Hi3520 PCI 级联应用

Application Notes

文档版本	01
发布日期	2009-12-23
BOM编码	N/A

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

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。

目 录

插图目录

表格目录

前 言

概述

本文分别从硬件环境准备、软件环境准备等方面介绍 Demo 板 PCI 级联操作的相关指导，同时介绍了 PCI 的基础知识、PCI 级联的业务实现和 PCI MPI 接口函数等，可为用户在应用 PCI 级联时提供参考。

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
Hi3520 H.264 编解码处理器	V100

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

约定

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。

命令行格式约定

格式	意义
粗体	命令行关键字（命令中保持不变、必须照输的部分）采用加粗字体表示。
斜体	命令行参数（命令中必须由实际值进行替代的部分）采用*斜体*表示。
[ ]	表示用“[ ]”括起来的部分在命令配置时是可选的。
{ x | y | ... }	表示从两个或多个选项中选取一个。
[x | y | ...]	表示从两个或多个选项中选取一个或者不选。
{ x | y | ... } *	表示从两个或多个选项中选取多个，最少选取一个，最多选取所有选项。
[x | y | ...] *	表示从两个或多个选项中选取多个或者不选。

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。

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
2009-12-23	01	正式发布。

修订日期	版本	修订说明
2009-12-21	00B20	第 1 章 Demo 板 PCI 级联操作指南 在 **1.1 硬件环境准备**中将“只有主片连接电源线，从片不需要再接电源”改为“只有主片连接电源线，从片不允许再接电源”。 在 **1.2 软件环境准备**中对 Hi3520 芯片的 2 个 ARM 核增加说明；对从片的镜像文件描述进行修改；将原 1.5 小节的内容提前导本小节进行描述；增加 SDK 发布包中 load_pci_host.sh 和 load_pci_device.sh 脚本的说明。 **1.3 从片启动**中步骤 1、步骤 2 和步骤 4 增加部分描述。 第 2 章 PCI 基础知识 **2.1 概述**中 PCI 的传输效率由 40～50％改为 30～50％。 **图 2-1** 的描述中增加相应主片和从片的描述。 **2.4 PCI MCC 消息应用**中关于“MCC 的消息缓存池使用 PCI 窗口中的可预取内存空间，且固定使用前 1M 的地址范围”的例子中，修改 MCC 模块使用的空间的起始地址。 第 4 章 PCI 开发参考 **4.2 PCIV MPI 参考**下的 HI_MPI_PCIV_Show 的描述中增加对显示的进一步说明，即传输；HI_MPI_PCIV_Hide 的描述中对隐蔽的进一步说明，即不传输。
2009-11-12	00B10	第 4 章中增加数据结构和错误码。
2009-10-31	00B03	增加第 2、3、4 章。
2009-09-30	00B02	增加 1.5 小节。
2009-09-02	00B01	第一次发布。

1 Demo 板 PCI 级联操作指南

1.1 硬件环境准备

PCI 级联调测，需要准备两块 Hi3520 Demo 板，一个 12V 电源，两根串口线，网线以及视频输入输出线若干。两块单板分别工作在 PCI 主模式（主片）和 PCI 从模式（从片），从片的金手指插入到主片的 PCI 插槽上，确保视频输入输出接口位于外侧即可。只有主片连接电源线，从片不允许再接电源。分别连接好两块单板的串口线、网线以及视频输入输出线。

Hi3520 Demo 板默认工作在 PCI 主模式，如果要工作在 PCI 从模式需要将启动模式更改为 DDR 启动模式（拨码开关 S1 的第 8 位拨到 R60 端），断开电阻 R373、R483 和 R484。调测前请先确认 PCI 从片上的以下内容已全部更改：

- 拨码开关 S1 位于单板正面，金手指的上方。
- 电阻 373 位于单板正面，紧贴 Hi3520 芯片，朝向金手指的一面。
- 电阻 483 位于单板反面，与电阻 373 正对。
- 电阻 484 位于单板正面，与电阻 373 并列相隔 8 个电阻（印刷的字符被一个小孔隔断，导致 R484 看着有点象 R404）。

详细的说明请参见《Hi3520 DMEB 原理图》和《Hi3520 Demo 单板用户指南》。

注意

- 从板的金手指插入到主片的 PCI 插槽后，需确保从片的视频输入输出接口位于外侧，否则就说明插反了。
- 只有主片需要连接电源线，从片由 PCI 供电，不能再接独立电源。

1.2 软件环境准备

Hi3520 芯片包括 2 个 ARM 核（主核为 ARM11，从核为 ARM9，以下统称为主 ARM、从 ARM），单颗芯片就需要两套 boot、内核以及文件系统。（可在发布包的 pub/images 目录下获得这些镜像文件。）

首先按照《Hi3520 SDK 安装以及升级使用说明》中的说明，烧写主片上的 boot、内核以及文件系统；主片成功启动后，配置网络及 NFS。

从片启动需要 6 个镜像文件，由于从片的用户程序运行在从片主 ARM 上，因此主 ARM 的文件系统需要重新制作成 cramfs.initrd 格式的镜像，其他五个文件可以与主片使用相同的镜像文件。所有镜像文件必须放在主片能够直接访问的一个目录内。6 个文件的名称，必须分别更改为引导程序默认的名称，如表 1-1 所示。

表1-1 启动文件清单

项目		文件名称	描述
主片	主 ARM	u-boot-hi3520v100_200M.bin	烧写到主片 Flash
		kernel-hi3520v100_full_release.img	烧写到主片 Flash
		rootfs-FULL_REL-Flash.jffs2	烧写到主片 Flash
	从 ARM	u-boot-hi3520v100_slave_200M.img	烧写到主片 Flash
		kernel-hi3520v100_full_release_slave.img	烧写到主片 Flash
		rootfs-SLV_FULL_REL.cramfs.initrd.img	烧写到主片 Flash
从片	主 ARM	u-boot-hi3520v100_200M.bin	与主片主 ARM 相同 **改名为 u-boot.bin**
		kernel-hi3520v100_full_release.img	与主片主 ARM 相同 **改名为 kernel.img**
		rootfs-FULL_REL-Inird.cramfs.initrd	需要重新制作 **改名为 cramfs.initrd.img**
	从 ARM	u-boot-hi3520v100_slave_200M.bin	与主片从 ARM 相同 **改名为 u-boot-slave.bin**

项目		文件名称	描述
		kernel-hi3520v100_full_release_slave.img	与主片从 ARM 相同 **改名为 kernel-slave.img**
		rootfs-SLV_FULL_REL.cramfs.initrd.img	与主片从 ARM 相同 **cramfs-slave.initrd.img**

