

目录

一、emWin 介绍-----	1
1.1 emWin 简介-----	1
1.2 emWin 应用的软件架构-----	2
二、《emWin 用户手册》介绍及使用-----	2
2.1 《emWin 用户手册》简介-----	2
2.2 根据函数名查找库函数说明-----	3
2.3 根据功能查找库函数-----	3
三、emWin 开发过程中调用 API 及功能-----	4
3.1 通用小工具 API-----	4
3.2 BUTTON API-----	5
3.3 EDIT API-----	6
四、搭建 emWin 的 Windows 仿真环境-----	7
4.1 仿真版本 emWin 文件介绍-----	7
4.2 使用 VS2010 仿真 emWin-----	8
4.3 emWin 仿真工程概览-----	9
五、GUI_Builder-----	18
5.1 简介-----	18
5.2 创建对话框-----	19
5.3 GUI_Builder 的输出-----	20
5.4 如何使用 C 文件-----	24
六、项目代码结构概述-----	24
七、emWin 的 keil 项目介绍-----	32
7.1 keil 环境搭建-----	32
7.2 keil 项目主要文件简介-----	32
7.3 硬件驱动配置（LCD、SDRAM）-----	34
八、Keil 环境下实板调试-----	36
九、问题总结-----	37
参考资料：-----	37

软件工具：Keil 4、VS2010

硬件工具：TFT-LCD7 寸、LPC1788 控制板

一、emWin 介绍

1.1 emWin 简介

emWin(大概是 embed win 的缩写)是 Segger 公司针对嵌入式平台开发的图形软件库，通过调用 emWin 提供的函数接口，开发嵌入式图形界面应用变得简单而快捷。

图形软件库就是一些与图形界面相关的函数，它提供接口给用户调用。例如，在没有使用图形软件库时，要控制 LCD 显示一个圆，需要用户自行根据 LCD 驱动编写画圆的函数，而使用图形界面软件库的话，这个画圆函数已经由该软件库提供给用户，用户只需要调用这个函数，设置圆的半径、圆心、颜色就可以画出自己想要的圆，不需要再考虑画圆的算法了。

emWin 并不只是提供了画圆之类的基本绘图函数库，它的主要功能如下：

- 绘制 2-D 图形：绘制圆、椭圆、多边形、弧线、线图和饼形图等；
- 显示位图文件：显示 BMP、JPEG、GIF 及 PNG 格式的图片文件；
- 显示文字：支持显示多国语言的文字，如中英文；
- 处理用户输入：如键盘、鼠标及触摸屏的人机交互输入处理；
- 提供各种窗口对象：这些窗口对象也被称为图形控件，最常见的如菜单控件、窗口控件、按钮控件、复选框及框架窗口控件等，有了这些控件，就非常容易地制作出类似 windows 系统的控制界面了。

emWin 的这些功能，就是为了解决人机交互应用的设计难题。而它对 ROM 和 RAM 的消耗相对来说并不高。据官方统计，它们提供的一个基本例程(移植了 emWin 函数库显示“HelloWorld”)，消耗 ROM 60KB，RAM 7KB，这在当前大多数嵌入式平台都是可以接受的。又因为 emWin 可以在多任务系统及裸机的平台上运行，尤其适合嵌入式乱七八糟的开发要求，自然使得 emWin 在嵌入式人机交互场合中大放异彩。

1.2 emWin 应用的软件架构



图 1-1

从最底层看起，首先是硬件层，硬件层就是指最基本的板级支持软件包(BSP)，也就是我们常说的底层驱动，它提供了对硬件平台资源最直接的访问；emWin 软件库承上启下，它运行在硬件层之上，与硬件层之间仅靠液晶驱动连接起来，而向应用层则提供了各种便于用户制作 GUI 应用的接口。

在没有使用 emWin 的软件架构中，很可能就是由硬件层直接过渡到应用层，甚至根本没有分层的概念，这样的应用程序将十分依赖于某个特定的硬件平台，不容易迁移到其它设备之中。从这个架构图可以看出，只需要编写好液晶驱动这个粘合 emWin 库和硬件 BSP 层的中间件，就可以正常使用 emWin 了，迁移应用到不同的平台也简单得多。

