用类/微驱动模型开发DSP视频驱动程序

■ 西安理工大学 胡涛 刘颖娟

摘 要 集成外设的增多，软件应用复杂性进一步的增加，导致开发外设驱动程序所需的工作量大大增加。TI公司提出了类/微驱动模型的驱动程序结构。实践结果表明，采用类/微驱动模型进行驱动程序设计后，应用软件可以复用绝大部分相似设备的驱动程序，从而极大地提高了驱动程序的开发效率。

关键词 DM642 I/O设备驱动 类/微驱动模型 DSP/BIOS实时操作系统

近年来，DSP运算能力的不断增强，使其在电子设备方面得到了广泛的应用。DSP/BIOS是TI公司推出的一个实时操作系统，与TI的CCS(Code Composer Studio)集成在一起。用DSP/BIOS可以大大简化DSP应用程序的开发和调试，其中与外围设备的I/O接口是DSP应用开发中不可缺少的重要部分。TI公司为C64x系列DSP[1]的开发者提供了一种类/微驱动模型(class/mini driver model)。通过对外围设备设计驱动程序，为高层应用程序提供统一的接口来操作底层硬件。只要是遵循此驱动程序接口标准开发的高层应用程序，都可以在具有相同接口的不同硬件平台上运行，从而使DSP软件系统与硬件系统相分离，提高了软件的可重用性、可维护性和可移植性，缩短了总体驱动程序的开发周期。

1 DSP的外设驱动开发模型

TI公司为开发DSP的外设驱动[2]程序定义了标准的设备驱动模型，并将设备驱动分为类驱动和微驱动，即依赖于硬件层和不依赖于硬件层。两层之间使用通用接口进行数据通信，并提供了一系列的API接口，用户应用程序通过调用API来访问相应的外部设备。外设驱动开发模型的建立，提高了外设驱动程序的可重用性和模块化程度，简化了驱动程序的开发。外设开发模型如图1所示。

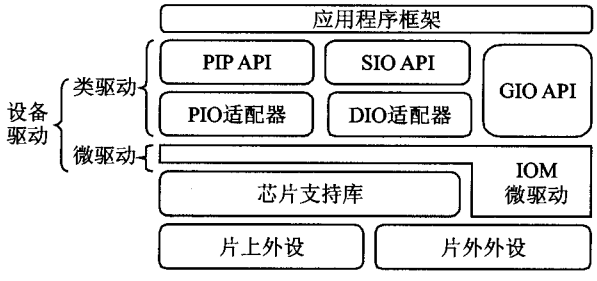

图1 DSP/BIOS类/微驱动模型

① 类驱动(class driver)。类驱动程序用来为应用程序提供接口。这部分程序与所使用的硬件设备无关，主要功能包括维护设备缓冲区，向上提供API接口供应用程序调用，向下提供适配层与微驱动层相连，实现API接口函数到微驱动层程序的映射。

② 微驱动(mini driver)。微驱动程序与外部硬件设备相关，所以设计微驱动程序是外设驱动开发的重点。微驱动程序与类驱动程序的接口格式是固定的，但微驱动程序对底层硬件的操作则须根据硬件平台的不同需要做相应的改动。微驱动通过接收类驱动层发出的调用命令来决定对底层硬件进行什么样的操作。

类驱动通过标准的微驱动接口调用微驱动控制硬件设备。到目前为止，TI共定义了3类驱动：① 流输入输出模块(SIO)，为每个DSP/BIOS线程提供一个独立的I/O机制，执行点到点的数据传送，支持动态创建，通过DIO适配模块与IOM连接；② 管道管理模块(PIP)，提供管理异步I/O的数据管道，每个管道对象都有一块同样大小的缓存，PIP模块通过缓存进行数据传输，通过PIO适配模块与IOM通信；③ 通用输入输出模块(GIO)，基于流输入/输出模式的同步I/O，适合大流量数据的传输，更适合文件系统。在用户应用程序中可直接调用GIO的API函数，GIO不需要额外的适配模块，可直接与IOM进行交互。GIO的这些优点使得通过GIO模块与外部设备进行数据流传输，操作简单、稳定，所以在视频采集的类驱动中采用了通用输入输出模块GIO。

