文章编号：1672-2892(2010)01-0063-04

基于驱动开发程序包的视频驱动开发

王 林，王正勇，卿粼波，徐 萍

(四川大学 电子信息学院，四川 成都 610064)

摘 要：驱动开发程序包(DDK)是 TI 公司提供的一套标准的 DSP/BIOS 驱动开发模型，基于此模型做视频驱动开发可简化驱动开发过程，增加代码的可移植性和兼容性。以 TMS320DM642 视频专用板为例，详述了视频驱动开发方法和开发过程，并以实例给出了调试结果。实践证明，基于 TMS320DM642 的 DDK 的视频驱动开发能为视频应用提供稳定的底层支持。

关键词：驱动开发程序包；视频驱动；TMS320DM642 器件；DSP/BIOS 驱动；类/微型驱动

中图分类号：TN911.72 文献标识码：A

Development of video driver based on Driver Developer's Kit

WANG Lin，WANG Zheng-yong，QING Lin-bo，XU Ping

(School of Electronics and Information Engineering，Sichuan University，Chengdu Sichuan 610064，China)

Abstract：Driver Developer's Kit(DDK) has provided a set of standard DSP/BIOS device driver model, based on which, the process of the video driver development can be simplified and the codes' compatibility and portability can be increased. Taking the TMS320DM642 video special board for example, the method of video driver's development and the process based on DDK were discussed in detail. The debugging result was given for the example. It has been proved that development of video driver based on DDK can work steadily for the video application.

Key words：Driver Developer's Kit ;video driver ;TMS320DM642 ;DSP/BIOS ;class driver/mini-driver

DDK 是 TI 公司提供的一套标准的 DSP/BIOS 驱动开发模型[1]，是为简化 TMS320 系列 DSP 及其相关 EVM (Evaluation Module)评估板上的外围设备驱动程序的开发而设计的，为各种外围器件提供完整的标准化驱动程序模型，使驱动程序可以很方便地移植到其他应用中，大大提高了驱动程序的开发效率。但实际应用中，不同视频的开发平台其外围芯片和电路设计略有不同，这使得 DDK 不能轻易移植到专用系统板上。鉴于此，本文以 TMS320DM642 为例，研究基于标准的 DSP/BIOS 驱动模型的开发过程。

1 硬件设计

TMS320DM642 作为视频专用板的核心芯片，是 TI 公司推出的一款针对多媒体处理领域应用的 DSP，它在 C64x 的基础上，增加了很多外围设备和接口，主频可高达 720 MHz。主要的片上外设有：3 个可配置的视频接口，VCXO 内插控制端口，I^2C 总线模块，增强型直接内存存取模块等。这些接口和外设使得 DM642 比其它系列的 DSP 处理器更适合处理视频码流[2]。此外，视频专用板的采集模块和显示模块分别采用了飞利浦的高性能视频解码芯片 SAA7113 和编码芯片 SAA7105。

视频专用板主要由 SAA7113 采集模块(完成复合视频广播信号(CVBS)的 A/D 转换)、DSP 视频处理模块、FPGA 转换模块(完成 YUV 信号转 RGB 信号)和 SAA7105 显示模块(完成视频信号的 D/A 转换)构成，系统框图如图 1 所示。

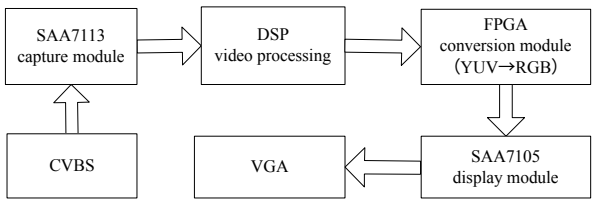


Fig.1 Diagram of system hardware
图 1 系统硬件框图

收稿日期：2009-08-10；修回日期：2009-10-13
基金项目：教育部科学技术研究重点资助项目(107094)

2 DDK 工作原理

DDK 为驱动开发定义了一个标准的 DSP/BIOS 驱动模型[3]，即类/微型驱动模型和一套 API 函数。

2.1 类/微型驱动模型

类/微型驱动模型是一种将设备驱动从机能上分为硬件无关层和硬件相关层的双层模型。这两层分别是类驱动和微型驱动。类驱动用来为应用程序提供接口，与微型驱动接口进行通信，一般提供对多线程 I/O 请求的串行化和同步，另外还负责设备实例的管理。微驱动程序与外部硬件设备相关，与类驱动程序的接口格式是固定的，但它对底层硬件的操作则是根据硬件平台的不同需要做相应的改动。只要微型驱动创建了规定的函数，应用程序就可以方便地通过类驱动调用。