二、《emWin 用户手册》介绍及使用

2.1 《emWin 用户手册》简介

《emWin 用户手册》就是关于 Segger 官方给出的 emWin 帮助文档，它的内容包括 emWin 简介、仿真、各种库函数详细信息、液晶驱动移植、配套工具软件的使用、emWin 性能及资源占用都一一进行了说明，对于没有开源代码的软件库，其帮助文档的重要性不言而喻。

2.2 根据函数名查找库函数说明

我们会经常遇到一些陌生的 emWin 库函数,库函数大多以 GUI 为前缀命名,如 GUI_Init、GUI_DispString 等 GUI_xxx 格式。当我们想要获这些库函数详细信息时,可在《emWin 用户手册》中搜索。如下图示:

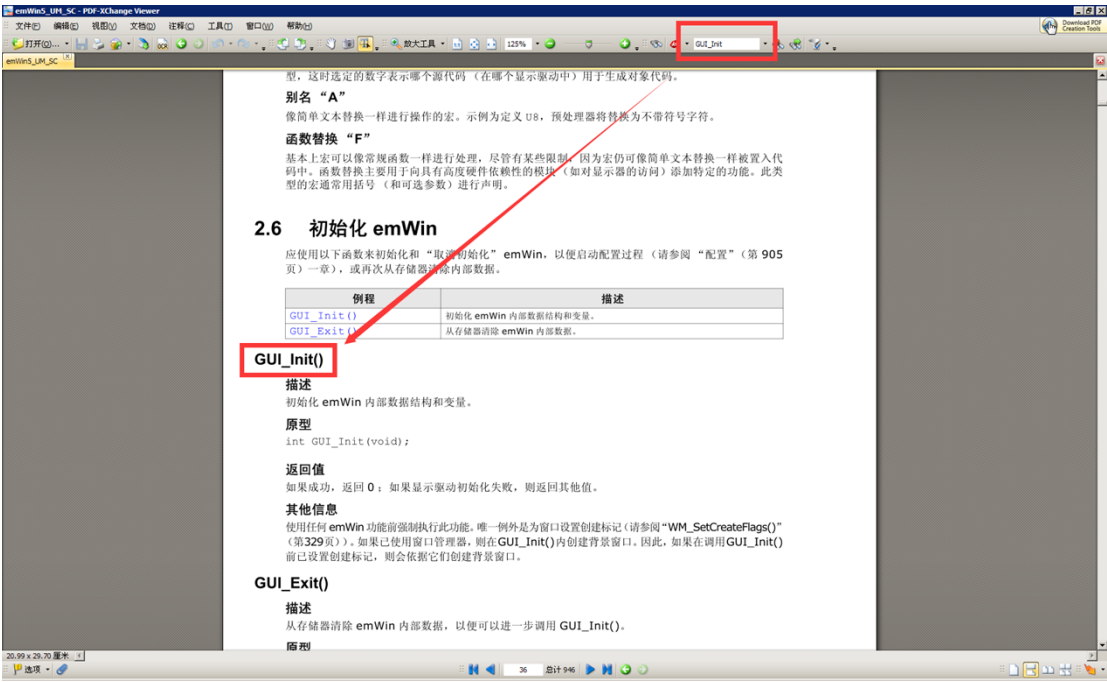


图 2-1

2.3 根据功能查找库函数

我们根本不知道 emWin 库有没有这种功能的函数,就算知道有,也常常忘记了函数的名字。这时候我们就可以根据《emWin 用户手册》的目录来找了,《emWin 用户手册》对它每种功能的函数集都有一个函数列表,简要介绍函数的功能,浏览相关的函数列表及其简介,就可以找到相关的库函数名了,这时再搜索该函数名就可以获得详细的函数说明。例如我们想查找某个在特定位置显示字符串的函数,就可以在用户手册的标签页中找到“文本 API”标签(其它功能函数如“评估 API(此处文档翻译有误,应为:数值 API)”、“图形 API”等),然后就可以在文本 API 函数列表中找到一个名叫 GUI_DispStringAt 的函数,其功能简介是“在指定位置显示字符串”,它正符合我们的要求,再根据给出的函数名,就可以搜索到该函数的详细说明,如下图示:

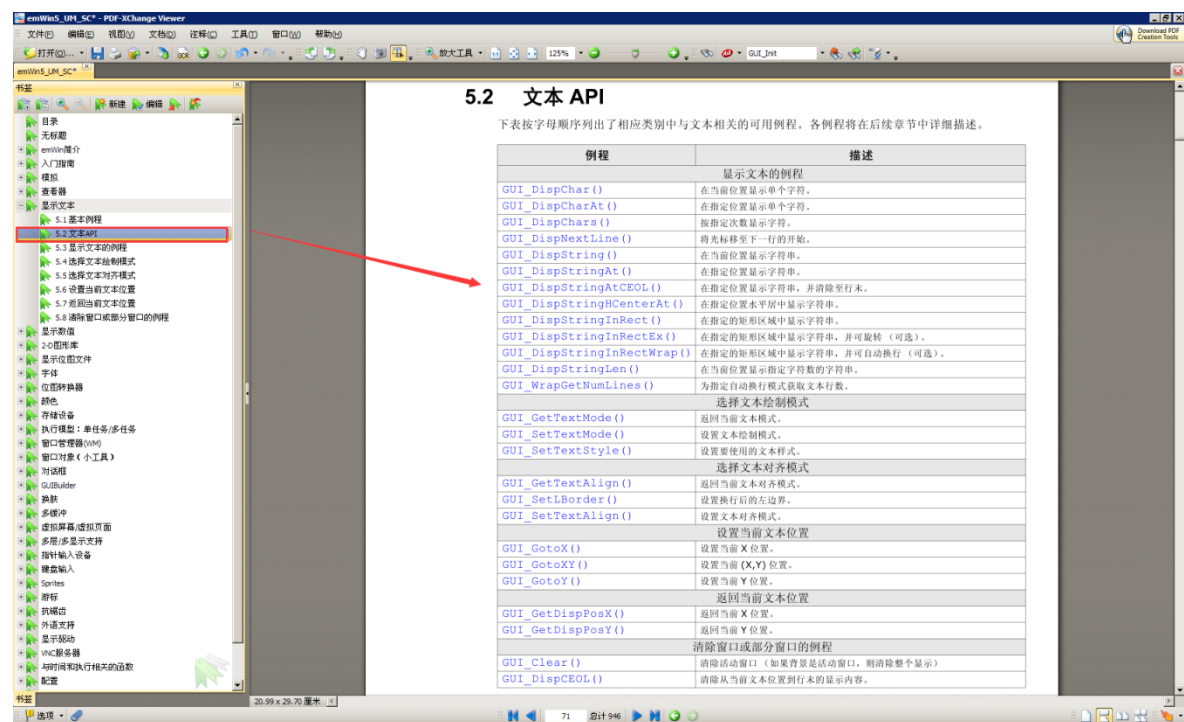


图 2-2

三、emWin 开发过程中调用 API 及功能

3.1 通用小工具 API

例程	描述
WM_DeleteWindow()	删除窗口
WM_DisableMemdev()	禁止使用存储设备进行重绘
WM_EnableMemdev()	启用存储设备用于重绘
WM_InvalidateWindow()	使窗口无效
WM_Paint()	立即绘制或重绘窗口

WM_DelectWindow()

原型

void WM_DeleteWindow(WM_HWIN hWin);

参数: hWin 描述: 窗口句柄

其他信息

删除窗口前，他将收一条 WM_DELETE 消息，此消息通常用于删除所使用的任何对象（小工具），并释放窗口动态内存分配。

如果指定的窗口具有任何现有子窗口，则在删除窗口前自动删除这些子窗口，因此不必分别删除子窗口，窗口删除前会发送一条 WM_NOTIFICATION_CHILD_DELETED 消息到其父窗口。

WM_DisableMemdev()

原型

void WM_DisableMemdev(WM_HWIN hWin);

参 数 : hWin 描 述 : 窗 口 句 柄

WM_EnableMemdev()

