 **SSP_SR**。

步骤 2 若 **SSP_SR[bsy]** 为 0，则写 **SSP_CR1[sse]**=0，进入步骤 3；若 **SSP_SR[bsy]** 为 1，则延时等待，返回步骤 1。

步骤 3 写 **SC_PERDIS[sspclkdis]**，关闭 SSP 工作时钟。

----结束

11.3.5.3 时钟配置

- SSP 的工作时钟的最小频率由此公式确定：

SSP 的工作时钟频率(min) $\geq 2 \times F_{SSP_SPICK}(\max)$

当 SSP 工作时钟确定，在 2 分频时 SSP 的输出时钟频率最大。

- SSP 工作时钟的最大频率由此公式确定：

SSP 的工作时钟频率(max) $\leq 254 \times 256 \times F_{SSP_SPICK}(\min)$

当 SSP 工作时钟确定，在 254×256 分频时，SSP 输出时钟频率最小。

- 输出的时钟频率 **SSP_SPICK** 为：

$F_{SSP_SPICK} = \text{SSP 的工作时钟频率} / (CPSDVSR \times (1 + SCR))$

配置不同的分频数 CPSDVS 和 SCR，可以得到不同的 SSP 输出时钟频率，请参见 [SSP_CR0](#)、[SSP_CPSR](#) 寄存器配置描述。

表 11-12 给出了 SSP 时钟分频的典型配置值。

表11-12 时钟分频的典型配置

SSP 工作时钟频率(MHz)	CPSDVS	SCR	SSP_SPICK(MHz)
135	2	1	33.75
135	2	4	13.5
27	2	1	6.75
27	2	4	2.7

11.3.5.4 软复位

对接器件容许的情况下，可以任意时间对 SSP 进行软复位，不会对系统造成影响。通过配置系统控制器 SC_PERCTRL0[ssp_srst]为 1，可实现对 SSP 的单独软复位。复位后各配置寄存器的值均复位为默认值，因此复位后需要重新对这些寄存器进行初始化配置。

11.3.5.5 中断或查询方式下的数据传输

初始化

初始化步骤如下：

- 步骤 1 向 [SSP_CR1](#)[sse]写 0，使 SSP 处于禁止状态。
- 步骤 2 写相应的配置值到 [SSP_CR0](#)，配置帧格式及传输数据位宽等参数。
- 步骤 3 配置 [SSP_CPSR](#) 寄存器，设定需要的时钟分频因子。
- 步骤 4 如果驱动程序采用中断方式则需设定 [SSP_INTMASK](#)，使能相应中断信号；采用查询方式时应禁止产生相应中断信号。
- 步骤 5 向 [SSP_CR1](#)[sse]写 1，使能 SSP，完成初始化配置。

----结束

数据发送

查询发送定长（以 4 个为例）数据的步骤如下：

- 步骤 1 配置 [SSP_CPSR](#) 寄存器，设置预分频因子 cpsdvsr。
- 步骤 2 配置 [SSP_CR0](#)，设置串行时钟率、帧格式（SPI、TI、MW）和帧长 dss。



步骤3 向 **SSP_CR1[see]**写 1, 使能 SSP。

步骤4 查询状态寄存器 **SSP_SR**, 判断 FIFO 空满状态。如果 **SSP_SR[tnf]**为 1, 说明发送 FIFO 不满, 则写 **SSP_DR** 寄存器, 共写 4 个数据。

步骤5 查询状态寄存器 **SSP_SR**, 判断 FIFO 空满状态, 直至完全发送。

步骤6 向 **SSP_CR1[see]**写 0, 禁止 SSP。

----结束

数据接收

查询接收定长（以 4 个为例）数据的步骤如下：

步骤1 配置 **SSP_CPSR[cpsdvsr]**, 设置预分频因子。

步骤2 配置 **SSP_CR0** 寄存器, 设置串行时钟率、帧格式（SPI、TI、MW）和帧长。

步骤3 向 **SSP_CR1[see]**写 1, 使能 SSP。

步骤4 查询状态寄存器 **SSP_SR**, 判断接收 FIFO 状态; 如果接收 FIFO 不空, 则读 **SSP_DR** 寄存器, 接收一个 half word, 共读 4 个数据。

步骤5 查询状态寄存器 **SSP_SR**, 直至不忙。

步骤6 向 **SSP_CR1[see]**写 0, 禁止 SSP。

----结束

11.3.5.6 DMA 方式下的数据传输

初始化

初始化步骤如下：

步骤1 向 **SSP_CR1[sse]**写 0, 使 SSP 处于禁止状态。

步骤2 写相应的配置值到 **SSP_CR0**, 配置帧格式及传输数据位宽等参数。

步骤3 配置 **SSP_CPSR** 寄存器, 设定需要的时钟分频因子。

步骤4 配置 **SSP_INTMASK** 寄存器, 禁止产生相应中断信号。

步骤5 配置 **SSP_DMACR** 寄存器, 使能 SSP 的 DMA 功能。

----结束

数据发送

数据发送步骤如下：

步骤1 获取一个 DMAC 的通道。

步骤2 配置这个 DMAC 通道的配置寄存器和控制寄存器中的相关参数。

- 步骤 3 启动 DMAC，响应 SSP 发送 FIFO 的 DMA 请求进行数据传输。
- 步骤 4 向 [SSP_CR1\[sse\]](#)写 1，使能 SSP，完成初始化配置。
- 步骤 5 如果配置的 DMAC 传输的数据长度为奇数个，DMAC 会先使用数据长度除以步骤 2 中配置的 burst 长度，确定需要响应的 burst 个数。这样虽然 SSP 会在所有发出 DMA burst 请求的同时产生 single 请求，但是 DMAC 只会先响应 burst 请求，直到最后不够一个 burst 长度时，才响应 SSP 的 single 请求完成全部的数据传输。
- 步骤 6 通过 DMA 中断上报，判断数据是否发送完成，如果完成则关闭 SSP 的 DMA 发送功能。
- 结束

数据接收

数据接收步骤如下：

- 步骤 1 获取一个 DMAC 通道。
- 步骤 2 配置该 DMAC 通道的配置寄存器和控制寄存器中的相关参数。
- 步骤 3 启动 DMAC，响应 SSP 接收 FIFO 的 DMA 请求进行数据传输。
- 步骤 4 向 [SSP_CR1\[sse\]](#)写 1，使能 SSP，完成初始化配置。
- 步骤 5 如果配置的 DMAC 传输的数据长度为奇数个，DMAC 会先使用数据长度除以步骤 2 中配置的 burst 长度，确定需要响应的 burst 个数。这样虽然 SSP 会在所有发出 DMA burst 请求的同时产生 single 请求，但是 DMAC 只会先响应 burst 请求，直到最后不够一个 burst 长度时响应 SSP 的 single 请求完成全部的数据传输。
- 步骤 6 通过 DMA 中断上报，判断数据是否接收完成，如果完成则关闭 SSP 的 DMA 接收功能。

----结束

11.3.6 寄存器概览

SSP 寄存器概览如[表 11-13](#) 所示。

表11-13 SSP 寄存器概览（基址是 0x101F_4000）

偏移地址	名称	描述	页码
0x000	SSP_CR0	控制寄存器 0	11-65
0x004	SSP_CR1	控制寄存器 1	11-66
0x008	SSP_DR	接收 FIFO 和发送 FIFO 的数据寄存器	11-67
0x00C	SSP_SR	状态寄存器	11-67
0x010	SSP_CPSR	时钟预分频寄存器	11-68



偏移地址	名称	描述	页码
0x014	SSP_INTMASK	中断屏蔽设置或清除寄存器	11-69
0x018	SSP_RINTSTATUS	原始中断状态寄存器	11-70
0x01C	SSP_MINTSTATUS	中断屏蔽状态寄存器	11-71
0x020	SSP_INTCLR	中断清除寄存器	11-71
0x024	SSP_DMACR	DMA 控制寄存器	11-72

11.3.7 寄存器描述

SSP_CR0

SSP_CR0 为控制寄存器 0, 用来控制 SSP 的各种功能。

Offset Address																Register Name			
0x000																SSP_CR0			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name																			
Reset																			
Bits	Access	Name				Description													
[15:8]	RW	scr				串行时钟率。SCR (System Clock Reference) 的值是用来产生 SSP 发送和接收的比特率的。比特率由此公式计算: FSSPCLK/(CPSDVS _R ×(1+SCR))。如果 SSPCPSR[CPSDVS _R]的值为一个 2~254 之间的偶数时, SCR 为一个 0~255 之间的值。													
[7]	RW	sph				SSP_SPICK 相位控制 (只对 Motorola SPI 帧格式适用)。 0: 在传输的第 1 个时钟沿捕获数据。 1: 在传输的第 2 个时钟沿捕获数据。													
[6]	RW	spo				SSP_SPICK 电平 (只对 Motorola SPI 帧格式适用)。 0: 在 SSP_SPICK 管脚产生一个稳定为低的值。 1: 如果没有数据传输, 在 SSP_SPICK 管脚产生一个稳定为高的值。													

[5:4]	RW	frf	帧格式选择。 00: Motorola SPI 帧格式。 01: TI 同步串行帧格式。 10: National Microwire 帧格式。 11: 保留。
[3:0]	RW	dss	数据大小选择。 0000~0010: 保留。 0011: 4bit。 0100: 5bit。 0101: 6bit。 0110: 7bit。 0111: 8bit。 1000: 9bit。 1001: 10bit。 1010: 11bit。 1011: 12bit。 1100: 13bit。 1101: 14bit。 1110: 15bit。 1111: 16bit。

SSP_CR1

SSP_CR1 为控制寄存器 1, 用来控制 SSP 的各种功能。

	Offset Address 0x004																Register Name SSP_CR1			Total Reset Value 0x0000		
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Name	reserved														ms	sse	lbt					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																			
[15:4]	-	reserved	保留。																			
[3]	RW	sod	slave 模式输出不使能。这一 bit 只与 slave 模式有关(ms=1)。在多 slave 的系统中, 可能 SSP master 采用广播的方式发信息给所有的 slave, 而确保只有一个 slave 将数据驱动到自己的串行输出线上, 这样从 slave 输出到																			



			<p>SSP_SPIDI 是连接在一起的。对于这样的系统操作，如果 slave 不准备驱动 SSP_SPIDO 的话，就设置 sod bit。</p> <p>0: SSP 可以在 slave 模式驱动 SSPTXD 输出。</p> <p>1: SSP 不能在 slave 模式驱动 SSPTXD 输出。</p>
[2]	RW	ms	<p>Master 或者 slave 模式选择，这一比特只能在 SSP 处于非使能状态的时候改变。</p> <p>0: 设备被配置成 master (默认)。</p> <p>1: 设备被配置成 slave。</p>
[1]	RW	sse	<p>同步串行接口使能。</p> <p>0: 禁止。</p> <p>1: 使能。</p>
[0]	RW	lbm	<p>环回模式。</p> <p>0: 正常的串行接口操作使能。</p> <p>1: 发送串行移位寄存器输出在内部连接在接收串行移位寄存器输入上。</p>

SSP_DR

SSP_DR 为数据寄存器，当该寄存器被读的时候就是 SSP 接收 FIFO 的出口。当该寄存器被写的时候就是 SSP 的发送 FIFO 的入口。在 Microwire 帧格式下，发送 FIFO 内数据位宽固定为 8bit，接收时无限制。当 [SSP_CRI1\[sse\]](#) 设置为 0 时，接收和发送 FIFO 并不被清 0，这样在我们启动 SSP 之前就可以将数据放在发送 FIFO 里。

	Offset Address								Register Name								Total Reset Value							
	0x008								SSP_DR								0x0000							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Name	data																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
Bits	Access	Name	Description																					
[15:0]	RW	data	读时为接收 FIFO，写时为发送 FIFO。																					

SSP_SR

SSP_SR 为 FIFO 状态寄存器，是一个只读状态寄存器。

Bit	Offset Address 0x00C												Register Name SSP_SR					Total Reset Value 0x0003				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Name	reserved												bsy	rff	rne	tnf	tfe					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1						
Bits	Access	Name	Description																			
[15:5]	-	reserved	保留。																			
[4]	RO	bsy	SSP 繁忙标记。 0: 空闲状态。 1: 繁忙状态。																			
[3]	RO	rff	接收 FIFO 满状态。 0: 未满。 1: 满。																			
[2]	RO	rne	接收 FIFO 空状态。 0: 空。 1: 非空。																			
[1]	RO	tnf	发送 FIFO 不满状态。 0: 满。 1: 未满。																			
[0]	RO	tfe	发送 FIFO 空状态。 0: 非空。 1: 空。																			

SSP_CPSR

SSP_CPSR 为时钟分频系数寄存器，它指定了一个分频因子，对输入的 SSP 工作时钟进行分频，产生 SSP_SPICK。这个因子必须是 2~254 之间的偶数，最低的 1bit 位必须是零。如果向这个寄存器写一个奇数，那么读回来的数据的最低的 1bit 位则必然为零。



Bit	Offset Address 0x010														Register Name SSP_CPSR		Total Reset Value 0x0000	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved														cpsdvsr			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description															
[15:8]	-	reserved	保留。															
[7:0]	RW	cpsdvsr	时钟预分频系数。该数值必须是 2~254 之间的偶数，与 SSP_CRO[scr] 一起对输入时钟 SSPCLK 的频率进行分频。CPSDVSER 最低位读作 0。															

