Windows CE 显示驱动在 DM6446 上的设计实现①

张俊臣　刘宏立　江　艳　李志娟　(湖南大学 电气与信息工程学院 湖南 长沙 410082)

摘　要： DM6446 是一款有着丰富外设资源和强大计算能力的多媒体处理芯片，一般需要通过操作系统对其复杂资源进行有效管理。由于多数多媒体应用需要有图形界面，但现有平台操作系统 Linux 其内核和图形界面是分离的，需要另外移植，相比之下 WINCE 本身自带了较为优秀的图形界面，只需对其进行显示驱动的开发就能拥有一个优秀的图形界面。详细阐述了 WINCE 显示驱动原理和在 DM6446 上的设计与实现。系统启动后 WINCE 图形界面运行稳定，表明驱动程序实现良好。

关键词： DM6446; WINCE; 显示驱动; MDD 和 PDD

Research and Design of Windows CE Display Driver Based on DM6446

ZHANG Jun-Chen, LIU Hong-Li, JIANG Yan, LI Zhi-Juan

(College of Electrical and Information Engineering, Hunan University, Changsha 410082, China)

Abstract: As a Multi-media processing chip in possession of rich resource and powerful computing peripherals, DM6446 usually manages effectively the complex resource through an operating system. Since most multimedia applications require a graphical interface, the existing platform operating system is Linux whose kernel and graphical interface is separate, needing to transplant one.and as opposed to Linux, Windows CE itself comes with a GUI, just aquiring a GUI by the development of display driver simply. This paper describes the design and implementation of Windows CE display driver on DM6446.After the system's startup, graphical interface runs stablely, indicating that the driver is fine.

Keywords: DM6446; WINCE; display driver ; MDD and PDD

作为一款多媒体处理芯片，TI 公司推出的 DM6446 采用 ARM+DSP 的双内核架构，有着丰富的外设资源和强大的计算能力，因此一般通过操作系统对其复杂资源进行有效管理。DM6446 现有平台操作系统主要基于嵌入式 linux 系统，但同时也有支持其他主流嵌入式操作系统的能力。

由于多媒体应用常常需要有图形界面，而现有平台操作系统 Linux 其内核和图形界面是分离的，需要进行另外移植，相比之下 Windows CE 本身自带了较为优秀的图形界面，只需对其进行显示驱动的开发就能拥有一个优秀的图形界面。因此选择在 DM6446 进行其他系统的移植开发无疑能使该平台程序开发具有更多的选择余地和更高的性价比。

Windows CE 是 Microsoft 公司专门针对嵌入式产品领域开发的嵌入式操作系统，具有图形用户界面出色、多任务处理能力、可裁剪性和可移植性、应用软件支持丰富、实时性良好等特点。本文选用的 Windows CE 版本为 Windows CE.NET 5.0，以下简

① 收稿时间:2010-04-16;收到修改稿时间:2010-05-10

称为 WINCE。

1 DM6446芯片及其显示模块介绍

1.1 DM6446 芯片简介

DM6446 芯片，如下图 1 所示，由 ARM 子系统、DSP 子系统、VICP 协处理器、视频处理子系统和众多的芯片外设组成。其中 ARM 核用作整个系统的控制功能，DSP 子系统用于复杂的数据和图像处理功能，视频处理子系统用于和图像输入和输出。这些模块的联系通过中心资源交换通道(Switch Central Resources，SCR) 进行管理。

