

## 引言

意法半导体与Segger微控制器合作提供STemWin库，这是一个基于Segger的emWin图形库产品。

STemWin库是一个专业级的图形栈库，在条件允许时该库可以利用STM32硬件加速功能，在任意STM32产品、任意LCD/TFT显示屏以及任意 LCD/TFT 控制器随时构建图形用户界面（GUI）。

STemWin库是一个功能丰富的全面解决方案，配备诸如JPG、GIF和PNG解码和众多小工具（复选框、按钮等等），以及VNC服务器，它允许远程显示本地画面，而且还包含GUIBuilder等通过鼠标拖放来创建对话框，以及字体转换器等等。

该图形库完全集成在STM32Cube固件包内（比如STM32CubeF2、STM32CubeF3和STM32CubeF4）。可在意法半导体的网站上免费下载 (<http://www.st.com/stm32cube>)

表1. 实用软件

类型	产品编号
MCU嵌入式软件	STemWin、STM32CubeF0、STM32CubeF1、STM32CubeF2、STM32CubeF3、STM32CubeF4、STM32CubeF7、STM32CubeL1、STM32CubeL4和STM32CubeH7

**目录**

<b>1</b>	<b>一般信息</b>	<b>6</b>
<b>2</b>	<b>库和包的说明</b>	<b>6</b>
2.1	许可信息	6
2.2	库的说明	6
2.3	库包组织	9
2.4	交付的二进制文件	9
<b>3</b>	<b>支持的板卡和示例</b>	<b>12</b>
<b>4</b>	<b>如何按部就班地使用 STemWin 库</b>	<b>14</b>
4.1	配置	14
4.1.1	GUIConf.c	15
4.1.2	LCDConf.c	15
4.1.3	硬件加速	15
4.1.4	GUI_X.c 或 GUI_X_OS.c	17
4.2	GUI初始化	17
4.3	核心函数	17
4.3.1	图像文件的显示	17
4.3.2	双向文本	18
4.3.3	Alpha 混合	18
4.3.4	游标和光标	18
4.4	内存设备	19
4.5	抗锯齿	20
4.6	窗口管理器	20
4.7	小工具程序库	20
4.8	VNC 服务端	21
4.8.1	要求	22
4.8.2	过程说明	23
4.9	字体	24
4.9.1	STemWin 库中包含的字体	24
4.9.2	如何添加新字体	27
4.9.3	语言支持	33

4.10	GUIBuilder .....	37
4.10.1	GUIBuilder 的基本使用 .....	38
4.10.2	创建程序 .....	38
4.10.3	用户自定义代码 .....	38
4.10.4	回调程序 .....	38
4.11	以不同的分辨率移植 GUI .....	39
4.12	如何添加新效果 .....	39
4.13	如何添加新的小部件 .....	40
<b>5</b>	<b>性能与内存开销 .....</b>	<b>41</b>
5.1	LCD 驱动程序性能 .....	41
5.2	STemWin 内存开销 .....	42
<b>6</b>	<b>FAQ（常见问题） .....</b>	<b>44</b>
<b>7</b>	<b>版本历史 .....</b>	<b>45</b>

表格索引

表1. 实用软件 ..... 1

表2. 所支持的 LCD 控制器 ..... 8

表3. 支持的板卡和示例 ..... 12

表4. 硬件加速API列表 ..... 16

表5. 字体 API ..... 26

表6. 速度测试列表 ..... 41

表7. FlexColor 和 Lin 驱动程序的速度测试 ..... 41

表8. 模块的内存开销 ..... 42

表9. 小工具内存开销 ..... 43

表10. FAQs ..... 44

表11. 文档版本历史 ..... 45

表12. 中文文档版本历史 ..... 45



## 图片索引

图1.	STemWin 结构层级	7
图2.	项目树	9
图3.	STemWin的应用结构	13
图4.	STemWin初始化	14
图5.	Alpha 混合效果	18
图6.	动画游标	18
图7.	光标	19
图8.	利用 memdev 实现缩放和旋转效果	19
图9.	图形抗锯齿	20
图10.	小工具示例	21
图11.	VNC 服务端的用途	22
图12.	VNC 客户端	23
图13.	字体转换器	28
图14.	字体类型和编码选项	29
图15.	字体、样式和大小选择	29
图16.	创建字体C文件	30
图17.	禁用/启用字体字符	31
图18.	字符范围选择	31
图19.	创建应用的不同文本	33
图20.	读取模式文件菜单	34
图21.	缺少所使用字体的字符代码	34
图22.	已启用模式字符	35
图23.	创建UTF-8模式	36
图24.	转换 UTF-8 模式为 "C" 代码	36
图25.	GUIBuilder 应用	38

## 1 一般信息

STemWin支持基于Arm<sup>®(a)</sup>的器件。



## 2 库和包的说明


STemWin库包由一套固件库程序以及用于构建基于GUI的高级专业应用程序的软件工具所组成。

### 2.1 许可信息

- ◆ STemWin库的GUI文件为对象格式，依照MCD-ST图像软件许可协议V2（“许可证”）进行授权；必须遵守该许可证的规定，才能使用这套库包。可从以下网址获取许可证副本：[www.st.com](http://www.st.com)。
- ◆ STemWin库的配置与头文件均为源码格式，依照MCD-ST自由软件许可协议V2（“许可证”）进行授权；必须遵守该许可证的规定，才能使用这套库包。可从以下网址获取许可证副本：[www.st.com](http://www.st.com)。

除非适用法律或书面协议的要求，否则在没有任何明示或暗示的担保或条款下，受到该许可证约束的软件必须“原样”发布。关于许可证所约束的具体语言管理权限和限制，请参见许可证的内容。

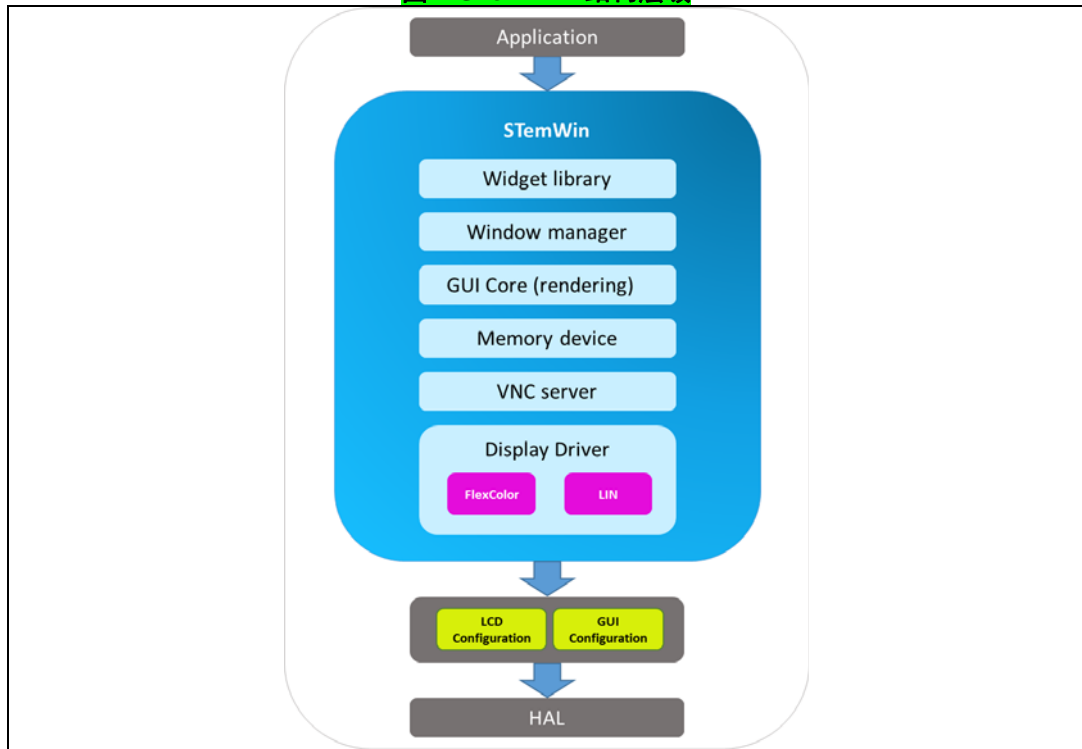
### 2.2 库的说明

 显示了STemWin的内部结构以及在整个项目中如何实现。

---

a. Arm是Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

图1. STemWin 结构层级



1. 使用该库之前，应启用 CRC 模块（在 RCC 外设时钟使能寄存器中）。

STemWin 库包含两个经过优化的驱动程序：

- ◆ 直接线性访问（LIN）驱动器。此类驱动器用于STM32F429、STM32F769、STM32H743 和任何基于LCD-TFT显示控制器（LTDC）或LTDC/DSI（显示屏串行接口）硬件的STM32。
- ◆ FlexColor（间接访问）驱动程序，适用于所有串行和并行总线外部LCD控制器。