2.2 GIO 类驱动

通用输入/输出(General Input/Output Manager，GIO)管理器类驱动提供了必需的同步读/写 API 接口以及扩展功能，并且在实现上采用了使代码量和数据量达到最小程度的设计。应用程序通过直接调用 GIO 的 API 函数和微型驱动程序进行交互，这些 GIO 的 API 函数可以看作是类驱动。图 2 给出了 GIO 类驱动和应用程序中其他部分之间的关系。

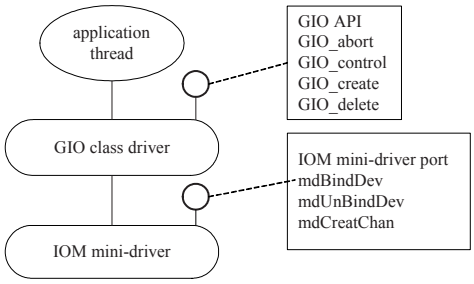


Fig.2 GIO driver port
图 2 GIO 驱动接口

2.3 视频类驱动

基于帧的视频模块(Frame Video，FVID)，是为支持视频应用，特别针对视频信息逐帧捕获和逐帧显示而进行的 GIO 类驱动 API 扩展。FVID 模块提供的设备驱动 API 函数不同于其他的设备驱动，因为它掌握数据缓冲区的所有权，应用程序按需求来分配缓冲区。

在视频驱动中，主要是通过调用 FVID 模块函数来完成类驱动代码的编写工作。这些函数是：FVID_create，分配并初始化 FVID 通道对象；FVID_control，发送一个控制命令到微型驱动；FVID_alloc，申请获得驱动程序缓存单元；FVID_exchange，交换应用程序和驱动程序的缓存空间；FVID_free，释放返回应用程序申请的缓存；FVID_delete，申请关闭一个 FVID channel 对象。

视频类驱动亦分为两层，上层为通用视频端口层，下层为指定编/解码芯片微型驱动层，如图 3 所示。这种结构的好处是：当使用不同的外围芯片时，只需要改动部分微型驱动，不需要将整个微驱动重新编写，使得驱动的复用性大大增强。

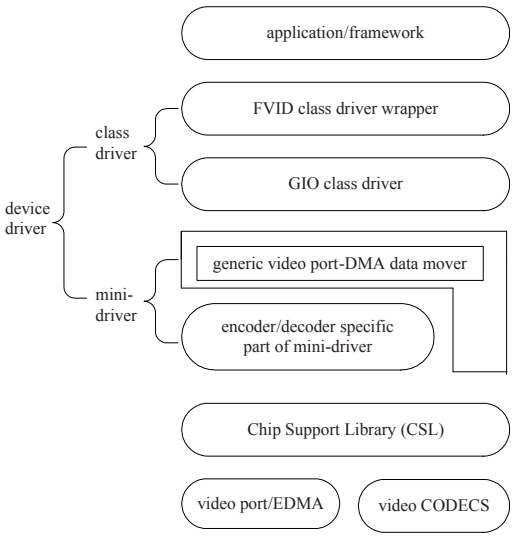


Fig.3 Video driver programming architecture
图 3 视频驱动程序模型

2.4 微型驱动

微型驱动主要通过一些函数来完成对外部设备的直接控制。标准的微型驱动 API 函数使得类驱动可以与微型驱动一起工作[4]。驱动开发者只要掌握一些简单的微型驱动 API 函数，就能开发出各种 DSP 外设的驱动。这些函数包括：mdBindDev()，绑定通道；mdCreatChan()/mdDeleteChan()，创建/删除通道；mdSubmitChan()，递交 I/O 请求；ISR，服务设备中断并完成 I/O 操作；mdControlChan()，控制设备。这些规定的函数将放入微型驱动的函数接口表(IOM_Fxns)中的相应位置，供应用程序通过适配模块或 GIO 类驱动调用。

