数字录像机界面的研究及实现

The study and implement of a GUI for digital video recorder

王保保　张 涛　(西安电子科技大学　710071)

摘要:BSP-15 是美国 Equator 公司新出的一款数字处理芯片,适用于单芯片、高带宽的解决方案。eCos 是一种开放源代码的实时操作系统。基于这样平台的数字录像机,是当前的一个研究方向。本文介绍图形用户界面的设计与实现。全部界面代码是用 C 语言写成。经过测试,这个界面运行稳定,效率高,进一步开发后可以做为产品使用。

关键词:图形用户界面 管理软件 嵌入式系统 程序设计 录像机 点阵字库

1 前言

数字录像机是一种获得视频输入(比如从摄像头),将其压缩存放在硬盘或其它存储媒体中的设备。用户可以方便的回放或者编辑这些视频。数字录像机有多种压缩编码方式。常用的编码有 mpeg2,mpeg4,mjpeg 等。Mpeg2 非常清晰,但是数据码率很大,很快就会填满存储空间。采用 mpeg4 和 mjpeg 等压缩编码可以在尽量保持图像质量的同时大大延长单次录像时间。基于安全方面的考虑,商场、银行、家庭等场所都能需要录像机。数字录像机在性能和功能上都优于传统录像机,所以最近几年发展得很快。

Equator BSP-15 处理器支持超长指令集,有高吞吐率的内存结构,以及数个不同功能的协处理器和大量的 I/O 接口。这款 DSP 特别适用于机顶盒,数字电视,视频会议系统,数字影像编辑设备等。

eCos 是一种可配置的实时操作系统。录像机要做的事情是很简单的,所以一个很小的操作系统就够了。编译 eCos 的时候,只把那些需要用到的特性编译进来,就得到一个很小的核。事实上,我们将编译得到的 eCos 的这个核和录像机的代码打成一个独立可启动的包。录像机加电的时候会去加载这个包,然后会在板子上建立一个操作系统,并运行录像机的代码。

2 BSP-15 的显示

2.1 BSP-15 的输出显示设备

BSP-15 支持两个输出接口:Vout 和 DRC(Display Refresh Controller)。Vout 接口接受 YUV422 packed(全部数据就是 1 个矩阵,Y,U,V 分量连续存放,如 y0u0v0y1y2u2v2y3…) 或者 planar(全部数据是 3 个矩阵 Y,U,V 分量各是一个矩阵)数据,并且驱动一个隔行扫描的 NTSC 或者 PAL 监视器。DRC 除了可以驱动 NTSC 或者 PAL 监视器外,还可以驱动模拟的 VGA 显示器和数字的 RGB 显示器。

2.2 DRC

DRC 输出设备支持模拟的 VGA,数字的 RGB 和 NTSC/PAL 视频输出。它支持 3 个通道:1 个 packed RGB 图像通道,1 个 YUV422/YUV420 planar 主视频通道和 1 个 packed RGB 或者 YUV422 次视频通道。主视频通道使用 video filter 来支持缩放。Video filter 是 BSP-15 的一个专门用于视频缩放的设备。

3 需求分析

首先,一套界面需要有方法来显示文字,擦除文字和刷新屏幕。因为我们并不是在 pc 上做显示,所以标准 C 语言里面的画图形和文本的函数都是不能用的。

其次,需要有方法来将文字和视频输出到显示设备上,比如模拟的 RGB 显示器或者电视等。

第三,录像机是根据很多设置比如录像的模式,比特率,视频质量,写到主硬盘还是次硬盘等来工作的,所以必须定义一套逻辑。还需要定义多个页面,将功能相近的设置放在一个页面上。

最后,我们要有方法来处理用户输入。

所有这些都是技术上可行的。

4 概要设计

4.1 输出设备

我们选择 DRC 做为输出设备。在录像机的界面里，我们在图像通道上显示菜单，文字，用户按下的键等等，在主视频通道上显示视频。

录像机的界面对于颜色的要求不高，所以我们在图像通道采用 RGB565 格式，即红色分量占 5 位，绿色分量占 6 位，蓝色分量占 5 位，加起来是 16 位，也就是 2 个字节。BSP－15 和 Intel x86 系列的 CPU 一样，都是低地址结尾的（little endian）. 所以红色分量的 5 位是在高地址，蓝色分量的 5 位在低地址，绿色分量的 6 位在中间的地址。我们可以用这样的模与（在 C 语言中操作符是 &）上一个 RGB565 的颜色值来得到各个分量的值：红色 0xF800，绿色 0x07E0，蓝色 0x001F。对颜色要求高的应用可以考虑采用 RGB888 格式。

