

μ C/GUI T6963C LCD 控制器驱动移植及矢量汉字显示

严龙军¹, 熊文卓²

(1. 中国科学院长春光学精密机械与物理研究所, 吉林 长春 130033;

2. 中国科学院研究生院, 北京 100039)

摘要: 介绍了 Micrium 公司开发的一个为嵌入式应用软件设计的通用图形软件库 μ C/GUI, 提出了在 μ C/GUI 中 T6963C LCD 控制器驱动移植及矢量汉字显示的实现方法。该方法为 μ C/GUI 的其他 LCD 控制器驱动移植及字库实现提供了参考, 同时为利用 μ C/GUI 实现更为复杂的图形用户界面打下了基础。

关键词: 图形用户界面; 矢量汉字; 液晶控制器

中图分类号: TP311.54

文献标识码: A

μ C/GUI T6963C LCD controller driver transplantation and vector characters display

YAN Long Jun¹, XIONG Wen Zhuo²

(1. Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China;

2. Graduate School of Chinese Academy of Sciences, Beijing 100039, China)

Abstract: In this paper, a software design for embedded applications of general-purpose graphics software library μ C/GUI is introduced, which is designed by Micrium corporation, and an implementation method about T6963C LCD controller driver transplantation and vector characters display in μ C/GUI is put forward. This method provides a reference implementation for other LCD controllers driver and font library transplantation in μ C/GUI. At the same time, this paper has laid a foundation for implementing more complex graphical user interface by using μ C/GUI.

Key words: graphical user interface; vector characters; LCD controller

在众多嵌入式系统中, 如手机、PDA 和数字机顶盒(STB)等, 系统使用者往往要求具有菜单、窗口和按钮等图形元素的人机交互界面 UI (User Interface), 而系统设计和实现者又迫切需要系统 UI 模块开发的支撑平台。该支撑平台以应用开发组件 (Application Development Component) 的方式来有效提高嵌入式应用的开发效率, 这种组件被称为嵌入式图形用户界面 (GUI)。

当前比较流行的 GUI 主要有 MicroWindows、MiniGUI、QT/Embedded、OpenGUI 等, 普遍采用客户/服务器结构、多线程概念, 主要用于嵌入式 Linux 系统中。本文介绍的 μ C/GUI 是 Micrium 公司开发的一个为嵌入式应用软件设计的通用图形软件库。它为任何一个使用图形 LCD 的应用提供一个有效的不依赖于处理器和 LCD 控制器的图形用户接口。它能工作于单任务或多任务的系统环

境下。 μ C/GUI 适用于使用任何 LCD 控制器和 CPU 的任何尺寸的物理和虚拟显示。同时, μ C/GUI 适用于所有的 CPU, 因为它是由 100% 的 ANSI 的 C 语言编写的。而且, μ C/GUI 适合于大多数使用黑白/彩色 LCD 的应用程序。它有一个很好的颜色管理器, 允许它处理灰阶。 μ C/GUI 还提供一个可扩展的 2D 图形库和一个视窗管理器, 用很少的 RAM 就能支持显示窗口。更重要的是 μ C/GUI 开放源代码, 以便了解更多 GUI 实现的具体细节。

1 μ C/GUI 的主要特点

(1) 一般特点: 适合于任何 8/16/32/64 位 CPU, 只需要有一个 ANSI 兼容的编译器; 适合于任何控制器支持 (如果有合适的驱动程序) 的任何单色/灰度级/彩色 LCD; 显示屏大小可配置; 使用配置宏定义各种接口; 大小和速度都优化过的函数; 清晰的结构; 支持虚拟显

示, 虚拟显示能够比实际显示表现更大尺寸的内容。

(2)在图形库方面: 支持不同色深的位图; 有效的位图转换器; 绝对没有浮点运算; 快速线/点绘制(不使用浮点指令); 快速的圆和多边形绘制。

(3)在字体方面: 支持多种不同的字体如 4×6 、 6×8 、 6×9 、 8×8 、 8×9 、 8×16 、 24×32 等, 支持像素高为 8、10、13、16 等比例字体; 可以定义新字体, 而且很容易被程序使用; 只有实际被应用程序用到的字体才会真正被链接到目标文件, 这样可以节省 ROM 的使用; 字体在横向(X轴)和纵向(Y轴)都可以自由伸缩。

(4)在字符串/值输出函数方面: 具有能够显示二进制、十进制、十六进制和任何字体的值的函数。

(5)在窗口管理器 WM 方面: 包括可裁减的完全的窗口支持, 改写超出一个窗口的区域是不可行的; 支持回调函数; WM 使用极少的内存, 例如每个窗口大约 20 B。