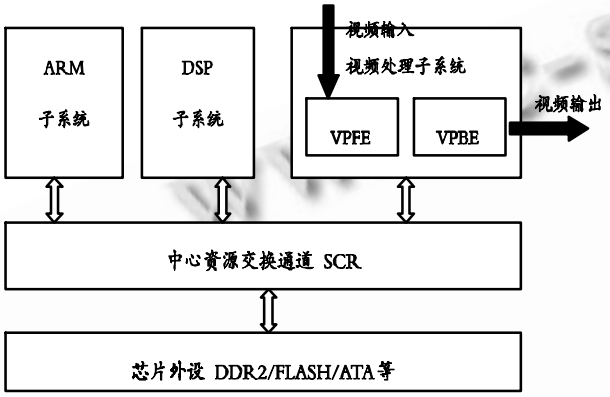

图 1 DM6446 芯片总体架构

1.2 芯片显示模块功能介绍

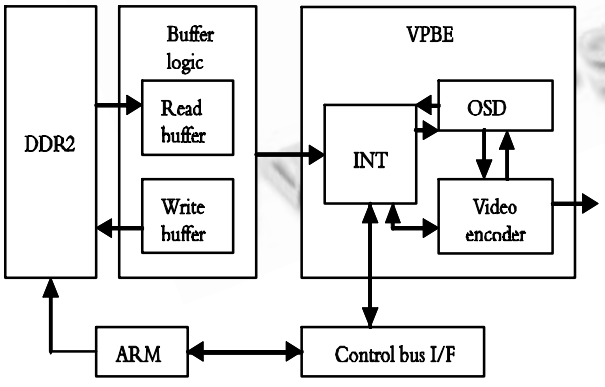

图 2 DM6446 显示模块总体架构

DM6446 显示模块又称为视频处理后端(VPBE，Video Processing Back End)，为芯片视频处理子系统的一部分。VPBE 总体结构如图 2 所示。由图 2 可以看出，VPBE 主要由 1 个 OSD (On Screen Display) 引擎和 1 个视频编码器(VENC，Video ENCoder)组成。OSD 引擎可以处理 2 个独立的视频窗口和两个独立的 OSD 窗口，VENC 视频编码器则能提供四路视频数据转换，工作频率高达 54MHz，兼容 NTSC/PAL 制式视频和 S-Video。

DM6446 视频编码器还能够向 RGB888 的显示设备提供 24 bit 的数字视频输出接口，支持 8/16 为的 BT.656 输出和垂直/水平同步分离的 CCIR.601。OSD 模块的视频信号在输出之前会经过合成然后送到 VENC 最终转变成 YCbCr 格式输出。视频数据是建立在外部存储器 DDR2 的，并直接送到显示设备作显示。从 DAC 出来就可以通过 RCA 端子接上 LCD 液晶电视。更详细的硬件说明可参考 TI 的官方数据手册 TMS320DM644x DMSoC Video Processing Back End (VPBE) User's Guide.pdf。

2 WINCE驱动架构分析

将 WINCE 移植到 DM6446 上面除了需要进行 OAL 层的代码和源码配置文件的编写以外，还需进行大量的设备驱动程序开发。

2.1 WINCE 驱动原理

设备驱动程序作为一个抽象物理设备或虚拟设备的功能程序，它管理设备的操作，并将设备的功能导出给应用程序和操作系统。因此用户程序访问这些硬件设备只需要通过调用驱动程序提供的接口函数。WINCE 的所有设备驱动程序都是以用户态下动态链接库(Dynamic Linkable Library，DLL)文件形式存在的。像所有的 Windows DLL 一样，DLL 是无法单独被加载和运行的[1]。如果要运行 DLL 中的代码，必须有一个 EXE 进程首先把该 DLL 加载到自己的地址空间内，然后才可以执行 DLL 中的代码。WINCE 下的驱动程序也必须被其他 EXE 加载。

2.2 WINCE 驱动分类

基于 WINCE 的驱动程序有两种模型：本地设备驱动程序(Native Device Driver)和流接口驱动(Streams Device Driver) 程序。本机设备驱动程序适用于集成到 WINCE 平台的设备，总是在 WINCE 的平台启动时被加载；流接口驱动程序也称为可安装的驱动程序，它们使用流接口驱动并借助于文件系统调用(如 Createfile，DeviceIoControl 等)从设备管理器

或应用程序获得命令。本文讨论的显示驱动属于本地设备驱动程序。

而从驱动实现方式来区分，无论流接口驱动还是本地驱动设备驱动，都可以采用两种实现方式：单体结构方式和分层结构方式，它们都向上提供 DDI (Device Driver Interface)调用，供其他模块或应用程序调用。无论采用哪种结构，驱动程序都必须与其控制设备的 DDI 相一致。DDI 是与 WINCE 系统的接口，流接口设备的 DDI 都是流接口函数。