SDK 发布包的 mpp_master\ko\目录下放有一个名为 mkimg_pci 的 Linux Shell 脚本，用户可以参考此脚本来提取和制作从片镜像文件。此脚本包含以下操作：

步骤 1 将启动从片需要的 6 个原始镜像文件从 pub/images 目录中拷贝到 pci_boot 目录下。

步骤 2 将从片主 ARM 的文件系统的压缩文件 rootfs-FULL_REL-Flash.tgz 从 pub/tarball 中拷贝到当前目录并解压。

步骤 3 将从片主 ARM 需要的 ko 和应用程序拷贝到文件系统的/root/目录下。

步骤 4 执行 mkimg.rootfs 制作 PCI 从片的文件系统镜像。

步骤 5 将从片主 ARM 文件系统镜像拷贝到 pci_boot 目录。

----结束

SDK 发布包的 mpp_master\ko\目录下有 load_pci_host.sh 和 load_pci_device.sh 以下 2 个 Linux Shell 脚本：

- load_pci_host.sh：用于在主片上引导从片启动、加载主片 PCI 内核模块和 MPP 相关模块。
- load_pci_device.sh：用于在从片上加载从片 PCI 内核模块和 MPP 相关模块。

说明
阅读“1.3 从片启动”～“1.5 PCIV 功能验证”时，可参考以上 2 个脚本。

“1.3 从片启动”～“1.5 PCIV 功能验证”主要描述如何在主片上引导从片启动、主从片分别加载相关内核模块以及如何运行相应测试程序和样例程序。阅读以下内容时，可以参看以上两个脚本。

1.3 从片启动

启动从片的步骤如下：

步骤 1 确认主片是否设置为 PCI 主模式（在主片 uboot 环境下检查 bootargs 下是否有配置 pcimod=host pciclksel=1）。

步骤 2 主片系统引导成功后，可以使用 cat /proc/bus/pci/devices 命令查看从片是否被正常的识别（proc 中会显示已经被正常识别的 PCI 从设备信息）。

步骤 3 主片插入 MMZ 模块。主片启动后，需要先插入 MMZ 模块，例如 modprobe mmz mmz=ddr,0,0xe4000000,32M。

步骤 4 主片插入 PCI 引导模块，例如 modprobe pci_multi_boot path=/mount/pci_boot/。其中 path 参数指定了从片启动所需镜像文件所在的目录（发布包的 pub/images 目录），也可以根据实际情况改动。

步骤 5 卸载 PCI 引导模块 rmmod pci_multi_boot。模块插入成功后，从片应该已经被引导起来，此时需要卸载引导模块，为后续工作清理环境。MMZ 模块也可以卸载掉（执行 rmmod mmz），需要的时候再重新配置。

步骤 6 从片启动后，建议也配置好网络及 NFS，方便调测。

----结束

1.4 数据传输功能验证

为了确保 PCI 多片之间的基本通讯正常，建议先通过以下的几个测试程序来验证 PCI 基本通讯功能，再启动完整的 PCIV 应用程序。

1.4.1 驱动加载顺序

主片上驱动模块加载顺序如下：

步骤 1 插入 PCI 硬件适配层模块 modprobe pci_hwhal_host。

步骤 2 插入 PCI 数据传输模块 insmod hi3520_pci_trans.ko。

----结束

从片上驱动模块加载顺序如下：

步骤 1 插入 PCI 硬件适配层模块 modprobe pci_hwhal_slave shm_phys_addr=0xC0000000 shm_size=0x1000000;这里带的参数是 PCI 共享内存的起始地址和长度，可以根据业务实际内存分布需要修改成其他值；shm_size 最大可配置成 16M，最小不应小于 1M。这块内存中，最前面 1M 的空间用于 MCC 消息通信，必须保留；后面 15M 用于数据传输，可以根据实际业务配置其大小。

步骤 2 插入 PCI 数据传输模块 insmod hi3520_pci_trans.ko。

----结束

1.4.2 PCI 数据传输功能验证

成功加载 PCI 驱动模块后，即可进行简单的功能验证。进入 source\drv\hisi-pci\pcit 目录，可以看到 pcit_test_device.c 和 pcit_test_host.c 两个测试程序，其中 pcit_test_host.c 在主片上运行，pcit_test_device.c 在从片上运行。在 pcit 上一级目录中的 Makefile 会编译这两个文件，并生成 device 和 host 两个可执行程序。将其复制到 NFS 可以访问的目录（如果从片没有网络 NFS，则需要事先将测试程序放入从片主 ARM 文件系统中）。

这两个程序可以完成下面三个功能验证：

- 通过 Window 窗共享内存将数据从主片传送到从片

  在主片运行./host 1；在从片运行./device 1。

主片会向共享内存中写入 0xa5，从片应该能够读取并打印出 0xa5；读取完毕后，程序自动退出。

- doorbell 中断互发测试

  在主片运行./host 2；在从片运行./device 2。

  主片循环向共享内存写入一个计数值，计数每增加一次就会向从片发送一次 doorbell 中断。从片接收到中断后，从共享内存中读取计数值并打印出来；再向主片回复一个 doorbell 中断。程序执行后，在主从片上都会打印出递增的计数值。如果要停止测试，需要 CTRL＋C 结束。

- 通过 DMA 方式将数据从从片传送到主片

  在主片运行./host 3；在从片运行./device 3。

  从片将递增的计数值先写入一块临时缓存中，再通过 PCI DMA 发送到主片的 DDR 中。主片接收到 PCI DMA 事件后，会从约定的缓存中读取计数值。在主从片上都会打印出递增的计数值。如果要停止测试，需要 CTRL＋C 结束。

  这个测试程序中使用的缓存都是约定好的，使用宏定义 TEST_DST_ADDR 和 TEST_SRC_ADDR 标识，运行测试前可能需要根据单板的内存分布，重新调整这两个值。

1.5 PCIV 功能验证

在硬件、软件环境准备好后，执行完整的 PCIV 功能验证（其中的脚本内容请直接阅读 SDK 发布包中相应脚本）的步骤如下：

步骤 1 主片进入 mpp_master/ko 目录，执行脚本 load_pci_host.sh，启动 PCI 从片，并加载 PCI 相关 ko 和 MPP 相关 ko。

步骤 2 从片进入/root/mpp_ko/目录，执行脚本 load_pci_device.sh，加载 PCI 相关 ko 和 MPP 相关 ko。

步骤 3 从片进入/root/目录，执行从片的 Sample 程序 sample_pciv_slave，此时终端上将打印 start check pci target id:0 ... ... 并被阻塞，等待主片启动。