关于所支持的全部显示控制器，请参见 表 2。

**注：** 只要实现“定制化”的驱动程序，仍可支持其它任意类型的 LCD。

除了主应用之外，用户必须设置和定制两个关键的接口文件：

- ◆ LCD 配置文件（LCDConf.c）
  - LCD 显示初始化和配置
  - LCD 显示驱动程序链接和定制化
  - 额外的硬件能力管理
- ◆ GUI 配置文件（GUIConf.c）
  - GUI管理

表2. 所支持的 LCD 控制器

驱动	所支持的 LCD 控制器	所支持的位数/像素
GUIDRV_Lin	该驱动程序支持每一款带有直接（全总线）接口线性可寻址视频存储器的显示控制器。这意味着视频RAM可通过CPU的地址线直接寻址。 驱动程序不含具体控制器的代码。所以，驱动程序也可用于无显示控制器，而仅须驱动程序来管理视频 RAM 的解决方案。	1、2、4、8、16、24、32
GUIDRV_FlexColor	<b>Epson S1D19122</b> <b>FocalTech FT1509</b> <b>Himax HX8353、HX8325A、HX8357、HX8340、HX8347、HX8352A、HX8352B、HX8301</b> <b>Hitachi HD66772</b> <b>Ilitek ILI9320、ILI9325、ILI9328、ILI9335、ILI9338、ILI9340、ILI9341、ILI9342、ILI9163、ILI9481、ILI9486、ILI9488、ILI9220、ILI9221</b> <b>LG Electronics LGDP4531、LGDP4551、GDP4525</b> <b>Novatek NT39122</b> <b>OriseTech SPFD5408、SPFD54124C、SPFD5414D</b> <b>Raio RA8870、RA8875</b> <b>Renesas R61505、R61516、R61526、R61580</b> <b>Samsung S6E63D6、S6D0117</b> <b>Sitronix ST7628、ST7637、ST7687、ST7735、ST7712</b> <b>Solomon SSD1284、SSD1289、SSD1298、SSD1355、SSD2119、SSD1963、SSD1961、SSD1351</b> <b>Syncoam SEPS525</b>	16, 18

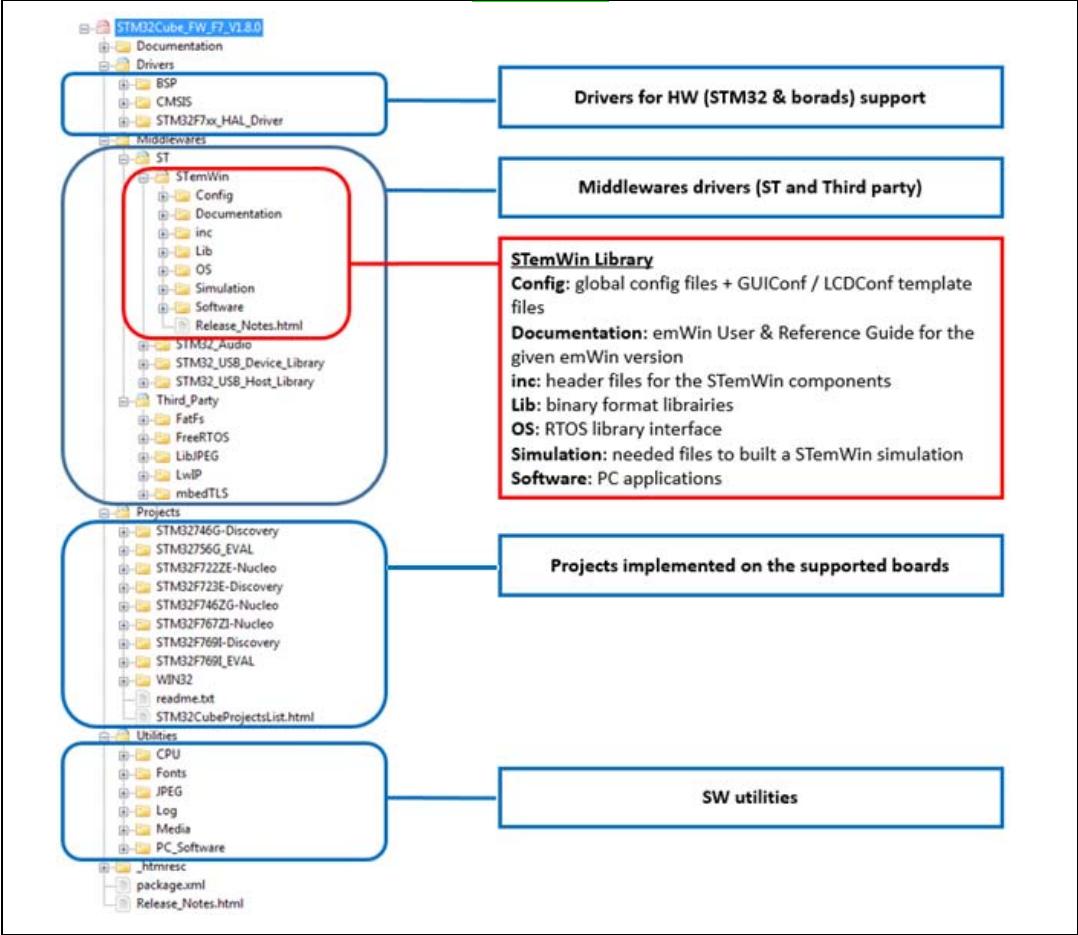


# 2.3

# 库包组织

图 2 显示了项目树结构。

图2. 项目树



# 2.4

# 交付的二进制文件

由意法半导体所发布的 STemWin 库作为对象代码库绑定于 STM32 产品。

该库编译用于CM0和CM3内核：

- ◆ 支持和不支持OS。
- ◆ 具有空间优化功能
- ◆ 使用 ABGR 和 ARGB 格式

该库编译用于 CM4 和 CM7 内核：

- ◆ 支持和不支持OS。
- ◆ 具有空间和时间优化功能
- ◆ 使用 ABGR 和 ARGB 格式
- ◆ FPU启动。

注：

该库依赖于编译器（IAR、Arm、GCC）。



该库同时编译用于WIN32模拟

- ✦ 使用 ABGR 和 ARGB 格式

**注：** 在相对于给定内核的STM32CubeFW中，只有为此内核生成的二进制文件和模拟内容包含在包中。

总而言之，生成74个二进制文件并拆分如下：

对于CMx（CM0和CM3）内核，有12个二进制文件：

- ✦ STemWinXY\_CMx\_GCC.a
- ✦ STemWinXY\_CMx\_GCC\_ARGB.a
- ✦ STemWinXY\_CMx\_IAR.a
- ✦ STemWinXY\_CMx\_IAR\_ARGB.a
- ✦ STemWinXY\_CMx\_Keil.lib
- ✦ STemWinXY\_CMx\_Keil\_ARGB.lib
- ✦ STemWinXY\_CMx\_OS\_GCC.a
- ✦ STemWinXY\_CMx\_OS\_GCC\_ARGB.a
- ✦ STemWinXY\_CMx\_OS\_IAR.a
- ✦ STemWinXY\_CMx\_OS\_IAR\_ARGB.a
- ✦ STemWinXY\_CMx\_OS\_Keil.lib
- ✦ STemWinXY\_CMx\_OS\_Keil\_ARGB.lib

对于CMx（CM4和CM7）内核，有24个二进制文件：

- ✦ STemWinXY\_CMx\_GCC.a
- ✦ STemWinXY\_CMx\_GCC\_ARGB.a
- ✦ STemWinXY\_CMx\_GCC\_ot.a
- ✦ STemWinXY\_CMx\_GCC\_ot\_ARGB.a
- ✦ STemWinXY\_CMx\_IAR.a
- ✦ STemWinXY\_CMx\_IAR\_ARGB.a
- ✦ STemWinXY\_CMx\_IAR\_ot.a
- ✦ STemWinXY\_CMx\_IAR\_ot\_ARGB.a
- ✦ STemWinXY\_CMx\_Keil.lib
- ✦ STemWinXY\_CMx\_Keil\_ARGB.lib
- ✦ STemWinXY\_CMx\_Keil\_ot.lib
- ✦ STemWinXY\_CMx\_Keil\_ot\_ARGB.lib
- ✦ STemWinXY\_CMx\_OS\_GCC.a
- ✦ STemWinXY\_CMx\_OS\_GCC\_ARGB.a
- ✦ STemWinXY\_CMx\_OS\_GCC\_ot.a
- ✦ STemWinXY\_CMx\_OS\_GCC\_ot\_ARGB.a
- ✦ STemWinXY\_CMx\_OS\_IAR.a
- ✦ STemWinXY\_CMx\_OS\_IAR\_ARGB.a
- ✦ STemWinXY\_CMx\_OS\_IAR\_ot.a
- ✦ STemWinXY\_CMx\_OS\_IAR\_ot\_ARGB.a
- ✦ STemWinXY\_CMx\_OS\_Keil.lib
- ✦ STemWinXY\_CMx\_OS\_Keil\_ARGB.lib
- ✦ STemWinXY\_CMx\_OS\_Keil\_ot.lib
- ✦ STemWinXY\_CMx\_OS\_Keil\_ot\_ARGB.lib