DRC 允许我们设置在图像通道中使用一个还是多个缓冲。如果设置成一个，那么这个缓冲区就在应用程序和 DRC driver 之间共享。应用程序往里面写，同时 DRC driver 会去读，然后就显示出来。这里录像机的界面就设置成使用一个缓冲。这样的好处是有一块地址固定的内存区域对应到屏幕上的全部像素，可以计算出每个像素的内存地址。更改这块内存区域的数据就可以更改屏幕显示。

主视频通道采用 YUV420 格式。录像机将来自摄像头的数据压缩后存放在硬盘上。播放的时候解压缩为 YUV420 格式，并送到 DRC 的主视频通道显示。

最后，在设置 DRC 的时候，要设置使用 Alpha 流。图像通道是在主视频通道之上的，这也叫 On－screen display. 如果不使用 Alpha 混合，那么图像通道将完全遮挡住主视频通道，将看不到视频。DRC 允许我们去设置图像通道每个像素的 Alpha 值。每个像素的 Alpha 值占用 4 位，即半个字节：0x0 为不透明，0xF 为全透明。在录像机的界面中，也是采用打点的方式来显示文字的。所以，我们设置那些文字所在像素的 Alpha 值为 0x0，设置其它像素的 Alpha 值为 0xF，就可以把文字显示在视频上。

4.2 页面

根据录像机所需要的设置，我们将录像机的界面分成这些页：startupPage，passwordInputPage，mainPage，recordSchedulePage，hardDriveSetupPage，sensorSetupPage，otherSetupPage，passwordChangePage，timeSetPage，protocolSetPage，hardDriveFormatPage，workingPage，timeSearchPage。将这些页都定义在一个枚举类型中。每一页完成特定的功能。比如，数字录像机有 3 种模式：普通，计划和警报。在普通模式下，用户按下"开始"键，它就开始工作。在计划模式下，录像机将会根据用户的设置，在某个时间段自动开始和停止录像。在警报模式下，如果它检测到镜头前有物体在移动，就自动开始录像。recordSchedulePage 就是用来设置在 1 天的 24 个小时，录像机分别处于哪个模式；startupPage 是在录像机启动时，将检测主硬盘和次硬盘的结果显示出来；passwordInputPage 是用来输入管理员密码，密码正确后进入录像机设置。

4.3 定义字符串数组

需要定义一个包含字符串和它显示的位置的结构，然后将每一页所需要用到的字符串都定义好，放在这个结构的数组中。每一页有两个数组，英文的一个，中文的一个，分别用在英文和中文的界面里。这样，要显示哪一页的时候，先读取设置，再根据设置，去数组里读取字符串来显示。比如，mainPage 中有一项是 video quality，可能的值是 high，low，normal。当我们要显示 mainPage 时，先去读取设置看 video quality 当前的设置是什么，比如是 normal，那么就去数组里读字符串和它的位置，并在这个位置上把它画出来。

定义字符串的结构和数组的另一个目的是为了定义焦点。在一个页面上可能有很多的字符串，但不是所有的字符串都是焦点。所谓焦点就是可以接受用户输入的地方。比如"video quality"这个字符串就不是焦点，而它的设置值 high，low，normal 所在的位置是焦点，用户可以按键来选择到底设置为 high 还是 low 还是 normal。一个页面里面可能有很多焦点，所以在显示的时候需要一个指示来告诉用户，当前焦点在哪里。还要为每个焦点定义好，当接受到某个按键时（比如上、下、左、右、加、减等），应该怎样处理。

4.4 逻辑

录像机根据很多设置来工作。这些设置存储在 flash 中，录像机启动时会读到内存中。用户更改设置后会保存到 flash 里，下次启动仍然有效。我们需要定

义一套逻辑来存储这些设置。如下：

```
#define FLAG1_CAMERA1_ON 0x00000001
#define FLAG1_CAMERA2_ON 0x00000002
#define FLAG1_CAMERA3_ON 0x00000004
#define FLAG1_CAMERA4_ON 0x00000008
#define FLAG1_MODE_EACH 0x00000100
#define FLAG1_MODE_QUAD 0x00000200
#define FLAG1_LANGUAGE_ENGLISH 0x00000400
#define FLAG1_LANGUAGE_CHINESE 0x00000800
#define FLAG1_VIDEO_QUALITY_HIGH 0x00001000
# define FLAG1 _ VIDEO _ QUALITY _ NORMAL 0x00002000
#define FLAG1_VIDEO_QUALITY_LOW 0x00004000
...
```