SSP_INTMASK

SSP_INTMASK 为中断屏蔽寄存器，写相应的比特位用于屏蔽或清除一个中断。

Bit	Offset Address 0x014														Register Name SSP_INTMASK		Total Reset Value 0x0000	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved														txim	rxim	rtim	ronim
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description															
[15:4]	-	reserved	保留。															
[3]	RW	txim	发送 FIFO 半空或更少时产生的中断。 0: 屏蔽该中断。 1: 不被屏蔽该中断。															
[2]	RW	rxim	接收 FIFO 半空或更少时产生的中断。 0: 屏蔽该中断。 1: 不屏蔽该中断。															
[1]	RW	rtim	接收 FIFO 不空并且在超时周期之前不读接收 FIFO 产生的超时中断。 0: 屏蔽该中断。 1: 不被屏蔽该中断；															



[0]	RW	rorim	接收 FIFO 在满情况下被写产生的溢出中断。 0: 屏蔽该中断。 1: 不被屏蔽该中断。
-----	----	-------	---

SSP_RINTSTATUS

SSP_RINTSTATUS 为原始中断状态寄存器，读该寄存器时，得到没有屏蔽的当前原始中断状态。写该寄存器时无效。

Offset Address												Register Name				Total Reset Value			
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved												txris	rxris	prris	rorris			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0			
Bits Access Name Description																			
[15:4]	-	reserved		保留。															
[3]	RO	txris		给出 SSPTXINTR 中断的原始（屏蔽之前的）中断状态。 0: 中断无效。 1: 中断有效。															
[2]	RO	rxris		给出 SSPRXINTR 中断的原始（屏蔽之前的）中断状态。 0: 中断无效。 1: 中断有效。															
[1]	RO	rtris		给出 SSPRTXINTR 中断的原始（屏蔽之前的）中断状态。 0: 中断无效。 1: 中断有效。															
[0]	RO	rorris		给出 SSPRORXINTR 中断的原始（屏蔽之前的）中断状态。 0: 中断无效。 1: 中断有效。															



SSP_MINTSTATUS

SSP_MINTSTATUS 为屏蔽后的中断状态寄存器，是只读状态寄存器，写无效。

Bit	Offset Address 0x01C														Register Name SSP_MINTSTATUS				Total Reset Value 0x0000			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Name	reserved														txmis	rxmis	rtmis	rormis				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
Bits	Access	Name	Description																			
[15:4]	-	reserved	保留。																			
[3]	RO	txmis	给出发送 FIFO 屏蔽中断状态（屏蔽之后）SSPTXINTR 中断。 0: 中断无效。 1: 中断有效。																			
[2]	RO	rxmis	给出接收 FIFO 屏蔽中断状态（屏蔽之后）SSPRXINTR 中断。 0: 中断无效。 1: 中断有效。																			
[1]	RO	rtmis	给出接收超时屏蔽中断状态（屏蔽之后）SSPRTINTR 中断。 0: 中断无效。 1: 中断有效。																			
[0]	RO	rormis	给出接收溢出屏蔽中断状态（屏蔽之后）SSPRORINTR 中断。 0: 中断无效。 1: 中断有效。																			

SSP_INTCLR

SSP_INTCLR 为中断清除寄存器，写 1 时清除相应的中断，写 0 无效。

Offset Address																Register Name		Total Reset Value	
0x020																SSP_INTCLR		0x0000	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved																rtic	roric	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																
[15:2]	-	reserved	保留。																
[1]	WO	rtic	SSPRTINTR 中断清除。 0: 不清除该中断。 1: 清除该中断。																
[0]	WO	roric	SSPRORINTR 中断清除。 0: 不清除该中断。 1: 清除该中断。																

SSP_DMCR

SSP_DMCR 为 DMA 功能使能寄存器，用于 SSP 中 DMA 请求功能的使能。

Offset Address																Register Name		Total Reset Value	
0x024																SSP_DMCR		0x0000	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Name	reserved																txdmae	rxdmae	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description																
[15:2]	-	reserved	保留。																
[1]	RW	txdmae	发送 FIFO 的 DMA 操作使能。 0: 禁止。 1: 使能。																
[0]	RW	rxdmae	接收 FIFO 的 DMA 操作使能。 0: 禁止。 1: 使能。																



11.4 红外接口

11.4.1 概述

红外遥控接收单元 IR (Infrared Remoter) 通过红外接口接收红外数据，可以支持 NEC with simple repeat code、NEC with full repeat code、SONY 和 TC9012 四种数据格式解码，以及接收数据错误检测和红外遥控唤醒等功能。

11.4.2 特点

IR 模块有以下特点：

- 软件可配置关闭红外遥控接收模块。
- 支持接收数据溢出中断、接收数据帧格式错误中断、接收数据帧中断、按键释放的中断，组合构成的组合中断。
- 支持初始中断状态查询和屏蔽后中断状态查询。
- 支持中断清除和屏蔽（写清）。
- 支持红外遥控唤醒。
- 支持参考时钟频率 1MHz~128MHz 可选，软件可编程控制分频因子使工作时钟预分频到 1MHz。

11.4.3 信号描述

IR 接口信号如表 11-14 所示。

表11-14 IR 接口信号表

信号名称	方向	描述	对应管脚
IR_RCV	I	从管脚接收到的串行红外数据。 ^a	IRRCV

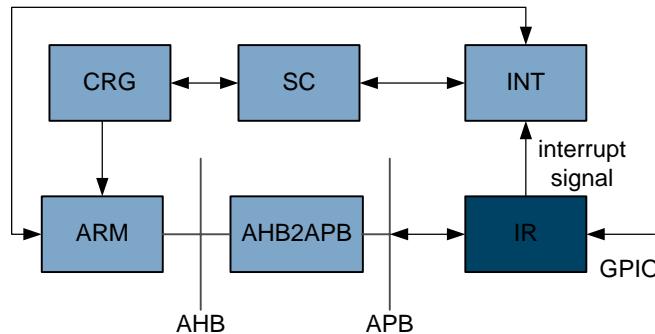
a: 作为红外接收功能时，IRRCV 管脚在单板上要求上拉。

11.4.4 功能描述

当 IR 模块接收到红外遥控器发射的红外信号时，便对其进行解码，然后传送给 ARM 系统。ARM 系统再根据接收到的不同的码来进行相应的操作，实现用户期望的功能。IR 模块连接在 ARM 子系统内的 APB 总线上，当芯片处于低功耗状态时（CPU 停止工作），IR 模块会在接收一个完整的帧数据后，产生中断信号送给 INT 模块，再由 INT 送出控制信号控制系统控制器，进而控制 CRG 唤醒 ARM 子系统，实现红外遥控唤醒功能。

IR模块功能框图如图 11-23所示。

图11-23 IR 模块功能框图



通过对多种红外遥控器发出的信号进行分析发现，不同的遥控器发出的红外指令中，引导码各不相同，而且后面的控制指令差别也很大，甚至指令码的位数也不相同，这是因为这些红外遥控器的设计没有遵循统一的红外遥控标准。但是基本的编码思想是相同的，都是采用不同的周期，不同占空比的脉冲来分别表示 0 和 1。不同遥控器占空比可能不同，且脉冲周期也不尽相同。根据这些不同，对一些码型类似的红外数据进行分类：NEC with simple repeat code 的数据格式、NEC with full repeat code 的数据格式、TC9012 的数据格式和 SONY 的数据格式。

红外接收数据码型统计情况如表 11-15~表 11-17 所示。

表11-15 红外接收数据码型的统计表-NEC 简单重复码

数据格式		NEC with simple repeat code			
		uPD6121G	D6121/BU5777/D1913	LC7461M-C13	AEHA
引导码(10μs)	LEAD_S	900	900	900	337.6
	LEAD_E	450	450	450	168.8
Bit0(10μs)	B0_L	56	56	56	42.2
	B0_H	56	56	56	42.2
Bit1(10μs)	B1_L	56	56	56	42.2
	B1_H	169	169	169	126.6
simple repeat code (10μs)	SLEAD_S	900	900	900	337.6
	SLEAD_E	225	225	225	337.6
burst(10μs)		55	55	55	42.2
帧长(10μs)		10800	10800	10800	8777.6~12828.8
有效数据位		32	32	42	48



表11-16 红外接收数据码型的统计表-NEC 完整重复码

数据格式		NEC with full repeat code						
		uPD6121G	LC7461 M-C13	MN602 4-C5D6	MN6014 -C6D6	MATNEW	MN6030	PANA SONIC
引导码 (10μs)	LEAD_S	900	900	337.6	349.2	348.8	349	352
	LEAD_E	450	450	337.6	349.2	374.4	349	352
Bit0 (10μs)	B0_L	56	56	84.4	87.3	43.6	87.3	88
	B0_H	56	56	84.4	87.3	43.6	87.3	88
Bit1 (10μs)	B1_L	56	56	84.4	87.3	43.6	87.3	88
	B1_H	169	169	253.2	174.6	130.8	261.9	264
simple repeat code (10μs)	SLEAD_S	无	无	无	无	无	无	无
	SLEAD_E							
burst(10μs)		55	55	84.4	87.3	43.6	87.3	88
帧长 (10μs)		10800	10800	10130	10470	12413.6~ 16594.4	10500	10400
有效数据位		32	42	22	24	48	22	22

表11-17 红外接收数据码型的统计表-TC9012 和 SONY 码

数据格式		TC9012	SONY			
		TC9012F/9243	SONY-D7C5	SONY-D7C6	SONY-D7C8	SONY-D7C13
引导码 (10μs)	LEAD_S	450	240	240	240	240
	LEAD_E	450	60	60	60	60
Bit0 (10μs)	B0_L	56	60	60	60	60
	B0_H	56	60	60	60	60
Bit1 (10μs)	B1_L	56	120	120	120	120
	B1_H	169	60	60	60	60
simple repeat code (10μs)	SLEAD_S	无	无	无	无	无
	SLEAD_E					
burst(10μs)		56	无	无	无	无
帧长(10μs)		10800	4500	4500	4500	4500

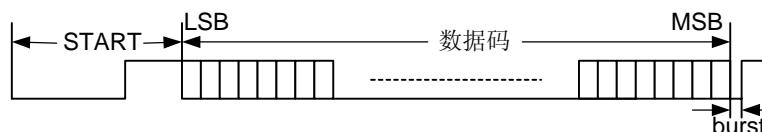
数据格式	TC9012	SONY			
	TC9012F/9243	SONY-D7C5	SONY-D7C6	SONY-D7C8	SONY-D7C13
有效数据位	32	12	13	15	20

11.4.4.2 NEC with simple repeat code 数据格式

帧格式

NEC with simple repeat code 数据格式是由 START (引导码)、数据码和 burst 三部分组成，其中 START 是由一个起始码（低电平），一个结束码（高电平）组成；数据码的有效位数以及某位表示的含义由具体的码型而定，其是按照 LSB first 的顺序接收的；burst 信号用于接收最后一个数据码位。发送单个 NEC with simple repeat code 的帧格式如图 11-24 所示。

图11-24 发送单个 NEC with simple repeat code 码的帧格式



如果按键时间持续超过一帧的时间，则再收到完整数据帧后，接下来收到的数据帧仅由简化的引导码和 burst 信号组成。引导码也是由起始码（低电平）和结束码（高电平）组成，如图 11-25 所示。

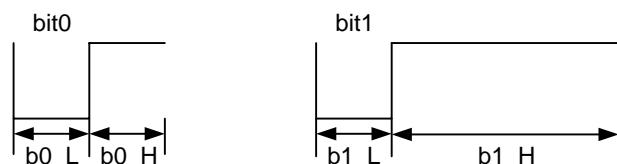
图11-25 持续按键连续发送 NEC with simple repeat code 码的帧格式



码格式

NEC simple repeat code 的 bit0 或 bit1 定义如图 11-26 所示。

图11-26 NEC simple repeat code 码 bit0 和 bit1 的位定义



NEC simple repeat code 单发代码格式和连发代码格式分别如图 11-27 和图 11-28 所示。



图11-27 NEC simple repeat code 码单发代码格式

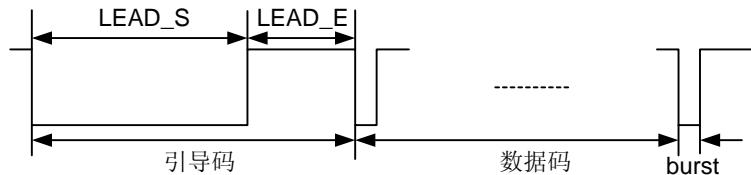
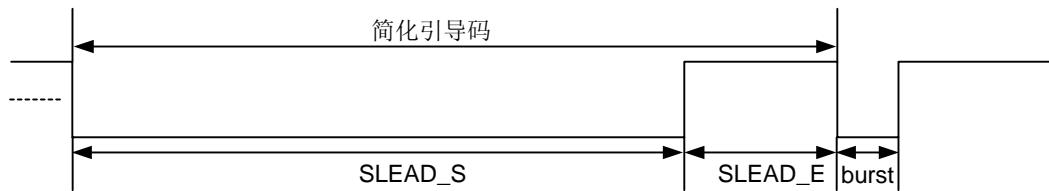