接下来，用户添加模拟二进制文件：

- ✦ STemWinXY\_WIN32.lib
- ✦ STemWinXY\_WIN32\_ARGB.lib

3 支持的板卡和示例

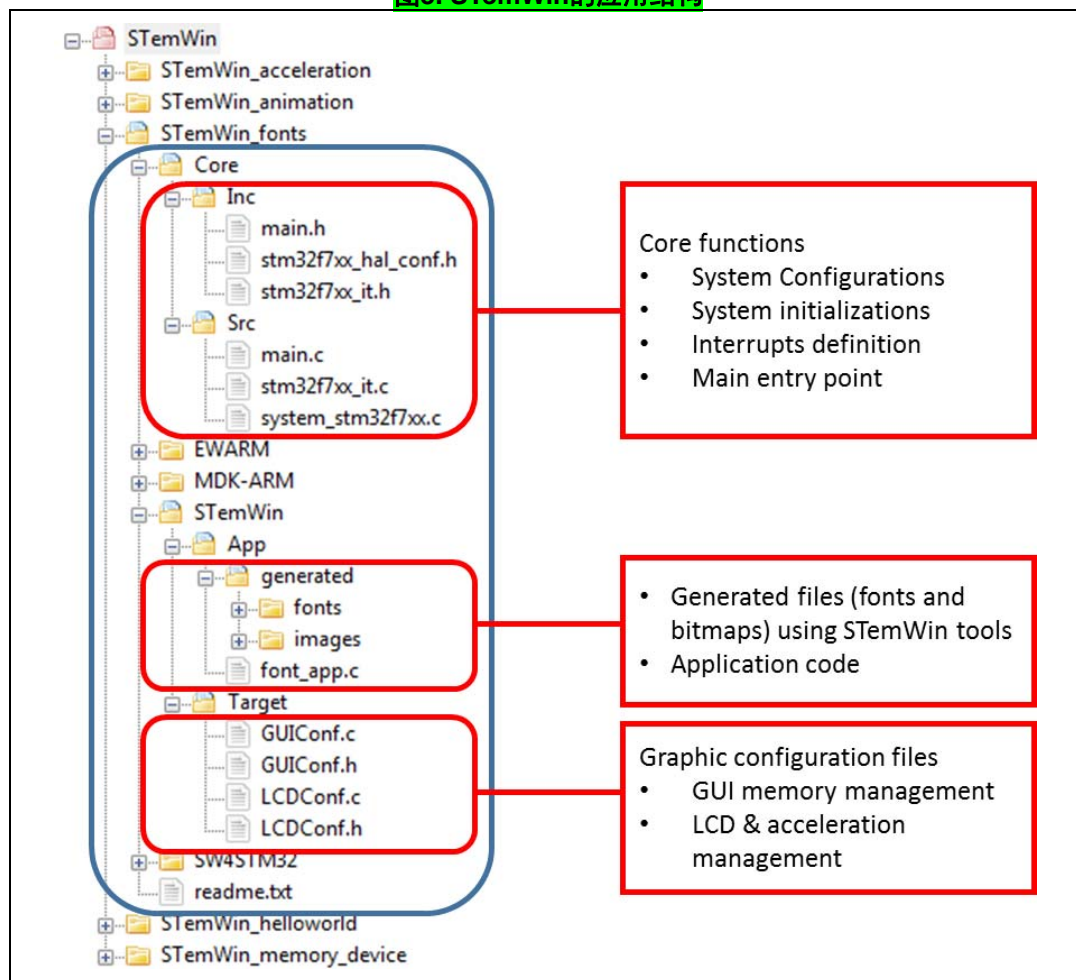
表 3 列出所支持的板卡和示例。

表3. 支持的板卡和示例			
STM32 系列/板卡		LCD接口	LCD驱动程序
STM32F0	STM32091C-EVAL	SPI	FlexColor
STM32F1	STM3210C-EVAL	SPI	FlexColor
	STM3210E-EVAL	FSMC	
STM32F2	STM322xG-EVAL	FSMC	FlexColor
STM32F3	STM32303C-EVAL	SPI	FlexColor
	STM32303E-EVAL	SPI	
	STM32373C-EVAL	SPI	
STM32F4	STM32F412G-DISCO	FMC	FlexColor
	STM32F413G-DISCO	FMC	FlexColor
	STM32F429I-DISCO	LTDC	Lin
	STM32F4x9I-EVAL	LTDC	Lin
	STM32F4xG-EVAL	FSMC	FlexColor
	STM32F446E-EVAL	FMC	FlexColor
	STM32F469I-EVAL	LTDC/DSI	Lin
	STM32F469I-DISCO	LTDC/DSI	Lin
STM32F7	STM32F723E-DISCO	FMC	FlexColor
	STM32F769I-EVAL	LTDC/DSI	Lin
	STM32F769I-DISCO	LTDC/DSI	Lin
	STM32F746G-DISCO	LTDC	Lin
	STM32F756G-EVAL	LTDC	Lin
STM32H7	STM32H743I-EVAL	LTDC	Lin
STM32L1	STM32L152D-EVAL	FSMC	FlexColor
STM32L4	STM32L49I-EVAL-MB1314	GFXMMU/LTDC/DSI	Lin
	STM32L49I-EVAL-MB1315	LTDC	Lin
	STM32L49I-DISCO	GFXMMU/LTDC/DSI	Lin
	STM32L476G-EVAL	FMC	FlexColor
	STM32L496G-DISCO	FMC	FlexColor

图 3 显示给定STemWin应用的架构。



图3. STemWin的应用结构



“Target”文件夹包含在 第 2.2 节所述的两个接口文件：

- **GUIConf.c/h**
- **LCDConf.c/h**

可以使用或不使用OS构建STemWin应用程序。上述应用架构是一个非OS案例。

## 4 如何按部就班地使用 STemWin 库

本节提供STemWin库的主要功能概览，以及主要设置及配置步骤。欲了解更多详细信息，请参见 Segger's emWin 用户手册。

### 4.1 配置

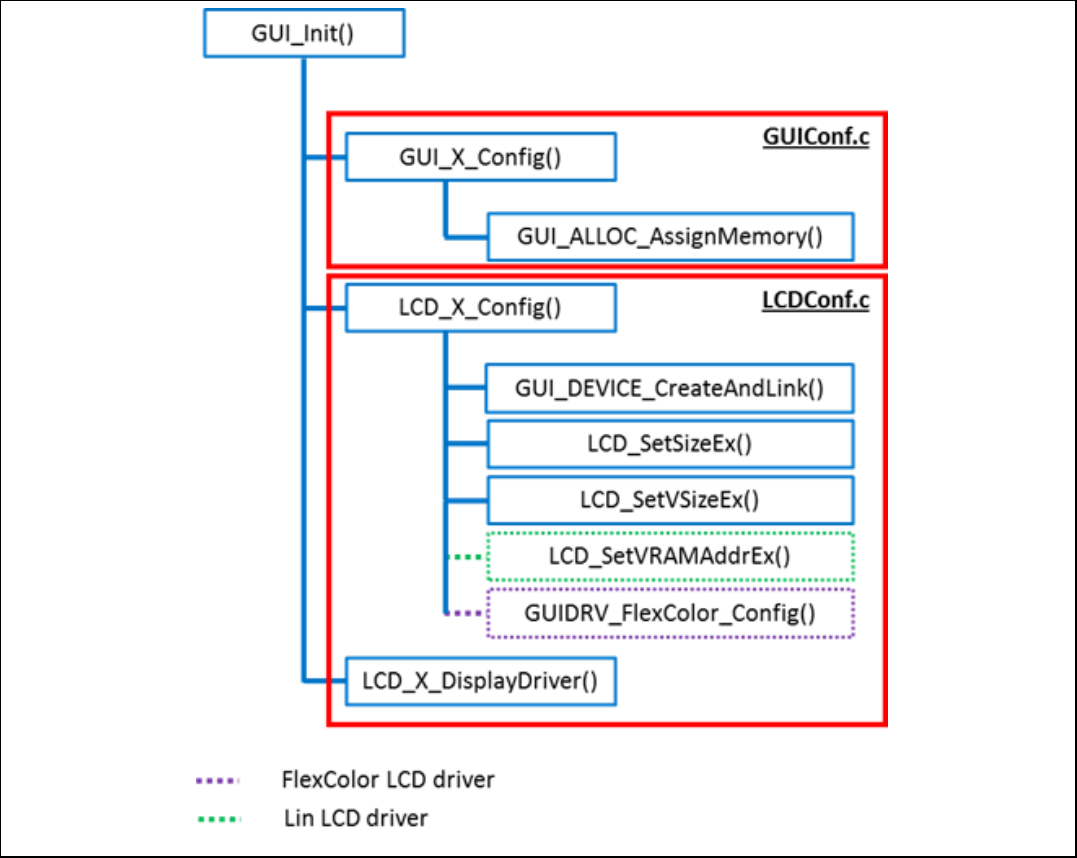
配置基本上分为两个部分：GUI 配置和 LCD 配置。

- GUI 配置包含了默认颜色和字体的配置，以及可用内存的配置。
- LCD配置更依赖于硬件，如果STM32支持Chrom-ART，用户即可定义不同的图形参数、显示驱动程序、要使用的颜色转换例程和绘图加速功能。