GIO模块实现GIO的类驱动，用于提供一个模块化的读写应用程序接口到应用程序。通过封装这部分代码，应用程序可以通过GIO提供的应用程序接口间接调用各种IOM微驱动来减小整体的代码大小，如图2所示。

GIO模块提供下述功能：提供模块化的读写应用程

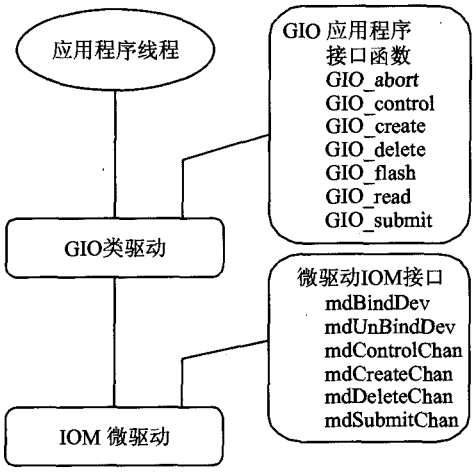

图 2 通用输入输出模块与微驱动接口

序；用 IOM 接口与指定设备微驱动实现程序通信；支持多个设备驱动；支持双向通道；允许用户配置模块化功能；支持应用程序增加新的应用领域（如视频）。其中，最后一项功能很重要。GIO_submit 函数对新增加的用户定制的应用程序接口（API）提供标准通道（如 video）。这种用户定制的类型包括用于文件系统的读写应用程序接口模块，例如 UART、DSP 视频帧等的应用。

传统的文件系统用读写应用程序接口来完成应用程序与文件之间的数据传输，需要由 GIO 类驱动和 IOM 微驱动来完成所需的双向通道。对 GIO 接口模块的扩展可以更加友好和高效地实现视频抓取和视频显示。这种扩展特别满足了视频设备存储空间（例如指定的帧缓存）的分配，而且通过简单的应用程序调用来更新视频帧缓存，提供视频驱动与应用程序之间最新的视频数据的更新。GIO 类驱动具有如下接口，在函数表中指定设备的操作模式：

```
typedef struct GIO_Obj {
  IOM_Fxns * fxns;  /* 微驱动函数表指针 */
  Uns mode;         /* IOM_INPUT, IOM_OUTPUT 或
                       IOM_INOUT 创建模式 */
  Uns timeout;      /* 模块调用所用的时间 */
  IOM_Packet syncPacket; /* 只用于同步操作的 IOM_Packet */
  QUE_Obj freeList; /* 用于异步输入/输出队列对象 */
  Ptr syncObj;      /* 同步对象的隐性指针 */
  Ptr mdChan;       /* 微驱动通道对象指针 */
} GIO_Obj, * GIO_Handle;
```

微驱动 IOM 通常包括如下函数：通道绑定函数（mdBindDev），通道创建函数（mdCreateChan），通道删除函数（mdDeleteChan），I/O 请求发送函数（mdSubitChan），通道解绑定函数（mdUnBindDev）和设备控制函数（mdControlChan）。类驱动通过调用这些底层函数完成相应外部设备与应用程序之间的数据传送通道的创建，以及外部设备和内存空间的分配，控制各个线程之间数据传送的同步等。这些规定好的底层函数将放入微驱动的函数接口表（IOM_Fxns）中的相应位置，供应用程序通过适配模块或直接由 GIO 类驱动调用。IOM 接口表的结构如下：

```
typedef struct IOM_Fxns {
    IOM_TmdBindDev      mdBindDev;
    IOM_TmdUnBindDev    mdUnBindDev;
    IOM_TmdControlChan  mdControlChan;
    IOM_TmdCreateChan   mdCreateChan;
    IOM_TmdDeleteChan   mdDeleteChan;
    IOM_TmdSubmitChan   mdSubmitChan;
} IOM_Fxns;
```