图11-28 NEC simple repeat code 码连发代码格式



注 1: 图中高低电平的脉宽宽度以及帧长均有各个具体码型决定, 请参见表 11-15~表 11-17。

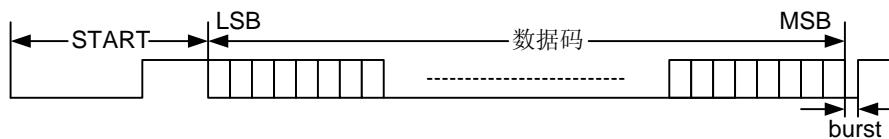
注 2: 帧长不能大于 160ms, 否则无法识别简化引导码。

11.4.4.3 NEC with full repeat code 数据格式

帧格式

NEC with full repeat code 的数据格式是由 START (引导码)、数据码和 burst 三部分组成。START 是由一个起始码 (低电平) 和一个结束码 (高电平) 组成; 数据码的有效位数以及某位表示的含义由具体的码型而定, 其是按照 LSB first 的顺序接收的; burst 信号用于接收最后一个数据码位。发送单个 NEC with full repeat code 帧格式如图 11-29 所示。

图11-29 发送单个 NEC with full repeat code 码的帧格式



如果按键时间持续超过一帧的时间, 则在收到完整数据帧 (第一帧) 后, 接下来收到的数据帧还是一个完整的数据帧格式 (即按照帧间隔重复发送第一帧数据), 如图 11-30 所示。

图11-30 持续按键连续发送 NEC with full repeat code 码的帧格式

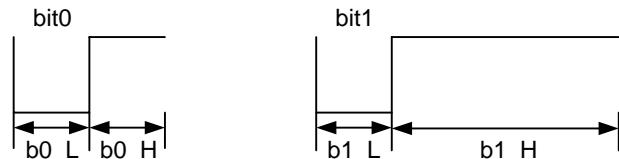


通过图 11-29 和图 11-30 可以看出：NEC with simple repeat code 与 NEC with full repeat code 唯一不同之处就是重复帧的格式，NEC with simple repeat code 发送的是简化的引导码，而 NEC with full repeat code 发送的是完整帧格式，第一帧和重复帧完全相同。

码格式

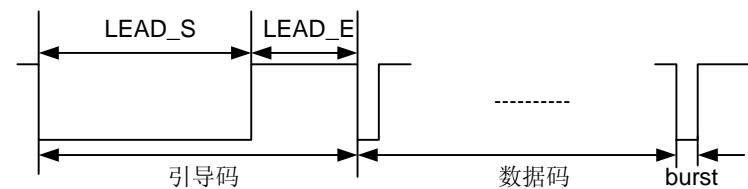
NEC full repeat code 码 bit0 或 bit1 定义如图 11-31 所示。

图11-31 NEC full repeat code 码 bit0 和 bit1 的位定义



NEC full repeat code 码单发代码格式如图 11-32 所示。

图11-32 NEC full repeat code 码单发代码格式



注：图中高低电平的脉宽宽度以及帧长均有各个具体码型决定。请参见表 11-15~表 11-17。

11.4.4.4 TC9012 数据格式

帧格式



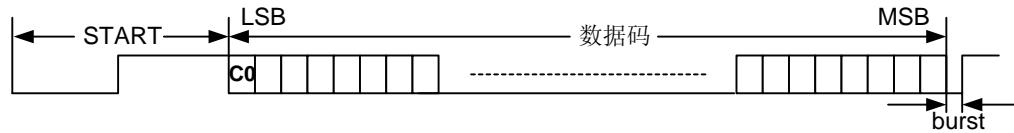
注意

根据 TC9012 码的数据格式特点，所有按键编码的第一位都必须全是 1 或者全是 0，否则会产生不需要的持续按键帧。

TC9012 的数据格式是由 START (引导码)、数据码和 burst 三部分组成，其中 START 是由一个起始码 (低电平)，一个结束码 (高电平) 组成；数据码的有效位数以及某位表示的含义由具体的码型而定，其是按照 LSB first 的顺序接收的；burst 信号用于接收最后一个数据码位。发送单个 TC9012 码的帧格式如图 11-33 所示。



图11-33 发送单个 TC9012 码的帧格式



如果按键时间持续超过一帧的时间，则再收到完整数据帧后，接下来收到的数据帧由引导码、一个数据位和 burst 信号三部分组成。引导码也是由起始码（低电平）和结束码（高电平）组成；该数据位是上一帧接收的第一个数据位（C0）的反码。发送连续 TC9012 码的帧格式如图 11-34 所示。

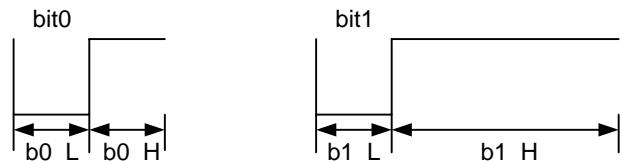
图11-34 持续按键连续发送 TC9012 码的帧格式



码格式

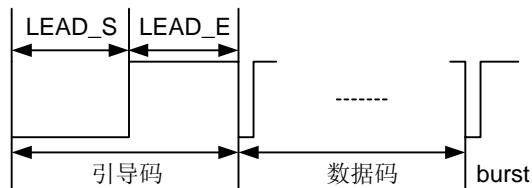
TC9012 码 bit0 或 bit1 定义如图 11-35 所示。

图11-35 TC9012 码 bit0 和 bit1 的位定义



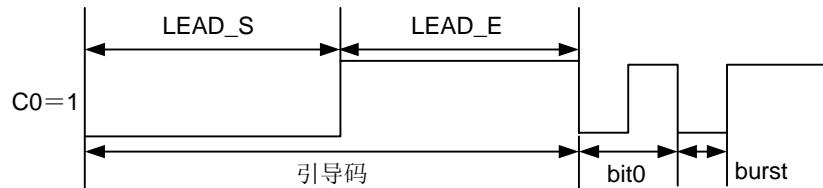
TC9012 码单发代码格式如图 11-36 所示。

图11-36 TC9012 码单发代码格式



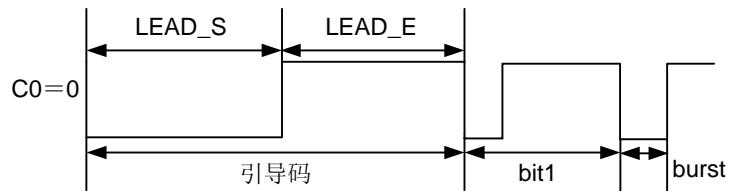
$C0=1$ 时，TC9012 码连发代码格式如图 11-37 所示。

图11-37 TC9012 码连发代码格式 (C0=1)



C0=0 时, TC9012 码连发代码格式如图 11-38 所示。

图11-38 TC9012 码连发代码格式 (C0=0)



注: 图中高低电平的脉宽宽度以及帧长均有各个具体码型决定。请参见表 11-15~表 11-17。另外值得注意的是帧长不能大于 160ms, 否则无法识别重复帧。

11.4.4.5 SONY 的数据格式

帧格式

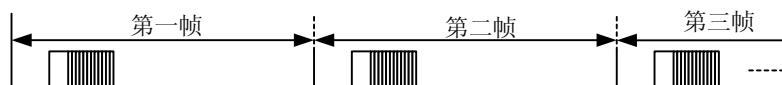
SONY 码数据格式是由 START (引导码) 和数据码两部分组成。其中 START 由一个起始码 (低电平), 一个结束码 (高电平) 组成; 数据码的有效位数以及某位表示的含义由具体的码型而定, 其是按照 LSB first 的顺序接收的。发送单个 SONY 码帧格式如图 11-39 所示。

图11-39 发送单个 SONY 帧格式



如果按键时间持续超过一帧的时间, 则再收到完整数据帧后, 接下来收到的数据帧还是一个完整的数据帧格式。持续按键连续发送 SONY 码帧格式如图 11-40 所示。

图11-40 持续按键连续发送 SONY 码帧格式图

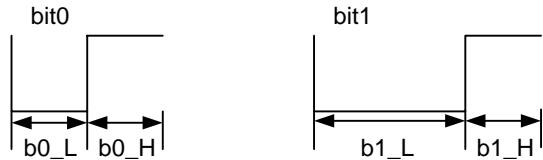




码格式

SONY 码 bit0 或 bit1 定义如图 11-41 所示。

图11-41 bit0 和 bit1 的位定义



注：图中高低电平的脉宽宽度以及帧长均有各个具体码型决定。请参见表 11-15~表 11-17。

11.4.5 工作方式

管脚复用配置

IR 管脚与 GPIO2[3]复用，使用 IR 前，需要配置系统控制器使能相应管脚的 IR 功能。
详细内容请参见 SC_PERCTRL1[17]配置说明。

时钟门控

通过配置系统控制器 SC_PEREN[8]为 1，可以使能 IR 模块的时钟 pclk；通过配置系统控制器 SC_PERDIS[8]为 1，可以关闭 IR 模块的时钟 pclk。

软复位

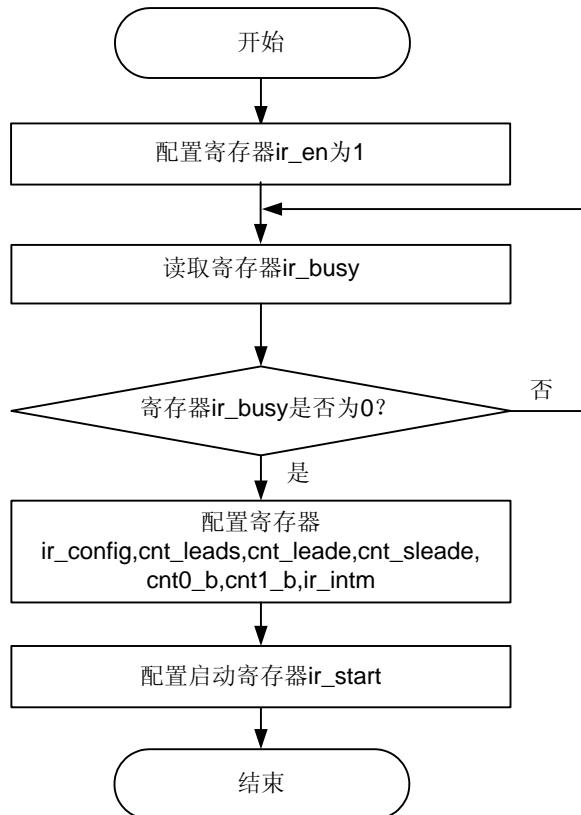
通过配置系统控制器 SC_PERCTRL0[7]为 1，可实现对 IR 模块的单独软复位。复位后各配置寄存器的值均复位为默认值，因此复位后需要重新对这些寄存器进行初始化配置。

寄存器配置实例

IR 模块初始化操作流程如图 11-42 所示。



图11-42 IR 模块初始化操作流程

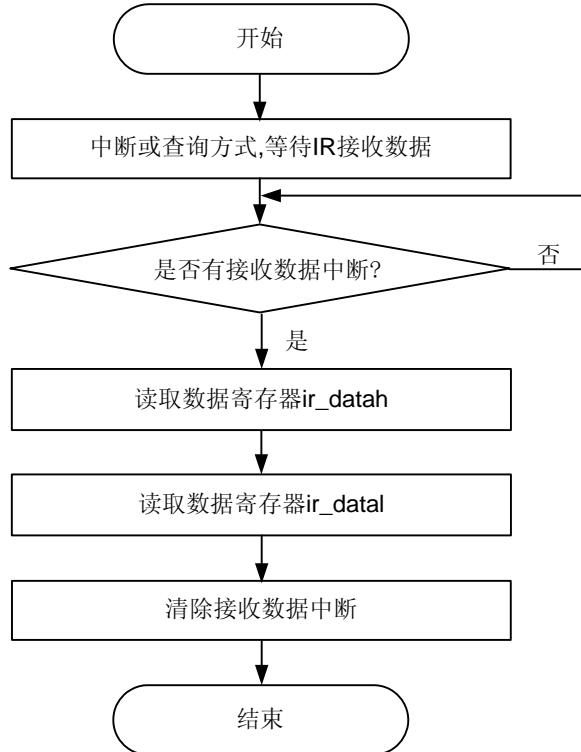


IR 模块初始化操作流程如下：

- 步骤 1 选中 IR 模块地址空间，开始 IR 初始化配置操作。
 - 步骤 2 配置寄存器 `IR_EN` 为 1，打开 IR 接收模块。
 - 步骤 3 读取寄存器 `IR_BUSY`，通过查询 `IR_BUSY` 来判断 IR 模块配置的当前状态：若读取的值为 1，表明 IR 模块处于配置忙状态，则返回到步骤 3，继续查询 `IR_BUSY`。**注意：**此时软件不要对 IR 模块的其他控制寄存器进行配置，否则配置无效；若读取的值为 0，表明 IR 模块处于配置空闲状态，则进入步骤 4。
 - 步骤 4 配置寄存器 `IR_CFG`、`IR_LEADS`、`IR_LEADE`、`IR_SLEADE`、`IR_B0`、`IR_B1`、`IR_INT_MASK`。**注意：**用户可以根据需要更新相应寄存器，如果不更新，则寄存器保持原值。
 - 步骤 5 配置 IR 启动寄存器 `IR_START`。`IR_START` 必须要等所有的 IR 控制寄存器都配置完成后，才能配置，因为它被用来产生启动信号，只要一配置，IR 模块就会根据控制寄存器的值进行红外数据接收。
- 结束