当需要支持新的LCD控制器时，必须创建/修改两个重要文件：GUIConf.c和LCDConf.c。

通过调用“GUI\_Init()”函数完成的栈初始化基于如图 4所示的两个文件。

图4. STemWin初始化



### 4.1.1 GUIConf.c

在这个文件中，用户应执行 GUI\_X\_Config() 函数，这是初始化过程中最先调用的程序。这个程序的主要任务是为 GUI 设置可用的内存空间，然后将这段空间分配到动态内存管理系统。

该操作通过 GUI\_ALLOC\_AssignMemory() 函数来完成：它传递一个指针到内存块，以及将它的大小（字节数）传递到内存管理器。

**注：** 内存必须可访问，而且必须是8位、16位、32位位宽。初始化过程中必须对内存访问进行检查。

### 4.1.2 LCDConf.c

该文件中的主要函数是 LCD\_X\_Config()，当 GUI\_X\_Config() 执行完毕后，便会立即调用该函数。LCD\_X\_Config() 可通过调用以下函数为每一层创建和配置显示驱动程序：

- GUI\_DEVICE\_CreateAndLink()，用于创建驱动设备并将其链接到设备链；
- LCD\_SetSizeEx() 和 LCD\_SetVSizeEx()，用于设置显示器尺寸配置；
- 其它针对特定驱动的配置程序，比如：
  - GUIDRV\_FlexColor\_Config()，与 FlexColor 驱动程序的使用相关联；
  - LCD\_SetVRAMAddrEx()，当使用线性可寻址内存时需要该函数。

如 第 2.2 节 所述，STemWin 库带有两个经过优化的驱动程序，GUIDRV\_Lin 和 GUIDRV\_FlexColor，覆盖了意法半导体 EVAL 板卡的所有嵌入式 LCD。

关于这些驱动程序在使用上的更多信息，请访问 Segger 的网站：

<http://www.segger.com>

一般来说，当需要支持新的 LCD 类型时，用户应检查是否已经支持相同的 LCD 控制器，然后必须更新已有的 LCDConf.c。

显示驱动程序还会调用另外一个 LCD\_X\_DisplayDriver() 函数，该函数有多种用途，比如在使用多重缓冲、平滑滚动或虚拟页面时需要用到。为了支持相应的任务，必须针对显示控制器对程序进行改编。

如果不支持所使用的显示控制器，用户只需要改编 Middlewares\ST\STemWin\Config 路径下 CubeFW 目录的 GUIDRV\_Template.c 文件，即可轻松创建个性化的驱动程序。

实际上，这个模板文件包含有显示驱动程序所需要用到的全部功能。需要适配改编的主要程序为 \_SetPixelIndex() 和 \_GetPixelIndex()。如果无法读取显示内容，应利用显示数据缓存来代替对 \_GetPixelIndex() 函数的调用。

### 4.1.3 硬件加速

为了减少 CPU 负荷，加快多项绘图操作，ST 在许多 STM32 产品中加入了硬件加速 IP，即 Chrom-ART。

另一方面，EmWin 提供了可以加速的 API 函数列表。

下表显示了当前在“LCDConf.c”实施的硬件加速 API 函数。

表4. 硬件加速API列表

功能	说明
GUICC_M1555I_SetCustColorConv()	该函数用于为相应的固定调色板模式设置颜色转换例程
GUICC_M565_SetCustColorConv()	
GUICC_M4444I_SetCustColorConv()	
GUICC_M888_SetCustColorConv()	
GUICC_M8888I_SetCustColorConv()	
LCD_DEVFUNC_COPYBUFFER	该函数用于设置复制缓冲区的自定义例程
LCD_DEVFUNC_FILLRECT	该函数用于设置填充矩形的自定义例程
LCD_DEVFUNC_COPYRECT	该函数用于设置复制矩形区的自定义例程
LCD_DEVFUNC_DRAWBMP_8BPP	该函数用于设置绘制8bpp位图的自定义例程
LCD_DEVFUNC_DRAWBMP_16BPP	该函数用于设置绘制16bpp位图的自定义例程
LCD_DEVFUNC_DRAWBMP_32BPP	该函数用于设置绘制32bpp位图的自定义例程
GUI_SetFuncAlphaBlending()	该函数用于设置执行Alpha混合操作的自定义函数
GUI_SetFuncGetpPalConvTable()	该函数用于将位图的调色板转换设置为索引值
GUI_SetFuncMixColorsBulk()	该函数用于设置批量混合操作
GUI_AA_SetpfDrawCharAA4()	该函数用于设置绘制4bpp Alpha字符
GUI_MEMDEV_SetDrawMemdev16bppFunc()	该函数用于将绘制16bpp位图设置为16bpp内存设备。
GUI_SetFuncDrawAlpha()	该函数用于设置2个加速，其中，第一个加速用于将具有alpha值的内存设备绘制到另一个内存设备中，第二个加速用于绘制具有alpha值的位图。

- 备注：
- ◆ 请注意，并非所有STM32产品都支持硬件加速。
  - ◆ 在当前提供的“LCDConf.c”上，尚未实现某些加速API。





#### 4.1.4 GUI\_X.c 或 GUI\_X\_OS.c

- GUI\_X.c 用于单任务执行：

“单任务”是指仅在单一任务中使用 STemWin 的项目。主要用途是为 STemWin 提供时序基础。OS\_TimeMS 须每毫秒递增。

- GUI\_X\_OS.c 用于多任务执行：

如果在多任务系统使用 STemWin，则此文件包含同步任务所需的其他例程。

该文件的所有函数都使用 CMSIS-RTOS API，它是基于 Arm®Cortex®-M 处理器设备的通用 RTOS 接口，为需要 RTOS 功能的软件组件提供标准化 API。

## 4.2 GUI 初始化

为了初始化 STemWin 的内部数据结构和变量，应使用 GUI\_Init()。

请注意，在初始化 GUI 之前，应启用 CRC 模块（在 RCC 外设时钟使能寄存器中）。

如下所示的简单“Hello world”程序显示了这一初始化过程。

### “Hello world”示例：

```
void Main(void) {
    int xPos, yPos;

    __HAL_RCC_CRC_CLK_ENABLE();
    GUI_Init();

    xPos = LCD_GetXSize() / 2;
    yPos = LCD_GetYSize() / 3;
    GUI_SetFont(GUI_FONT_COMIC24B_ASCII);
    GUI_DispStringHCenterAt("Hello world!", xPos, yPos);
    while(1);
}
```

## 4.3 核心函数

### 4.3.1 图像文件的显示

STemWin 目前支持 BMP、JPEG、GIF 和 PNG 等文件格式。程序库中包含有适用于这些格式的丰富 API（关于完整的文档说明，请参见 STemWin 用户手册）。各种图像类型所需的内存资源近似值如 [第 5 节：性能与内存开销](#) 所示。

### 4.3.2 双向文本

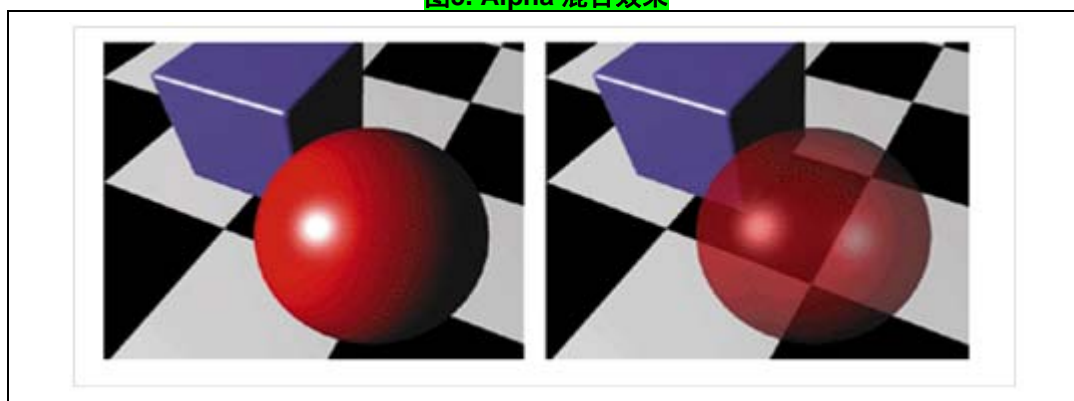
利用 STemWin 绘出阿拉伯文字或希伯来文字是相当容易的事情，每个基于文本的功能均自动支持这些文字。该功能只需使用以下一次性命令即可启用：

```
GUI_UC_EnableBIDI()
```