步骤 4 主片进入 mpp_master/sample/pciv/目录，执行主片的 Sample 程序 sample_pciv_host，完成与从片的握手过程后，即启动 PCIV 视频预览业务。

步骤 5 如果需要停止 Sample 程序，可以在主片上输入两次回车键，主片会发消息给从片销毁相关业务，并退出本片程序；从片需要按 CTL+C 退出。

----结束

2 PCI 基础知识

2.1 概述

PCI 是外围设备互连（Peripheral Component Interconnect）的简称，作为一种通用的总线接口标准，在目前的计算机系统中得到了非常广泛的应用。PCI 总线的时钟频率一般使用 33MHz，在 32bit 系统中，理论极限速度可以达 132MB/s（32×33M/8）；但 PCI 无法一直维持在峰值传输的状态，一般只能保持在 30～50％的传输效率，即 40MB/s～60MB/s 的速度。如果需要更高的传输速度，可以使用 66MHz 的时钟频率，但随着频率的增加，对硬件设计的要求也会更高。

PCI 设备上有三种地址空间，对应三种 PCI 总线命令，具体如表 2-1 所示。

说明
CPU 可以访问 PCI 设备上的所有地址空间。

表2-1 PCI 地址空间和命令

地址空间	描述	命令	
I/O 空间	提供给设备驱动程序使用	I/O 操作命令	对设备对应的 I/O 地址空间进行访问，此类访问不可预取。
存储空间		Memory 操作命令	对设备的 Memory 空间进行访问，其中 Memory 操作命令又可分为 Prefechable（可预取）和 Non-prefechable（不可预取）两种类型。
配置空间	提供 Linux 内核中的 PCI 初始化代码使用	配置访问命令	对设备的配置空间进行读写访问，用来初始化设备，给设备分配资源。

内核在启动时负责对所有 PCI 设备进行初始化，配置所有的 PCI 设备，包括中断号以及 I/O 基址，并在文件/proc/bus/pci/devices 中列出所有找到的 PCI 设备，以及这些设备的参数和属性。

请查阅有关 PCI 规范的资料获取 PCI 协议详细说明。本文重点介绍关注业务应用中常用的知识。

2.2 PCI DMA 方式数据传输

Hi3520 PCI 模块内建 DMAC，可直接由 PCI 接口发起 DMA 操作，此时不需要 ARM 的干预，可获得更好的系统性能。

支持 PCI 从到主、主到从以及从到从的 DMA 读写传输，主要用于传输预览图像、解码图像以及码流等数据。预览图像和解码图像的传输由 PCIV 模块在内核态调用接口完成，而码流数据的传输则需要用户调用 PCIV 模块封装的接口来完成。

软件提供的 DMA 传输接口需要以下输入参数：

- 目标物理地址（即 PCI 地址）
- 源物理地址（即 AHB 地址）
- 传输长度

注意

不建议使用 DMA 读操作。

从到主的 DMA 写操作：源地址使用从片的 AHB 地址（即 DDR 地址），目标地址使用主片的 DDR 地址（即其 PCI 地址）。

主到从的 DMA 写操作：源地址使用主片的 AHB 地址（即 DDR 地址），目标地址使用从片的窗口 PF 地址空间对应的 PCI 地址。

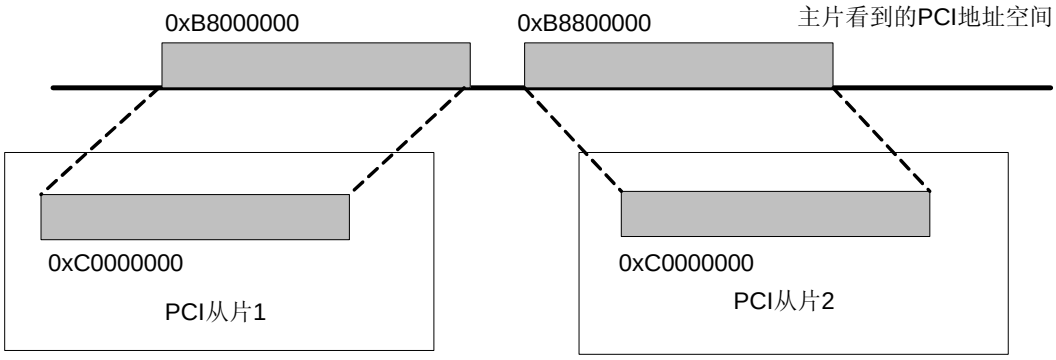
2.3 PCI 共享内存方式数据传输

作为 PCI Host 的 Hi3520 可以通过 AHB-PCI window 实现 ARM core 对 PCI 总线上的其它设备的访问。

AHB-PCI window 上存在三种地址空间，分别是非可预取内存空间（NP Memory）、可预取内存空间（PF Memory）和 IO 空间（IO）；

PCI 主设备由操作系统统一分配从设备配置空间中的 BAR3 和 BAR4 寄存器的基址，并通过配置空间访问写入；从设备则可以在 AHB 通过寄存器配置 NP Memory 和 PF Memory 的 AHB 侧基地址以及范围大小（最大 16M）。因此主设备上看到的各个从设备的窗口的 PCI 地址是与从设备本身的 AHB 侧基地址一一对应，即可使用 Window 中的可预取内存空间实现主从片间的内存共享，具体如图 2-1 所示。

图2-1 PCI 地址与 AHB 侧地址的映射关系



如图 2-1 所示，主片的 PCI 地址 0xB8000000 与从片 1 上的 AHB 地址 0xC0000000 可以认为存在映射关系，在主片上对 0xB8000000 区域的读写访问可以通过 PCI 总线反应到从片的 0xC0000000 区域；主片上的 PCI 基址 0xB8000000 是通过读取从片 1 的 BAR3 寄存器而得到（用户程序可以通过 PCIV 模块封装的接口获取），而从片上的 AHB 基址 0xC0000000 是由驱动程序写入到 PCIAHB_ADDR_PF 寄存器中的（用户可以在加载 pci_hwhal_slave.ko 模块时通过修改模块参数来更改此 AHB 基址）。

注意：PF Memory 在从片上的地址范围是可以不断移动的（即窗口的移动），但鉴于整个系统的稳定性以及 PCI 消息模块对 NP 基址的依赖性，不建议在从片启动后再移动窗口。