3 DDK 的移植

3.1 DSP/BIOS 初始化

DSP/BIOS 的 System→Global Settings 对目标板进行初始化[5]。其中，EVMDM642_init 是用户调用的初始化函数，它对应的位置在 TI 提供的文件 evmdm642.c 中。此函数根据 EVM 开发板的板级设计进行某些模块的初始

化，如 EMIFA,I^2C 等。此时，用户根据自己目标板的资源进行修改，如，视频专用板上没有用到与 FPGA 相连的 FLASH 资源，则修改相关配置语句："EVMDM642_fpgaLoad(EVMDM642_FPGAFLASH_BASE);"。

3.2 类/微型驱动程序分析

DSP/BIOS 系统初始化时，mdBindDev 被调用，绑定外部设备到指定的端口，EDC function(需要用户编写)将被传到其参数 devParam 指针。编写不同于 EVM 板的编解码器的驱动程序主要是编写 EDC_Fxns 类型的函数表。以 SAA7113 为例，它需要编写 3 个函数供微型驱动程序调用：SAA7113_open(),SAA7113_close()和 SAA7113_ctrl()。其中，SAA7113_open()返回的是解码芯片的 I^2C 地址；SAA7113_ctrl()主要对该解码芯片的寄存器进行配置，这也是编程的重点；SAA7113_close()返回关闭器件后的布尔值。这些函数在视频驱动中的调用过程如下。

类驱动调用 FVID_create()创建视频通道时，可以看到这样的调用过程：FVID_create→GIO_create→mdCreateChan→SAA7113_open。此过程中，微型驱动还为 VP 采集口配置了增强型直接内存存取。接着类驱动调用 FVID_control()配置器件，通过控制命令 VPORT_CMD_EDC_BASE+ EDC_CONFIG 实现对解码芯片的控制，调用过程是：FVID_control→GIO_control→mdControlChan→SAA7113_control。

同理，这种底层驱动编写方法对其他的芯片也适用。

3.3 辅助调试函数

为了跟踪程序运行，方便调试，有必要将 gio.c,gio_crea.c,giosbmt.c,gio_cntl.c 等 API 函数添加到工程下的 Source 中。这样，在调试遇到问题时程序员能很快找到问题的所在，并且更清楚地知道数据的流向。

4　视频驱动的实现

4.1 视频驱动的设计

注册微型驱动。为了在 DSP/BIOS 应用程序中注册并使用一个微型驱动，用户必须配置应用程序使用该微型驱动，可以通过 DSP/BIOS 配置工具完成[6]。创建一个新的设备对象后，对其属性主要设置设备号、设备参数指针、设备实例的驱动函数表以及该驱动函数表的类型。

视频驱动流程图见图 4。FVID 函数会在设备表中查找已注册的微型驱动，并调用微型驱动函数完成对外部设备的操作。接下来需要创建 FVID 采集、显示通道，配置 SAA7105, SAA7113，分配相应的缓冲区，对得到的视频帧进行处理等操作。其中，交换应用程序和驱动程序的缓存空间，是为了保证视频数据能够源源不断地供应给应用程序使用。根据上述流程，给出部分代码：

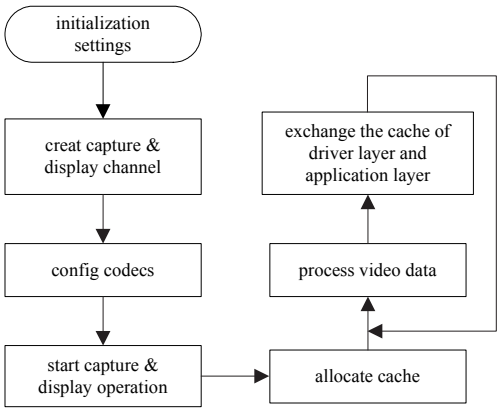
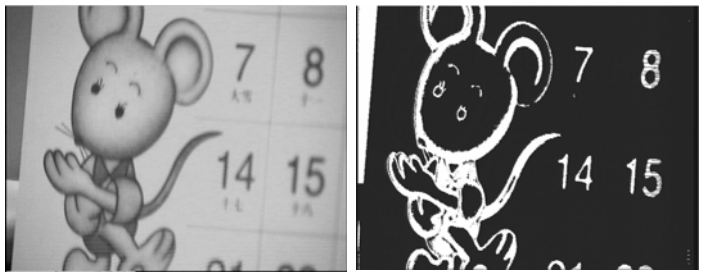