### 4.3.3 Alpha 混合

Alpha 混合是为了生成半透明效果，而将 alpha 通道与图像的其他图层相组合的一种方法（参见 [图 5](#)）。

图5. Alpha 混合效果



用户可以利用以下命令来启用自动 alpha 混合功能：

```
GUI_EnableAlpha()
```

它还能给出 alpha 值，用于确定像素的可见程度以及背景的显示程度：

```
GUI_SetUserAlpha()
```

### 4.3.4 游标和光标

游标是能够显示在画面中所有其它图形之上的一幅图像。

游标维护它所覆盖的画面区域。它可以随时移动，或者移除以恢复显示画面内容。通过使用多幅图像可以生成动画效果。

图6. 动画游标

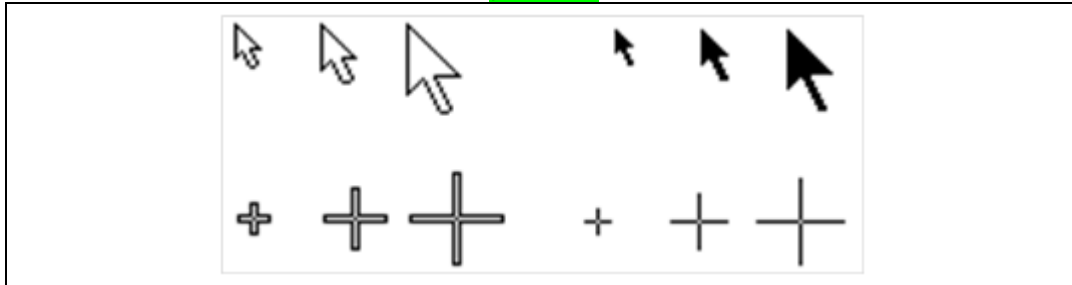


游标可以通过调用 `GUI_SPRITE_CreateAnim()` 而动画化（[图 6](#)）。

请注意，游标可以自动管理背景。

STemWin 还包含一个系统级光标 (图 7)，利用 `GUI_CURSOR_SetAnim()` 也可将光标设为动画。实际上基于子图的光标。

图7. 光标



尽管光标一直存在，但默认处于隐藏状态。直到显示函数被调用之后，光标才变得可见 (`GUI_CURSOR_Show()`)，而当调用 (`GUI_CURSOR_Hide()`) 时，光标会再次隐藏。

## 4.4

### 内存设备

内存设备是绘图操作中独立于硬件的目标设备。

若已创建内存设备（通过调用 `GUI_MEMDEV_Create()`）并生效（通过调用 `GUI_MEMDEV_Select()`）则所有的绘图操作均在内存中执行。仅当完成所有操作之后，才会在画面上显示最终结果。该动作通过调用 `GUI_MEMDEV_CopyToLCD()` 来完成。

内存设备可用于：

- ◆ 避免（向显示器直接绘图而产生的）闪烁效果，
- ◆ 作为解压图像的容器，
- ◆ 用于旋转操作（`GUI_MEMDEV_Rotate()`）和缩放操作（图 8），
- ◆ 用于淡入淡出效果，
- ◆ 用于窗口动画，
- ◆ 用于透明效果。

图8. 利用 memdev 实现缩放和旋转效果



由于内存设备需要使用大量内存空间（参见 表 8 中的“内存设备”组件），如果有条件的话建议使用外部存储器。

## 4.5 抗锯齿

抗锯齿功能通过将背景颜色与前景颜色相“混合”的方法使曲线和斜线变得平滑。“混合”的作用是在对象与背景之间添加中间颜色来实现的。

### 图形抗锯齿

STemWin 支持绘制以下抗锯齿图形：

- 文字（需要字体转换器来创建 AA 字体）
- 弧形（GUI\_AA\_DrawArc()）
- 圆形（GUI\_AA\_FillCircle()）
- 线条（GUI\_AA\_DrawLine()）
- 多边形（GUI\_AA\_DrawPolyOutline() 和 GUI\_AA\_FillPolygon()）

图9. 图形抗锯齿



## 4.6 窗口管理器

窗口管理器的说明：

- 用于分层窗口结构的管理系统：
  - 每一层都有专属的桌面窗口。每个桌面窗口都具有用于子窗口的分层结构树。
- 基于回调机制的系统：
  - 通信功能基于事件驱动的回调机制。所有绘图操作均应在WM\_PAINT事件中完成。
- 小工具程序库的基础：
  - 所有小工具均是基于窗口管理器的功能。

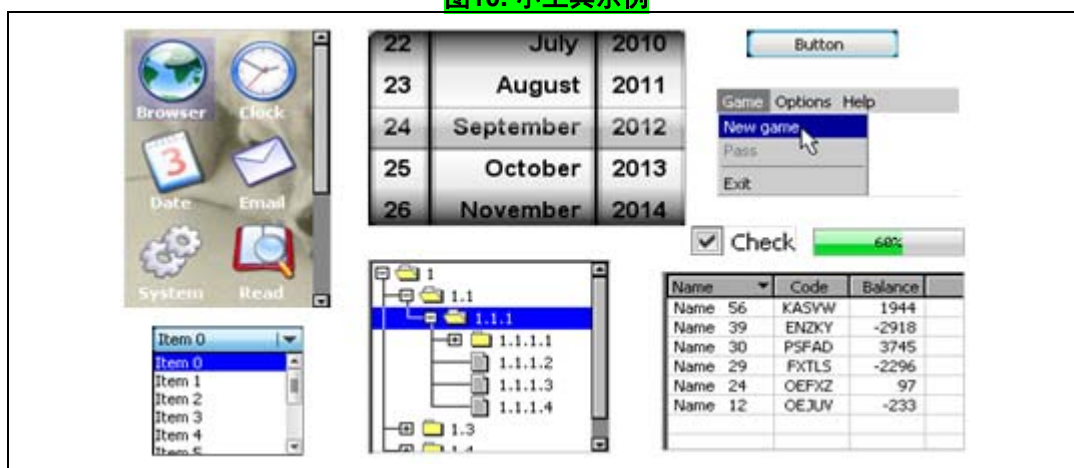
## 4.7 小工具程序库

小工具（窗口+ 小配件）是带有对象类型属性的窗口。它们都需要用到窗口管理器。

STemWin 程序库所提供的小工具可在以下网址找到：<http://www.segger.com>

创建了小工具之后，会将其视作窗口来对待。窗口管理器确保小工具在必要时能够正确显示（以及重绘）。

图10. 小工具示例



### 小工具的创建

仅须一行代码即可创建一个小工具。

创建小工具基本上有两种方式：

#### ♦ 直接创建：

利用每个小工具现成的创建函数：

- `<WIDGET>_CreateEx()`：创建过程无需用户数据。
- `<WIDGET>_CreateUser()`：使用用户数据来创建小工具。

#### ♦ 间接创建：

“间接”表示使用对话框创建函数和包含间接创建程序指针的 `GUI_WIDGET_CREATE_INFO` 结构体来创建小工具：

- `<WIDGET>_CreateIndirect()`：通过对话框创建函数来创建。

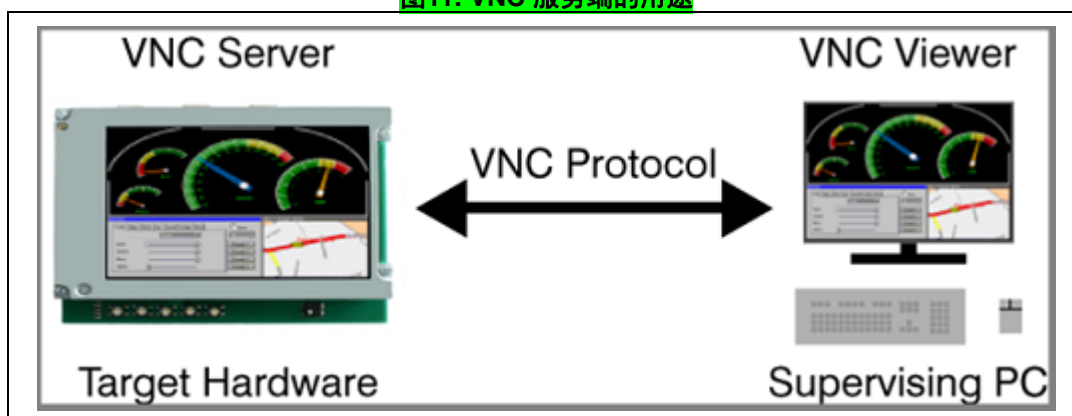
## 4.8 VNC 服务端

VNC 表示“虚拟网络计算 (Virtual Network Computing)”。VNC 服务器通过 TCP/IP 协议将嵌入式目标连接至网络机，该功能用于：