原型

void WM_EnableMemdev(WM_HWIN hWin);

参数: hWin 描述: 窗口句柄

WM_InvalidateWindow()

原型

void WM_InvalidateWindow(WM_HWIN hWin);

参数: hWin 描述: 窗口句柄

WM_Paint()

原型

void WM_Paint(WM_HWIN hWin);

参数: hWin 描述: 窗口句柄

3.2 BUTTON API

例程	描述
BUTTON_Create()	创建 BUTTON 小工具
BUTTON_SetBkColor()	设置按钮背景色
BUTTON_SetDefaultFont()	设置 BUTTON 小工具默认字体
BUTTON_SetDefaultTextAlign()	设置 BUTTON 小工具默认文本对齐方式
BUTTON_SetBitmap()	设置显示指定按钮时要使用的位图

BUTTON_Create()

描述

在指定位置创建指定尺寸大小的 BUTTON 小工具

原型

BUTTON_Handle BUTTON_Create(int x0,int y0,int xSize ,ySize int Id,int Flags);

参数: x0 描述: 按钮的最左像素（在父窗口中）
 y0 按钮的最上像素（在父窗口中）
 xSize 按钮的水平尺寸（单位: 像素）
 ySize 按钮的垂直尺寸（单位: 像素）
 Id 按钮按下时要返回的 ID
 Flags 窗口创建标记, 为了使小工具可见, 通常是 WM_CF_SHOW

返回值

已创建 BUTTON 小工具的句柄, 函数失败时为 0;

BUTTON_SetBkColor()

原型

void BUTTON_SetBkColor(BUTTON_Handle hObj, unsiged int Index,GUI_COLOR Color);

参数: hObj 描述: 按钮的句柄
 Index 颜色的索引
 Color 要设置的颜色

BUTTON_SetDefaultFont()

描述

设置用于显示 BUTTON 小工具文本 GUI_FONT 结构的指针

原型

```
void BUTTON SetDefaultFont(const GUI_FONT * pFont);
```

参数: pFont 描述: 要使用的 GUI_FONT 指针

BUTTON_SetDefaultTextAlign()

描述

设置用于显示 BUTTON 小工具文本的默认对齐方式

原型

```
void  BUTTON  SetDefaultTextAlign(int Align);
```

参数: Align 描述: 要使用的文本对齐方式

BUTTON_SetBitmap()

原型

```
Void  BUTTON_SetBitmap(BUTTON_Handle  hObj, unsigned int  Index,const GUI_BITMAP  *
pBitmap);
```

参数: hObj	描述: 按钮的句柄
----------	-----------

Index 位图的索引

pBitmap 位图结构的指针

其他信息

如果仅设置了未按下状态的位图，则当按钮按下或者禁用时，按钮也将显示该位图。

3.3 EDIT API

例程	描述
EDIT_Create()	创建 EDIT 小工具
EDIT_GetText()	获取用户输入
EDIT_SetFont()	选择文本字体
EDIT_SetMaxLen()	设置编辑字段的最大字符
EDIT_SetVlaue()	设置当前值

EDIT_Create()

原型

```
EDIT Handle EDIT Create(int x0,int y0,int xsize,int ysize,int Id,int ,MaxLen,int Flags);
```

参数: x0	描述: 编辑字段的最左像素 (在父坐标中)
--------	-----------------------

v0	编辑字段的最上像素（在父坐标中）
----	------------------

xsize	编辑字段的水平尺寸（单位：像素）
-------	------------------

ysize	编辑字段的垂直尺寸（单位：像素）
-------	------------------

Id	将返回的 ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

MaxLen 最大字符数

Flags 窗口创建标记

EDIT GetText()

原型

```
Void EDIT GetText(EDIT Handle hObj,char * sDest,int MaxLen);
```

参数: hObj	描述:编辑字段句柄
----------	-----------

sDest	指向缓冲区指针
-------	---------

MaxLen	缓冲区的大小
--------	--------

原型

参数: hObj	描述: 编辑字段的句柄
pFont	字体的指针

原型

参数: hObj	描述: 编辑字段的句柄
MaxLen	字符数

原型

参数: hObj	描述: 编辑字段的句柄
Value	新值

4.1 仿真版本 emWin 文件介绍

The diagram illustrates the file structure of the 'SimulationTrial' project and provides instructions on how to open it in Visual Studio 2013.

