应用笔记

emWin 移植向导

Rev. <MM.mm> — 12 May 2011 应用笔记

Document information

Info	Content
Keywords	SWD,SW-DP, AHB-AP, Core debug, IAP, Flash Programming
Abstract	This specification describes how to program the on-chip flash memory of Cortex-M based LPC MCUs, including the background theory, a layered implementation model, and provide a reference implementation with source code.

Revision history

Rev	Date	Description
<M.m>	<yyyymmdd>	<revision description (delete row if not required)>
<M.m>	<yyyymmdd>	<revision description, including important changes re. previous version; no author name(s); latest information in top row>

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

NXP 有些 MCU 不带 LCD 控制器，也可免费使用 emWin。但是本文中的 porting 目前假设读者使用的是自带 LCD 控制器的 MCU 系列，包括 LPC178x, LPC18xx, LPC43xx, 以及老的 LPC247x。

1. Porting overview

移植 emWin 的主要工作涉及配置 emWin 的功能、驱动电路板，以及与执行环境对接。可以从零开始完成这些工作，也可以克隆一个现有的与新目标板类似的移植，并且作必要的修改。毫无疑问，只要可行就应尽量使用“克隆+修改”的方式，即便条件不允许，也应积极搜寻一些可用的代码——从硬件供应商或者开源项目。本文以 NXP 提供的一个移植作为蓝本，着重介绍克隆和修改的过程，但也对涉及到的函数作一介绍，以便当需要亲自实现时作为参考。不管使用哪种方法，在开始前，值得先了解一些移植的背景知识。移植 emWin 主要包括 3 个方面

1.1 剪裁与配置

emWin是可剪裁的，前提是以源代码的方式购买了emWin。在NXP的MCU上免费使用的emWin是以二进制库提供的，因此实际上不存在可剪裁性。不过，emWin的功能均已包含，只是不支持多个任务调用emWin，并且不支持鼠标。
但仍然有些可配置的参数。例如，选择LCD控制器类型，以及为emWin分配内存堆。emWin需要从几十KB到几MB的RAM来管理内部数据，用量主要取决于图形的复杂度。一般情况下我们在外部扩展的SDRAM/SRAM中定义一个C数组来分配一块内存。

1.2 启动目标板

基本的初始化代码，涉及芯片级的时钟、电源、引脚，以及其它基本外设，如调试用的串口。还有十分关键的初始化外部扩展存储器控制器的代码。如果连接了常用的 SDRAM，还包括专用于 SDRAM 的初始化代码。这些代码在 emWin 自身初始化之前执行，可自行定义结构与调用顺序。
LCD 与触摸屏（通常都会有）的初始化代码。这些代码是在 emWin 的初始化过程中由 emWin 调用的，需要遵循 emWin 要求的接口与功能。

1.3 与执行环境适配

emWin 可在传统的前后台式执行环境中使用，也可以配合 RTOS 使用。后者还有单个任务调用 emWin 与多个任务调用 emWin 的区别。需要根据实际使用的方式来完成与系统时钟和任务互斥相关的接口函数，以使 emWin 与底层协调工作。

2. emWin 的配置

2.1 GUIConf.h - 选择功能

这个文件中有多个宏定义来配置 emWin 的功能、执行环境以及默认字体。

GUI_NUM_LAYERS	正整数	设置图层数。只有图形硬件支持多图层，以及软件仿真时有效，在 MCU 上运行时无需关心。
GUI_OS	0/1	未使用RTOS或只有单个任务/线程调用emWin设置为0，否则设置为1。
GUI_MAX_TASK	正整数	仅当 GUI_OS 为 1 时有效，指出最多允许几个任务调用 emWin。可在运行期使用

		GUITASK_SetMaxTask()函数另设
GUI_SUPPORT_TOUCH	0/1	1=支持触摸屏
GUI_SUPPORT_MOUSE	0/1	1=支持鼠标
GUI_WINSUPPORT	0/1	1=支持窗口，会加入窗口管理器与 widgets 的代码
GUI_SUPPORT_MEMDEV	0/1	1=支持存储器设备
GUI_SUPPORT_AA	0/1	1=支持反锯齿功能
GUI_SUPPORT_STATIC_MEMORY	0/1	1=窗口管理器支持静态存储器设备
GUI_DEFAULT_FONT	字体变量名	设置默认字体，可在运行期间使用 GUITASK_SetMaxTask()另设