在调用 IOM 微驱动之前，必须要先在 DSP/BIOS[3] Config 中注册。在 Device Drivers 中右击选择插入一个设备驱动，命名为 VP1ACAPTURE，并进行各个属性（如函数表指针、函数表类型和设备 ID 号等）的配置，指明 IOM_Fxns 函数表地址和设备参数地址，如图 3 所示。

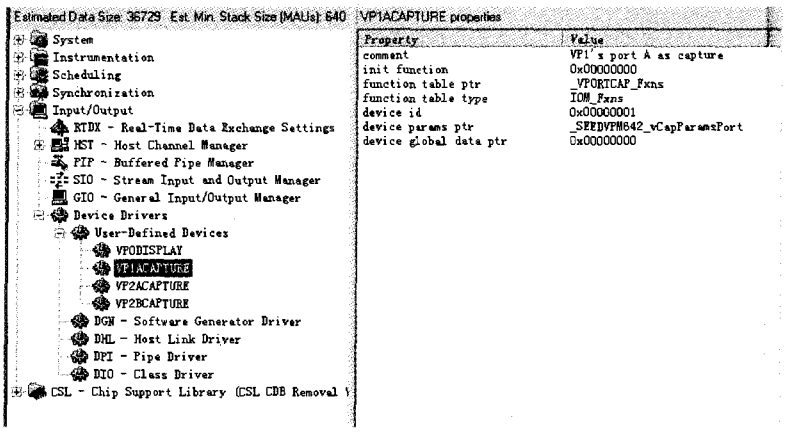

图 3 DSP/BIOS 配置

2 DM642 芯片视频驱动程序设计

视频驱动程序的设计主要包括微驱动、类驱动和中间接口 3 方面的设计。为了最大程度的提高视频驱动[4]代码的复用性和通用性，在视频采集驱动程序实例中，在 GIO 类驱动程序基础上进一步封装成 FVID 类。将微驱动细分为视频端口类和指定的编解码芯片微驱动，二者之间通过外部设备控制接口（EDC）实现对外围芯片的打开、控制和关闭等操作。这样一来，即使所使用的板卡上集成了不同的视频编解码芯片，也只需改变特定编解码芯片的微驱动。视频采集驱动程序的类/微驱动模型如图 4 所示。

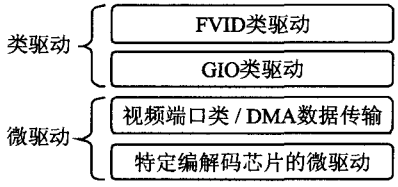

图 4 视频采集驱动程序的类/微驱动模型

例如，对 VPORT_PortParams 接口的说明属于对 VPORT 端口类的说明，而对 SAA7121 接口的说明则属于对特定编码芯片的说明。如果把 VPORT 类用于不同的芯片，则只需对 SAA7121 进行修改。对 EDC 的接口说明如下(指定要对外围设备进行打开、关闭等的操作)：

```
typedef struct EDC_Fxns {
    EDC_Handle ( * open)(String name, Arg optArg);
    Int ( * close)(Ptr devHandle);
    Int ( * ctrl)(Ptr devHandle, Uns cmd, Arg arg);
} EDC_Fxns;
```