2.4 PCI MCC 消息应用

PCI MCC 消息模块基于 PCI 的 Window 窗口机制以及 PCI doorbell 中断，实现 PCI 主从设备间的消息通讯功能。

用户态接口包括：获取 PCI 本地以及远端的 ChipId 号，主从片间的相互检测机制（即通讯握手），消息端口的打开、关闭，消息的读和写、以及 Select 接口等。

从 MCC 模块获取到的 ChipId 号，在 PCI 主设备上为 0，在 PCI 从设备上则为 PCI slot 号加 1（由于 PCIV 模块内部的消息通讯也基于 MCC，因此 PCIV 中的 ChipId 与此一致）。

MCC 的消息缓存池使用 PCI 窗口中的可预取内存空间，且固定使用前 1M 的地址范围，例如加载 pci_hwhal_slave.ko 模块时，配置窗口范围为 0xC0000000 的 16M 地址范围，则从 0xC0000000 开始的 1M 空间分配给 MCC 模块使用，用户程序不应该再去使用它。

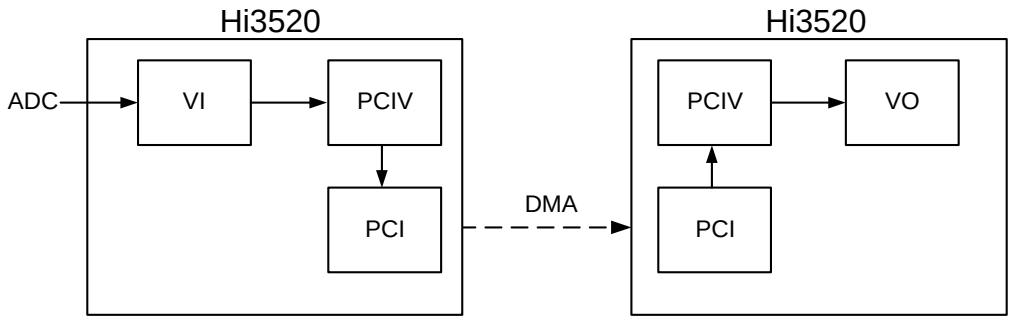
如果用户需要使用 PCI doorbell 中断（一般情况下不需要使用），那么需要注意 MCC 模块固定使用了 15 号 doorbell 中断，请不要与其冲突。

3 PCI 级联业务实现

3.1 视频预览

视频预览是将 Hi3520 的 VI 图像传送到 PCI 总线上的其他 Hi3520 的 VO 设备上显示。基本的数据流处理如图 3-1 所示。

图3-1 视频预览数据流处理流程

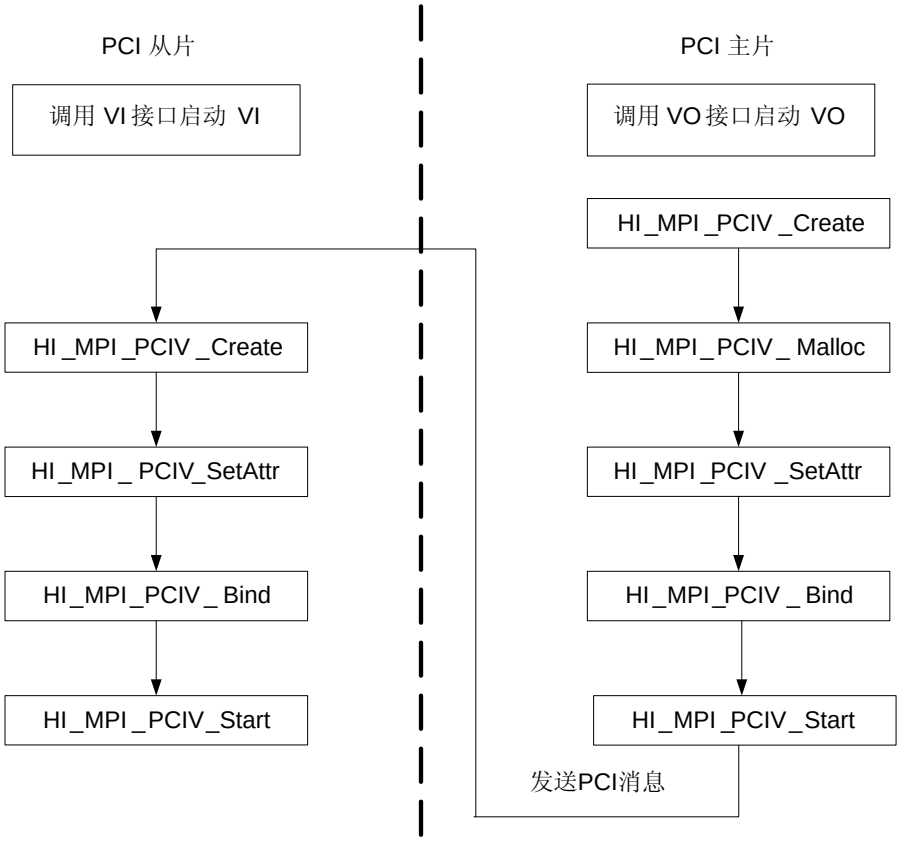


数据流的控制和传输由 MPP 系统在内核态完成，用户只需要调用 MPI 接口完成相应配置、使用 PCI 消息机制完成部分命令的传递。传输通路建立以后，正常图像传输则不需用户干预。具体接口调用流程图如图 3-2 所示（以从片到主片的预览业务为例）。

说明

- PCIV 相关接口的详细说明和注意事项请参见“4 PCIV 开发参考”。
- PCI 消息的相关接口则使用 mcc 模块提供的 iotcl 命令。

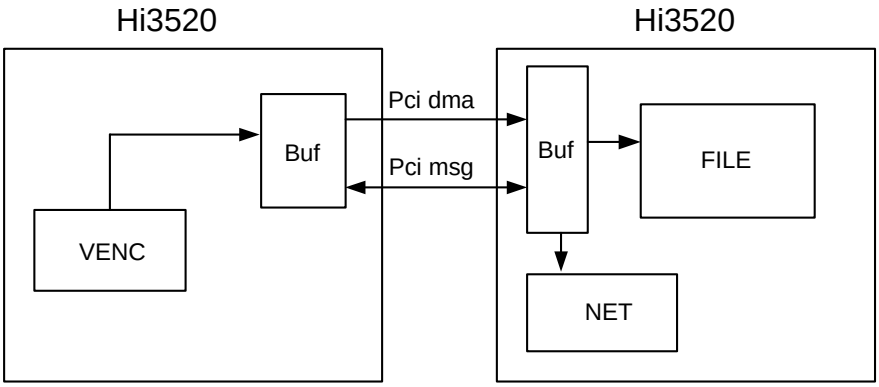
图3-2 视频预览时接口调用流程



3.2 码流传输

码流传输是将 Hi3520 的视频编码码流数据传送到 PCI 总线上的其他 Hi3520 设备。基本的数据流处理如图 3-3 所示。

图3-3 码流传输数据流处理流程



码流发送端首先从 VENC 通道中获取编码码流数据，将其拷贝至准备好的 stream buffer 中，然后通过 PCI 的 DMA 将码流数据发送到 PCI 对端的 stream buffer 中，对端再将码流取出通过网络发送或存文件，每次传输的读写位置信息可以通过 PCI 消息发送到对端以便进行发送和接收的同步控制。发送端和接收端的 stream buffer 需要用户自行实现，推荐采用不定长的循环 buffer，每次传输多帧数据。