2.2 GUIConf.c: 为 emWin 分配内存

emWin 在初始化期间会调用分配内存有关的函数"GUI_X_Config"，位于这个文件中。我们在移植时不必修改代码，只要设置以下宏

GUI_NUMBYTES	正整数	供 emWin 在运行期可管理的内存堆容量。本文件会定义一个数组，容量正是由此宏指定，并把数组的地址传给 emWin。
GUI_BLOCKSIZE	正整数	如果应用要使用含有大量条目的列表视图，则可将块平均尺寸设置为较小的值。另一方面，如果应用主要将存储器管理用于一些存储器设备或图像解压，则应把平均尺寸设为较大的值。建议范围为 32 至 1024

这是 emWin 在运行时管理的内存堆，在其上动态分配与回收内存。比如，当新创建一个窗口或存储器设备时，都会分配一部分内存。因此，这部分内存的用量是在运行时期根据显示的图形复杂度动态决定的，编译期的配置选项只给出了堆的容量，并不决定实际的用量。

GUI_X_Config 函数的功能在附录中也有参考。

3. LCDConf.c: 驱动 LCD 与触摸屏硬件

如果待移植平台使用相同的 LCD 与触摸屏，并且引脚配置相同，则这个文件完全不必改动。否则，需要根据改动的内容作相应的修改。

3.1 配置 LCD 与触摸屏参数

在 NXP 的实现中，把与 LCD 相关的参数定义成由多个 C 语言的宏，在 LCDConf.c 文件中。下表的宏与 emWin 交互，在调用一些 emWin 的函数时它们作为参数，如下所示：

XSIZE_PHYS	正整数	LCD 每一行的像素数
YSIZE_PHYS	正整数	LCD 的行数
VXSIZE_PHYS	正整数	虚拟屏幕每一行的像素数，很少使用，定义成 XSIZE_PHYS 即可。
VYSIZE_PHYS	正整数	虚拟屏幕的行数，很少使用，定义成 YSIZE_PHYS 即可。
COLOR_CONVERSION	GUICC_开头的宏名	emWin 在内部使用 32 位 ARGB 格式表达色彩，根据实际使用的液晶屏选择色彩转换方式。常用的有 GUICC_M565 以及红蓝交换的 GUICC_565。

PIXEL_WIDTH	1/2/4	帧缓存中存储每个像素占用的字节数，常用 2
TOUCH_AD_LEFT TOUCH_AD_TOP TOUCH_AD_RIGHT TOUCH_AD_BOTTOM	正整数	触摸屏的校准参考坐标
DISPLAY_DRIVER	&GUIDRV_Lin_开头的变量名	为使用片载的 LCD 控制器，需选用 emWin 内置的线性寻址设备驱动程序（直接存取显存），它们有一系列旋转与镜像处理的变体，均以 GUI_DRV_Lin 开头

下面的宏是 LCD 的时序参数，需要根据 LCD 的器件手册查出它们的值。如果器件手册中提供了一个范围的值，最好先取一个平均值。这些数值相当关键，如果设置不当可能导致显示错乱甚至没有显示！

PPL	正整数	每行像素数
HSW	正整数	水平同步（HSYNC）脉冲的宽度，以像素时钟数为单位
HFP	正整数	水平前肩的像素时钟数
HBP	正整数	水平后肩的像素时钟数
LPP	正整数	LCD 面板的行数
VSW	正整数	垂直同步（VSYNC）脉冲的宽度，以行数为单位
VFP	正整数	垂直前肩的行数
VBP	正整数	垂直后肩的行数
ACB	正整数	（仅用于 STN LCD）AC 偏置频率
IVS	0/1	VSYNC 信号反相(低电平有效)
IHS	0/1	HSYNC 信号反相(低电平有效)
IPC	0/1	像素时钟反相
CPL	正整数	每行的像素时钟数。对于 TFT，与 PPL 相同。对于单色/灰度 STN，等于 PPL /LCD 总线宽度；对于彩色 STN，等于 PPL*3/LCD 总线宽度
BPP	正整数	每像素比特数的代码，用于初始化 LCD 控制器。常用的 RGB565 代码是 6
BGR	0/1	(只用于 TFT)是否交换红色与蓝色