图11-43 读取解码数据的操作流程



读取解码数据的操作流程如下：

- 步骤 1 选中 IR 模块地址空间。
 - 步骤 2 中断或查询方式等待接收数据帧。中断方式下，当 CPU 接收到 IR 模块的中断请求信号时，查询 `IR_INT_STATUS[intms_rcv]` 的值，如果为 1 表明 IR 模块接收到一个数据帧，则进入步骤 3；否则进入步骤 2，继续等待中断。查询方式下，软件不停（或每间隔一定时间）读取寄存器 `IR_INT_STATUS[intrs_rcv]` 的值，如果为 1 表明 IR 模块接收到一个数据帧，则进入步骤 3；当读取的值为 0 时，说明尚未接收到数据帧，否则进入步骤 2，继续查询。
 - 步骤 3 读取数据寄存器 `IR_DATAH`。（如果一帧内的数据位数不大于 32 位，可以省略此步骤。）
 - 步骤 4 读取数据寄存器 `IR_DATA1`。
 - 步骤 5 清除接收数据中断。
- 结束

11.4.6 寄存器概览

IR 寄存器概览如表 11-18 所示。

表11-18 IR 寄存器概览（基址是 0x101E_9000）

偏移地址	名称	描述	页码
0x00	IR_EN	IR 接收使能控制寄存器	11-84
0x04	IR_CFG	IR 配置寄存器	11-85
0x08	IR_LEADS	引导码起始位裕量配置寄存器	11-86
0x0C	IR_LEADE	引导码结束位裕量配置寄存器	11-87
0x10	IR_SLEADE	简化引导码结束位裕量配置寄存器	11-88
0x14	IR_B0	数据 0 的判断电平裕量配置寄存器	11-89
0x18	IR_B1	数据 1 的判断电平裕量配置寄存器	11-90
0x1C	IR_BUSY	配置忙标志寄存器	11-91
0x20	IR_DATAH	IR 接收解码数据的高 16 位寄存器	11-92
0x24	IR_DATAL	IR 接收解码数据的低 32 位	11-92
0x28	IR_INT_MASK	IR 中断屏蔽寄存器	11-93
0x2C	IR_INT_STATUS	IR 中断状态寄存器	11-93
0x30	IR_INT_CLR	IR 中断清除寄存器	11-95
0x34	IR_START	IR 启动配置寄存器	11-96

11.4.7 寄存器描述

IR_EN

IR_EN 为 IR 接收使能控制寄存器。当寄存器 IR_EN[0]=0 时，其他寄存器只可读不可写，且读出值为寄存器的复位值。



注意

软件必须先配置寄存器 IR_EN[0]=1，才能配置其他寄存器，否则配置无效。



Offset Address			Register Name			Total Reset Value																										
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												ir_en			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:1]	-	reserved	保留。																													
[0]	RW	ir_en	IR 接收模块的使能。 0: 关闭 IR 接收模块。 1: 打开 IR 接收模块。																													

IR_CFG

IR_CFG 为 IR_CFGIR 配置寄存器。

IR 支持的参考时钟频率为 1MHz~128MHz, 其与分频因子 ir_freq 的对应关系是:

- 当参考时钟频率为 1MHz, 分频因子 ir_freq 需配置为 0x00。
- 当参考时钟频率为 128MHz, 分频因子 ir_freq 需配置为 0x7F。

当 IR 的参考时钟是 1~128MHz 内的非整数倍频率时, 选用四舍五入的方法选择相应的分频因子。举例: 参考为 12.1MHz, 选用分频因子为 0x0B; 参考为 12.8MHz, 选用分频因子为 0x0C。

对于频偏和计数偏差的关系: 基频 f, 频率变化 D_f , 则频偏率 $ratio = D_f/f$; 计数器计数偏差 Dcnt; 判断电平宽度 s (μs 为单位), 则计数偏差: $Dcnt = \lceil 0.1 \times s \times ratio \rceil$ 。所以在时钟有频偏的情况下, 参数值的有效范围要移位, 如果频率上升的话, 相应的裕量值应改为: [min+Dcnt, max+Dcnt], 其中 min 和 max 为无偏移时的裕量值; 如果频率下降的话, 范围位移为: [min-Dcnt, max-Dcnt]。以引导码的起始位裕量来举例来说: 假如基频为 100MHz, 频率上漂 0.1MHz, 那么 $ratio = 0.1/100 = 0.001$, 则假设 $s = 9000\mu s$, 则 $Dcnt = \lceil 0.1 \times 9000 \times 0.001 \rceil = 1$, 则 ir_leads 的裕量值应改为[0x033D, 0x3CD]。

如果软件对 ir_bits 位配置 0x30~0x3F 范围内的值, 配置无效, 寄存器 ir_bits 保持原值不变。

关于具体码型属于哪类码族, 请参见表 11-15~表 11-17。



注意

必须在确保 ir_busy=0 并且 ir_en=1 时，才能配置此寄存器，否则配置无效，寄存器保持原值。

Offset Address		Register Name		Total Reset Value																
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8	0x004	IR_CFG	0x0000_1F0B																
Name	reserved															ir_format	ir_bits	reserved	ir_freq	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1																		
Bits	Access	Name	Description																	
[31:16]	-	reserved	保留。																	
[15:14]	RW	ir_format	数据码型。 00: NEC with simple repeat code 的数据格式。 01: TC9012 的数据格式。 10: NEC with full repeat code 的数据格式。 11: SONY 的数据格式。																	
[13:8]	RW	ir_bits	一帧内的数据位。 0x00~0x2F: 分别对应一帧内包含 1~48 个数据位。 0x30~0x3F: 保留。																	
-	RW	reserved	保留。																	
[6:0]	RW	ir_freq	工作时钟分频因。 0x00~0x7F: 分别对应工作时钟分频因子 1~128。																	

IR_LEADS

IR_LEADS 为引导码起始位裕量配置寄存器。

为了准确判断引导码的起始位，需要在具体码型的典型值左右考虑一定的裕量，具体码型的典型值请参见表 11-15~表 11-17 中 LEAD_S 的值。

- 对于典型值不小于 400 (其精度为 10μs) 的脉宽，建议裕量范围设为典型值的 8%。举例说明：D6121 码型，其 LEAD_S 的典型值为 900，那么相应的 $ir_leads_min=900 \times 92\% = 828 = 0x33C$, $ir_leads_max=900 \times 108\% = 972 = 0x3CC$ 。
- 对于典型值小于 400 (其精度为 10μs) 的脉宽，建议裕量范围设为典型值的 20%。举例说明：SONY-D7C5 码型，其 LEAD_S 的典型值为 240，那么相应的 $ir_leads_min=240 \times 80\% = 192 = 0xC0$, $ir_leads_max=240 \times 120\% = 288 = 0x120$ 。



基本的配置原则是：ir_leads_max 不小于 ir_leads_min，并且 cnt_leads_min 大于 cnt0_b_max 和 cnt1_b_max。



注意

必须在确保 ir_busy=0 并且 ir_en=1 时，才能配置此寄存器，否则配置无效，寄存器保持原值；另外寄存器保留值配置无效，仍然保持原值不变。

Offset Address			Register Name																Total Reset Value													
0x008			IR_LEADS																0x033C_03CC													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved		cnt_leads_min										reserved		cnt_leads_max																	
Reset	0	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0	0	
Bits	Access	Name	Description																													
[31:26]	-	reserved	保留。																													
[25:16]	RW	cnt_leads_min	引导码起始位的最小脉宽。 0x000~0x007：保留。																													
[15:10]	-	reserved	保留。																													
[9:0]	RW	cnt_leads_max	引导码起始位的最大脉宽。 0x000~0x007：保留。																													

IR_LEADE

IR_LEADE 为引导码结束位裕量配置寄存器。

为了准确判断引导码的结束位，需要在具体码型的典型值左右考虑一定的裕量，裕量范围约为典型值的 8%。具体码型的典型值请参见表 11-15~表 11-17 中 LEAD_E 的值。

- 对于典型值不小于 400（其精度为 10μs）的脉宽，建议裕量范围设为典型值的 8%。举例说明：D6121 码型，其 LEAD_E 的典型值为 450，那么相应的 $cnt_leade_min=450 \times 92\% = 414 = 0x19E$, $cnt_leade_max=450 \times 108\% = 486 = 0x1E6$ 。
- 对于典型值小于 400（其精度为 10μs）的脉宽，建议裕量范围设为典型值的 20%。举例说明：SONY-D7C5 码型，其 LEAD_E 的典型值为 60，那么相应的 $cnt_leade_min=60 \times 80\% = 48 = 0x030$, $cnt_leade_max=60 \times 120\% = 72 = 0x048$ 。

基本的配置原则是：cnt_leade_max 不小于 cnt_leade_min 的值。



注意

- 必须在确保 ir_busy=0 并且 ir_en=1 时，才能配置此寄存器，否则配置无效，寄存器保持原值；另外寄存器保留值配置无效，仍然保持原值不变。
- 对于 NEC with simple repeat code 的码族，其 cnt_sleade 的裕量范围和 cnt_leade 的裕量范围设置不能重合，否则当实际计数值落入重合范围内时，无法识别简化引导码，会导致帧格式错误。

Offset Address			Register Name	Total Reset Value	
Bit	0x00C			IR_LEADE	0x019E_01E6
Name	reserved	cnt_leade_min	reserved	cnt_leade_max	
Reset	0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 0				
Bits	Access	Name	Description		
[31:25]	-	reserved	保留。		
[24:16]	RW	cnt_leade_min	引导码结束位的最小脉宽。 0x000~0x007: 保留。		
[15:9]	-	reserved	保留。		
[8:0]	RW	cnt_leade_max	引导码结束位的最大脉宽。 0x000~0x007: 保留。		

IR_SLEADE

IR_SLEADE 为简化引导码结束位裕量配置寄存器。

为了准确判断简化引导码的结束位，需要在具体码型的典型值左右考虑一定的裕量。具体码型的典型值请参见表 11-15~表 11-17 中 SLEAD_E 的值。

- 对于典型值不小于 225（其精度为 10 μ s）的脉宽，建议裕量范围设为典型值的 8%。举例说明：D6121 码型，其 SLEAD_E 的典型值为 225，那么相应的 cnt_sleade_min=225×92%=207=0xCF，cnt_sleade_max=225×108%=243=0xF3。
- 对于典型值小于 225（其精度为 10 μ s）的脉宽，建议裕量范围设为典型值的 20%。举例说明：比如某种码型其 SLEAD_E 的典型值为 60，那么相应的 cnt_sleade_min=60×80%=48=0x30，cnt_sleade_max=60×120%=72=0x48。

基本的配置原则是：cnt_sleade_max 不小于 cnt_sleade_min 的值。



注意

- 必须在确保 ir_busy=0 并且 ir_en=1 时，才能配置此寄存器，否则配置无效，寄存器保持原值；另外寄存器保留值配置无效，仍然保持原值不变。
 - 对于 NEC with simple repeat code 的码族，cnt_sleade 的裕量范围和 cnt_leade 的裕量范围设置不能重合，否则当实际计数值落入重合范围内时，无法识别简化引导码，会导致帧格式错误。
 - 对于 NEC with simple repeat code 的数据格式，才需配置此寄存器；对于其他格式，无需配置此寄存器。

IR_B0

IR_B0 为数据 0 的判断电平裕量配置寄存器。

为了准确判断 bit0，需要在具体码型的典型值左右考虑一定的裕量，裕量范围约为典型值的 20%。

- 对于 NEC with simple repeat code、NEC with simple repeat code 和 TC9012 这三类码型，其包含的具体码型的典型值请参见表 11-15~表 11-17 中 $B0_H$ 的值。举例说明：D6121 码型，其 $B0_H$ 的典型值为 56（其精度为 $10\mu s$ ），那么相应的 $cnt0_b_min=56\times80\%=45=0x2D$ ， $cnt0_b_max=56\times120\%=67=0x43$ 。
 - 对于 SONY 的数据格式，其包含的具体码型的典型值请参见表 11-15~表 11-17 中 $B0_L$ 的值。举例说明：SONY-D7C5 码型，其 $B0_L$ 的典型值为 60（其精度为 $10\mu s$ ），那么相应的 $cnt0_b_min=60\times80\%=48=0x30$ ， $cnt0_b_max=60\times120\%=72=0x48$ 。

基本的配置原则是: `cnt0_b_max` 不小于 `cnt0_b_min` 的值。



注意

- 必须在确保 ir_busy=0 并且 ir_en=1 时，才能配置此寄存器，否则配置无效，寄存器保持原值；另外寄存器保留值配置无效，仍然保持原值不变。
- 对于四类码型，bit0 判断电平裕量范围和 bit1 判断电平范围设置不能重合，否则当实际计数值落入重合范围内时，无法识别 bit1，只能被误解接收到 bit0。

Offset Address		Register Name		Total Reset Value	
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	IR_B0		0x002D_0043	
Name	reserved	cnt0_b_min	reserved	cnt0_b_max	
Reset	0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1	Bits	Access	Name	Description
[31:23]	-	reserved		保留。	
[22:16]	RW	cnt0_b_min		bit0 判断电平的最小脉宽。 0x00~0x07: 保留。	
[15:7]	-	reserved		保留。	
[6:0]	RW	cnt0_b_max		bit0 判断电平的最大脉宽。 0x00~0x07: 保留。	