Fig.4 Flow chart of video driver
图 4 视频驱动流程图

```
capChan=FVID_create("/VP0CAPTURE/A/0",IOM_INPUT,&status,(Ptr)&EVMDM642_vCapParamsChan,NULL);
//创建采集通道
disChan=FVID_create("/VP2DISPLAY",IOM_OUTPUT,&status,(Ptr)&EVMDM642_vDisParamsChan,NULL);
//创建显示通道
FVID_control(disChan,VPORT_CMD_EDC_BASE+EDC_CONFIG,(Ptr)&EVMDM642_vDisParamsSAA7105);
配置 SAA7105
FVID_control(capChan,VPORT_CMD_EDC_BASE+EDC_CONFIG,(Ptr)&EVMDM642_vCapParams SAA7113);
//配置 SAA7113
FVID_control(disChan,VPORT_CMD_START, NULL); //开始显示操作
FVID_control(capChan,VPORT_CMD_START, NULL); //开始采集操作
FVID_alloc(disChan,&disFrameBuf); //分配显示缓冲区
FVID_alloc(capChan,&capFrameBuf); //分配采集缓冲区
while(1){
    ProcessVideo(&capFrameBuf,disFrameBuf,numPixel,…); //处理视频帧
    FVID_exchange(capChan,&capFrameBuf); //交换采集缓冲区
    FVID_exchange(disChan,&disFrameBuf); //交换显示缓冲区
};
```

4.2 系统调试结果

本文在专用视频处理系统上完成了对 DDK 的移植。图 5 所示的灰度图像，即为本系统基于 DDK 视频驱动开发程序所采集并显示出的图像。如上节所述，ProcessVideo 模块是视频帧处理模块，这种模块化设计有利于后期视频处理程序的开发。图 5 所示的二值化图像，为本系统通过 ProcessVideo 模块进行二值化处理后传送给 VGA 显示的图像。只需要给 ProcessVideo 模块传递视频采集的目的地址空间和视频显示的源地址，就可完成处理模块的嵌入。图像大小均为 720×576。从调试结果可以得出，基于 DDK 的视频驱动开发，为应用程序提供了稳定的底层支持，并有利于模块化开发。



(a) gray-scale image (b) binary image

Fig.5 Gray-scale image and binary image
图 5 灰度图像和二值化图像

5 结论

以 DDK 驱动模型做视频驱动开发，可提高驱动程序的可重用性和可移植性，简化视频驱动程序的开发，从而极大地提高驱动程序的开发效率。实践证明，基于 DDK 的视频驱动程序运行稳定，与应用程序实现了无缝连接，能为各种视频处理算法提供稳定的底层支持，是一种高效可靠的开发方法。

参考文献：

[1] Texas Instruments Incorporated. The DSP/BIOS Driver Developer's Guide[Z]. 2002.
[2] Texas Instruments Incorporated. TMS320DM64x DSP Video Port/VCXO Interpolated Control(VIC) Port reference Guide[Z]. 2003.
[3] 何伟,陈彬,张玲. DSP/BIOS 在基于 DM642 的视频图像处理中的应用[J]. 信息与电子工程, 2006,4(1):60-62.
[4] Texas Instruments Incorporated. The TM320DM642 Video Port Mini-Driver[Z]. 2003.
[5] 吴江,迟学芬,刘娜,等. 基于 DDK 的音视频编码器驱动的设计[J]. 吉林大学学报(信息科学版), 2007,25(3):246-250.
[6] 金朝辉. 基于 TMS320DM642 驱动模型的驱动程序开发[J]. 单片机与嵌入式系统应用, 2006(6):44-46.

作者简介：

王 林(1985-)，男，山西朔州人，在读硕士研究生，主要研究方向为图像的采集与处理及图像通信等.email:w_leon@163.com.

王正勇(1969-)，女，成都市人，副教授，硕士生导师，主要研究方向为信号与信息处理、图像处理与模式识别、智能系统与设计.

卿粼波(1982-)，男，四川省资阳市人，博士，讲师，主要研究方向为模式识别、图像处理和图像通信等.

徐 萍(1985-)，女，四川省眉山市人，在读硕士研究生，主要研究方向为图像采集与处理及图像通信等.