3.2 接口函数介绍

emWin 对这个文件只规定了两个接口函数。尽管只有两个函数，但要完成的工作有很多，并且 emWin 未规定实现标准。NXP 提供的开发包已经实现了这些函数。
注: 所有在移植时需要实现的函数均以 LCD_X 或 GUI_X 开头。

3.2.1.1 LCD_X_Config 函数 （无需修改）

emWin 会在初始化期间调用此函数。此函数需要创建并连接设备对象。另外，emWin 内置了若干种 LCD 控制器的模型并一一提供驱动程序，这个函数还需要为选定的 LCD 控制器模型驱动程序调用相应的初始化配置函数。NXP 的参考实现是针对带有 LCD 控制器的 NXP MCU 的，选用的模型驱动程序是”GUIDRV_Lin_”开头的一系列变体。

3.2.1.2 LCD_X_DisplayDriver 函数

emWin 在初始化期间会多次以不同的命令码调用这个函数，它要做的就是响应命令码。必须响应的只有

LCD_X_INITCONTROLLER: 初始化 LCD 以及触摸屏控制器。LCD 控制器统一使用 MCU 片载的（各型号间的接口相同），初始化代码通常无需修改；但触摸屏控制器若没有使用片载的 A/D 转换器，则需要根据实际选用的来适配代码。NXP 的参考实现编写了_InitController()函数，在响应此命令码时只调用了这个函数。

LCD_X_SETORG: 设置帧缓存的起始地址。无需修改

3.3 初始化 LCD 与触摸屏

在 NXP 的参考实现中，初始化的时机是：

GUI_Init ->

用户程序调用 GUI_Init()初始化 emWin

emWin 在初始化期间以 LCD_X_INITCONTROLLER 命令码调用 LCD_X_DisplayDriver()函数

调用_InitController()函数来响应命令码

调用_**InitLCD()**函数来初始化控制器

调用_**InitTouch()**函数来初始化触摸屏

3.3.1 _InitLCD()函数

_InitLCD 函数初始化 MCU 的片载 LCD 控制器。如上所述，LPC17xx, LPC18xx, LPC43xx, LPC24xx 共享相同的 LCD 控制器。但是由于 LCD 子板设计的区别，如背光控制，以及额外要求的初始化顺序等，这个函数可能还要做少量修改。因此，NXP 的参考实现把初始化分为两步。

第一步，在禁用 MCU 片载 LCD 控制器的情况下分配引脚并初始化 LCD 控制器，这部分的代码无需改动。

第二步，这也是可能要改动的，初始化 LCD 子板，包括打开背光等，最后使能 MCU 片载的 LCD 控制器。LCD 子板的初始化取决于硬件，在移植时也应尽量寻找子板供应商提供的参考实现。

3.3.2 _InitTouch()函数

_InitTouch()函数用来初始化触摸屏控制器并提供校准信息。在不同的实现中，有可能在上一步初始化 LCD 子板时也同时初始化了触摸屏控制器。因此，这个函数中必然含有的只是无需修改的公用的内容，包括

- 调用 emWin 的 GUI_TOUCH_SetOrientation()函数来设置触摸屏的方位，为简单起见通常设置为 0，即没有旋转，而是在提供参考的校准坐标中应用方位信息。
- 分两次调用 emWin 的 GUI_TOUCH_Calibrate()函数以分别提供 X 和 Y 方向的校准值。
- emWin 在运行期间会定期扫描触摸屏，在这里初始化一个定时器以提供时钟源，并在它的中断服务例程中完成扫描工作。参考实现使用了定时器 1。