说明

- PCI DMA 的传输使用 PCIV 模块提供的相应接口。
- PCI 消息则使用 MCC 模块提供的相应接口。

详细操作及流程可以参考 SDK 中的样例程序。

3.3 解码图像显示

解码图像是将 Hi3520 的 VDEC 解码后图像传送到 PCI 总线上的其他 Hi3520 的 VO 设备上显示。具体的数据流处理和接口调用与 PCI 预览流程类似，主要区别如下：

- 用户需要创建解码通道并向其发送码流进行解码；
- PCIV 相应传输通路建立后，PCIV 模块内部从 VDEC 通道取解码后图像数据，处理后通过 PCI DMA 发送到对端。

对端的数据接收及 VO 显示与预览流程一致。

3.4 内存配置

与 PCI 业务相关的内存配置时，需要注意以下事项：

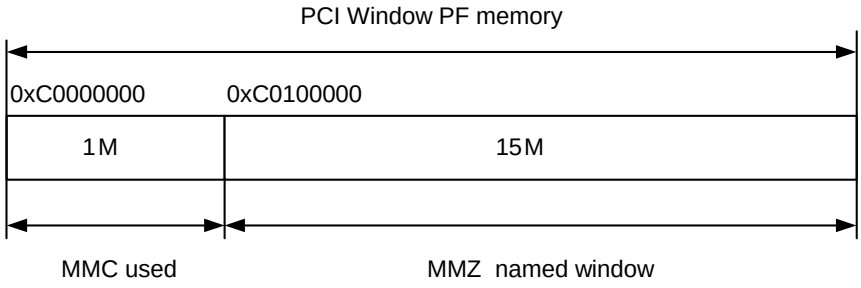
- PCI 从设备加载 pci_hwhal_slave.ko 模块时，配置窗口 PF 地址范围最大为 16M。需要注意，前 1M 固定分配给 MCC 模块。
- 图像或码流的传输过程中，使用的源地址和目标地址都是由 MMZ 管理，用户可通过 MMZ 的 VideBuffer 相关接口获取。其中图像的传输由 PCIV 模块内部从 VideBuff 中获取内存，而码流传输需要用户在用户态调用接口获取 VideBuffer 内存。

- 从到主的 DMA 数据传输（包括图像和码流）时，源地址和目标地址都可以是各自 DDR 上的任意可用地址。
- 主到从或者从到从的 DMA 数据传输时，目标地址必须使用 PCI 窗口映射的 PCI 地址。

  以主到从的码流传输为例，主片上的码流发送 buffer 可位于 DDR 上任意有效地址空间（可以使用 VideBuffer 相关接口申请内存）；而从片上的码流接收 buffer 则必须在 PCI 窗口的 PF Memory 空间范围内，获取其内存可以调用 PCIV 相应接口：首先调用 HI_MPI_PCIV_WinVbCreate 创建基于 window mmz 的缓存池，然后调用 HI_MPI_PCIV_Malloc 接口从其中获取缓存块。

  如图 3-4 所示，命名为 window 的 MMZ 区域位于整个 PF 窗口区域的后 15M 范围内，此 MMZ 区域是在从片的 ARM9 的初始化加载脚本中创建，基址为 0xC0100000，对应的整个从片 PCI 窗口的 PF 基址是 0xC0000000，如果用户需要修改 PCI 窗口 PF 基址，则必须同时修改 window MMZ 的基址。

图3-4 PCI 窗口 PF 区域和 MMZ 区域的示意



4 PCIV 开发参考

4.1 PCIV 概述

PCIV 模块主要提供 PCI 多片间图像数据传输等相关 MPI 接口。具体包括：

说明
PCIV 模块不提供 PCI 消息通讯、编码与 PCIV 通道的绑定等接口，这些接口由其他模块提供或者由用户实现。

- 预览图像的传输：图像发送端绑定 VI 通道，图像接收端绑定 VO 通道，发送端和接收端进行相应配置并启动后，即可将发送端的 VI 图像发送到接收端的 VO 通道进行显示。
- 解码图像的传输：图像发送端绑定 VDEC 通道，图像接收端绑定 VO 通道，发送端和接收端进行相应配置并启动后，即可将 VDEC 通道的解码后图像发送到接收端的 VO 通道进行显示。
- 码流数据的传输：提供启动 PCI DMA 传输的 MPI 接口，用户可以调用此接口将编码码流或者其他数据通过 PCI DMA 传输到 PCI 远端。

4.2 PCIV MPI 参考

HI_MPI_PCIV_Create

【描述】

创建 PCIV 通道。

【语法】

```
HI_S32 HI_MPI_PCIV_Create(PCIV_CHN pcivChn);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

重复创建通道会返回错误。

HI_MPI_PCIV_Destroy

【描述】

销毁 PCIV 通道。

【语法】

```
HI_S32 HI_MPI_PCIV_Destroy (PCIV_CHN pcivChn);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_NOT_PERM	操作不允许。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 如果未创建通道，直接返回成功。
- 销毁通道前必须先停止通道，否则返回错误。