IR_B1

IR_B1 为数据 1 的判断电平裕量配置寄存器。

为了准确判断 bit1，需要在具体码型的典型值左右考虑一定的裕量，裕量范围约为典型值的 20%。

- 对于 NEC with simple repeat code、NEC with simple repeat code 和 TC9012 这三类码族，其包含的具体码型的典型值请参见表 11-15~表 11-17 中 B1_H 的值。举例说明：D6121 码型，其 B1_H 的典型值为 169（其精度为 10μs），那么相应的 cnt1_b_min=169×80%=135=0x87，cnt1_b_max=169×120%=203=0xCB。
- 对于 SONY 的数据格式，其包含的具体码型的典型值请参见表 11-15~表 11-17 中 B1_L 的值。举例说明：SONY-D7C5 码型，其 B1_L 的典型值为 120（其精度为 10μs），那么相应的 cnt1_b_min=120×80%=96=0x60，cnt1_b_max=120×120%=144=0x90。

基本的配置原则是：cnt1_b_max 不小于 cnt1_b_min 的值。



注意

- 必须在确保 ir_busy=0 并且 ir_en=1 时，才能配置此寄存器，否则配置无效，寄存器保持原值；另外寄存器保留值配置无效，仍然保持原值不变。
- 对于四类码型，bit0 判断电平裕量范围和 bit1 判断电平范围设置不能重合，否则当实际计数值落入重合范围内时，无法识别 bit1，只能被误解接收到 bit0。

Offset Address			Register Name	Total Reset Value	
Bit	0x018			IR_B1	0x0087_00CB
Name	reserved	cnt1_b_min	reserved	cnt1_b_max	
Reset	0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 1				
Bits	Access	Name	Description		
[31:25]	-	reserved	保留。		
[24:16]	RW	cnt1_b_min	bit1 判断电平的最小脉宽。 0x000~0x007: 保留。		
[15:9]	-	reserved	保留。		
[8:0]	RW	cnt1_b_max	bit1 判断电平的最大脉宽。 0x000~0x007: 保留。		

IR_BUSY

IR_BUSY 为配置忙标志寄存器。

Offset Address			Register Name	Total Reset Value	
Bit	0x01C			IR_BUSY	0x0000_0000
Name	reserved	reserved	reserved	ir_busy	
Reset	0 0				
Bits	Access	Name	Description		
[31:1]	-	reserved	保留。		

[0]	RO	ir_busy	忙状态标志。 0: 空闲状态, 可以配置数据。 1: 忙状态, 软件不可以配置数据。
-----	----	---------	--

IR_DATAH

IR_DATAH 为 IR 接收解码数据的高 16 位寄存器。

IR_DATAH 是接收到的解码数据的高 16 位, IR_DATAL 是接收到的解码数据的低 32 位。具体哪些数据位有效取决于具体码型一帧内包含的有效数据位数, 请参见表 11-15~表 11-17 的有效数据位。

数据存储原则: 按照由高到低的顺序存储在 IR_DATAH 和 IR_DATAL 中 (MSB……LSB), 先存满 IR_DATAL, 然后再存放 IR_DATAH, 未用到的高位作为保留位。软件读取数据的顺序必须是: 先读 IR_DATAH, 然后再读 IR_DATAL。

对于具体每个数据位表示什么含义, 硬件不做判断, 只是单纯负责接收所有数据位, 最终由软件进行统一处理。

	Offset Address																																Register Name	Total Reset Value
	0x020																														IR_DATAH	0x0000_0000		
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	reserved																														ir_datah			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	Bits	Access	Name	Description																														
	[31:16]	-	reserved	保留。																														
	[15:0]	RO	ir_datah	接收到的解码数据的高 16 位数据。																														

IR_DATAL

IR_DATAL 为 IR 接收解码数据的低 32 位寄存器。

	Offset Address																																Register Name	Total Reset Value
	0x024																															IR_DATAL	0x0000_0000	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name	ir_datal																																ir_datal	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	Bits	Access	Name	Description																														
	[31:0]	RO	ir_datal	接收到的解码数据的低 32 位数据。																														

IR_INT_MASK

IR_INT_MASK 为 IR 中断屏蔽寄存器。如果中断全部屏蔽后，无法支持红外遥控唤醒功能。



必须在确保 **IR_EN[0]=1** 时，才能配置此寄存器，否则配置无效，寄存器保持原值。

Offset Address		Register Name	Total Reset Value					
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	IR_INT_MASK	0x0000_0000					
Name	reserved		intm_release	intm_overflow	intm_framerr	intm_rcv		
Reset	0 0							
Bits	Access	Name	Description					
[31:4]	-	reserved	保留。					
[3]	RW	intm_release	禁止或允许按键释放的中断请求。 0: 允许中断请求。 1: 禁止中断请求。					
[2]	RW	intm_overflow	禁止或允许接收数据溢出错的中断请求。 0: 允许中断请求。 1: 禁止中断请求。					
[1]	RW	intm_framerr	禁止或允许接收帧格式错误的中断请求。 0: 允许中断请求。 1: 禁止中断请求。					
[0]	RW	intm_rcv	禁止或允许接收到数据帧的中断请求。 0: 允许中断请求。 1: 禁止中断请求。					

IR INT STATUS

IR_INT_STATUS 为 IR 中断状态寄存器。

寄存器中涉及到的中断定义如下：

- 接收数据溢出中断

如果出现当前帧的数据 CPU 没有及时响应取走，而下一帧数据也已经收到的情况，下一帧数据将会覆盖当前帧数据，同时上报屏蔽前接收数据溢出错中断请求。

- 接收数据帧格式错误中断

如果接收到的数据帧不完整以及数据脉宽不满足裕量范围，则会上报屏蔽前的接收帧格式错误中断请求。

- 接收数据帧中断

当接收到一个完整的帧数据后，则会上报屏蔽前接收到数据帧中断请求。

- 支持按键释放的检测中断

对于 NEC with simple repeat code 和 TC9012 码族的数据格式，在检测到一个有效起始同步码之后的 160ms 内，如果没有再次检测到起始同步码，或者检测到非简化引导码而是有效数据帧时，则会上报屏蔽前遥控器按键释放中断。对于 NEC with full repeat code 和 SONY 两种码制不支持按键释放中断。

硬件没有中断优先级仲裁，任何一个或多个屏蔽后的中断源有效，都会产生中断。

		Offset Address	Register Name	Total Reset Value																													
		0x02C	IR_INT_STATUS	0x0000_0000																													
Bit	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0																																
Name	reserved																reserved																
Reset	0 0																																
Bits	Access	Name	Description																														
[31:20]	-	reserved	保留。																														
[19]	RO	intms_release	屏蔽后的按键释放的中断状态。 0: 未产生中断。 1: 已产生中断。																														
[18]	RO	intms_overflow	屏蔽后的接收数据溢出错中断状态。 0: 未产生中断。 1: 已产生中断。																														
[17]	RO	intms_framerr	屏蔽后的接收帧格式错误中断状态。 0: 未产生中断。 1: 已产生中断。																														



[16]	RO	intms_rcv	屏蔽后的接收到数据帧中断状态。 0: 未产生中断。 1: 已产生中断。
[15:4]	RO	reserved	保留。
[3]	RO	Intrs_release	屏蔽前的按键释放的中断状态。 0: 未产生中断。 1: 已产生中断。
[2]	RO	intrs_overflow	屏蔽前的接收数据溢出错中断状态。 0: 未产生中断。 1: 已产生中断。
[1]	RO	intrs_framerr	屏蔽前的接收帧格式错误中中断状态。 0: 未产生中断。 1: 已产生中断。
[0]	RO	intrs_rcv	屏蔽前的接收到数据帧中断状态。 0: 未产生中断。 1: 已产生中断。

IR_INT_CLR

IR_INT_CLR 为 IR 中断清除寄存器。

	Offset Address			Register Name	Total Reset Value																											
	0x030			IR_INT_CLR	0x0000_0000																											
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																									intc_rcv				
[31:4]	-	reserved	保留。																									intc_framerr				
[3]	WO	intc_release	清除遥控器按键释放中断请求。 0: 不清除。 1: 清除。																								intc_overflow					

[2]	WO	intc_overflow	清除接收数据溢出错中断请求。 0: 不清除。 1: 清除。
[1]	WO	intc_framerr	清除接收帧格式错误中断请求。 0: 不清除。 1: 清除。
[0]	WO	intc_rcv	清除接收到数据帧中断请求。 0: 不清除。 1: 清除。 如果接收数据帧中断请求产生后, 软件未读走 IR_DATA1 中的数据就直接对本位进行写 1 操作, 无法清除该中断请求。

IR_START

IR_START 为 IR 启动配置寄存器。在其他寄存器的值配置完成之后, 启动 IR 模块时, 只要往该地址进行一次写操作 (写操作数可以为任意值), 就可以启动配置寄存器。

	Offset Address		Register Name	Total Reset Value																												
	0x034		IR_START	0x0000_0000																												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reserved																												ir_start			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description																													
[31:1]	-	reserved	保留。																													
[0]	WO	ir_start	IR 启动配置寄存器。 写入任何值, 启动 IR; 读该寄存器返回 IR 的启动状态。 0: 未启动。 1: 启动。																													



11.5 通用目的输入输出接口

11.5.1 概述

Hi3511/Hi3512 支持 8 组 GPIO (General Purpose Input/Output)，每组 GPIO 提供 8 个可编程的输入输出管脚。每个管脚可以配置为输入或者输出。这些管脚用于生成特定应用的输出信号或采集特定应用的输入信号。作为输入管脚时，GPIO 可作为中断源；作为输出管脚时，每个 GPIO 都可以独立地清 0 或置 1。每个 GPIO 管脚均与其他管脚复用，管脚复用说明请参见“2.2 管脚复用”，相关的控制请参见“3.10 系统控制器”。

11.5.2 特点

GPIO 模块有以下特点：

- 每个 GPIO 管脚均可配置为输入或输出。
 - 当 GPIO 作为输入管脚时，可作为中断源，每个 GPIO 管脚都具有独立的中断控制。
 - 当 GPIO 作为输出管脚时，每个 GPIO 管脚都可以独立地清 0 或置 1。
- GPIO 的中断通过 [GPIO_IS](#) 等 7 个寄存器进行控制。通过这些寄存器可以选择中断源、极性以及边沿特性。GPIO 对应的中断寄存器请参见“3.4 中断系统”。
 - 当有多个中断同时发生的时候，将会统一汇集成一个中断进行上报，GPIO 的中断映射关系请参见“表 3-13 中断映射关系表”。
 - [GPIO_IS](#)、[GPIO_IBE](#)、[GPIO_IEN](#) 三个寄存器共同决定了中断源的特性和中断触发类别。
 - 通过 [GPIO_RIS](#) 和 [GPIO_MIS](#) 分别读取中断的原始状态和屏蔽后的状态。通过 [GPIO_IE](#) 可以控制中断的最终上报情况。此外还提供了单独的 [GPIO_IC](#) 用于对中断状态进行清除控制。

11.5.3 信号描述

GPIO 接口信号如表 11-19 所示。

表11-19 GPIO 接口信号描述

信号名称	方向	描述	对应管脚
GPIO0_0	I/O	GPIO 双向数据信号。与 VI 水平同步信号、VI0 的高清模式数据输入信号、以及中断输入信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0HS
GPIO0_1	I/O	GPIO 双向数据信号。与 VI 垂直同步信号、VI0 的高清模式数据输入信号、以及 SPI 片选信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0VS



信号名称	方向	描述	对应管脚
GPIO0_2	I/O	GPIO 双向数据信号。与 SIO1 的 I ² S 接收左右声道选择信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	SIO1RFS
GPIO0_3	I/O	GPIO 双向数据信号。与 SIO1 接收位流时钟信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	SIO1RCK
GPIO0_4	I/O	GPIO 双向数据信号。与 SIO1 的数据输入信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	SIO1DI
GPIO0_5	I/O	GPIO 双向数据信号。与 VI 水平同步信号、VI0 的高清模式数据输入信号、以及 UART 数据接收信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI2HS
GPIO0_6	I/O	GPIO 双向数据信号。与 VI 垂直同步信号、VI0 的高清模式数据输入信号、以及 UART 数据发送信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI2VS
GPIO0_7	I/O	GPIO 双向数据信号。与延迟 PCB 板反馈输出信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	DDRMRCVO
GPIO1_0	I/O	GPIO 双向数据信号。与延迟 PCB 板反馈输入信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	DDRMRCVI
GPIO1_1	I/O	GPIO 双向数据信号。与 SMI 片选 1 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	EBICS1N
GPIO1_2	I/O	GPIO 双向数据信号。与 SMI 地址总线信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	EBIADDR24
GPIO1_3	I/O	GPIO 双向数据信号。与 MMC 命令信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	SDIOCMD
GPIO1_4	I/O	GPIO 双向数据信号。与 MMC 数据 0 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	SDIODAT0
GPIO1_5	I/O	GPIO 双向数据信号。与 MMC 数据 1 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	SDIODAT1