与硬件相关的触摸屏初始化则需要自行实现，推荐只使用一个接口函数并且在初始化定时器之前调用它。

3.3.3 GUI_X_Init()函数

emWin 在初始化硬件之前，还会调用一个名为 GUI_X_Init()的函数，参考实现中它是空的，也可视需要自行添加内容，此函数在附录中也有介绍。

4. 集成 emWin 到执行环境中

emWin 可以在多种环境下执行：

- 前后台系统

- 多任务环境下单个任务调用 emWin
- 多个任务调用 emWin

emWin 还有一个方便好用的 GUI_Delay()函数，可用于所有的执行环境中。

4.1 使 GUI_Delay()函数工作

GUI_Delay()的使用十分方便，并且在 emWin 的内部也常常在多种场合下使用它来定期更新图形。GUI_Delay()函数在被调用时，会在内部调用 GUI_Exec()，如果执行 GUI_Exec()耗时还不到指定的延时时间，则 GUI_Delay()会继续等待直到延时时间已到期。由此可见，这个函数是“阻塞式”的：如果未使用 RTOS，在 GUI_Delay()执行期间 CPU 无法处理其它任务。如果使用了 RTOS，则当前任务的在这里空转，不影响其它更高优先级的任务，但更低优先级的任务则会被累及。

为使 GUI_Delay()可以工作，需要实现它依赖的几个函数，如下所示：

4.1.1.1 int GUI_X_GetTime(void)

调用顺序: GUI_Delay -> GUI_GetTime -> 本函数。GUI_Delay()是我们可以直接使用的 emWin 接口函数。事实上，在 emWin 内部的定时器机制，窗口管理器的窗口特效功能（如移动，淡入等），以及一些控件的动画功能也会最终调用本函数。

完成功能: 获取系统当前已开机的毫秒数。在实现中一般使用一个定时器以固定频率产生中断，作为时基。

4.1.1.2 void GUI_X_Delay(int ms)

调用顺序: GUI_Delay -> _Delay -> 本函数。

完成功能: 完成真正的延时功能。在前后台系统中可使用一循环来不断读取当前系统时间直到已经过要求的间隔。如果有 RTOS 支持则一般是调用 RTOS 提供的任务延时服务。

4.1.1.3 void GUI_X_ExecIdle(void)

弃用，不做任何事直接返回即可。

以上函数在附录中也有介绍。

4.2 集成到前后台系统

在前后台系统中，没有使用 RTOS，应用程序逻辑在一个无穷的主循环中不停地轮转调用，这是最简单的情况。在这类系统中，定义 GUI_OS 宏为 0，并且在主循环中执行 GUI_Exec()函数。如果仍然希望 emWin 在执行一段时间后返回，可使用 GUI_Delay()函数。但是这个函数返回前无法处理主循环中的其它工作！

4.3 集成到多任务系统——单个任务调用 emWin

对 emWin 来说，这与前后台系统并无二致，对系统而言，只是使用任务的主循环来替代原来后台的主循环。在任务的主循环中使用 GUI_Delay()函数只会使这个任务在返回前无法处理其它工作，不会影响系统的其它方面，因此使用起来更加得心应手。这种情况下，仍然定义 GUI_OS 宏为 0。

与图形有关的操作通常不具备实时性——花费数百毫秒来渲染一帧画面是常有的事，因此务必把调用 emWin 的任务的优先级调低，不要干扰到实时任务。

4.4 集成到多任务系统——多个任务调用 emWin

这与前两种情况有本质的区别，需要定义 GUI_OS 为 1 并且使用 GUITASK_SetMaxTask()设置最大任务数。除此之外，需要实现 emWin 与 RTOS 的接口函数以实现任务间互斥地调用 emWin。它们包括：

4.4.1 void GUI_X_InitOS(void)

调用顺序: GUI_Init -> GUITASK_Init -> 本函数。

完成功能: 创建任务互斥对象，在 RTOS 中常常被称为“互斥体”或“互斥型信号量”。这个互斥对象用于在多个任务调用 emWin 的任务间执行互斥。

4.4.2 void GUI_X_Lock(void)

调用顺序: 几乎在 emWin 的所有函数中被最终调用。
完成功能: 锁住互斥对象。

4.4.3 void GUI_X_Unlock(void)