HI_MPI_PCIV_SetAttr

【描述】

设置 PCIV 通道的属性。

【语法】

```
HI_S32 HI_MPI_PCIV_SetAttr(PCIV_CHN pcivChn, PCIV_ATTR_S *pPcivAttr);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入
pPcivAttr	PCIV 通道属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。

返回值	描述
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_UNEXIST	通道不存在。
HI_ERR_PCIV_NOT_PERM	操作不允许。
HI_ERR_PCIV_ILLEGAL_PARAM	参数错误。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 设置属性前必须先创建通道。
- PCIV 通道属性包含以下参数：
  - 目标图像属性（stPicAttr）：配置图像显示端的图像宽、高、像素格式等。
  - 图像 buffer 所属 PCI Chip 号（s32BufChip）：启动 PCI 图像传输，需要先在图像接收方分配好图像 buffer，s32BufChip 即为图像接收方的 chip 号。
  - 图像缓冲块大小（u32BlkSize）：每块图像缓冲块应该与一帧目标图像大小一致。
  - 图像缓冲块个数（u32Count）：取值范围为 2～4。
  - 图像缓冲块物理地址（u32PhyAddr）：图像接收方的 buffer 中每块缓冲块的物理地址，接收方通过调用接口 HI_MPI_PCIV_Malloc 获取，然后用户可以通过 PCI 消息将地址信息传递给图像发送方。
  - 对端 PCI 通道信息（stRemoteObj）：与当前本地 PCI 通道对应的远端 PCI chip 号及 PCIV 通道号。

HI_MPI_PCIV_GetAttr

【描述】

获取 PCIV 通道的属性。

【语法】

```
HI_S32 HI_MPI_PCIV_GetAttr(PCIV_CHN pcivChn, PCIV_ATTR_S *pPcivAttr);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入
pPcivAttr	PCIV 通道属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_NOT_PERM	操作不允许。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

必须先设置通道属性，然后才能获取。

HI_MPI_PCIV_Bind

【描述】

绑定 PCIV 通道。

支持与 VI、VDEC、VO 通道进行绑定：

- 如果传输预览图像，则在图像发送方绑定 VI 通道，在图像接收方绑定 VO 通道。
- 如果传输解码图像，则在图像发送方绑定 VDEC 通道，在图像接收方绑定 VO 通道。

【语法】

```
HI_S32 HI_MPI_PCIV_Bind(PCIV_CHN pcivChn, PCIV_BIND_OBJ_S *pBindObj);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入
pBindObj	PCIV 绑定关系结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_UNEXIST	通道不存在。
HI_ERR_PCIV_NOT_PERM	操作不允许。
HI_ERR_PCIV_ILLEGAL_PARAM	参数错误。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 必须先创建通道，否则返回错误。
- 必须先绑定再启动，即通道启动后，不允许调用此接口绑定。
- PCIV 通道可以与 VI、VDEC、VO 绑定，但同时只能与一种进行绑定。
- 一个 VI 通道可以与多个 PCIV 通道绑定。
- 一个 VDEC 通道可以与多个 PCIV 通道绑定。
- 一个 PCIV 通道可以与多个 VO 通道绑定，但反过来一个 VO 通道不能同时被多个 PCIV 通道绑定。

HI_MPI_PCIV_UnBind

【描述】

解绑定 PCIV 通道。

【语法】

```
HI_S32 HI_MPI_PCIV_UnBind(PCIV_CHN pcivChn, PCIV_BIND_OBJ_S *pBind);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入
pBind	PCIV 绑定关系结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_NULL_PTR	空指针错误。
HI_ERR_PCIV_NOT_PERM	操作不允许。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 必须先停止通道再解绑定，否则返回错误。
- 如果通道未绑定，则直接返回成功。

HI_MPI_PCIV_Start

【描述】

启动 PCIV 通道。

启动通道的具体行为取决于其绑定的通道类型：

- 绑定的是 VI 通道：先将 VI 图像加工处理为配置的目标图像，然后通过 PCI 发送到远端。
- 绑定的是 VDEC 通道：先从 VDEC 通道获取解码后图像，加工处理为配置的目标图像，然后通过 PCI 发送到远端。
- 绑定的是 VO 通道：从配置的 buffer 物理地址处取出帧图像，将其发送到绑定的 VO 通道进行显示。

【语法】

```
HI_S32 HI_MPI_PCIV_Start (PCIV_CHN pcivChn);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_UNEXIST	通道不存在。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 启动通道前必须先创建通道、设置属性并绑定，否则返回错误。
- 如果已经启动通道，则直接返回成功。

HI_MPI_PCIV_Stop

【描述】

停止 PCIV 通道。

【语法】

```
HI_S32 HI_MPI_PCIV_Stop(PCIV_CHN pcivChn);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

如果未启动通道，则直接返回成功。

HI_MPI_PCIV_EnumBindObj

【描述】

列举 PCIV 通道的绑定关系。

【语法】

```
HI_S32 HI_MPI_PCIV_EnumBindObj(PCIV_CHN pcivChn,
    PCIV_BIND_OBJ_S astBindObj[PCIV_MAX_BINDOBJ], HI_U32 *pu32Count);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入
astBindObj	绑定关系结构体数组指针。	输出
pu32Count	绑定的个数。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

如果没有任何绑定关系，绑定个数 pu32Count 返回 0。

HI_MPI_PCIV_DmaTask

【描述】

创建 PCI DMA 传输任务。

用于调用者自行启动一次或一系列的 PCI DMA 传输任务。

【语法】

```
HI_S32  HI_MPI_PCIV_DmaTask(PCIV_DMA_TASK_S *pTask);
```

【参数】

参数名称	描述	输入/输出
pTask	PCI 任务结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_BUSY	PCI 任务忙。
HI_ERR_PCIV_NOMEM	系统内存不足。
HI_ERR_PCIV_TIMEOUT	任务超时。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 此接口为阻塞接口，即等到 DMA 任务完成后接口才返回。
- 目前只支持 DMA 写操作，不建议使用 DMA 读操作。
- 源地址和目标地址必须是物理地址。

HI_MPI_PCIV_Malloc

【描述】

申请 PCI 相关内存。

用于申请 PCI 主从片间数据传输的相关内存，可以一次申请多块指定大小的内存块。

【语法】

```
HI_U32  HI_MPI_PCIV_Malloc(HI_U32 u32BlkSize, HI_U32 u32BlkCnt, HI_U32
u32PhyAddr[]);
```

【参数】

参数名称	描述	输入/输出
u32BlkSize	内存块大小。	输入
u32BlkCnt	内存块个数。	输入
u32PhyAddr	物理地址数组。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 在 PCI 主片上调用此接口时，直接从 MPP 的公共 VideoBuffer 中申请缓存块。需要保证 VideoBuffer 公共缓存池中有足够满足条件的缓存块，否则返回错误，VideoBuffer 公共缓存池的相关概念请参见《Hi3520 媒体处理软件开发参考》中系统控制章节相关内容。
- 在 PCI 从片上调用此接口时，必须先创建 PCIV 专用缓存池（使用 HI_MPI_PCIV_WinVbCreate 创建），然后从 PCIV 专用缓存池中申请缓存块。

HI_MPI_PCIV_Free

【描述】

释放 PCIV 相关内存。

【语法】