信号名称	方向	描述	对应管脚
GPIO1_6	I/O	GPIO 双向数据信号。与 MMC 数据 2 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SDIODAT2
GPIO1_7	I/O	GPIO 双向数据信号。与 PCI 总线申请信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	PCIREQ4N
GPIO2_0	I/O	GPIO 双向数据信号。与 PCI 总线仲裁信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	PCIGRANT4N
GPIO2_1	I/O	GPIO 双向数据信号。与 MMC 数据 3 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SDIODAT3
GPIO2_2	I/O	GPIO 双向数据信号。与 MMC 输出时钟信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SDIOCK
GPIO2_3	I/O	GPIO 双向数据信号。与红外遥控输入信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	IRRCV
GPIO2_4	I/O	GPIO 双向数据信号。与 UART1 数据接收信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	URXD1
GPIO2_5	I/O	GPIO 双向数据信号。与 UART1 数据发送信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	UTXD1
GPIO2_6	I/O	GPIO 双向数据信号。与 UART1 输出 RTS 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	URTSN1
GPIO2_7	I/O	GPIO 双向数据信号。与 UART1 输入 CTS 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	UCTSN1
GPIO3_0	I/O	GPIO 双向数据信号。与 MII 接收数据错误指示信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	ERXERR
GPIO3_1	I/O	GPIO 双向数据信号。与 MII 冲突载波侦听信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	ECRS
GPIO3_2	I/O	GPIO 双向数据信号。与 MII 冲突指示信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	ECOL



信号名称	方向	描述	对应管脚
GPIO3_3	I/O	GPIO 双向数据信号。与 I ² C 总线数据/地址信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”）	SDA
GPIO3_4	I/O	GPIO 双向数据信号。与 I ² C 总线时钟信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”）	SCL
GPIO3_5	I/O	GPIO 双向数据信号。与可编程输出时钟信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	ACKOUT
GPIO3_6	I/O	GPIO 双向数据信号。与 VI 656 数据比特 2 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI2DAT2
GPIO3_7	I/O	GPIO 双向数据信号。与 VI 656 数据比特 6 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI2DAT6
GPIO4_0	I/O	GPIO 双向数据信号。与 VI 数据 2 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT2
GPIO4_1	I/O	GPIO 双向数据信号。与 VI 数据 3 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT3
GPIO4_2	I/O	GPIO 双向数据信号。与 VI 数据 4 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”）	VI0DAT4
GPIO4_3	I/O	GPIO 双向数据信号。与 VI 数据 5 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT5
GPIO4_4	I/O	GPIO 双向数据信号。与 VI 数据 6 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT6
GPIO4_5	I/O	GPIO 双向数据信号。与 VI 数据 7 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT7
GPIO4_6	I/O	GPIO 双向数据信号。与 VI 数据 8 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT8
GPIO4_7	I/O	GPIO 双向数据信号。与 VI 数据 9 信号复用。（复用时的配置信息请参见“ 管脚复用配置 ”。）	VI0DAT9



信号名称	方向	描述	对应管脚
GPIO5_0	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 0 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT0
GPIO5_1	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 1 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”)	VI3DAT1
GPIO5_2	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 2 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT2
GPIO5_3	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 3 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT3
GPIO5_4	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 4 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT4
GPIO5_5	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 5 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT5
GPIO5_6	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 6 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT6
GPIO5_7	I/O	GPIO 双向数据信号。与 VI3 接口接收数据 7 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI3DAT7
GPIO6_0	I/O	GPIO 双向数据信号。与 SPI 输入数据信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SPIDI
GPIO6_1	I/O	GPIO 双向数据信号。与 SPI 片选 0 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”)	SPICSN0
GPIO6_2	I/O	GPIO 双向数据信号。与 SIO0 的 I2S 或 PCM 发送帧同步信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SIO0XFS
GPIO6_3	I/O	GPIO 双向数据信号。与 SIO0 的 I2S 或 PCM 接口发送位流时钟信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SIO0XCK
GPIO6_4	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 0 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT0



信号名称	方向	描述	对应管脚
GPIO6_5	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 1 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT1
GPIO6_6	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 2 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT2
GPIO6_7	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 3 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT3
GPIO7_0	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 4 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT4
GPIO7_1	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 5 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”)	VI1DAT5
GPIO7_2	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 6 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT6
GPIO7_3	I/O	GPIO 双向数据信号。与 VI1 接口接收数据 7 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI1DAT7
GPIO7_4	I/O	GPIO 双向数据信号。与 SMI 等待输入信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	EBIRDYN
GPIO7_5	I/O	GPIO 双向数据信号。与 SPI 时钟信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SPICK
GPIO7_6	I/O	GPIO 双向数据信号。与 SPI 输出数据信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	SPIDO
GPIO7_7	I/O	GPIO 双向数据信号。与 VI 656 数据比特 4 信号复用。(复用时的配置信息请参见“ 管脚复用配置 ”。)	VI2DAT4

11.5.4 功能描述

每组 GPIO 提供 8 个可编程的输入输出管脚。每个管脚可以配置为输入或者输出。这些管脚用于生成或采集特定应用的输出或输入信号。



作为输入管脚时，可作为中断源；作为输出管脚时，每个 GPIO 都可以独立地清 0 或置 1。GPIO 可以根据电平或跳变值产生可屏蔽的中断。GPIOINTR (General Purpose Input Output Interrupt) 给中断控制器一个指示，表示有中断发生。用户可以配置中断，使中断可以是电平触发也可以是边沿触发的。

11.5.5 工作方式

管脚复用配置

GPIO0_0 与 VI 水平同步信号、VI0 的高清模式数据输入信号、以及中断输入信号复用，使用 GPIO0[0]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[1:0]配置说明。

GPIO0_1 与 VI 垂直同步信号、VI0 的高清模式数据输入信号、以及 SPI 片选信号复用，使用 GPIO0[1]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[1:0]配置说明。

GPIO0_2 与 SIO1 的 I²S 接收左右声道选择信号复用，使用 GPIO0[2]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[5]配置说明。

GPIO0_3 与 SIO1 接收位流时钟信号复用，使用 GPIO0[3]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[5]配置说明。

GPIO0_4 与 SIO1 的数据输入信号复用，使用 GPIO0[4]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[5]配置说明。

GPIO0_5 与 VI 水平同步信号、VI0 的高清模式数据输入信号、以及 UART 数据接收信号复用，使用 GPIO0[5]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[8:7]配置说明。

GPIO0_6 与 VI 垂直同步信号、VI0 的高清模式数据输入信号、以及 UART 数据发送信号复用，使用 GPIO0[6]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[8:7]配置说明。

GPIO0_7 与延迟 PCB 板反馈输出信号复用，使用 GPIO0[7]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[15]配置说明。

GPIO1_0 与延迟 PCB 板反馈输入信号复用，使用 GPIO1[0]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[15]配置说明。

GPIO1_1 与 SMI 片选 1 信号复用，使用 GPIO1[1]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[20]配置说明。

GPIO1_2 与 SMI 地址总线信号复用，使用 GPIO1[2]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[21]配置说明。

GPIO1_3 与 MMC 命令信号复用，使用 GPIO1[3]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[9]配置说明。

GPIO1_4 与 MMC 数据 0 信号复用，使用 GPIO1[4]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[9]配置说明。

GPIO1_5 与 MMC 数据 1 信号复用，使用 GPIO1[5]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[9]配置说明。

GPIO1_6 与 MMC 数据 2 信号复用, 使用 GPIO1[6]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[9]配置说明。

GPIO1_7 与 PCI 总线申请信号复用, 使用 GPIO1[7]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[13]配置说明。

GPIO2_0 与 PCI 总线仲裁信号复用, 使用 GPIO2[0]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[13]配置说明。

GPIO2_1 与 MMC 数据 3 信号复用, 使用 GPIO2[1]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[9]配置说明。

GPIO2_2 与 MMC 输出时钟信号复用, 使用 GPIO2[2]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[9]配置说明。

GPIO2_3 与红外遥控输入信号复用, 使用 GPIO2[3]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[17]配置说明。

GPIO2_4 与 UART1 数据接收信号复用, 使用 GPIO2[4]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[16]配置说明。

GPIO2_5 与 UART1 的数据发送信号复用, 使用 GPIO2[5]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[16]配置说明。

GPIO2_6 与 UART1 输出 RTS 信号复用, 使用 GPIO2[6]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[16]配置说明。

GPIO2_7 与 UART1 输出 CTS 信号复用, 使用 GPIO2[7]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[16]配置说明。

GPIO3_0 与 MII 接收数据错误指示信号复用, 使用 GPIO3[0]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[24]配置说明。

GPIO3_1 与 MII 冲突载波侦听信号复用, 使用 GPIO3[1]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[24]配置说明。

GPIO3_2 与 MII 冲突指示信号复用, 使用 GPIO3[2]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[24]配置说明。

GPIO3_3 与 I²C 总线数据/地址信号复用, 使用 GPIO3[3]需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[14]配置说明。

GPIO3_4 与 I²C 总线时钟信号复用, 使用 GPIO3[4]配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[14]配置说明。

GPIO3_5 与可编程输出时钟信号复用。使用 GPIO3[5]配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[19]配置说明。

GPIO3_6 与 VI 656 数据比特 2 信号复用, 使用 GPIO3[6]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[3]配置说明。

GPIO3_7 与 VI 656 数据比特 6 信号复用, 使用 GPIO3[7]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[3]配置说明。

GPIO4_0 与 VI 数据 2 信号复用, 使用 GPIO4[0]前, 需要配置系统控制器使能相应管脚的 GPIO 功能, 详见 SC_PERCTRL1[2]配置说明。



GPIO4_1 与 VI 数据 3 信号复用，使用 GPIO4[1]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO4_2 与 VI 数据 4 信号复用，使用 GPIO4[2]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO4_3 与 VI 数据 5 信号复用，使用 GPIO4[3]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO4_4 与 VI 数据 6 信号复用，使用 GPIO4[4]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO4_5 与 VI 数据 7 信号复用，使用 GPIO4[5]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO4_6 与 VI 数据 8 信号复用，使用 GPIO4[6]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO4_7 与 VI 数据 9 信号复用，使用 GPIO4[7]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[2]配置说明。

GPIO5_0 与 VI3 接口接收数据 0 信号复用，使用 GPIO5[0]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_1 与 VI3 接口接收数据 1 信号复用，使用 GPIO5[1]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_2 与 VI3 接口接收数据 2 信号复用，使用 GPIO5[2]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_3 与 VI3 接口接收数据 3 信号复用，使用 GPIO5[3]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_4 与 VI3 接口接收数据 4 信号复用，使用 GPIO5[4]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_5 与 VI3 接口接收数据 5 信号复用，使用 GPIO5[5]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_6 与 VI3 接口接收数据 6 信号复用，使用 GPIO5[6]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO5_7 与 VI3 接口接收数据 7 信号复用，使用 GPIO5[7]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[6]配置说明。

GPIO6_0 与 SPI 输入数据信号复用，使用 GPIO6[0]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[10]配置说明。

GPIO6_1 与 SPI 片选 0 信号复用，使用 GPIO6[1]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[10]配置说明。

GPIO6_2 与 SIO0 的 I2S 或 PCM 发送帧同步信号复用，使用 GPIO6[2]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[18]配置说明。

GPIO6_3 与 SIO0 的 I2S 或 PCM 接口发送位流时钟信号复用，使用 GPIO6[3]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[11]配置说明。

GPIO6_4 与 VII 接口接收数据 0 信号复用，使用 GPIO6[4]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO6_5 与 VII 接口接收数据 1 信号复用，使用 GPIO6[5]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO6_6 与 VII 接口接收数据 2 信号复用，使用 GPIO6[6]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO6_7 与 VII 接口接收数据 3 信号复用，使用 GPIO6[7]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO7_0 与 VII 接口接收数据 4 信号复用，使用 GPIO7[0]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO7_1 与 VII 接口接收数据 5 信号复用，使用 GPIO7[1]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO7_2 与 VII 接口接收数据 6 信号复用，使用 GPIO7[2]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO7_3 与 VII 接口接收数据 7 信号复用，使用 GPIO7[3]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[23]配置说明。

GPIO7_4 与 SMI 等待输入信号复用，使用 GPIO7[4]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[4]配置说明。

GPIO7_5 与 SPI 时钟信号复用，使用 GPIO7[5]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[10]配置说明。

GPIO7_6 与 SPI 输出数据信号复用，使用 GPIO7[6]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[10]配置说明。

GPIO7_7 与 VI 656 数据比特 4 信号复用，使用 GPIO7[7]前，需要配置系统控制器使能相应管脚的 GPIO 功能，详见 SC_PERCTRL1[3]配置说明。

接口复位

上电复位时，所有的寄存器都被清 0。因此管脚默认为输入。

复位信号有效时，GPIO 有如下状态：

- 通过清除 **GPIO_IE** 中相应的比特位使中断无效。
- 所有的寄存器被清 0。
- 所有的管脚都被配置为输入。
- 原始中断寄存器都被清 0。
- 中断被设为边沿触发的中断。

通用输入输出

每个管脚可以配置为输入或者输出，具体步骤如下：



- 步骤 1 选择要使用的 GPIO 管脚，将复用管脚配置选择为 GPIO 功能，配置系统控制器使能相应管脚的 GPIO 功能，请参见“[管脚复用配置](#)”。
- 步骤 2 设置要使用的 GPIO 管脚的方向，通过配置寄存器 [GPIO_DIR](#)，选择 GPIO 是输入还是输出。
- 步骤 3 当 GPIO 用于输入时，外部信号通过 GPIO 管脚送进来，可通过 [GPIO_DATA](#) 寄存器查看输入信号值，需要注意的是，输入的信号会同时送到和 GPIO 复用的管脚上；当 GPIO 用于输出时，可以通过写 [GPIO_DATA](#) 寄存器，GPIO 会将写进此寄存器的值通过 GPIO 输出，需要注意的是，此时若使能了 GPIO 中断功能，则当输出信号满足触发条件时，也会产生中断。