调用顺序: 几乎在所有函数中被最终调用。
完成功能: 解锁互斥对象。

4.4.4 U32 GUI_X_GetTaskId(void)

调用顺序: 未知。
完成功能: 获取当前任务的 ID 号。

注：以上函数在附录中也有介绍。
还有一个要强调的是，GUI_Delay()函数是让 emWin()响应事件、处理例行事务的关键入口，为避免混乱，应仅从一个任务调用这个函数。在系统中内存富余的情况下，甚至可以专门开设一个任务只循环调用 GUI_Delay()一件事。

4.5 可选实现的函数

以下功能在移植时并不是必须要实现的，但是实现它们可以改善性能，或启动调试功能。

4.5.1 （可选）利用 RTOS 的事件机制改善性能

涉及到事件发送与事件等待函数，这两个函数需要相互配合完成工作。因此，要么一个也不实现，要么双双实现。

4.5.1.1 void GUI_X_WaitEvent(void)

emWin 经常需要等待键盘以及触摸屏的输入，主要用于编辑框等待键盘输入，以及显示对话框。这需要在一个循环中等待，每循环一次调用一次 GUI_Exec()函数以保持 emWin 正常运作。在循环等待期间，调用 emWin 的任务，或者是主循环(对于前后台系统)无法处理其它工作，系统也不可能进入待机状态。为改善性能，emWin 支持与事件相关的等待与发送函数——这里的事件通常与 RTOS 中的“事件”有一致的含义。如果移植时实现了本函数，emWin 就会在循环等待期间调用事件等待函数。我们可以充分利用 RTOS 提供的事件机制来与之对接。对于前后台系统，也可以写这样一个函数作为占位，并在其中执行一些需例行处理的工作。
为使 emWin“知道”移植时实现了本函数，需要执行 GUI_SetWaitEventFunc(GUI_X_WaitEvent)来注册，注册的时机最好是在 GUI_X_Init()中。
这个函数还有一个改进版本，可以指定等待事件的最久时间(毫秒数)：GUI_X_WaitEventTimed(int ms)
当然，是否以它的改进版本来实现，也是可选的。

4.5.1.2 void GUI_X_SignalEvent(void)

如果实现并注册了本函数，则 emWin 在检测到键盘与触摸屏时，以及在无效化窗口时，会调用本函数。正确的实现应使正在等待事件的任务因此而就绪。RTOS 的发送事件服务往往要求指定接收事件的任务，应在执行 GUI_X_WaitEvent()时记录当时的任务(RTOS 都会提供获取当前任务指针的服务)。如果有多个任务调用 emWin，则最好广播此事件给所有相关的任务。

4.5.2 (可选)输出调试信息

下列函数用于调试目的，如果实现了它们，就可以查看 emWin 输出的日志、警告，以及错误信息。常见的实现方式是使用一个串口来输出，并且在 PC 上用终端软件来接收。

4.5.2.1 void GUI_X_Log (const char *s)
4.5.2.2 void GUI_X_Warn (const char *s)
4.5.2.3 void GUI_X_ErrorOut(const char *s)

这里略过“启动最小系统”（包括 SDRAM）代码对应的文件，只列出与 emWin 有关的移植代码所在的文件,并以文件为单位一一介绍。

GUIConf.h: 对 emWin 功能的裁剪。

GUIConf.c: 为 emWin 分配内存。

LCDConf.c: LCD 与触摸屏的参数配置，以及与硬件相关的驱动代码。

LCDConf.h: 自行定义。

GUI_X.c: emWin 与执行环境集成的适配代码。

5. 附录 1

移植 emWin 时，所有需实现（或可选实现）的函数均以 GUI_X 或 LCD_X 开头。这些函数在本文中的不同位置介绍。为方便读者参考，本附录把它们归纳起来。

5.1 以 GUI_X 开头的函数

这些函数是 emWin 与移植的接口函数，它们全部位于 GUI_X.c 中。并非每个函数都需要，视具体的软硬件环境而定。

5.1.1 基本函数

5.1.1.1 void GUI_X_Config(void)

调用顺序: GUI_Init -> GUI__Config -> 本函数

完成功能:

GUI_ALLOC_AssignMemory(_aMemory, GUI_NUMBYTES);