这样，我们通过一些与或操作，就可以知道某个设置。

4.5 字符显示

类似很多 C 语言编写的中文系统，我们必须制作点阵字库。通过写代码解析 Windows 自带的标准字体，就可以制作出字库来。首先确定字体的大小，然后对这个大小的 TrueType 字体进行扫描，把得到的点阵信息存在数组里，就可以了。比如制作宽 14，高 27 的点阵字库，其中，'画'字可以这样存储：将一个相应大小的'画'字放在 14 * 27 的网格中进行采样，如果某网格被'画'覆盖了，就记为 1，否则记为 0。0 和 1 的信息在计算机里存储只需要 1 位。扫描网格的顺序是从左到右，从上到下。这样，每行 14 个网格，需要 14 位。总共 27 行，需要 14 ×27 位。为了方便读写，每行可以按字节对齐。14 位用两个字节，多余的两位填 0。一个 14 * 27 的'画'字的数据是这样的：{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x3f, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x30, 0x00, 0x00, 0xff, 0xf8, 0x00, 0x00, 0xc6, 0x33, 0x00, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xff, 0xf3, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xc6, 0x33, 0x80, 0x1c, 0xff, 0xf3, 0x80, 0x1c, 0xc0, 0x33, 0x80, 0x1c, 0x00, 0x03, 0x80, 0x1f, 0xff, 0xff, 0x80, 0x18, 0x00, 0x03, 0x80, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}。数据做好以后，还需要做一个索引，将每个字符和它的数据对应起来。

字库做好了以后实际上是常量的数组，就放在代码里。再写一个函数 setPixel(int x,int y,int color)，用于将(x,y)处的像素值设为颜色 color。在这个函数基础上写另外一个函数，用于显示一个字符。显示一个字符的过程是：根据字符去上面提到的索引里查找，找到字符的数据；计算每个点的位置，调用 setPixel 将数据为 1 的点设置为前景色，将数据为 0 的点设置为背景色，或者不做处理。

需要注意的是，录像机的界面只需要全部数字，全部英文字母，部分常用符号和部分中文。所以只把要用到的字符做成字库就可以了，这样可以节省一些存储空间。

4.6 字符串显示

```
static void drawText( struct stringData text);
static void drawTextEx( int x,int y,char * text);
static void drawTextBlink( int x,int y,char * text);
static void clearText( struct stringData text);
static void clearTextEx( int x,int y,char * text);
```

这里是一些用来显示字符串的函数。drawText( ) 和 drawTextEx( ) 的功能都是将一个字符串画在指定的位置。它们的区别是接受不同的参数，stringData 是一个包含字符串和它的位置(top,left) 的结构。

clearText( ) 和 clearTextEx( ) 的功能都是将擦除一个字符串，区别也是接受不同的参数。擦除也就是用背景色填充。比如，当前 video quality 设置为 high，那么屏幕上显示的是"high"。现在用户按了一下键，要改为"normal"，那么需要先擦掉"high"，再写上"normal"，不然"high"和"normal"会叠在一起显示。

drawTextBlink( ) 函数的功能是画一个闪烁的字符串。有时候需要提示用户一些信息，可以用这个函数。

5 结论

Equator BSP－15 提供的设备驱动程序使我们可以很方便的将数据输出到显示设备上。支持 Alpha

（下转第 85 页）

(上接第 94 页)

blending(透明混合)使得在显示图形界面的同时显示下面的 video 层成为了可能。这个图形界面代码的结构比较简单,清晰,容易添加代码以支持新的功能。界面和精简过的 eCos 的核编译在一起以后,在开发板上运行良好,相信经过完善以后就可以应用到产品中去。

参考文献

1 杨磊、李峰、付龙等,电视监控实用技术[M],北京 机械工业出版社,2002。

2 蒸春棣,嵌入式系统的构建[M],北京 清华大学出版社,2004。

3 蒋句平,嵌入式可配置实时操作系统 eCos 开发与应用[M],北京 机械工业出版社,2004。

4 陈翌、田捷、王金刚,嵌入式软件开发技术[M],北京 国防工业出版社,2003。

5 刘富强,数字视频监控系统开发及应用[M],北京 机械工业出版社,2003。

6 雷雨权,多媒体电视监控与报警系统[M],北京 国防工业出版社,2003。

7 (美)Wayne Wolf(孙玉芳、梁彬、罗保国等译),嵌入式计算系统设计原理[M],北京 机械工业出版社,2002。

8 (美)Arnold Berger(吕骏译),嵌入式系统设计[M],北京 电子工业出版社,2002。

9 沈绪榜、何立民,2001 嵌入式系统及单片机国际学术交流会论文集[M],北京航空航天出版社,2001。