File Structure:

- Application:** 综合应用例程的C文件
- Backup:** VS环境配置文件
- Config:** LCD及emWin配置文件
- Exe:** 综合应用例程的可执行文件
- GUI:** GUI库及头文件
- Sample:** 入门独立小例程
- Simulation:** GUI仿真库文件
- Tool:** 字体、图片转换等软件小工具

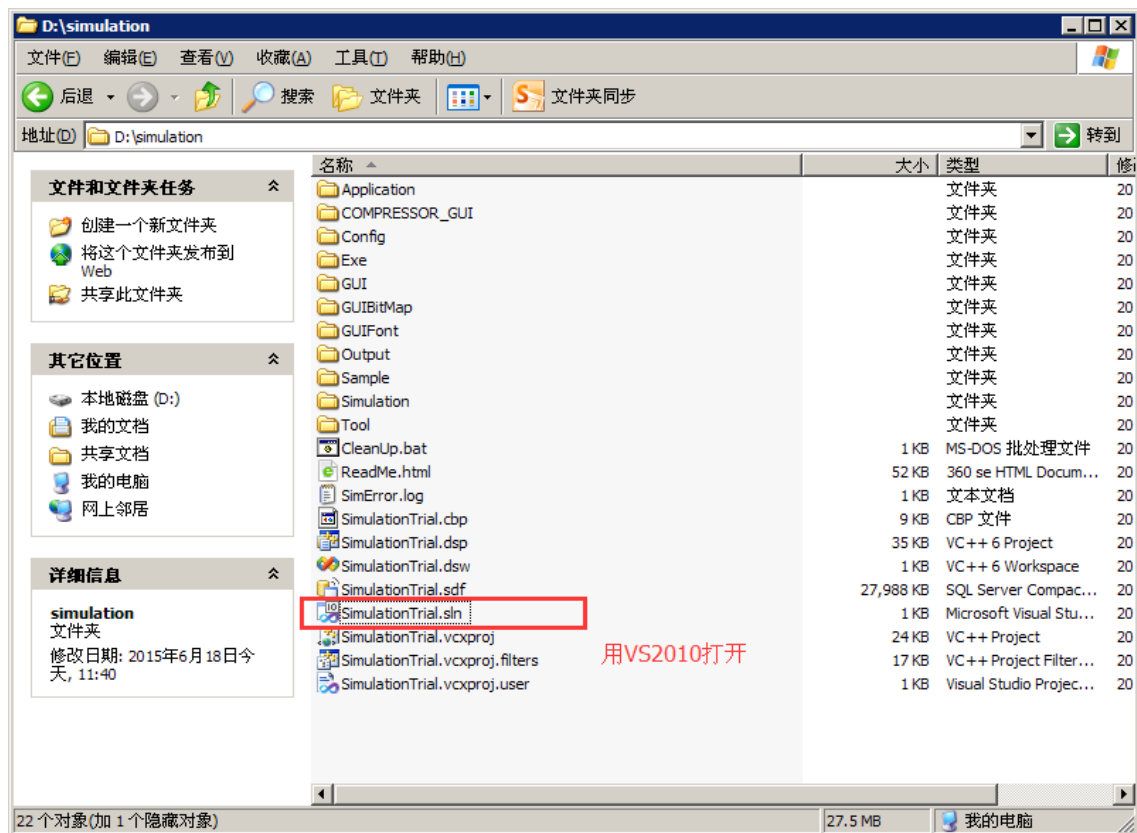
VS的工程文件, 使用VS2013打开

SimulationTrial.sln 即可进行仿真

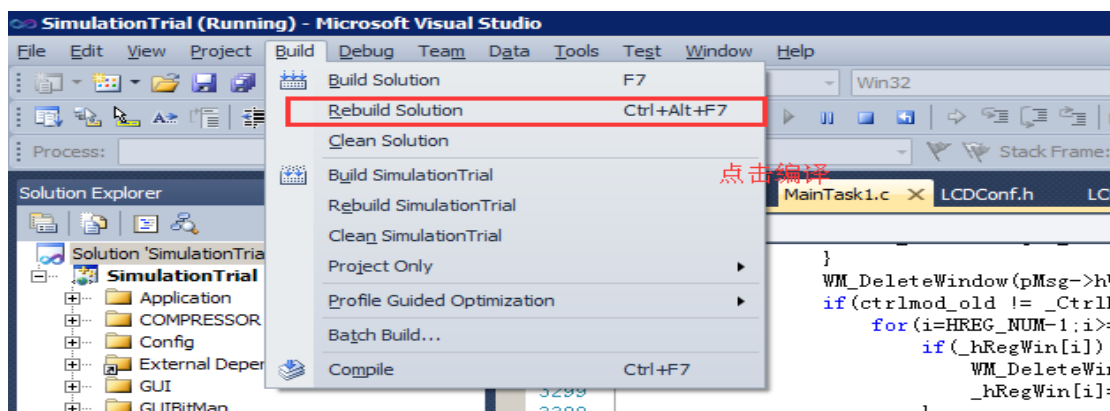
4.2 使用 VS2010 仿真 emWin

仿真版本压缩包，注意解压后的文件路径不要出现中文，否则在编译时会出错的；

2) 打开仿真工程。安装了 VS2013 集成环境后，打开 emWin 仿真版本的目录，找到“SimulationTrial.sln”文件，双击打开，打开后会弹出一个对话框提示 VC++版本问题的，直接选择“是”即可，如下图所示：