GUI_ALLOC_SetAvBlockSize(GUI_BLOCKSIZE); 如果应用要使用含有大量条目的列表视图，则可将块平均尺寸设置为较小的值。另一方面，如果应用主要将存储器管理用于一些存储器设备或图像解压，则应把平均尺寸设为较大的值。建议范围为 32 至 1024。

5.1.1.2 void GUI_X_Init(void)

调用顺序: GUI_Init -> 本函数

完成功能: emWin 在处理了配置信息后，初始化图形硬件之前调用这个函数。本意是完成一些附加的初始化操作。但一般我们都在系统的基本功能初始化完毕后再调用 GUI_Init()函数，所以这个函数往往为空。如果实现了可选的事件管理功能（见下文介绍），也可以把等待事件与发送事件的函数在这里注册。

5.1.2 时间管理相关函数

在 emWin 中，常常在多种场合下使用 GUI_Delay()来定期更新图形。如果未使用 RTOS，在 GUI_Delay()执行期间 CPU 无法处理其它任务。如果使用了 RTOS，则只有当前任务的执行被暂停，不影响其它任务。

5.1.2.1 int GUI_X_GetTime(void)

调用顺序: GUI_Delay -> GUI_GetTime -> 本函数。GUI_Delay()是我们可以直接使用的 emWin 接口函数。事实上，在 emWin 内部的定时器机制，窗口管理器的窗口特效功能（如移动，淡入等），以及一些控件的动画功能也会最终调用本函数。

完成功能: 获取系统当前已开机的毫秒数。在实现中一般使用一个定时器以固定频率产生中断，作为时基。

5.1.2.2 void GUI_X_Delay(int ms)

调用顺序: GUI_Delay -> _Delay -> 本函数。

完成功能: 完成真正的延时功能。在前后台系统中可使用一循环来不断读取当前系统时间直到已经过要求的间隔。如果有 RTOS 支持则一般是调用 RTOS 提供的任务延时服务。

5.1.2.3 void GUI_X_ExecIdle(void)

弃用，不做任何事直接返回即可。

5.1.3 支持多任务的相关函数

以下函数仅当 GUI_OS 宏定义为 1 时才被条件编译——此时，可以有多个任务调用 emWin。在实现时，需使用 RTOS 的接口函数。

5.1.3.1 void GUI_X_InitOS(void)

调用顺序: GUI_Init -> GUITASK_Init -> 本函数。
完成功能: 创建任务互斥对象，在 RTOS 中常常被称为“互斥体”或“互斥型信号量”。这个互斥对象用于在多个任务调用 emWin 的任务间执行互斥。

5.1.3.2 void GUI_X_Lock(void)

调用顺序: 几乎在 emWin 的所有函数中被最终调用。
完成功能: 锁住互斥对象。

5.1.3.3 void GUI_X_Unlock(void)

调用顺序: 几乎在所有函数中被最终调用。
完成功能: 解锁互斥对象。

5.1.3.4 U32 GUI_X_GetTaskId(void)

调用顺序: 未知。
完成功能: 获取当前任务的 ID 号。

5.1.4 （可选）利用 RTOS 的事件机制改善性能

涉及到事件发送与事件等待函数，这两个函数需要相互配合完成工作。因此，要么一个也不实现，要么双双实现。

5.1.4.1 void GUI_X_WaitEvent(void)

emWin 经常需要等待键盘以及触摸屏的输入，主要用于编辑框等待键盘输入，以及显示对话框。这需要在一个循环中等待，每循环一次调用一次 GUI_Exec()函数以保持 emWin 正常运作。在循环等待期间，调用 emWin 的任务，或者是主循环(对于前后台系统)无法处理其它工作，系统也不可能进入待机状态。为改善性能，emWin 支持与事件相关的等待与发送函数——这里的事件通常与 RTOS 中的“事件”有一致的含义。如果移植时实现了本函数，emWin 就会在循环等待期间调用事件等待函数。我们可以充分利用 RTOS 提供的事件机制来与之对接。对于前后台系统，也可以写这样一个函数作为占位，并在其中执行一些需例行处理的工作。
为使 emWin“知道”移植时实现了本函数，需要执行 GUI_SetWaitEventFunc(GUI_X_WaitEvent)来注册，注册的时机最好是在 GUI_X_Init()中。
这个函数还有一个改进版本，可以指定等待事件的最久时间(毫秒数)：
GUI_X_WaitEventTimed(int ms)
当然，是否以它的改进版本来实现，也是可选的。