```
HI_S32  HI_MPI_PCIV_Free(HI_U32 u32BlkCnt, HI_U32 u32PhyAddr[]);
```

【参数】

参数名称	描述	输入/输出
u32BlkCnt	内存块个数。	输入
u32PhyAddr	物理地址数组。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

与接口 HI_MPI_PCIV_Malloc 配合使用，否则会造成释放失败或程序异常。

HI_MPI_PCIV_GetBaseWindow

【描述】

获取 PCI 窗口信息。

- 在主片上可以获取所有从片的 PCI 窗口信息。
- 在从片上则只能获取本片的可预取空间的 AHB 地址信息。

【语法】

```
HI_S32  HI_MPI_PCIV_GetBaseWindow(HI_S32 s32ChipId, PCIV_BASEWINDOW_S
*pBase);
```

【参数】

参数名称	描述	输入/输出
s32ChipId	PCI ID 号。	输入
pBase	PCI 窗口信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

- 在主片上调用此接口时，s32ChipId 为指定从片的 PCI ID 号（大于 0）。
- 在从片上调用此接口时，s32ChipId 必须置为 0。

HI_MPI_PCIV_WinVbCreate

【描述】

创建 PCIV 专用缓存池。

一般用于在 PCI 从片上创建一个专用的 VideoBuffer 缓存池（基于名称为“window”的 MMZ 区域）。此专用 VideoBuffer 缓存池与接口 HI_MPI_VB_SetConf 配置的缓存池概念类似；创建此缓存池的目的是用于实现 PCI 主到从或从到从的数据传输及共享。

【语法】

```
HI_S32 HI_MPI_PCIV_WinVbCreate(PCIV_WINVBCFG_S *pCfg);
```

【参数】

参数名称	描述	输入/输出
pCfg	PCIV 缓存池结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

无。

HI_MPI_PCIV_WinVbDestroy

【描述】

销毁 PCIV 专用缓存池。

【语法】

```
HI_S32 HI_MPI_PCIV_WinVbDestroy(HI_VOID);
```

【参数】

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

与 HI_MPI_PCIV_WinVbDestroy 接口配对使用。

HI_MPI_PCIV_Show

【描述】

显示（传输）PCIV 图像。

与 HI_MPI_PCIV_Hide 接口配合使用，当 PCIV 通道与 VO 绑定时，用于控制是否 DMA 传输图像 YUV 数据，默认情况下传输图像 YUV 数据。

【语法】

```
HI_S32 HI_MPI_PCIV_Show(PCIV_CHN pcivChn);
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

无。

HI_MPI_PCIV_Hide

【描述】

隐藏（不传输）PCI 图像。

与 HI_MPI_PCIV_Show 接口配合使用，当 PCIV 通道与 VO 绑定时，用于控制是否 DMA 传输图像 YUV 数据，默认情况下传输图像 YUV 数据。

【语法】

```
HI_S32 HI_MPI_PCIV_Hide(PCIV_CHN pcivChn);;
```

【参数】

参数名称	描述	输入/输出
pcivChn	PCIV 通道号。 取值范围：[0, PCIV_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【错误码】

接口返回值	含义
HI_SUCCESS	成功。
HI_ERR_PCIV_EXIST	通道已经存在。
HI_ERR_PCIV_NOMEM	系统内存不足。

【需求】

- 头文件：hi_comm_pciv.h、mpi_pciv.h
- 库文件：libmpi.a libpciv.a

【注意】

无。

4.3 数据类型

PCIV 数据类型定义如下：

- PCIV_VIDEVICE_S：定义 PCIV 中 VI 设备数据类型。
- PCIV_VODEVICE_S：定义 PCIV 中 VO 设备数据类型。
- PCIV_VDECDEVICE_S：定义 PCIV 中 VDEC 设备数据类型。
- PCIV_BIND_TYPE_E：定义 PCIV 绑定类型。
- PCIV_BIND_OBJ_S：定义 PCIV 绑定关系结构体。
- PCIV_REMOTE_OBJ_S：定义 PCIV 远端目标结构体。
- PCIV_PIC_ATTR_S：定义 PCIV 图像属性结构体。
- PCIV_ATTR_S：定义 PCIV 属性结构体。
- PCIV_WINVBCFG_S：定义 PCI Window VideoBuffer 结构体。
- PCIV_BASEWINDOW_S：定义 PCI Window 信息结构体。
- PCIV_DMA_BLOCK_S：定义 PCI DMA 任务块结构体。
- PCIV_DMA_TASK_S：定义 PCI DMA 任务结构体。

PCIV_VIDEVICE_S

【说明】

定义 PCIV 中 VI 设备数据类型。

【定义】

```
typedef struct hiPCIV_VIDEVICE_S
{
    VI_DEV viDev;
    VI_CHN viChn;
```

```
} PCIV_VIDEVICE_S;
```

【成员】

成员名称	描述
viDev	VI 设备号。 取值范围：[0, VIU_MAX_DEV_NUM)。
viChn	VI 通道号。 取值范围：[0, VIU_MAX_CHN_NUM)。

PCIV_VODEVICE_S

【说明】

定义 PCIV 中 VO 设备数据类型。

【定义】

```
typedef struct hiPCIV_VODEVICE_S
{
    VO_DEV voDev;
    VO_CHN voChn;
} PCIV_VODEVICE_S;
```

【成员】

成员名称	描述
voDev	VO 设备号。 取值范围：[0,VO_MAX_DEV_NUM)。
voChn	VO 通道号。 取值范围：[0, VO_MAX_CHN_NUM)。

PCIV_VDECDEVICE_S

【说明】

定义 PCIV 中 VDEC 设备数据类型。

【定义】

```
typedef struct hiPCIV_VDECDEVICE_S
{
    VDEC_CHN vdecChn;
} PCIV_VDECDEVICE_S;
```

【成员】

成员名称	描述
vdecChn	VDEC 通道号。 取值范围： [0,VDEC_MAX_CHN_NUM)。

PCIV_BIND_TYPE_E

【说明】

定义 PCIV 绑定类型。

【定义】

```
typedef enum hiPCIV_BIND_TYPE_E
{
    PCIV_BIND_VI    = 0,
    PCIV_BIND_VO    = 1,
    PCIV_BIND_VDEC  = 2,
    PCIV_BIND_BUTT
} PCIV_BIND_TYPE_E;
```