2 T6963C LCD 控制器驱动移植

作为一个通用的图形函数软件库, $\mu\text{C}/\text{GUI}$ 能够适用于各种 LCD 控制器和 LCD, 但是, 针对具体的目标系统, 由于 LCD 及其控制器的访问方式不同, 以及显示缓存的访问方式不同, 还是要做具体的移植工作。对于内存映射的 LCD 控制器, 只需要在 LCDConf.h 文件中设置 LCD 的访问函数; 对于端口/缓存模式的 LCD 控制器, 必须在 LCD_X.C 文件中定义访问 LCD 的接口函数。 $\mu\text{C}/\text{GUI}$ 的移植主要是设置子文件夹 config 中的头文件, 其中包括 LCD 宏(定义 LCD 显示屏的属性)和 LCD 控制器宏的设置(定义如何访问 LCD 控制器)。

LCD 控制器有两种基本类型的总线接口: 完全和简单的总线接口。大多数小一些的显示屏(通常最大为 240×128 或 320×240) 的 LCD 控制器使用简单总线接口与 CPU 连接。对于简单总线, 只有一个地址位(通常是 A0)连接到 LCD 控制器。这样的控制器比较慢, 硬件设计者可以将其连接到 I/O 脚而不是地址总线。图 1 为带有简单总线的 LCD 控制器的方框图。

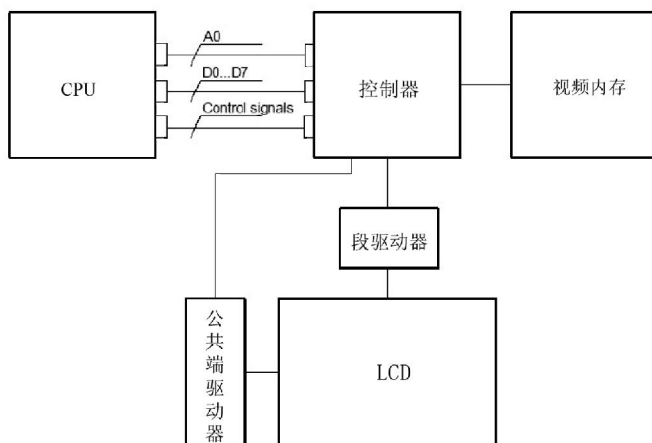


图 1 简单总线 LCD 控制器框图

用于简单总线接口控制器的宏:

(1)LCD_READ_A0 在 A0(C/D) 引脚为低电平时, 从 LCD 控制器读取 1B;

(2)LCD_READ_A1 在 A0(C/D) 引脚为高电平时, 从 LCD 控制器读取 1B;

(3)LCD_WRITE_A0 在 A0(C/D) 引脚为低电平时, 向 LCD 控制器写 1B;

(4)LCD_WRITE_A1 在 A0(C/D) 引脚为高电平时, 向 LCD 控制器写 1B。

液晶显示驱动控制器 T6963C, 是日本东芝公司的产品, 目前被广泛应用于内置控制器型液晶显示模块。T6963C 的最大特点是具有独特的硬件初始值设置功能, 显示驱动所需的参数如占空比系数、驱动传输的字节数/行及字符的字体选择等均有引脚电平设置, 这样 T6963C 的初始化在上电时就已经基本设置完成, 软件操作的主要精力就可全部用于显示画面的设计。

T6963C 属于具有简单总线接口的 LCD 驱动控制器。以 Philips 公司的 LPC2104 ARM 芯片为 CPU 控制 T6963C 为例来说明具体的移植过程。LPC2104 的 P0.0 ~ P0.7 与 T6963C 的 D0 ~ D7 相连, P0.8 与 WR 相连, P0.9 与 RD 相连, P0.10 与 C/D 相连。根据 T6963C 的时序图(如图 2 所示), 可以很方便地修改 LCD 底层驱动函数。