----结束

中断操作

如果要产生中断，那么必需按照下面的初始化顺序才能避免假中断：

- 步骤 1 配置 [GPIO_IS](#) 选择边沿触发或电平触发。
- 步骤 2 配置 [GPIO_IEN](#)，选择下降沿/上升沿触发和高电平/低电平触发。
- 步骤 3 如果选择了边沿触发，正确配置 [GPIO_IBE](#)，选择单沿或双沿触发方式。
- 步骤 4 保证 GPIO 数据线在这些操作的过程中保持稳定。
- 步骤 5 通过往寄存器 [GPIO_IC](#) 写 0xFF，清中断。
- 步骤 6 配置 [GPIO_IE](#)，使能中断。

----结束

GPIO 的中断设置由 7 个寄存器控制。当有一个或多个 GPIO 管脚产生中断，一个组合中断输出会送到中断控制器。对于边沿触发的中断，软件必需清除该中断以使能更深的中断；对于电平触发的中断，外部中断源应该保持该电平直到处理器识别到该中断。

11.5.6 寄存器概览

8 组 GPIO 的基地址如[表 11-20](#) 所示。

表11-20 8 组 GPIO 寄存器基地址

寄存器	基地址
GPIO0	0x101E_4000
GPIO1	0x101E_5000
GPIO2	0x101E_6000
GPIO3	0x101E_7000
GPIO4	0x101F_7000

寄存器	基地址
GPIO5	0x101F_8000
GPIO6	0x101F_9000
GPIO7	0x101F_A000

表 11-21 是单组 GPIO 内部寄存器的偏移地址以及定义，GPIO0~GPIO7 具有相同的寄存器组。GPIO_n 对应的寄存器地址为：GPIO_n 基地址+该寄存器偏移地址。

表11-21 GPIO 寄存器概览

偏移地址	寄存器名称	描述	页码
0x000~0x3FC	GPIO_DATA	GPIO 数据寄存器	11-108
0x400	GPIO_DIR	GPIO 方向控制寄存器	11-109
0x404	GPIO_IS	GPIO 中断触发寄存器	11-110
0x408	GPIO_IBE	GPIO 双沿触发中断寄存器	11-110
0x40C	GPIO_IEV	GPIO 触发中断条件寄存器	11-111
0x410	GPIO_IE	GPIO 中断屏蔽寄存器	11-111
0x414	GPIO_RIS	GPIO 原始中断状态寄存器	11-111
0x418	GPIO_MIS	GPIO 屏蔽状态中断寄存器	11-112
0x41C	GPIO_IC	GPIO 中断清除寄存器	11-112
0x420	RESERVED	保留	-

11.5.7 寄存器描述

GPIO_DATA

GPIO_DATA 为 GPIO 数据寄存器，用来对输入/输出数据进行缓存。

	Offset Address	Register Name	Total Reset Value					
	0x000~0x3FC	GPIO_DATA	0x00					
Bit	7	6	5	4	3	2	1	0
Name	gpio_data							
Reset	0	0	0	0	0	0	0	0
Bits	Access	Name	Description					



[7:0]	R/W	gpio_data	当 GPIO 配置为输入模式时, 为 GPIO 输入数据; 当 GPIO 配置为输出模式时, 为输出数据。各比特均可独立控制。与 GPIO_DIR 配合使用。
-------	-----	-----------	---

当配置 [GPIO_DIR](#) 中对应位为输出时, 写入该寄存器的值将会输出到相应的管脚 (注意配置正确的管脚复用); 如果配置为输入时, 将会读取相应输入管脚的值。



注意

该寄存器利用 PADDR[9:2]实现了读写寄存器比特的屏蔽操作。该寄存器对应 256 个地址空间。

PADDR[9:2]分别对应 GPIO_DATA[7:0], 当相应的 bit 为高时, 则可以对相应的位进行读写操作; 反之, 若对应 bit 为低则不能进行操作。

例如:

- 若地址为 0x3FC (0b11_1111_1100), 则对该寄存器 8bit 操作全部有效。
- 若地址为 0x200 (0b10_0000_0000), 则仅能对 GPIO_DATA[7]进行有效操作。



注意

当 [GPIO_DIR](#) 相应的比特配置为输入时, 有效读取的结果将返回管脚的值; 当配置为输出的时候, 有效读取的结果将返回写入的值。

GPIO_DIR

GPIO_DIR 为 GPIO 方向控制寄存器, 用来配置 GPIO 管脚方向。

	Offset Address		Register Name		Total Reset Value			
	0x400		GPIO_DIR		0x00			
Bit	7	6	5	4	3	2	1	0
Name	gpio_dir							
Reset	0	0	0	0	0	0	0	0
Bits	Access	Name	Description					
[7:0]	RW	gpio_dir	GPIO 方向控制寄存器, bit[7:0]分别对应 GPIO_DATA [7:0], 用于控制该比特是作为输入信号还是作为输出信号, 各比特独立控制, 含义如下:					

			0: 输入。 1: 输出。
--	--	--	------------------

GPIO_IS

GPIO_IS 为 GPIO 中断触发寄存器，用来配置 GPIO 管脚触发电平方式。

	Offset Address								Register Name	Total Reset Value
	0x404								GPIO_IS	0x00
Bit	7	6	5	4	3	2	1	0		
Name	gpio_is									
Reset	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description							
[7:0]	RW	gpio_is	GPIO 中断触发控制寄存器，bit[7:0]分别对应 GPIO_DATA[7:0]，各比特独立控制，含义如下： 0: 边沿触发中断。 1: 电平触发中断。							

GPIO_IBE

GPIO_IBE 为 GPIO 双沿触发中断寄存器，用来配置 GPIO 管脚沿触发方式。

	Offset Address								Register Name	Total Reset Value
	0x408								GPIO_IBE	0x00
Bit	7	6	5	4	3	2	1	0		
Name	gpio_ibc									
Reset	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description							
[7:0]	RW	gpio_ibc	GPIO 中断沿触发控制寄存器，bit[7:0]分别对应 GPIO_DATA[7:0]，各比特独立控制，含义如下： 0: 单边沿触发中断，具体是上升沿还是下降沿触发由 GPIO_IEV 控制。 1: 双边沿触发中断。							



GPIO_IEV

GPIO_IEV 为 GPIO 触发中断条件寄存器，用来配置 GPIO 管脚触发中断条件。

Offset Address								Register Name	Total Reset Value
Bit	0x40C							GPIO_IEV	0x00
Name	7	6	5	4	3	2	1	0	
gpio_iev									
Reset	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description						
[7:0]	RW	gpio_iev	GPIO 触发中断条件寄存器，bit[7:0]分别对应 GPIO_DATA[7:0]，各比特独立控制，含义如下： 0: 下降沿或低电平触发中断。 1: 上升沿或高电平触发中断。						

GPIO_IE

GPIO_IE 为 GPIO 中断屏蔽寄存器，用来屏蔽 GPIO 管脚中断。

Offset Address								Register Name	Total Reset Value
Bit	0x410							GPIO_IE	0x00
Name	7	6	5	4	3	2	1	0	
gpio_ie									
Reset	0	0	0	0	0	0	0	0	
Bits	Access	Name	Description						
[7:0]	RW	gpio_ie	GPIO 中断屏蔽寄存器，bit[7:0]分别对应 GPIO_DATA[7:0]，各比特独立控制，含义如下： 0: 屏蔽中断。 1: 不屏蔽中断。						

GPIO_RIS

GPIO_RIS 为 GPIO 原始中断状态寄存器，用来查询 GPIO 管脚原始中断状态。

Offset Address	Register Name	Total Reset Value
0x414	GPIO_RIS	0x00

Bit	7	6	5	4	3	2	1	0
Name	gpio_ris							
Reset	0	0	0	0	0	0	0	0
Bits	Access	Name	Description					
[7:0]	RO	gpio_ris	GPIO 原始中断寄存器, bit[7:0]分别对应 GPIO_DATA[7:0], 指示未屏蔽的中断状态。该状态不受 GPIO_IE 寄存器屏蔽控制。 0: 已发生中断。 1: 未发生中断。					

GPIO_MIS

GPIO_MIS 为 GPIO 屏蔽状态中断寄存器, 用来查询 GPIO 管脚屏蔽后的中断状态。

Offset Address									Register Name	Total Reset Value
0x418									GPIO_MIS	0x00
Bit	7	6	5	4	3	2	1	0		
Name	gpio_mis									
Reset	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description							
[7:0]	RO	gpio_mis	GPIO 屏蔽后中断寄存器, bit[7:0]分别对应 GPIO_DATA[7:0], 指示经屏蔽后的中断状态。该状态受 GPIO_IE 寄存器屏蔽控制。 0: 中断无效。 1: 中断有效。							

GPIO_IC

GPIO_IC 为 GPIO 中断清除寄存器, 用来清除 GPIO 管脚产生的中断, 同时清除 GPIO_RIS 寄存器和 GPIO_MIS 寄存器。

Offset Address									Register Name	Total Reset Value
0x41C									GPIO_IC	0x00
Bit	7	6	5	4	3	2	1	0		
Name	gpio_ic									
Reset	0	0	0	0	0	0	0	0		
Bits	Access	Name	Description							



[7:0]	WC	gpio_ic	GPIO 中断清除寄存器, bit[7:0]分别对应 GPIO_DATA[7:0] , 各比特独立控制。 0: 无影响。 1: 清除中断。
-------	----	---------	---

RESERVED

RESERVED 为 GPIO 保留寄存器, 必须按下面要求配置。

	Offset Address								Register Name	Total Reset Value	
	0x420								RESERVED	0x00	
Bit	7	6	5	4	3	2	1	0			
Name	reserved										
Reset	0	0	0	0	0	0	0	0			
Bits	Access	Name	Description								
[7:0]	RW	reserved	必须设为 0x00。								



12 测试接口

关于本章

本章描述内容如下表所示。

标题	内容
12.1 概述	介绍测试接口的功能。
12.2 工作模式	介绍 Hi3511/Hi3512 芯片的工作模式。
12.3 JTAG 调试	介绍 JTAG 接口的信号和调试模式。



12.1 概述

Hi3511/Hi3512 的测试接口 JTAG 符合 IEEE (Institute of Electrical and Electronics Engineers) 1149.1 标准，可用于 ARM 软件调试和板级测试。

12.2 工作模式

Hi3511/Hi3512 有 2 种工作模式，通过芯片管脚 TESTMODE0 的配置可以实现这两种模式之间的切换。

各工作模式说明如表 12-1 所示。

表12-1 Hi3511/Hi3512 工作模式说明表

TESTMODE0	模式说明
0	Hi3511/Hi3512 正常工作，此时可通过 JTAG 对 ARM 软件进行调试。
1	Hi3511/Hi3512 处于测试模式，此时可以进行芯片 DFT (Design For Test) 测试和板级互连测试。

12.3 JTAG 调试

12.3.1 接口信号

Hi3511/Hi3512 的 JTAG 接口信号与标准的 JTAG 接口信号完全一致，Hi3511/Hi3512 的 JTAG 接口信号的描述如表 12-2 所示。

表12-2 Hi3511/Hi3512 的 JTAG 接口信号表

信号名	信号描述
TCK	JTAG 时钟输入，芯片内部下拉。建议单板下拉。
TDI	JTAG 数据输入，芯片内部上拉。建议单板上拉。
TMS	JTAG 模式选择输入，芯片内部上拉。建议单板上拉。
TRSTN	JTAG 复位输入，芯片内部下拉。正常工作建议单板下拉。如果通过 JTAG 口连接 Realview-ICE 等调试器，建议单板上拉。
TDO	JTAG 数据输出。建议单板上拉。

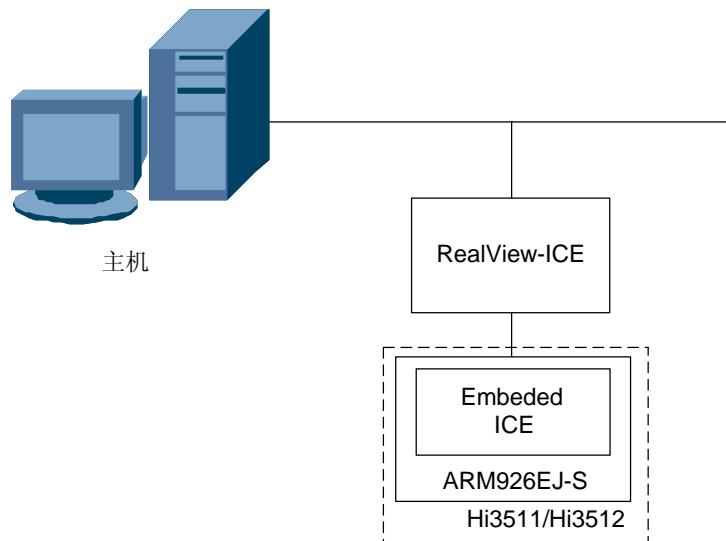


12.3.2 调试模式

对 ARM 进行调试

用一个能够支持 JTAG 接口的 ICE 设备（如 RealView-ICE）和主机相连，并采用相应的调试软件，就可以通过 JTAG 接口对 ARM 进行调试，如图 12-1 所示。