- ♦ 在远程 PC 显示器上查看 LCD 显示内容，
- ♦ 利用鼠标控制嵌入式环境。

换言之，嵌入式设备的显示内容可以在运行着 VNC 客户端（比如网络 PC）的机器屏幕上显示；鼠标和键盘可用于控制目标。

图11. VNC 服务端的用途



#### 4.8.1

#### 要求

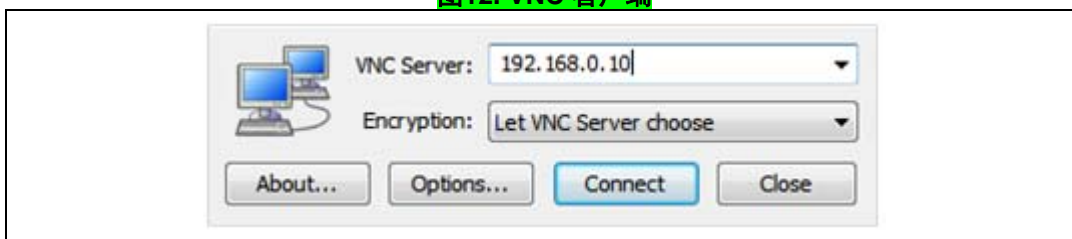
- 具备嵌入式以太网 IP 的 STM32 设备，比如 STM32F107xx、STM32F2x7xx 或 STM32F4x7xx。
- 目标机器应具备 TCP/IP 协议栈。在所交付的演示程序中采用的是 LwIP。
- VNC 服务端应运行于单独的线程。因此，emWin VNC 服务端应运行在多任务系统中。在本例中，演示程序包运行在 FreeRTOS 中。
- 监控 PC 上应具备 VNC 查看器（比如 RealVNC、TightVNC、UltraVNC...）。

## 4.8.2 过程说明

- ♦ 通过以太网线缆将目标硬件连接至网络（如需本地连接，则先连接到 PC）。
- ♦ 运行演示程序。
  - VNC\_Server\_task 是 Background\_Task 的一项子任务。
  - 硬件（LED、触摸屏、SRAM.....）和 GUI 初始化完毕后，会对 TCP/IP (LwIP) 协议栈进行初始化。
  - 然后调用 GUI\_VNC\_X\_StartServer() 用于：
    - a) 初始化 VNC 上下文并将其连接到一个层级上，
    - b) 创建 VNC 服务端任务，监听 5900 端口，直至检测到接入连接，然后运行实际的服务端（通过调用 GUI\_VNC\_Process()）。
- ♦ 如果启用 DHCP（“#define USE\_DHCP” in main.h）：
  - 等待 DHCP 服务器分配 IP 地址；信息会显示在“VNC 服务端”页面（紧接在“Intro（介绍）”页面之后）。
  - 如果无法检索到任何 IP 地址（DHCP 超时），便会分配并显示预定义的静态 IP 地址。
- ♦ 如果 DHCP 被禁用（或者 DHCP 超时）：
  - 等待显示静态 IP 地址。
  - 利用目标硬件中的同级地址来配置 PC 的 IP 地址和子网掩码。
- ♦ 启动 VNC 查看器。
  - 连接至目标硬件的 IP 地址（参见 [图 12](#)）。
  - 演示程序便会显示在 PC 上。
- ♦ 利用 VNC 查看器，用户可以：
  - 在显示器上查看所运行的演示程序（实时流数据）；
  - 在 PC 上控制目标硬件（利用鼠标）；
  - 对演示程序进行截图（比如用于指南介绍）。

**注：** 断开与重连查看器与服务端之间的连接，并不会丢失任何数据。

图12. VNC 客户端



## 4.9 字体

STemWin 程序库中包含最为常见的字体作为标准字体包。所有字体包均包含 ASCII 字符集，大部分字体包还包含 ISO 8859-1 字符。

嵌入字体的完整清单如下所示（摘自 GUI.h）。

**注：** STemWin 程序库的默认字体为 GUI\_Font6x8。

### 4.9.1 STemWin 库中包含的字体

```
//
// 比例字体
//
#define GUI_FONT_8_ASCII           &GUI_Font8_ASCII
#define GUI_FONT_8_1               &GUI_Font8_1
#define GUI_FONT_10S_ASCII        &GUI_Font10S_ASCII
#define GUI_FONT_10S_1            &GUI_Font10S_1
#define GUI_FONT_10_ASCII         &GUI_Font10_ASCII
#define GUI_FONT_10_1             &GUI_Font10_1
#define GUI_FONT_13_ASCII         &GUI_Font13_ASCII
#define GUI_FONT_13_1             &GUI_Font13_1
#define GUI_FONT_13B_ASCII        &GUI_Font13B_ASCII
#define GUI_FONT_13B_1            &GUI_Font13B_1
#define GUI_FONT_13H_ASCII        &GUI_Font13H_ASCII
#define GUI_FONT_13H_1            &GUI_Font13H_1
#define GUI_FONT_13HB_ASCII       &GUI_Font13HB_ASCII
#define GUI_FONT_13HB_1           &GUI_Font13HB_1
#define GUI_FONT_16_ASCII         &GUI_Font16_ASCII
#define GUI_FONT_16_1             &GUI_Font16_1
#define GUI_FONT_16_HK            &GUI_Font16_HK
#define GUI_FONT_16_1HK           &GUI_Font16_1HK
#define GUI_FONT_16B_ASCII        &GUI_Font16B_ASCII
#define GUI_FONT_16B_1           &GUI_Font16B_1
#define GUI_FONT_20_ASCII         &GUI_Font20_ASCII
#define GUI_FONT_20_1             &GUI_Font20_1
#define GUI_FONT_20B_ASCII        &GUI_Font20B_ASCII
#define GUI_FONT_20B_1           &GUI_Font20B_1
#define GUI_FONT_24_ASCII         &GUI_Font24_ASCII
#define GUI_FONT_24_1             &GUI_Font24_1
#define GUI_FONT_24B_ASCII        &GUI_Font24B_ASCII
#define GUI_FONT_24B_1           &GUI_Font24B_1
#define GUI_FONT_32_ASCII         &GUI_Font32_ASCII
#define GUI_FONT_32_1             &GUI_Font32_1
#define GUI_FONT_32B_ASCII        &GUI_Font32B_ASCII
#define GUI_FONT_32B_1           &GUI_Font32B_1
```



```

//
// 比例字体，有框架
//
#define GUI_FONT_20F_ASCII &GUI_Font20F_ASCII

//
// 等宽字体
//
#define GUI_FONT_4X6 &GUI_Font4x6
#define GUI_FONT_6X8 &GUI_Font6x8
#define GUI_FONT_6X8_ASCII &GUI_Font6x8_ASCII
#define GUI_FONT_6X8_1 &GUI_Font6x8_1
#define GUI_FONT_6X9 &GUI_Font6x9
#define GUI_FONT_8X8 &GUI_Font8x8
#define GUI_FONT_8X8_ASCII &GUI_Font8x8_ASCII
#define GUI_FONT_8X8_1 &GUI_Font8x8_1
#define GUI_FONT_8X9 &GUI_Font8x9
#define GUI_FONT_8X10_ASCII &GUI_Font8x10_ASCII
#define GUI_FONT_8X12_ASCII &GUI_Font8x12_ASCII
#define GUI_FONT_8X13_ASCII &GUI_Font8x13_ASCII
#define GUI_FONT_8X13_1 &GUI_Font8x13_1
#define GUI_FONT_8X15B_ASCII &GUI_Font8x15B_ASCII
#define GUI_FONT_8X15B_1 &GUI_Font8x15B_1
#define GUI_FONT_8X16 &GUI_Font8x16
#define GUI_FONT_8X17 &GUI_Font8x17
#define GUI_FONT_8X18 &GUI_Font8x18
#define GUI_FONT_8X16X1X2 &GUI_Font8x16x1x2
#define GUI_FONT_8X16X2X2 &GUI_Font8x16x2x2
#define GUI_FONT_8X16X3X3 &GUI_Font8x16x3x3
#define GUI_FONT_8X16_ASCII &GUI_Font8x16_ASCII
#define GUI_FONT_8X16_1 &GUI_Font8x16_1

//
// 数字
//
#define GUI_FONT_D24X32 &GUI_FontD24x32
#define GUI_FONT_D32 &GUI_FontD32
#define GUI_FONT_D36X48 &GUI_FontD36x48
#define GUI_FONT_D48 &GUI_FontD48
#define GUI_FONT_D48X64 &GUI_FontD48x64
#define GUI_FONT_D64 &GUI_FontD64
#define GUI_FONT_D60X80 &GUI_FontD60x80
#define GUI_FONT_D80 &GUI_FontD80

//

```

```
// 漫画字体
//
#define GUI_FONT_COMIC18B_ASCII &GUI_FontComic18B_ASCII
#define GUI_FONT_COMIC18B_1 &GUI_FontComic18B_1
#define GUI_FONT_COMIC24B_ASCII &GUI_FontComic24B_ASCII
#define GUI_FONT_COMIC24B_1 &GUI_FontComic24B_1
```