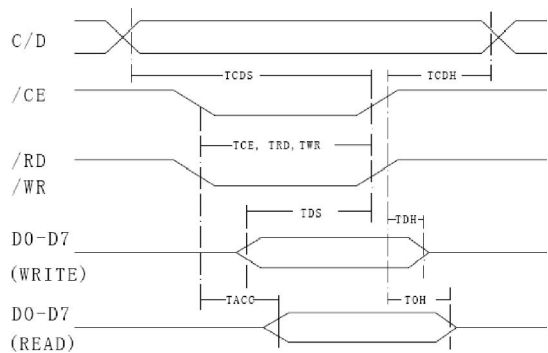


图 2 T6963C 时序图

```
#define wr 1<<8
#define rd 1<<9
#define cd 1<<10
/* 读取 T6963 状态字 */
uint8 LCD_READ_A1(void)
{
    uint8 temp;
    IODIR=0x000000700;
    // 设置 P0.8 ~ P0.10 为输出, P0.0 ~ P0.7 为输入
    IOSET=cd; // 把 C/D 引脚置位
    IOCLR=rd; // 把 RD 引脚置低电平
```

```

temp=IOPIN;           // 读数据总线上的数
IOSET=rd;             // 把 RD 引脚置位
return(temp);
}
/* 向 T6963C 写一个字节 (控制指令) */
void LCD_WRITE_A1(int Byte)
{
    IODIR=0x000007ff;    // 设置 P0.0 ~ P0.10 为输出
    IOSET=cd;            // 把 C/D 引脚置位
    IOCLR=0xff;          // 清除数据总线上数据
    IOCLR=wr;            // 把 WR 引脚置低电平
    IOSET=Byte;          // 把要发送的控制指令放数据总线上
    IOSET=wr;            // 把 WR 引脚置位
}
/* 向 T6963C 写一个字节 (数据) */
void LCD_WRITE_A0(int Byte)
{
    IODIR=0x000007ff;    // 设置 P0.0 ~ P0.10 为输出
    IOCLR=cd;            // 把 C/D 引脚置低电平
    IOCLR=0xff;          // 清除数据总线上数据
    IOCLR=wr;            // 把 WR 引脚置低电平
    IOSET=Byte;          // 把要发送的数据放数据总线上
    IOSET=wr;            // 把 WR 引脚置位
}

```

图 3 为 T6963C 指令写入的流程图, 对该流程图的理解是通过 CPU 向 T6963C 发送指令的关键。当计算机写指令或一次读/写数据时, S0 和 S1 要同时有效, 即“准备好”状态。对 T6963C 的软件操作每一次之前都要要进行判“忙”, 只有在不“忙”的状态下 CPU 对 T6963C 的操作才有效。

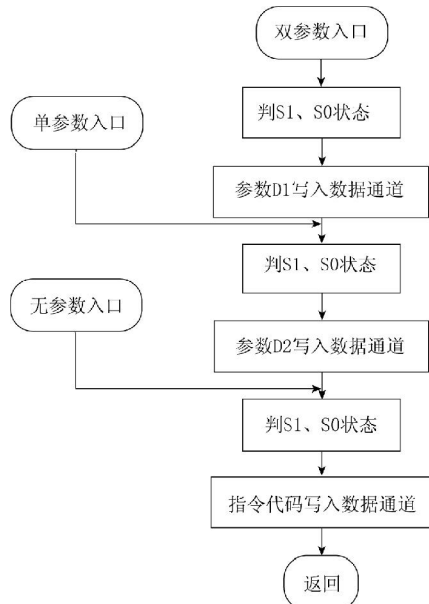


图 3 T6963C 指令写入流程图

```

/* 判定 S0 和 S1 的状态, “忙”即等待 */
#define LCD_WAIT() while ((LCD_READ_A1()&3)!=3)
/* 写命令宏 */
#define LCD_WRITECMD(cmd)
{ LCD_WAIT(); LCD_WRITECMD0(cmd); }
/* 写数据宏 */
#define LCD_WRITEDATA(Data)
{ LCD_WAIT(); LCD_WRITEDATA0(Data); }
/* T6963C 初始化 */
void LCD_T6963_Init(void)
{
    /* 设置文本显示区首地址 */
    LCD_WRITEDATA(0x00);           // 参数 1
    LCD_WRITEDATA(0x00);           // 参数 2
    LCD_WRITECMD(0x40);             // 指令代码
    /* 设置文本显示区宽度 */
    LCD_WRITEDATA(0x14); // 一行显示所占的字节数
    LCD_WRITEDATA(0x00);
    LCD_WRITECMD(0x41);
    /* 设置图形显示区首地址 */
    LCD_WRITEDATA(0x40);           // 低字节
    LCD_WRITEDATA(0x01);           // 高字节
    LCD_WRITECMD(0x42);
    /* 设置图形显示区宽度 */
    LCD_WRITEDATA(0x14); // 一行显示所占的字节数
    LCD_WRITEDATA(0x00);
    LCD_WRITECMD(0x43);
    /* 显示开关设置开文本和图形显示 */
    LCD_WRITECMD(0x9c);
    /* 显示方式设置逻辑或合成 */
    LCD_WRITECMD(0x80);
    /* 设置光标显示形状 */
    LCD_WRITECMD(0xa1);
}

```

在 LCDConf.h 中, 可以根据实际情况更改设置, 如:

```

#define LCD_XSIZE (160)           // LCD x 坐标方向的分辨率
#define LCD_YSIZE (128)           // LCD y 坐标方向的分辨率
#define LCD_CONTROLLER (6963)     // LCD 控制器型号
#define LCD_BITSPERPIXEL (1)      // LCD 每像素的位数

