lkdGui使用文档

lkdGui介绍：

lkdGui是一款为单色显示屏制作的图形化界面，lkdGui主要定位于工业控制领域，用于简单漂亮的图形界面设计。它的源代码全部使用c语言编写，符合c99标准。当然，移植也非常简单，后面会详细介绍。

一：lkdGui的移植

1.修改GuiDrawPoint（）函数。GuiDrawPoint（）是最底层的打点函数，用于向显示屏打点。由于此函数在运行过程中会被频繁调用，因此，用户在编写的时候要好好优化优化此函数。

在介绍此函数之前先和大家谈谈单色显示屏，大多数单色显示屏的打点并不十分友好，比如笔者用的一款160X160的单色显示屏，它必须一次性打三个点，如果只打一个点它就不会显示，这便给要画一个漂亮界面的程序员狠狠的刺了一刀。为了避免这个不友好的打点，笔者建议对较小的单色显示屏，先给它分配这么多点的内存，比如像笔者的160X160的显示屏，给它分配了lkdGuiBuff[160][160/8] = 3K多一点的内存，看上去的确浪费了不少内存，但个人感觉如果内存充足的话还是挺划算的。然后将lkdGuiBuff[160][160/8]当作显示屏的缓存，缓存里的每个点的位置及当前值就代表显示屏的每个点的位置和当前颜色。然后在GuiDrawPoint（）中实现根据参数的位置和颜色修改内存里对应的值。当然你也可以直接通过GuiDrawPoint（）向屏幕打点。

void GuiDrawPoint(lkdCoord x, lkdCoord y, lkdColour color)

{

//根据自己的显示屏的实现打点

}

2.修改 GuiUpdateDisplayAll（）函数。GuiUpdateDisplayAll（）是显示函数。如果GuiDrawPoint（）使用的是利用缓存的模式，在画完界面之后就可以调用此函数让所有缓存中的点输出的显示屏上，如果GuiDrawPoint（）直接将点打到了显示屏上，这个函数为空就行了。

void GuiUpdateDisplayAll(void)

{

}

移植好这两个函数lkdGui图形库就可以使用了。但此时只能使用正常的画图画线等基本函数，要想实现字体显示功能，请参看字体使用章节。

对于GuiReadPoint（）目前为暂留接口，用户不用管。GuiRangeUpdateDisplay（）是只更新部分显示区域，用法和GuiUpdateDisplayAll（）差不多。如果用户只使用GuiUpdateDisplayAll（），那么此函数也可以不用管，或者里面直接调用GuiUpdateDisplayAll（）。