大多数情况下，这些字体已经够用。然而在有需要的情况下，STemWin 还可以支持多种外部字体格式：

- 系统独立字体 (SIF) 格式
- 外部位图字体 (XBF) 格式
- TrueType 字体 (TTF) 格式

为此，STemWin 库包含丰富的字体 API。请参见表 5。

表5. 字体 API

程序	说明
与字体函数有关的 C 语言文件	
GUI_SetDefaultFont()	设置默认字体。
GUI_SetFont()	设置当前字体。
与字体函数有关的'SIF'文件	
GUI_SIF_CreateFont()	通过传递指针至系统独立字体数据，来创建和选择字体。
GUI_SIF_DeleteFont()	通过 GUI_SIF_CreateFont() 来删除先前所创建的字体。
与字体函数有关的'TTF'文件	
GUI_TTF_CreateFont()	根据 TTF 字体文件创建 GUI 字体。
GUI_TTF_DestroyCache()	删除 TTF 引擎中的缓存。
GUI_TTF_Done()	释放 TTF 引擎的所有动态分配内存。
GUI_TTF_GetFamilyName()	返回字体的系列名称。
GUI_TTF_GetStyleName()	返回字体的样式名称。
GUI_TTF_SetCacheSize()	用于设置 TTF 缓存的默认大小。
与字体函数有关的'XBF'文件	
GUI_XBF_CreateFont()	通过传递指针至回调函数，从 XBF 字体文件中提取数据，从而创建和选择字体。
GUI_XBF_DeleteFont()	通过 GUI_XBF_CreateFont() 来删除先前所创建的字体。
常见的字体相关函数	
GUI_GetCharDistX()	返回当前字体特定字符的宽度方向像素（X 方向大小）。
GUI_GetFont()	返回指向当前选中字体的指针。
GUI_GetFontDistY()	返回当前字体的 Y 方向间隔。
GUI_GetFontInfo()	返回包含字体信息的结构体。
GUI_GetFontSizeY()	返回当前字体的高度方向像素（X 方向大小）。



表5. 字体 API（续）

程序	说明
GUI_GetLeadingBlankCols()	返回给定字符前方空白的像素数。
GUI_GetStringDistX()	返回采用当前字体的一段文本在 X 方向的尺寸。
GUI_GetTextExtend()	计算采用当前字体的一段文本的尺寸。
GUI_GetTrailingBlankCols()	返回给定字符后方空白的像素数。
GUI_GetYDistOfFont()	返回特定字体的 Y 方向间隔。
GUI_GetYSizeOfFont()	返回特定字体的 Y 方向尺寸。
GUI_IsInFont()	评估某具体字符是否属于特定字体。

4.9.2 如何添加新字体

为了能够在构建应用程序时使用任何所需的字体，STemWin提供了一个“Font Converter”工具。

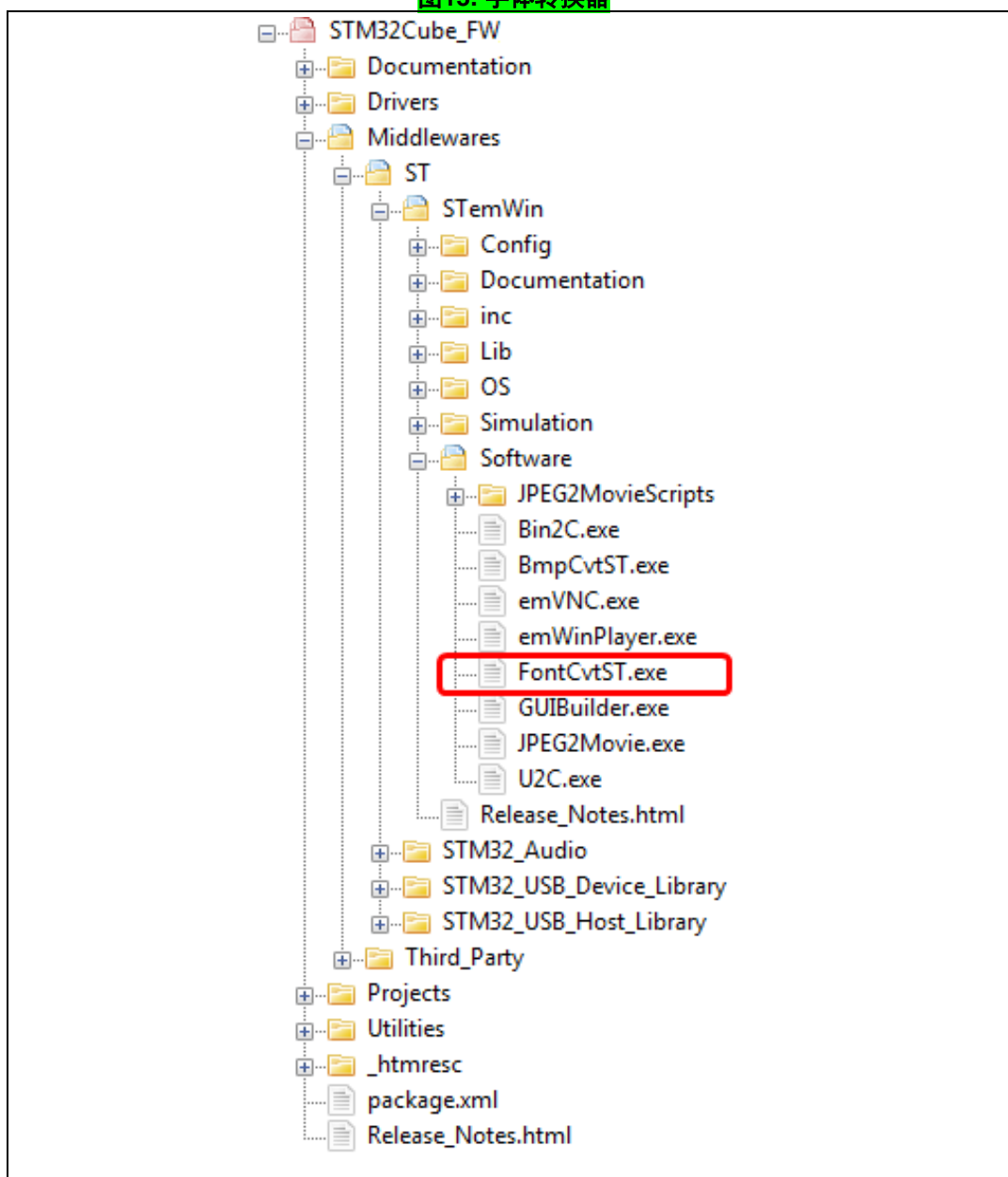
作为Windows程序，它可以方便地将PC已安装的任何字体转换为STemWin（位图）字体，该字体可以轻松集成到我们的应用程序中。

生成的字体将定义为C文件，或者包含系统独立字体（SIF）或外部位图字体（XBF）的二进制文件。

此部分仅详细说明如何将PC上已安装字体生成C文件。

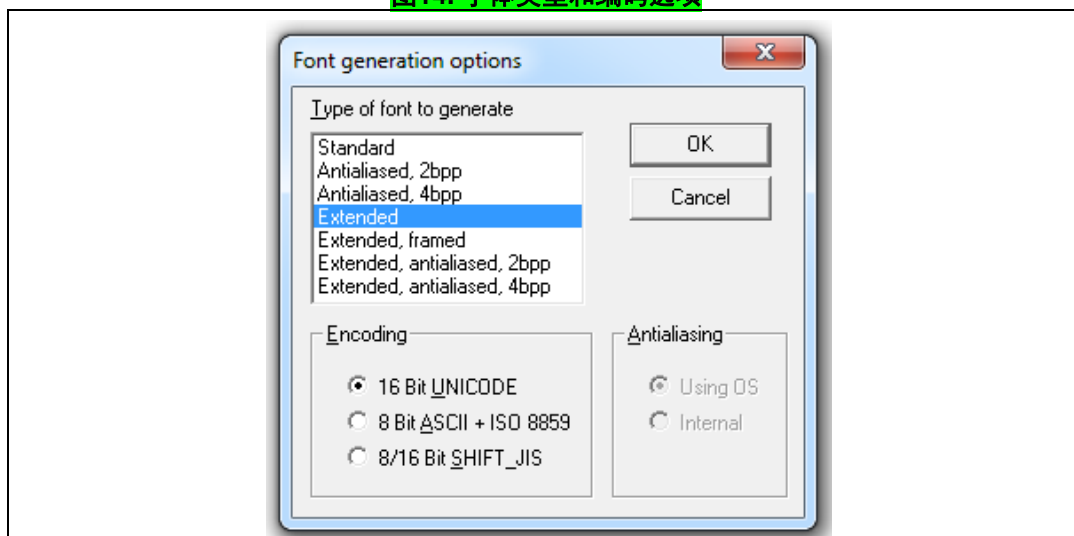
在STM32CubeFW上，转到以下路径“Middlewares \ ST \ STemWin \ Software”并执行“FontCvtST.exe”