```

3 在 μ C/GUI 中矢量汉字的显示

μ C/GUI 中字体的显示是通过查找字模的方式实现的。字体库中的每一个字母都有其对应的字模。所有的字母的字模由 GUI_FONT 和 GUI_FONT_PROP 这 2 个结构体来进行统一管理。GUI_FONT 结构体中定义了该字母的点阵大小(比如 16×16 或者 8×8)和 GUI_FONT_PROP 结构体的入口地址。GUI_FONT_PROP 这个结构体建立了字库中字母编码(比如字母 A 在 ASCII 中的字母编码为 33)和字模数据存放地址的映像。GUI_FONT_PROP 中 pNext 指针指向下一个 GUI_FONT_PROP 数据的入口地址,这为解决在字母编码不连续的情况下,保证字模数据在程序段的存储连续这一问题提供了一个良好的解决方案。设计人员可以定义多个 GUI_FONT_PROP 结构,只要使上一个结构体的 pNext 指针指向下一个 GUI_FONT_PROP 结构体,并且保证该指针最终指向零地址空间即可。掌握这一原理后,可以方便地制作自己的汉字字库。

利用 μ C/GUI 汉字库生成器可以生成指定字体矢量汉字的字模文件。该文件与 μ C/GUI 字库文件格式相同,但是该文件包含所有 GB2312 中的汉字,字号越大文件会变得很大,只需找出自己需要的汉字字模,并按 μ C/GUI 中字体文件的格式得到自己的字体文件,这个过程可以编程实现。图 4 是用 Proteus 仿真的结果。

本文通过对 μ C/GUI T6963C LCD 控制器驱动移植及 μ C/GUI 中实现矢量汉字的介绍,为 μ C/GUI 其他 LCD 控制器驱动的移植提供了参考,同时也为利用 μ C/GUI 实现更为复杂的图形用户界面打下了基础。

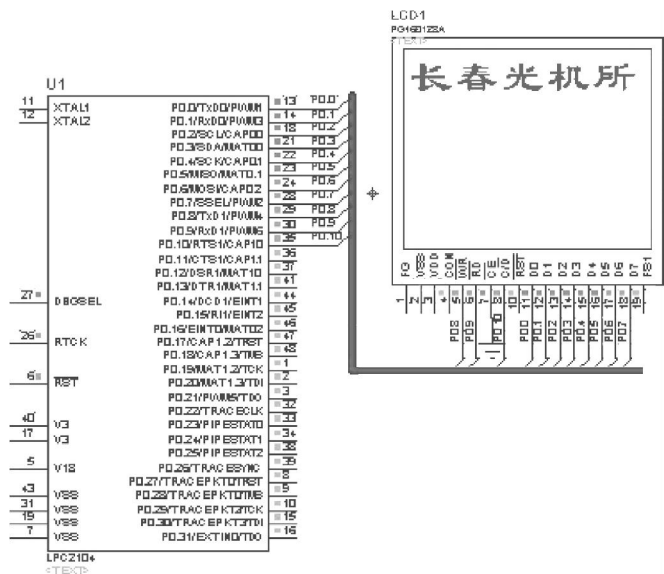


图 4 用 Proteus 进行的 μ C/GUI 仿真

参考文献

- [1] 王孝平.嵌入式GUI性能评测[D].四川 电子科技大学 2007.
- [2] 陈钦梧 张泉 周林峰. μ C/GUI在博创ARM300-S上的移植[J]. 计算机工程与设计 2006 27(19):3697-3700.
- [3] 张磊 江海河 储焰南.基于 μ C/OS-II的嵌入式GUI研究与应用[J].嵌入式操作系统应用 2008 24(6-2):50-52.
- [4] T6963C数据手册.日本东芝公司.
- [5] μ C/GUI user manual.Micrium Technologies Corporation.

(收稿日期:2009-05-30)

(上接第38页)

为简单,是一个逆过程,在此不用实例进行表述。

ZigBee 技术是最新的短距离无线通信技术之一,其安全方面越来越受到人们的关注。本文主要针对 ZigBee 的 MAC 层安全进行分析,介绍了 MAC 层的安全结构。对其中的一种安全方案 CCM 算法进行了详细的研究,该算法提供了数据完整性检查、加密等功能,可以提供高质量的安全防护。通过对 MAC 层的安全研究,了解 MAC 层所提供的安全功能,便于进一步开展 ZigBee 技术的安全研究。

参考文献

- [1] ZigBee specification. <http://www.zigbee.org>. 2008.
- [2] 胡江.ZigBee无线传感器网络安全性分析[J].科技论坛,2007

(09):103-105.

- [3] 周公博 韩振铎 胡宁宁 ZigBee标准的密钥协商机制分析[J]. 电子技术应用 2007 33(10):61-69.
- [4] 吕治安 ZigBee网络原理与应用开发[M] 北京 北京航空航天大学出版社 2008:76-79.
- [5] 瞿雷 刘盛德 胡咸斌 ZigBee技术及应用[M] 北京 北京航空航天大学出版社 2007.

(收稿日期:2009-06-21)