3　显示驱动的实现

3.1 显示驱动的加载管理

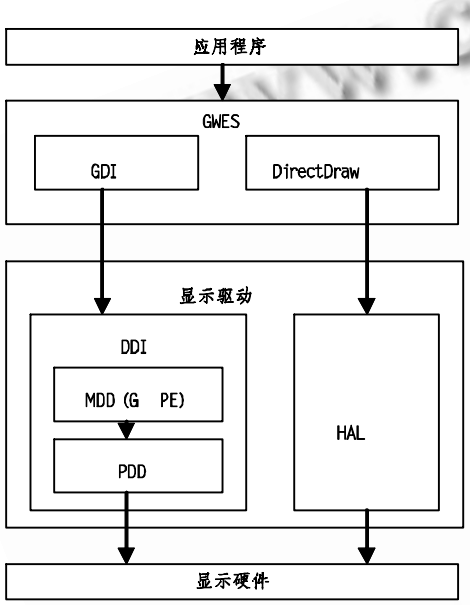

图 3　WINCE 显示驱动主体架构示意图

WINCE 下的驱动程序必须被其他 EXE 加载，显示驱动也不例外。WINCE 显示驱动在系统启动时由 GWES.exe 加载和管理，并驻留在 GWES 的进程地址空间内。如图 3 所示，GWES 子系统，由 GDI 和 DirectDraw 两部分组成，为运行在操作系统之上的应用程序提供图形功能的系统调用，例如 CreateDC，ReleaseDC 等等[2]。GWES 加载显示驱动的具体过程如下：GWES 启动时将去访问候选显示设备列表(该列表在注册表 HKEY_LOCAL_MACHINE\System\GDI\DisplayCandidates 下面)，看看是否有驱动程序已经在本机上实例化，如果有的话 GWES 会使用它找到的第一个已经实例化的驱动；如果驱动程序没有在本机上实例话或者找不到合适的驱动程序，接下来 GWES 尝试加载 Ddi.dll。默认情况下加载的是 Ddi.dll，但如果存在 KEY_LOCAL_MACHINE\System\GDI\Drivers\Display 项，GWES 会加载此注册表项所指定的显示驱动。

3.2 显示驱动主要组成部分

WINCE 的显示驱动程序如图 3 所示，由 DDI(Display Device Interface) 和 HAL(Hardware Abstraction Layer)两部分组成。HAL 主要为 DirectDraw 服务，只需要在驱动中向 GDI 导出 HALinit()即可，因此本文研究的重点是 DDI 部分，即通常的显示驱动部分。由于在显示中存在大量硬件无关操作，显示驱动通常采用分层结构，采用分层结构有助于降低代码复杂度提高代码效率，其中 MDD 层实现缺省的绘图功能，由微软提供的图形原语引擎模块(GPE，Graphics Primitive Engine)组成，如果要支持 Directdraw，则要使用 DDGPE 模块；而 PDD 层与硬件具体相关，则是显示驱动的主要内容，一般由 OEM 厂商或独立硬件商实现。

WINCE 上层程序通过一组(约 20 多个)显示驱动接口函数同显示驱动打交道，因此显示设备驱动程序必须实现这些显示驱动接口函数，GDI 通过调用这组函数初始化显示设备驱动程序和将图形输出到显示设备上[3]。由于采用分层结构，显示驱动由 MDD 层负责对上层的 GWES 模块提供函数接口，但是这些函数并不是直接提供出来的，实际上只是通过一个 DrvEnableDriver( )函数来完成的。作为 DDI 部分的一个导出函数，DrvEnableDriver 会在 GDI 初始化时被调用。

DrvEnableDriver 在 MDD 层中没有实现，所以需要在 PDD 层中定义，主要代码如下：

```
BOOL APIENTRY DrvEnableDriver
(ULONG engineVersion,ULONG cj,
DRVENABLEDATA *data,PENGCALLBACKS
```

```
engineCallbacks)
    {
        BOOL fOk = FALSE;
        if(gszBaseInstance[0] != 0)
        {
          fOk = GPEEnableDriver(engineVersion, cj, data, engineCallbacks);
        }
        return fOk;
    }
```

这里 GPEEnableDriver 是微软预先编写的一个 MDD 层函数。该函数位于源文件 ddi_if.cpp 里，因此我们只需简单调用就可以了。GPEEnableDriver 函数通过执行语句 memcpy(pded, &pDrvFn, cj) 将一个预先定义好的 DRVENABLEDATA 结构体变量 pDrvFn 的地址传给一个上层结构体指针 pded。而在结构体变量 pDrvFn 中预先已包含了 20 多个底层显示驱动函数指针，这样 GWES 就可以通过这些指针操纵底层显示硬件了。例如应用程序想创建一个到图形设备的连接时可以通过 GWES.exe 调用 CreateDC()，而该函数会调用 DrvEnablePDEV() 函数，当应用程序需要从显示设备上断开时则会调用 DeleteDC()，DeleteDC() 则会调用 DrvDisablePDEV()。DrvEnablePDEV() 和 DrvDisablePDEV()就属于这 20 多个被 GWES 调用的底层显示驱动函数。

以上这些底层显示驱动函数大部分跟硬件密切相关，因此需要进一步调用 PDD 层函数。由于不同的显示硬件特点都不尽相同，因此势必造成 PDD 层暴露给 MDD 层的接口函数各不相同，这样势必会增加代码的复杂性。为此微软设计了一个 GPE 类，一个 GPE 类实例代表一个显示设备硬件，其所有数据成员都对应于一个显示设备的属性数据，并设计了多个成员函数用以操纵这些数据成员。考虑到硬件的多样性，GPE 类的有些函数并为全部实现，或为空函数或者虚函数，需要其子类实现或者覆盖。因此不能直接定义 GPE 类型的变量，只能以先构造 GPE 类为父类的继承类，然后才能定义实例。