3) 开始编译工程。可以直接按快捷键“F5”，或者点击编译器中的调试按钮，编译并运行工程，编译器会弹出一个提示框，我们在该对话框打上勾，并选择“是”，让它以后都不用再提示我们选择生成新项目。接下来就开始了编译过程，正常的话都会编译通过，然后弹出一个 emWin 仿真运行的界面，如下图所示：





4.3 emWin 仿真工程概览

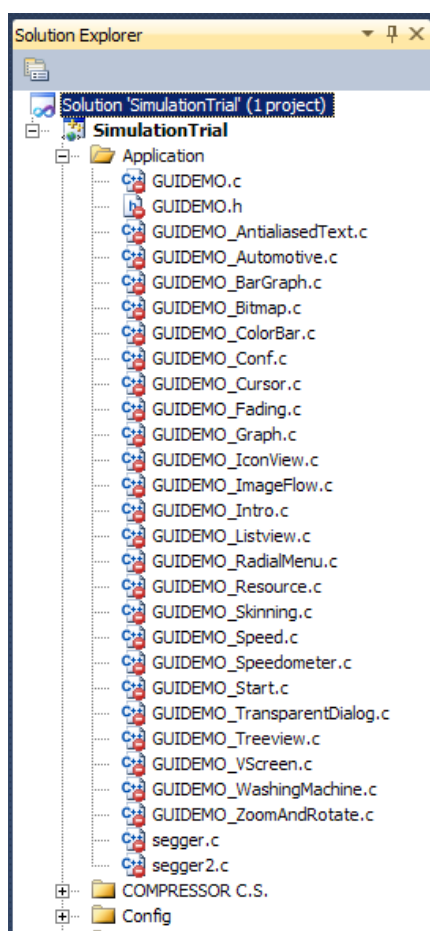
主要包含了 Application、Config、GUI、Sample、Simulation 目录及一些外部依赖项文件，如下图示：



4.3.1 Application 文件夹

Application 文件夹是演示例程的应用文件包。我们编译的文件都是在这个 Application 目录下的，它包含了官方演示例程的应用文件，展开 Application 目录后，可以到这些文件命名都很规范，基本上每个文件对应一个演示例程的小环节，如果看到很炫酷的示例，可以到这里找找它的源代码来学习(演示例程学习难度较大，适合有一定 emWin 基础的用户)。