5.1.4.2 void GUI_X_SignalEvent(void)

如果实现并注册了本函数，则 emWin 在检测到键盘与触摸屏时，以及在无效化窗口时，会调用本函数。正确的实现应使正在等待事件的任务因此而就绪。RTOS 的发送事件服务往往要求指定接收事件的任务，应在执行 GUI_X_WaitEvent()时记录当时的任务(RTOS 都会提供获取当前任务指针的服务)。如果有多个任务调用 emWin，则最好广播此事件给所有相关的任务。

5.1.5 (可选)输出调试信息

下列函数用于调试目的，如果实现了它们，就可以查看 emWin 输出的日志、警告，以及错误信息。常见的实现方式是使用一个串口来输出，并且在 PC 上用终端软件来接收。

5.1.5.1 void GUI_X_Log (const char *s)

5.1.5.2 void GUI_X_Warn (const char *s)

5.1.5.3 void GUI_X_ErrorOut(const char *s)

5.2 以 LCD_X 开头的函数

这些函数全部位于 LCDConf.c 中。NXP 的参考实现也把驱动物理设备的代码放到了这个文件中。如果目标板较复杂，为使目录结构清晰，也可以另设专门的目录来存储这些硬件相关的代码，而在这个文件中只调用它们。

5.2.1 LCD_X_Config 函数

emWin 会在初始化期间调用此函数。此函数需要创建并连接设备对象。另外，emWin 内置了若干种 LCD 控制器的模型并一一提供驱动程序，这个函数还需要为选定的 LCD 控制器模型驱动程序调用相应的初始化配置函数。NXP 的参考实现是针对带有 LCD 控制器的 NXP MCU 的，选用的模型驱动程序是"GUIDRV_Lin_"开头的一系列变体。

5.2.2 LCD_X_DisplayDriver 函数

emWin 在初始化期间会多次以不同的命令码调用这个函数，它要做的就是响应命令码。必须响应的只有

LCD_X_INITCONTROLLER: 初始化 LCD 以及触摸屏控制器。LCD 控制器统一使用 MCU 片载的（各型号间的接口相同），初始化代码通常无需修改；但触摸屏控制器若没有使用片载的 A/D 转换器，则需要根据实际选用的来适配代码。NXP 的参考实现编写了_InitController()函数，在响应此命令码时只调用了这个函数。

LCD_X_SETORG: 设置帧缓存的起始地址。无需修改

```
SystemInit
    Clock, Power, Pin
    SDRAM
AppLogicInit
GUI_Init
    GUI_X_Config
    LCD_X_Config
```

6. 附录 2，参考的目录结构

BSP_<目标板的名字>/	硬件相关的驱动代码
Chip/	芯片相关，包括 Cortex-M 内核与片上外设的驱动，一般地把 CMSIS 库中的相关文件拷贝进来即可。
Board/	目标板相关，存储板上设备的驱动程序。建议把 LCD 子板的驱动实现也放在这里。可包含子目录，尤其是当引用了现有的代码时。
emWin/	emWin 的全部头文件(.h)与库文件(.lib)
Port/	移植到本目标板上的文件。如果目标板可连接不同类型的 LCD 子板，推荐使用宏定义来选择使用哪种。
GUIConf.c	
GUIConf.h	
LCDConf.c	
LCDConf.h	
GUI_X.c	
<OS 名>/	如果系统中使用了嵌入式操作系统，把相关目录与文

	件放到这里
<其它基础软件，如 FS, Network 等>/	如果系统中还使用了其它基础软件，如文件系统，USB，网络协议栈等，可一一放置

7. Legal information

7.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

7.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

7.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

7.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP B.V.

8. Index

No index entries found.

9. List of figures

No table of figures entries found.

10. List of tables

No table of figures entries found.