二：基本文件和函数介绍

在lkdGui文件夹中可以看到三个文件夹：

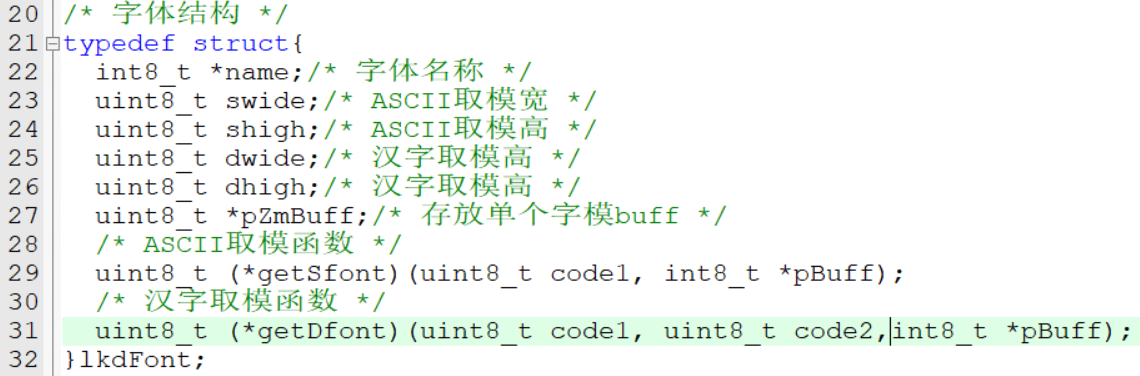
其中Driver文件夹中是一些显示屏移植好的函数，大家可以参考一下。

Include文件夹中是lkdGui的.h存放位置，需要包含到工程路径中去。

Source文件夹中存放的是lkdGui的.c文件。下面将一一介绍每个.c文件。

1. lkdGuiBasic.c 此文件主要实现了画点、线、矩形、位图等基本函数。每个函数的用法文件中说明的很详细，此处不在累赘。

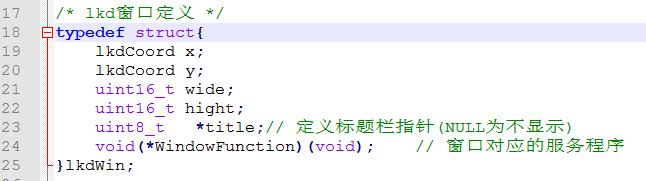
2. lkdGuiFont.c 此文件是字体的接口文件。大家可以根据自己的需求指定字模库，然后利用此文件中的统一接口接入。lkdGuiFont.h中有这样一个重要的结构体，见下图



这个结构体就是用户自定义字体与lkdGui连接的桥梁。结构体成员name是字体的名称。用用户给自定义字体取名。swide，shigh是ASCII字符的宽度和高度，getSfont函数指针是指向用户定义好的ASCII字模读取函数，通过第一个参数输入ASCII码的码值，通过pBuff获取到ASCII码的字模点阵。同理，dwide,dhigh,是汉字的宽度和高度，getDfont函数指针是指向用户定义好的汉字字模读取函数。Code1为汉字低位编码，code2为高位编码。成员pZmBuff是lkdGui读取字模时临时存放字模的地方，它指向的数组一定要能容纳下最大字模。特别注意，字模的宽度一般都是以字节为单位的。比如6\*8的字体，其实它的宽度还是8，所以获取的字模应该是8\*8，那么pZmBuff指向的数组应该为buff[64]的全局数组。

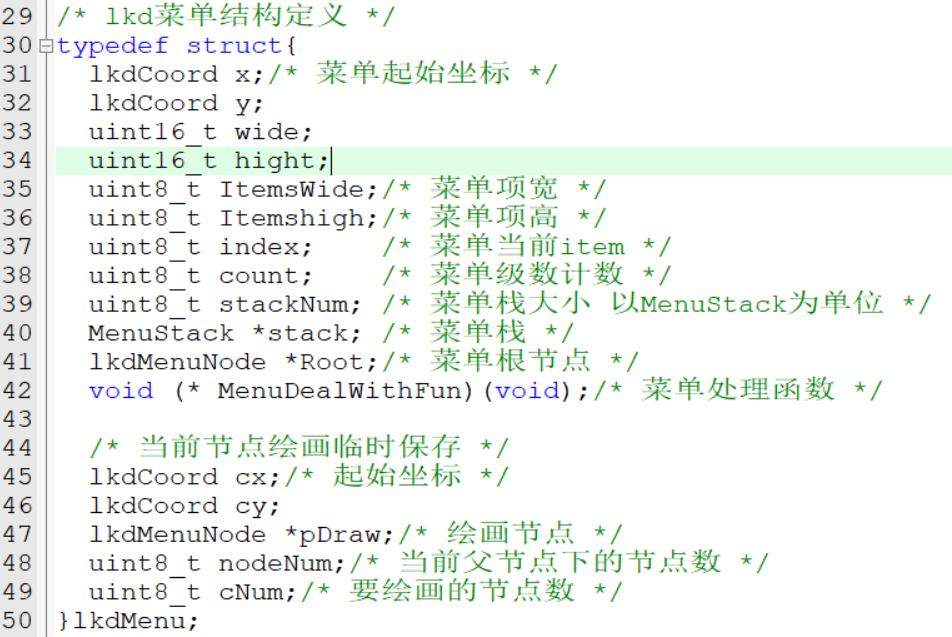
当大家把这个结构体定义的变量的所有成员都设置好之后，就可以通过GuiFontSet（）来将当前字体切换成用户定义的字体呢。通过GetCurrentFont（）可以获取到当前使用的字体指针。这个文件中的写文本函数GuiText（）和写行文本函数GuiRowText（）有相应的列子程序，请大家参考相应程序。