图12-1 对 ARM 进行调试的系统示例图



通过 JTAG 接口单元（如 ARM RealView-ICE）进行初始设置时，需要将 TESTMODE0 设为 0b0。

板级测试模式

Hi3511/Hi3512 除了支持通过 JTAG 接口进行软件调试外，还支持在单板上的一些互连测试，如 Hi3511/Hi3512 与其他芯片的连接测试等。板级互连测试通过标准的 JTAG 控制器实现。在进行板级互连测试时，需要将 TESTMODE0 设置为 0b1。



13 视频处理

关于本章

本章描述内容如下表所示。

标题	内容
13.1 视频编解码器	介绍视频编解码器的功能和特点。
13.2 图形加速	介绍图形加速模块的功能和特点。
13.3 图形缩放	介绍图形缩放模块的功能和特点。



13.1 视频编解码器

13.1.1 概述

视频编解码器是一个支持 H.264、JPEG/MJPEG 协议的视频编码和视频解码的处理单元，它是由运行于 ARM 处理器上的 Video Firmware 和内嵌的视频编解码硬件加速引擎构成，支持单独进行视频编码或单独进行视频解码，也支持同时进行视频编码和视频解码、即同编同解。

13.1.2 特点

视频编解码器有以下特点：

- 支持 H.264 Main Profile@Level 3.0（以及 Level2 和 Level1）编解码
 - 单独编码或者单独解码时最大支持 75fps@D1
 - 同编同解时最大支持 30fps@D1
- 支持 JPEG/MJPEG Baseline 编解码
 - 最大支持 300 万象素的编码，帧率 5fps
 - 支持同编同解
- 支持大小码流同时编码
 - 大小码流支持 H.264/H.264、H.264/JPEG、JPEG/H.264、JPEG/JPEG 四种协议组合
 - 大小码流使用相同的源图像，大码流直接由源图像编码得到，小码流由源图像缩小后编码得到
 - 大小码流编码图象尺寸的水平、垂直比例均可设置为 1:1、2:1、4:1
 - 小码流编码图像支持的最大尺寸为 CIF
- 支持视频前端 de-interlace
 - de-interlace 功能可使能/禁止
- 支持时域滤波
 - 时域滤波功能可开关
- 支持视频前端 OSD 叠加处理
 - 支持最多 4 个区域的编码前 OSD 叠加
 - 支持任意大小，任意位置（不超出图像大小和位置）OSD 叠加
 - 支持 129 级的 alpha 叠加
 - OSD 叠加功能可开关
- 支持运动检测
 - 支持 SAD（Sum Of Absolute Difference）值输出
 - 支持运动矢量 MV 输出
- H264 支持 CBR/VBR 码率控制，16kbit/s~20Mbit/s



13.2 图形加速

13.2.1 概述

2D 图形加速引擎 TDE (Two Dimensional Engine) 利用硬件进行图形绘制，可以减少对 CPU 的占用，同时提高内存带宽的资源利用率。

TDE 通过 AHB Master 总线接口读写位图数据，通过 AHB Slave 总线接口获得 CPU 的寄存器配置信息。

13.2.2 特点

TDE 模块有以下特点：

- 源位图、目标位图和输出位图支持 RGB 444、ARGB 4444、RGB 555、RGB 565、ARGB 1555、RGB 888、ARGB 8888 像素格式。
- 支持源位图、目标位图和输出位图格式分别可配。
- 支持数据小端格式。
- 支持在固定矩形区域的 pattern fill。
- 支持硬件画垂直、水平直线。
- 支持硬件矩形填充。
- 支持 alpha blending 操作。
- 支持 1/4/8 bit 的 alpha 值，支持像素 alpha 值的操作和全局 alpha 值操作。
- 支持源、目的位图的 color space 操作。
- 支持 ROP (Raster Operation) 操作。
- 支持图像首地址按像素对齐，word 定义为 32 比特。
 - 16 比特像素格式的图像首地址 half word 地址对齐
 - 32 比特像素格式的图像首地址 word 地址对齐
- 提供任务完成中断。
- 支持垂直、水平二阶抗闪烁。

13.3 图形缩放

13.3.1 概述

DSU (Dedicated Scaling Unit) 完成对视频和图形数据的缩放、数据格式之间的相互转换、空域去噪、色彩增强以及对比度拉伸。DSU 为独立功能单元，输入输出基于 AHB 总线。

13.3.2 特点

- 图像数据搬移。
- 支持 8 倍以内比例的水平、垂直缩放。



- 支持同时进行水平垂直缩放。
- 支持水平/垂直亮度、色度各 32 组 6 阶滤波系数，系数可配置。
- 支持亮度分量的单独处理。
- 支持色度分量的单独处理。
- 支持 YUV4:2:2 到 YUV4:2:0 的相互转换。
- 支持色彩增强和对比度拉伸。
- 支持亮度分量的空域去噪。



14 Hi3511 与 Hi3512 差异说明

关于本章

本章描述内容如下表所示。

标题	内容
14.1 硬件接口差异	介绍 Hi3511 与 Hi3512 硬件接口差异。
14.2 管脚差异	介绍 Hi3511 与 Hi3512 管脚差异。



14.1 硬件接口差异

Hi3511 与 Hi3512 硬件接口差异如表 14-1 所示。

表14-1 Hi3511 与 Hi3512 硬件接口差异

类别	Hi3511	Hi3512
硬件接口	4 个 54MHz BT656 接口	2 个 54MHz BT656 接口

14.2 管脚差异

除视频输入 VI 的个别管脚不同外, Hi3511 与 Hi3512 管脚完全兼容, 管脚复用关系完全相同, 具体如表 14-2 所示。

表14-2 Hi3511 与 Hi3512 VI 管脚差异

Pin	Hi3511		Hi3512	
	管脚名称	管脚描述	管脚名称	管脚描述
Y2	VI2CK	VI2 接口的时钟。	NC	NC (悬空)。
AB1	VI3CK	VI3 接口的时钟。	NC	NC (悬空)。
W4	VI3DAT0	VI3 接口的数据 DAT0。	GPIO5_0	通用 GPIO。
W3	VI3DAT1	VI3 接口的数据 DAT1。	GPIO5_1	通用 GPIO。
Y3	VI3DAT2	VI3 接口的数据 DAT2。	GPIO5_2	通用 GPIO。
AA2	VI3DAT3	VI3 接口的数据 DAT3。	GPIO5_3	通用 GPIO。
AB2	VI3DAT4	VI3 接口的数据 DAT4。	GPIO5_4	通用 GPIO。
AC2	VI3DAT5	VI3 接口的数据 DAT5。	GPIO5_5	通用 GPIO。
AC3	VI3DAT6	VI3 接口的数据 DAT6。	GPIO5_6	通用 GPIO。
AB3	VI3DAT7	VI3 接口的数据 DAT7。	GPIO5_7	通用 GPIO。



A 缩略语

A

ACD	Auto Command Done	自动停止指令完成
AES	Advanced Encryption Standard	先进的加密标准
AHB	Advanced High-performance Bus	-
AMBA	Advanced Microcontroller Bus Architecture	先进的微处理器总线结构
ARM	ARM	ARM 公司的 RISC Core

B

BVACT	Bottom Vertical Active Area	底场垂直活动有效区域
BVBB	Bottom Vertical Back Blank	底场垂直后消隐
BVFB	Bottom Vertical Front Blank	底场垂直前消隐

C

CBC	Cipher Block Chaining	密码分组链接
CD	Command Done	指令完成
CFB	Cipher Feedback	密码反馈
CL	CAS Latency	读延迟
CPU	Central Processing Unit	中央处理单元
CRC	Cyclic Redundancy Check	循环冗余校验
CRG	Clock Reset Generation	时钟复位产生模块
CTR	Counter	计数器

**D**

DCRC	Data CRC Error	数据 CRC 错误
DDR	Double Data-Rate	双数据速率
DES	Data Encryption Standard	数据加密标准
DFT	Design For Test	可测试性设计
DLL	Delay Locked Loop	延迟锁相环
DMA	Direct Memory Access	直接存储器存取
DMAC	Direct Memory Access Controller	直接存储器存取控制器
DQS	Data Strobe	数据选通信号
DRTO	Data Read Timeout	数据读超时
DTO	Data Transfer Over	数据传输完成
DVR	Digital Video Recorder	硬盘录像机

E

EBE	End-bit error	结束位错误
EBI	External Bus Interface	外部总线接口
ECB	Electronic Codebook	电子密码书
EOF	End Of Frame	帧结束
EOP	End Of Packet	包结束
ETH	Ethernet MAC	以太网接口

F

FIFO	First In First Out	先入先出
FIQ	Fast Interrupt Request	快速中断请求
FRUN	FIFO Underrun/Overrun Error	FIFO 溢出错误

G

GPIO	General Purpose Input/Output	通用目的输入/输出
-------------	------------------------------	-----------

H

HACT	Horizontal Active Area	水平活动有效区域
-------------	------------------------	----------



HCCA	Host Controller Communication Area	主机控制器通信区域
HFB	Horizontal Front Blank	水平前消隐
HLE	Hardware Locked Error	硬件锁定错误
HPW	Horizontal Pulse Width	水平脉冲宽度
HTO	Data starvation-by-host timeout	控制器读写数据超时
HBB	Horizontal Back Blank	水平后消隐
I		
I²C	Inter-Integrated Circuit	一种串行总线协议标准
IEEE	Institute of Electrical and Electronics Engineers	美国电气和电子工程师协会
I²S	Inter-IC Sound	I ² S 音频输入输出接口
IR	Infrared Remoter	红外遥控接口
IRQ	Interrupt Request	中断请求
ISR	Interrupt Service Routine	中断服务程序
ITCM	Instruction TCM	指令紧耦合存储器
IV	Initialization Vector	初始向量
J		
JTAG	Joint Test Action Group	联合测试行动小组
L		
LSB	Least Significant Bit	结尾 bit 位
M		
MAC	Media Access Control	媒体访问控制
MCU	Micro Controller Unit	微型控制单元
MDIO	Management Data Input/Output	控制数据输入输出接口
MII	Media Independent Interface	介质无关接口
MMC	Multi-media Card	多媒体卡
MSB	Most Significant Bit	起始 bit 位

**N**

NTSC	National Television Systems Committee	国家电视系统委员会制式
-------------	---------------------------------------	-------------

O

OFB	Output Feedback	输出反馈
OHCI	Open Host Controller Interface	公开主机控制器接口
OSD	On Screen Display	屏幕显示图形层
OTG	On-The-Go	-

P

PAL	Phase Alternating Line	逐行倒向制式
PCB	Printed Circuit Board	印刷电路板
PCI	Peripheral Component Interconnect	一种通用的本地总线
PCM	Pulse Code Modulation	脉冲编码调制
PID	Packet ID	包标识
PSRAM	Pseudo Static Random Access Memory	伪静态随机存储器

Q

QXGA	Quantum Extended Graphics Array	昆腾扩展图像序列是一种计算机图形显示标准格式
-------------	---------------------------------	------------------------

R

RAM	Random-Access Memory	随机存取存储器
RCRC	Response CRC error	响应 CRC 错误
RE	Response error	响应错误
ROM	Read Only Memory	只读存储器
ROP	Raster Operation	光栅操作
RTO	Response Timeout	响应超时
RXDR	Receive FIFO data request	接收 FIFO 数据请求

S

SAD	Sum Of Absolute Difference	图象绝对误差和
------------	----------------------------	---------



SBE	Start-bit Error	起始位错误
SCL	Serial Clock	串行时钟
SCR	System Clock Reference	系统时钟参考
SD	Secure Digital	安全数字
SDA	Serial Data	串行数据
SDIO	secure digital Input/Output	安全数字输入输出接口
SDRAM	Synchronous Dynamic Random Access Memory	同步动态随机存取存储器
SFD	Start of Frame Delimiter	帧前导码
SI	Specific Information	特定信息
SIO	Sonic Input/Output	音频输入输出接口
SMI	Static Memory Interface	静态存储器接口
SOF	Start Of Frame	起始帧
SPI	Synchronous Peripheral Interface	同步外设接口
SRAM	Static Random Access Memory	静态随机存储器
SSP	Synchronous Serial Port	同步串口

T

TCM	Tightly-Coupled Memory	紧耦合存储器
TDE	Two Dimension Engine	2D 引擎
TVACT	Top Vertical Active Area	顶场垂直活动有效区域
TVBB	Top Vertical Back Blank	顶场垂直后消隐
TVFB	Top Vertical Front Blank	顶场垂直前消隐
TXDR	Transmit FIFO Data Request	发送 FIFO 数据请求

U

UART	Universal Asynchronous Receiver Transmitter	通用异步收发器
USB	Universal Serial Bus	通用串行总线

V

VACT	Vertical Active Area	垂直活动有效区域
VBB	Vertical Back Blank	垂直后消隐



VBI	Vertical Blanking Interval	垂直消隐间隔
VEDU	Video Encoding Decoding Unit	视频编解码器
VFB	Vertical Front Blank	垂直前消隐
VIU	Video Input Unit	视频输入单元
VOU	Video Output Unit	视频输出单元
VPW	Vertical Pulse Width	垂直脉冲宽度