【成员】

成员名称	描述
PCIV_BIND_VI	绑定 VI 类型。
PCIV_BIND_VO	绑定 VO 类型。
PCIV_BIND_VDEC	绑定 VDEC 类型。

PCIV_BIND_OBJ_S

【说明】

定义 PCIV 绑定关系。

【定义】

```
typedef struct hiPCI_BIND_OBJ_S
{
    PCIV_BIND_TYPE_E enType;
    union
    {
        PCIV_VIDEVICE_S   viDevice;
```

```
        PCIV_VODEVICE_S   voDevice;
        PCIV_VDECDEVICE_S vdecDevice;
    } unAttachObj;
} PCIV_BIND_OBJ_S;
```

【成员】

成员名称	描述
enType	绑定类型。
unAttachObj	绑定关系。

PCIV_REMOTE_OBJ_S

【说明】

定义 PCIV 远端目标结构体。

【定义】

```
typedef struct hiPCIV_REMOTE_OBJ_S
{
    HI_S32   s32ChipId;
    PCIV_CHN pcivChn;
} PCIV_REMOTE_OBJ_S;
```

【成员】

成员名称	描述
s32ChipId	远端目标的 PCI 芯片序号。 取值范围：[0,PCIV_MAX_CHIPNUM)。
pcivChn	远端目标的 PCIV 通道号。 取值范围： [0,PCIV_MAX_CHN_NUM)。

PCIV_PIC_ATTR_S

【说明】

定义 PCIV 图像属性结构体。

【定义】

```
typedef struct hiPCIV_PIC_ATTR_S
{
    HI_U32   u32Width;
```

```
    HI_U32   u32Height;
    HI_U32   u32Stride[3];
    VIDEO_FIELD_E   u32Field;
    PIXEL_FORMAT_E  enPixelFormat;
} PCIV_PIC_ATTR_S;
```

【成员】

成员名称	描述
u32Width	图像宽度。
u32Height	图像高度。
u32Stride	图像跨度。
u32Field	图像帧场选择。
enPixelFormat	图像像素格式。

PCIV_ATTR_S

【说明】

定义 PCIV 属性结构体。

【定义】

```
typedef struct hiPCIV_ATTR_S
{
    PCIV_PIC_ATTR_S   stPicAttr;
    HI_S32           s32BufChip;
    HI_U32           u32BlkSize;
    HI_U32           u32Count;
    HI_U32           u32PhyAddr[PCIV_MAX_BUF_NUM];
    PCIV_REMOTE_OBJ_S stRemoteObj;
} PCIV_ATTR_S;
```

【成员】

成员名称	描述
stPicAttr	图像属性。
s32BufChip	图像缓存所在的芯片 PCI 序号。 取值范围：[0,PCIV_MAX_CHIPNUM)。
u32BlkSize	图像缓存块的大小。

成员名称	描述
u32Count	图像缓存块的个数。 取值范围：[1, 4]。
u32PhyAddr	图像缓存块的物理地址。
stRemoteObj	远端目标信息。

PCIV_WINVBCFG_S

【说明】

定义 PCI Window VideoBuffer 结构体。

【定义】

```
typedef struct hiPCIV_WINVBCFG_S
{
    HI_U32 u32PoolCount;
    HI_U32 u32BlkSize[PCIV_MAX_VBCOUNT];
    HI_U32 u32BlkCount[PCIV_MAX_VBCOUNT];
} PCIV_WINVBCFG_S;
```

【成员】

成员名称	描述
u32PoolCount	缓存池个数。
u32BlkSize	缓存块大小。
u32BlkCount	缓存块个数。

PCIV_BASEWINDOW_S

【说明】

定义 PCI Window 信息结构体。

【定义】

```
typedef struct hiPCIV_BASEWINDOW_S
{
    HI_S32 s32ChipId;
    HI_U32 u32NpWinBase;
    HI_U32 u32PfWinBase;
    HI_U32 u32CfgWinBase;
    HI_U32 u32PfAHBAddr;
```

```
} PCIV_BASEWINDOW_S;
```

【成员】

成员名称	描述
s32ChipId	PCI 芯片序号。 取值范围：[0,PCIV_MAX_CHIPNUM)。
u32NpWinBase	NP 基地址。
u32PfWinBase	PF 基地址。
u32CfgWinBase	CFG 基地址。
u32PfAHBAddr	PF AHB 侧基地址。

PCIV_DMA_BLOCK_S

【说明】

定义 PCI DMA 任务块结构体。

【定义】

```
typedef struct hiPCIV_DMA_BLOCK_S
{
    HI_U32  u32SrcAddr;
    HI_U32  u32DstAddr;
    HI_U32  u32BlkSize;
} PCIV_DMA_BLOCK_S;
```

【成员】

成员名称	描述
u32SrcAddr	源地址。
u32DstAddr	目标地址。
u32BlkSize	任务块大小。 取值范围：[0, 16M)。

PCIV_DMA_TASK_S

【说明】

定义 PCI DMA 任务结构体。

【定义】

```
typedef struct hiPCIV_DMA_TASK_S
```

```
{
    HI_U32  u32Count;
    HI_BOOL bRead;
    PCIV_DMA_BLOCK_S *pBlock;
} PCIV_DMA_TASK_S;
```

【成员】

成员名称	描述
u32Count	任务个数。
bRead	是否 DMA 读操作。
pBlock	DMA 任务块结构体指针。

4.4 错误码

错误代码	宏定义	描述
0xA01A8002	HI_ERR_PCIV_INVALID_CHNID	PCIV 通道号无效
0xA01A8003	HI_ERR_ PCIV_INVALID_PARA	PCIV 参数设置无效
0xA01A8004	HI_ERR_PCIV_EXIST	PCIV 通道已经存在
0xA01A8005	HI_ERR_PCIV_UNEXIST	PCIV 通道不存在
0xA01A8006	HI_ERR_PCIV_NULL_PTR	输入参数空指针错误
0xA01A8007	HI_ERR_PCIV_NOT_CONFIG	PCIV 通道属性未配置
0xA01A8008	HI_ERR_PCIV_NOT_SUPPORT	操作不支持
0xA01A8009	HI_ERR_PCIV_NOT_PERM	操作不允许
0xA01A800C	HI_ERR_PCIV_NOMEM	分配内存失败
0xA01A800D	HI_ERR_PCIV_NOBUF	分配 VideoBuffer 失败
0xA01A800E	HI_ERR_PCIV_BUF_EMPTY	PCIV 缓存为空
0xA01A800F	HI_ERR_PCIV_BUF_FULL	PCIV 缓存为满
0xA01A8010	HI_ERR_PCIV_SYS_NOTREADY	系统未初始化
0xA01A8012	HI_ERR_PCIV_BUSY	系统忙
0xA01A8040	HI_ERR_PCIV_TIMEOUT	任务超时