MDD 层的底层显示驱动函数通过实例化一个 GPE 继承类的实例就可以直接调用 PDD 层代码了，这一般是通过 SafeGetGPE 函数来实现的。SafeGetGPE 由微软设计实现，位于 MDD 层的 ddi_if.cpp，一般无须改动。在 SafeGetGPE 函数中调用了 GetGPE 函数，这个函数 MDD 层没有，需要我们在 PDD 层实现。GetGPE 函数可以简单实现如下：

```
GPE* GetGPE (void )
{  //gGPE 是 PDD 中定义好的一个全局变量
    if ( NULL == gGPE )
    {
        gGPE = new DM6446VPBE;
//DM6446VPBE 是一个 GPE 子类，与 DM6446 的显示硬件密切相关
    }
    return  gGPE;
}
```

这里代码利用了 C++的多态性和继承性。在 C++中父类或更上一级的类的指针可以引用继承类中相同的变量，并且对数据成员和成员函数的引用以继承类的实现或定义优先。这样在 MDD 中使用指针 gGPE 所指向的数据或函数时得到的都是类 DM6446VPBE 的成员变量和成员函数。由此可以看出 GetGPE 函数是显示驱动中联系 MDD 和 PDD 的桥梁，通过它 MDD 可以直接调用 PDD 的代码。

3.3 GPE 继承类的实现

通过上面的分析可以看出，WINCE 的显示驱动主要部分在于 PDD 层，而 PDD 层除了向 MDD 导出一些接口函数外如 DrvEnableDriver，其余主要是构建一个 GPE 或是 DDGPE 的子类(如果要实现 DirectDraw)。由于 DDGPE 的父类是 GPE，因此无论是 DDGPE 还是 GPE 的子类差别并不大。构建一个 GPE 的子类其实就是实现一个有具体数据和函数并且具体准确的反映了特定显示设备硬

件属性的 GPE 类的子类，并通过该子类去实例化一个对象。

一个 GPE 子类通常需要重载 GPE 类中的同名函数和实现 GPE 中的虚函数以及子类独有的一些函数如初始化构造函数[3]。子类构造函数主要是初始化硬件和子类成员变量，譬如视频处理时钟寄存器设置，OSD Window 的大小和坐标，VENC 的输出模式，以及子类的成员变量如显示宽度 m_nScreenWidth 和显示高度 m_nScreenHeight 等等。子类要 GPE 类中的函数包括 GPE 的空函数和虚函数，这些函数实际上就是 MDD 调用 PDD 层驱动中需要实现的函数，主要函数包括：SetMode()，用于设置一个显示设备能够支持的显示模式；GetPhysicalVideoMemory()，用于获取显示设备内存的系统基地址和内存大小；以及 AllocSurface() SetPointerShape() BltComplete() SetPalette()等。这些函数具体可以参考微软提供的驱动示例代码，它们位于 Public\ Common\OAK\Drivers\Display\ 目录下[1]。除了这些函数外 PDD 还需实现一个 MDD 层函数 DrvGetMask，但比较简单，只需要定义一个全局数组 gBitMasks，该数组内容是代表 RGB 的所占的位域，与具体的显示硬件有关。

3.4 驱动程序与应用程序的通信

不同于其他流式驱动可以由应用程序直接调用，显示驱动由操作系统调用，应用程序不能直接访问。具体来说,应用程序不是通过 CreateFile 等这些文件系统API接口来访问，而是通过GDI接口间接访问。对于GDI调用而言，对应的后台服务进程是GWES.exe，然后GWES.exe再进一步调用MDD和PDD函数，即WINCE底层显示驱动。例如如果要画一个矩形，则可以调用SetRect、GetDC和FillRect等函数在图形界面上面进行显示，而要在图形界面上输出一段文字只需调用 DrawText函数就可以了，至于显示驱动调用就可以交给GDI就可以了[4]。

4 结束语

本文阐述和分析了 DM6446 显示硬件原理和 Windows CE 驱动模型，剖析了显示驱动程序的工作原理和显示工作流程。本文的创新点在于完整的阐述了 WINCE 显示驱动程序在 DM6446 上的设计实现，而以往 WINCE 的显示驱动都是基于 LCD，因此本文对编写同类驱动程序的开发人员将有一定的参考价值。WINCE 启动运行后，图形界面运行稳定，并可支持 Windows CE 下的应用软件运行，表明驱动程序设计良好。

参考文献

1 何宗键. Windows CE 嵌入式技术. 北京: 北京航空航天大学出版社, 2006:194－203.
2 苏宁.基于 Windows CE.net 的 PDA 显示驱动程序设计[硕士学位论文].北京:北京交通大学, 2007.
3 李大为.Windows CE 工程实践完全解析. 北京: 中国电力出版社, 2008:158-160,211－232.
4 周立功. ARM & WinCE 实验与实践--基于 S3C2410. 北京: 北京航空航天大学出版社, 2007:77－84.