第21章 高层次配置

高层次配置相当简单。在一开始，你通常可以使用已有的配置文件（例如，那些在仿真器当中使用的）。只有在需要对系统进行调整，或减小内存消耗时，配置文件 GUIConf.h 才需要修改。该文件通常位于你的工程根目录的“Config”子目录下。使用 GUIConf.h 进行任意的高层次配置。

在你的硬件上使用μC/GUI 要做的第二件事情是修改依赖于硬件的函数，位于文件 Sample\GUI_X\GUI_X.c 中。

21.1 有效的GUI 配置宏

下表列出了用于μC/GUI 的高层次配置的有效宏。

类型	宏	默认值	说明
N	GUI_ALLOC_SIZE	1000	定义可选择的动态存储器的尺寸（有效的字节数)。动态存储器只有窗口和存储设备才需要。该存储器只能在 GUIAlloc 模块连接时才使用。而这种连接也只是在有动态存储器需求时，才发生。
S	GUI_DEBUG_LEVEL		定义哪个消息要通过 GUI_X_Log()传递。你应该在本章后面有关 GUI_X_Log()的内容中找到详细的描述。
N	GUI_DEFAULT_BKCOLOR		定义默认背景颜色。
N	GUI_DEFAULT_COLOR		定义默认前景颜色。
S	GUI_DEFAULT_FONT	&GUI_Font6x8	定义在执行 GUI_Init()后，哪一种字体作为默认字体。如果你没有使用默认字体，那么改变为不同的默认值是很有意义。一旦默认字体被代码引用，它会因此总是处于连接状态。
N	GUI_MAXTASK	4	当多任务支持被启用时，定义调用μC/GUI 访问显示屏的的最大任务的数量（请参阅第 11 章“执行模型：单任务/多任务”)。
B	GUI_OS	0	多个任务调用μC/GUI 时激活启用多任务的支持（请参阅第 11 章“执行模型：单任务/多任务”)。
B	GUI_SUPPORT_MEMDEV	1	启用可选择的存储器设备支持。不使用存储器设备会节省大约 40 字节用于函数指针的 RAM，同时轻微地增加运行速度。
B	GUI_SUPPORT_TOUCH	0	启用可选择的触摸屏支持。
B	GUI_SUPPORT_UNICODE	1	启用含有 8 位字符串的 Unicode 字符支持。请注意：Unicode 字符始终显示，而字符代码始终被当作 16 位处理。
B	GUI_WINSUPPORT	0	启用可选择的视窗管理器支持。

如何配置GUI

我们推荐使用下面的操作步骤

1. 将原始配置文件备份。
2. 检查所有的配置开关。
3. 删除配置中不使用的部分。

配置范例

以下是一个简略的 GUI 配置文件（ConfigSample\GUIConf.h）：

```
/*********************************************************************
*                          期望功能的配置                           *
*********************************************************************/

#define     GUI_WINSUPPORT        (1)    /* 如果为真（1），使用视窗管理器。*/
#define     GUI_SUPPORT_TOUCH     (1)    /* 使用触摸屏 */

/*********************************************************************
*                          动态存储器的配置                         *
*********************************************************************/

动态存储器用于存储器设备和视窗管理器。
如果你不使用这些特性，没有必要使用动态存储器，它可以完全关闭。（这部分可以删除）
*/

#ifndef GUI_ALLOC_SIZE
#define GUI_ALLOC_SIZE  5000     /* 动态存储器的尺寸 */
#endif

/*********************************************************************
*                          字体的配置                               *
*********************************************************************/

如果你建立了附加字体（通常使用μC/GUI-FontConvert 来做到），为了能使用它们，这些
字体需要按外来字体定义。这里是做这项工作的好地方。*/

#define     GUI_DEFAULT_FONT     &GUI_Font6x8     /* 该字体用作默认字体 */
```

21.2 GUI_X 函数参考

当在你的硬件上使用μC/GUI 的时候，在你的工程当中必须有几个依赖于硬件的函数。使用仿真器时，库已经包含了它们。可以找到一个范例文件，Sample\GUI_X\GUI_X.c。下表在各自的分类中按字母顺序列出了有效的依赖于硬件的函数。这些函数的详细描述在本章稍后部分给出。

函　数	说　明
初始化函数	
GUI_X_Init()	从 GUI_Init()调用，能用于初始化硬件。
时间函数	
GUI_X_Delay()	在一个指定的时间段后返回。
GUI_X_ExecIdle()	只从视窗管理器的非阻塞函数调用。
GUI_X_GetTime()	返回当前系统时间，以毫秒为单位。
内核接口函数	
GUI_X_InitOS()	初始化内核接口模块（建立一个资源旗语/互斥）。
GUI_X_GetTaskId()	返回一个当前任务/线程唯一的 32 位标识符。
GUI_X_Lock()	锁上 GUI（阻塞资源旗语/互斥）。
GUI_X_Unlock()	解锁 GUI（解除资源旗语/互斥阻塞）。
GUI_X_Log()	返回调试信息。如果启用记录的话，这是必需的。

21.3 初始化函数

GUI_X_Init()

描述

从 GUI_Init()调用，能用于初始化硬件。

函数原型

```
void GUI_X_Init (void) ;
```

21.4 时间函数

GUI_X_Delay()

描述

在一个指定的时间段（以毫秒为单位）后返回。

函数原型

```
void GUI_X_Delay(int Period)
```

参数	含义
Period	以毫秒为单位的周期。

GUI_X_ExecIdle()

描述

只从视窗管理器的非阻塞函数调用。

函数原型

```
void GUI_X_ExecIdle (void) ;
```

附加信息

当不再有要求处理的消息时调用。在这一点是，GUI 是最新的。

GUI_X_GetTime()

描述

用于函数 GUI_GetTime，返回当前系统时间（以毫秒为单位）。

函数原型

```
int GUI_X_GetTime (void)
```

返回值

以毫秒为单位的当前系统时间，整数类型。

21.5 内核接口函数

这些函数的详细描述请参阅第 11 章“执行模型：单任务/多任务”。

21.6 调试

GUI_X_Log()

描述

从μC/GUI 返回调试信息。

函数原型

```
void GUI_X_Log(const char * s);
```

参数	含义
s	要发送的字符串的指针。

附加信息

μC/GUI 调用该函数进行错误消息和警告的传送，如果启用记录的话，它是必需的。GUI 调用这个函数依赖于配置宏 GUI_DEBUG_LEVEL。下表列出了 GUI_DEBUG_LEVEL 允许的值：

值	说明
GUI_DEBUG_LEVEL_NOCHECK	没有执行运行时间检测。
GUI_DEBUG_LEVEL_CHECK_PARA	执行参数检测以避免碰撞。
GUI_DEBUG_LEVEL_CHECK_ALL	执行参数检测和一致性检测。
GUI_DEBUG_LEVEL_LOG_ERRORS	记录错误。
GUI_DEBUG_LEVEL_LOG_WARNINGS	记录错误和警告。
GUI_DEBUG_LEVEL_LOG_ALL	记录错误、警告和消息。