下面举例说明对 FVID 类驱动的视频应用程序接口(video API)的操作。就应用程序而言，对设备驱动程序的操作可分为 3 个阶段：创建、处理和删除。创建是在应用程序与外部设备之间建立一个数据输入/输出的逻辑通道，对应于 FVID_create；处理是在应用程序与外部设备之间进行数据的传送，并对外设进行相应的控制，对应于 FVID_control 和 FVID_alloc；外设使用完毕后，相应地删除原先所建立的通道，对应于 FVID_delete。具体的调用函数如下：

```
#define FVID_BASE        IOM_USER
#define FVID_ALLOC       (FVID_BASE + 0)
#define FVID_FREE        (FVID_BASE + 1)
#define FVID_EXCHANGE (FVID_BASE + 2)
#define FVID_alloc(gioChan, bufp) \GIO_submit(gioChan, FVID_ALLOC, bufp, NULL, NULL)
#define FVID_control(gioChan, cmd, args) \GIO_control(gioChan, cmd, args)
#define FVID_create(name, mode, status, optArgs, attrs) \GIO_create(name, mode, status, optArgs, attrs)
#define FVID_delete(giochan) \GIO_delete(gioChan)
```

结 语

本文介绍了 TI 公司开发的类/微驱动模型和改进后 DM642 的视频类/微驱动模型。实践表明，DM642 的视频类/微驱动模型降低了系统中软硬件之间的耦合性，提高了驱动程序的可重用性和可移植性，简化了视频驱动程序的开发。

参考文献

[1] 李方慧，王飞. TMS320C6000 系列 DSPs 原理与应用[M]. 第 2 版. 北京：电子工业出版社，2003.

[2] Texas Instruments Incorporated. The DSP/BIOS Driver Developer's Guide. Literature Number: SPRU616, November 2002.

[3] 彭启琮，管庆，等. DSP 集成开发环境——CCS 及 DSP/BIOS 的原理与应用[M]. 北京：电子工业出版社，2004.

[4] Texas Instruments Incorporated. The TMS320DM642 Video Port Mini - Driver. Literature Number: SPRU918a, August 2003.

胡涛(教授)，主要研究方向为图像处理和模式识别；刘颖娟(硕士)，主要研究方向为基于 DSP 的人脸识别。

(收修改稿日期：2006-12-28)

11 得到广泛的应用。其中一个最关键的因素就是预测的可靠性。没有一种预测算法是 100%准确的，也没有一种算法可以应用于所有的程序；而对于实时类的应用(如音频、视频等)，预测失败的结果是不可接受的。因为实时类的应用都有一个 Deadline，错过 Deadline，就意味着程序的运行出了问题。比如音频或视频帧的播放时间错过以后，用户就能明显地感觉到音频或视频的不连贯，这会极大地影响用户的体验，从而也会影响用户对 DVFS 的信心。作者在进行 DVFS 的测试时，就碰到过这些问题。IEM 测试中采用的简单移动平均算法只对单一应用程序有效。但是i. MX31内置的移动指数平均算法 EMA 也不是万能的。对于 Pink Floyd 的某些音乐，它就不能平滑地播放(也许通过修改一些加权参数，可以播放)。

但是作者相信，随着预测算法的进步，DVFS 技术必将得到广泛的应用，因为它能够节省很多能量。而节能对许多便携式设备来说，常常是第一要求。

参考文献

[1] Freescale. i. MX31 Multimedia Application Processor Reference Manual. Rev 1. 2006 - 02.

[2] Freescale. Boris Bobrov & Michael Priel, 6/2005, i. MX31Power Management White Paper. Rev 0.

[3] Krisztian Flautner, et al. OSDI 2002, Vertigo: Automatic Performance - Setting for Linux. ARM.

[4] Krisztian Flautner, et al. DesignConn 2003, A Combined Hardware - Software Approach for Low - Power SoCs: Applying Adaptive Voltage Scaling and Intelligent Energy Management Software. ARM.

[5] Suji Velupillai, Ken Tough. Intelligent Energy Manager (IEM) Benchmarking on a Freescale's i. MX31 Multimedia Processor. Intrinsyc, 2006.

卢春鹏(工程师)，主要研究方向为嵌入式系统、移动网络、多媒体应用。

(收修改稿日期：2007-01-15)