文件列表如下图：



4.3.2 Config 文件夹

Config 文件夹是显示配置文件包。在 Config 文件夹下，有 GUIConf.c、GUIConf.h、LCDConfig.c、LCDConfig.h 及 SIMConfig.c 文件，其中前三个文件在实际的嵌入式平台是非常重要的，它们分别起到了分配存储空间、裁剪 emWin 功能模块和 LCD 底层驱动实现。而 LCDConfig.h 文件没有内容，SIMConfig.c 文件是 VS2013 仿真环境需求的一些特定设置，在此我们不作探讨。关于 GUIConf.c、GUIConf.h 与 LCDConfig.c 文件的内容见代码清单如下所示：

GUIConf.c 文件内容

```

/*****
*          SEGGER Microcontroller GmbH & Co. KG          *
*    Solutions for real time microcontroller applications    *
*****/
*
*    (c) 1996 - 2015 SEGGER Microcontroller GmbH & Co. KG    *
*
*    Internet:      Support:      *
*
*****/
```

**** emWin V5.28 - Graphical user interface for embedded applications ****
emWin is protected by international copyright laws. Knowledge of the
source code may not be used to write a similar product. This
only be used in accordance with a license and should not be re-
distributed in any way. We appreciate your understanding and fairness.

File : GUIConf.c
Purpose : Display controller initialization

-----END-OF-HEADER-----

*/

#include "GUI.h"

/*****

*

* Defines

*

*****/

*/

//

// Define the available number of bytes available for the GUI

//

#define GUI_NUMBYTES 0x400000

//

// Define the average block size

//

#define GUI_BLOCKSIZE 0x80

/*****

*

* Public code

*

*****/

*/

/*****

*

* GUI_X_Config

*

* Purpose:

* Called during the initialization process in order to set up the

* available memory for the GUI.

*/

void GUI_X_Config(**void**) {

```

//
// 32 bit aligned memory area
//
static U32 aMemory[GUI_NUMBYTES / 4];
//
// Assign memory to emWin
//
GUI_ALLOC_AssignMemory(aMemory, GUI_NUMBYTES);
GUI_ALLOC_SetAvBlockSize(GUI_BLOCKSIZE);
}

```

/****** End of file *****/

GUIConf.c 文件主要包含了一个 GUI_X_Config 函数，在这个函数中向系统申请了一个大小为 4MB(0x400000) 的静态空间 aMemory，接着调用 emWin 的库函数 GUI_ALLOC_AssignMemory 把这个空间分配给 emWin，并使用库函数 GUI_ALLOC_SetAvBlockSize 设置存储块的平均大小。所以 emWin 的库函数使用的存储空间都是 aMemory 中分配的。在嵌入式平台上，我们需要根据其限制控制分配给 emWin 库的 aMemory 大小，修改宏 GUI_NUMBYTES 即可。

LCDConfig.c文件内容

```

/*****
*                               *
*   SEGGER Microcontroller GmbH & Co. KG                               *
*   Solutions for real time microcontroller applications                 *
*                               *
*                               *
*   (c) 1996 - 2015 SEGGER Microcontroller GmbH & Co. KG               *
*                               *
*   Internet:   Support:   *
*                               *
*****/

** emWin V5.28 - Graphical user interface for embedded applications **
emWin is protected by international copyright laws. Knowledge of the
source code may not be used to write a similar product. This
only be used in accordance with a license and should not be re-
distributed in any way. We appreciate your understanding and fairness.

-----
File      : LCDConf.c
Purpose   : Display controller configuration (single layer)
-----END-OF-HEADER-----
*/

#include "GUI.h"

```

```

/*****
*
*      Layer configuration (to be modified)
*
*****/
//
// Physical display size
//
#define XSIZE_PHYS  800
#define YSIZE_PHYS  480
#define VYSIZE_PHYS (YSIZE_PHYS << 1)

//
// Color conversion
//
#define COLOR_CONVERSION GUICC_888

//
// Display driver
//
#define DISPLAY_DRIVER GUIDRV_WIN32

/*****
*
*      Configuration checking
*
*****/
*/
#ifndef VXSIZE_PHYS
    #define VXSIZE_PHYS XSIZE_PHYS
#endif
#ifndef VYSIZE_PHYS
    #define VYSIZE_PHYS YSIZE_PHYS
#endif
#ifndef VRAM_ADDR
    #define VRAM_ADDR 0
#endif

#ifndef XSIZE_PHYS
    #error Physical X size of display is not defined!
#endif
#ifndef YSIZE_PHYS
    #error Physical Y size of display is not defined!