1. lkdGuiColour.c此文件是颜色控制文件。定义了两个全局变量，一个前景色forecolor ，一个背景色backcolor 。可以通过GuiSetForecolor（）和GuiSetbackcolor（）设置前景色以及背景色。通过GuiExchangeColor（）交换前景色和背景色。
2. lkdGuiControl.c此文件是控件库。包含进度条，滚动条，按钮等常用控件。具体的使用见相关例子。
3. lkdGuiWin.c此文件是窗口管理库，lkdGui的窗口实现了用户不同界面的独立化，使用户的开发变得十分方便，所以建议用户使用lkdGui时，以窗口为单位进行开发。

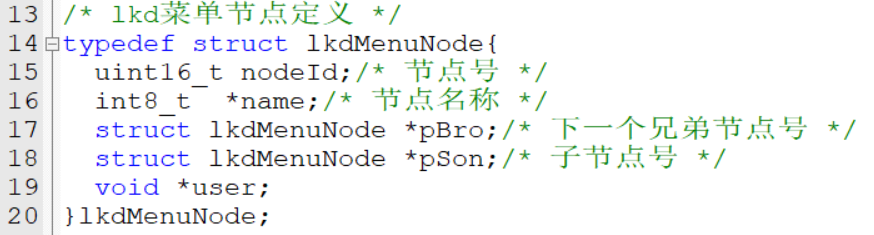


上图是lkdGui的窗口结构体lkdWin。x,y是窗口的坐标，wide，hight是窗口的大小,title是窗口的名称。显示在窗口的标题栏中，如果为NULL,那么标题栏就不会绘制出来。函数指针WindowFunction是窗口的内容函数，具体要干的事就在这个函数里面干。当此窗口为最顶层窗口时就会一直调用此函数。函数管理是通过压栈的方式管理，在Gui线程或者主函数中循环调用GuiWinDisplay()，在需要添加窗口的时候调用GuiWinAdd()添加一个窗口，那么GuiWinDisplay()就会一直调用当前添加的窗口对应服务程序。当要退出窗口时通过GuiWinDeleteTop（）就可以退出当前窗口。宏GUIWINMANAGE\_NUM为最大可以嵌套的窗口数量，如果嵌套较多，可以自行修改这个宏的大小。

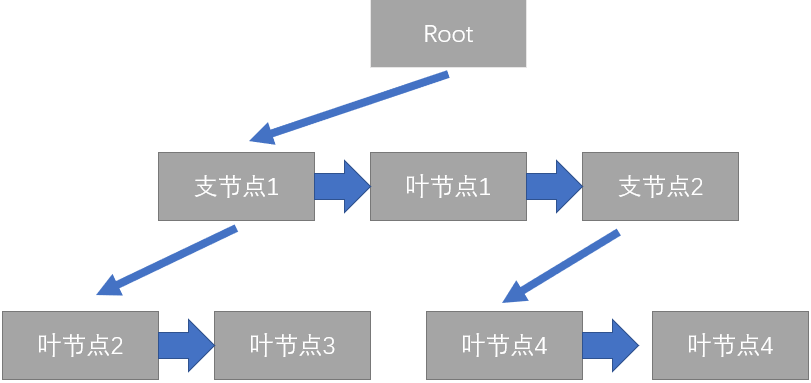
6. lkdGuiMenu.c此文件是菜单管理库。下图是lkdGui的菜单结构体



x,y为菜单的起始坐标，wide,hight为菜单的大小。ItemsWide,ItemsHigh为菜单项的宽度和高度，用户根据自己的菜单项大小来初始化。Index是菜单当前菜单项的索引。Count是菜单压如栈的计数。Stack是菜单的栈指针，由用户分配内存，大小是以结构体MenuStack为单位的，stackNum为栈大小，由用户根据stack的大小赋值。Root是菜单和菜单项连接的桥梁，后面会具体介绍。函数指针MenuDealWithFun是菜单控制的回调函数，由用户提供，菜单例子中有模板。后面的是lkdGui管理菜单使用的，用户不必理会。下面说一下菜单节点结构体



nodeId是节点标识，用户应该确保它是唯一的，name是节点名称，也就是菜单项的名称。pBro是此节点的下一个兄弟节点。pSon是此节点的儿子节点。lkdGui的菜单节点连接使用的是树结构。



图中，支节点1的pBro为叶节点1，PSon为叶节点2。叶节点1的pBro为支节点2，叶节点1的pSon为NULL。关于菜单的使用参见菜单使用例程。