```

```

#endif
#ifndef COLOR_CONVERSION
    #error Color conversion not defined!
#endif
#ifndef DISPLAY_DRIVER
    #error No display driver defined!
#endif

/*****
*
*      Public code
*
*****/
*/
/*****
*
*      LCD_X_Config
*
* Purpose:
*   Called during the initialization process in order to set up the
*   display driver configuration.
*
*/
void LCD_X_Config(void) {
    //
    // Set display driver and color conversion for 1st layer
    //
    GUI_DEVICE_CreateAndLink(DISPLAY_DRIVER, COLOR_CONVERSION, 0, 0);
    //
    // Display driver configuration
    //
    LCD_SetSizeEx    (0, XSIZE_PHYS,  YSIZE_PHYS);
    LCD_SetVSizeEx   (0, VXSIZE_PHYS, VYSIZE_PHYS);
    LCD_SetVRAMAddrEx(0, (void *)VRAM_ADDR);
    //
    // Set user palette data (only required if no fixed palette is used)
    //
    #if defined(PALETTE)
        LCD_SetLUTEx(0, PALETTE);
    #endif
}

/*****
*

```

```

*      LCD_X_DisplayDriver
*
* Purpose:
*   This function is called by the display driver for several purposes.
*   To support the according task the routine needs to be adapted to
*   the display controller. Please note that the commands marked with
*   'optional' are not cogently required and should only be adapted if
*   the display controller supports these features.
*
* Parameter:
*   LayerIndex - Index of layer to be configured
*   Cmd        - Please refer to the details in the switch statement below
*   pData      - Pointer to a LCD_X_DATA structure
*
* Return Value:
*   < -1 - Error
*   -1 - Command not handled
*   0 - Ok
*/
int LCD_X_DisplayDriver(unsigned LayerIndex, unsigned Cmd, void * pData) {
    int r;

    switch (Cmd) {
        //
        // Required
        //
        case LCD_X_INITCONTROLLER: {
            //
            // Called during the initialization process in order to set up the
            // display controller and put it into operation. If the display
            // controller is not initialized by any external routine this needs
            // to be adapted by the customer...
            //
            // ...
            return 0;
        }
        case LCD_X_SETVRAMADDR: {
            //
            // Required for setting the address of the video RAM for drivers
            // with memory mapped video RAM which is passed in the 'pVRAM' element of p
            //
            LCD_X_SETVRAMADDR_INFO * p;
            p = (LCD_X_SETVRAMADDR_INFO *)pData;
            //...

```

```

        return 0;
    }
    case LCD_X_SETORG: {
        //
        // Required for setting the display origin which is passed in the 'xPos' and 'yPos'
        element of p
        //
        LCD_X_SETORG_INFO * p;
        p = (LCD_X_SETORG_INFO *)pData;
        //...
        return 0;
    }
    case LCD_X_SETLUTENTRY: {
        //
        // Required for setting a lookup table entry which is passed in the 'Pos' and
        'Color' element of p
        //
        LCD_X_SETLUTENTRY_INFO * p;
        p = (LCD_X_SETLUTENTRY_INFO *)pData;
        //...
        return 0;
    }
    case LCD_X_ON: {
        //
        // Required if the display controller should support switching on and off
        //
        return 0;
    }
    case LCD_X_OFF: {
        //
        // Required if the display controller should support switching on and off
        //
        // ...
        return 0;
    }
    default:
        r = -1;
    }
    return r;
}

```

/****** End of file *****/

LCDConfig.c 文件包含了 LCD_X_Config 和 LCD_X_DisplayDriver 两个函数
 LCD_X_Config 函数调用 emWin 库函数 GUI_DEVICE_CreateAndLink 配置了显示驱动及颜

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。

如要下载或阅读全文，请访问：

<https://d.book118.com/007140014006006110>