图13. 字体转换器



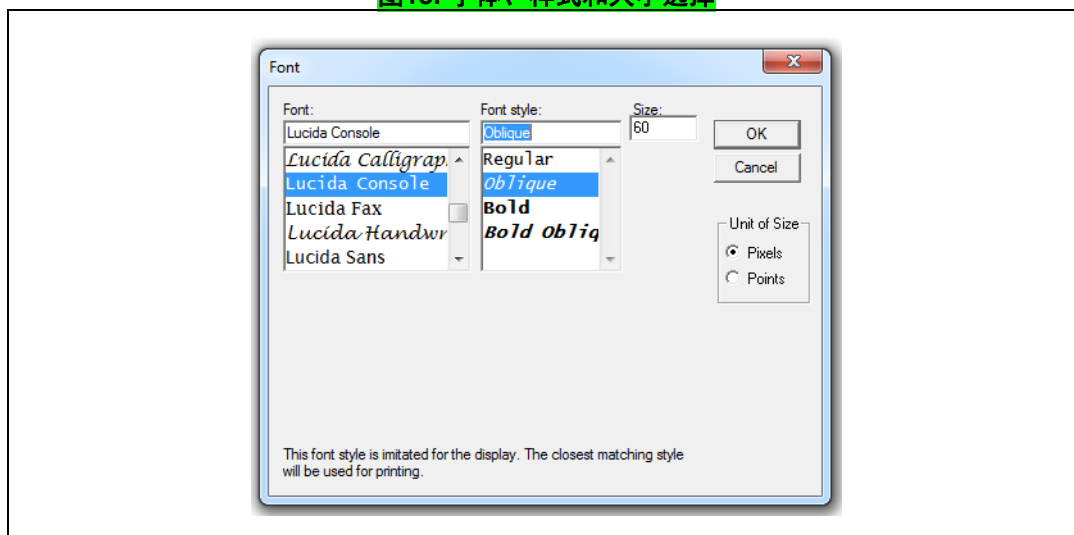
打开后，您需要选择要生成的字体类型和编码选项。

图14. 字体类型和编码选项



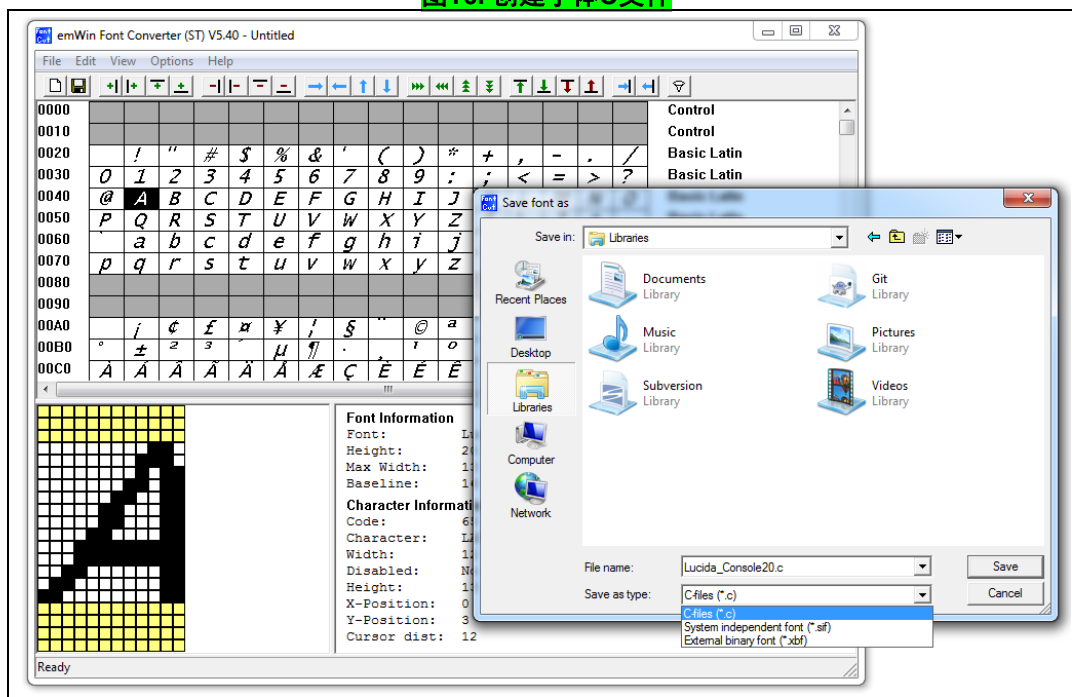
接下来，选择您所需的字体、样式和大小。

图15. 字体、样式和大小选择



最后，您需要将此字体保存为一个C文件，并根据字体的相关信息为文件设置一个有具体含义的名称。

图16. 创建字体C文件



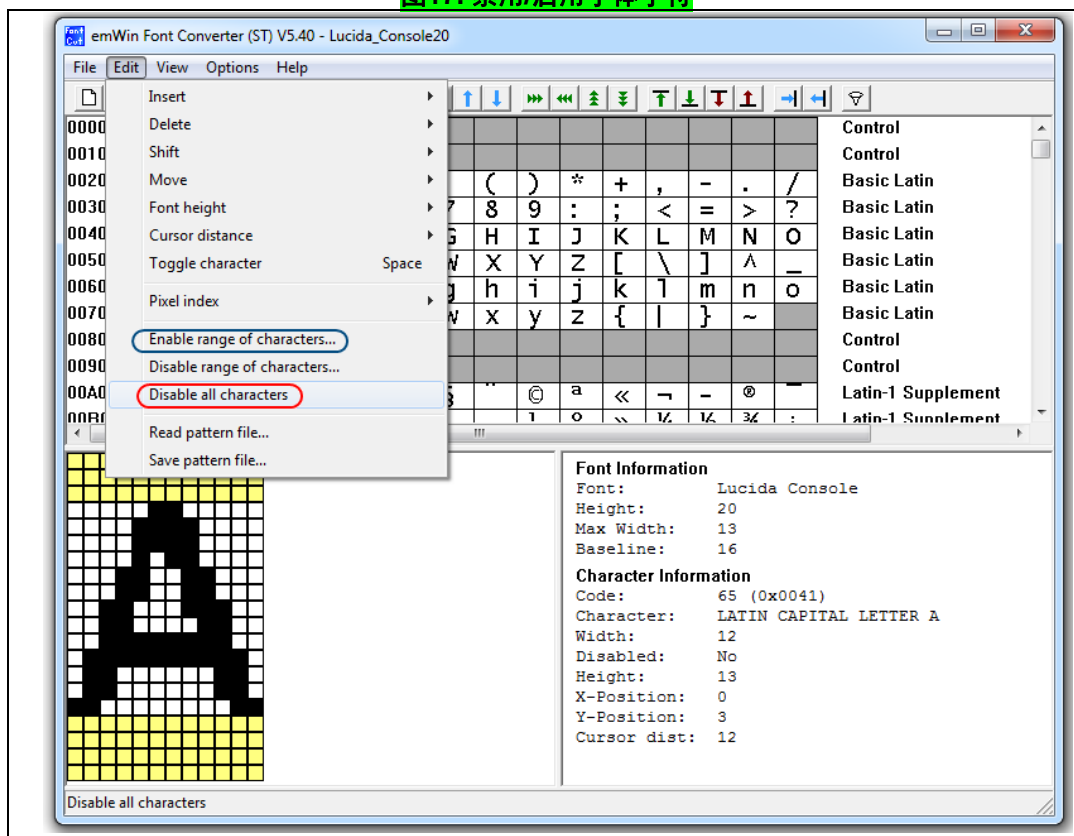
## 优化

此处的问题在于，生成的“C”文件将包含字体的所有字符和符号。由于“C”文件将与应用程序一起编译，因此用户需要足够的可寻址存储器，以便存储字体数据。

为解决这一问题，可以使用字体转换器禁用不需要的某些符号和字符，仅生成所需字体需要的块。

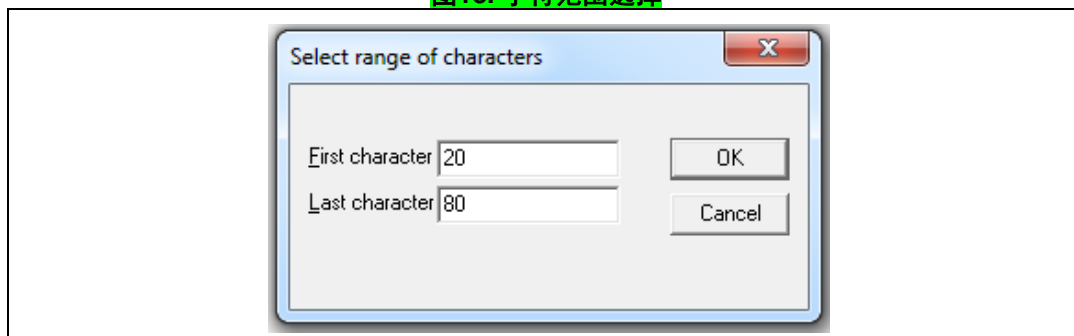
首先对于已打开的字体，用户需要禁用所有字符（下图的红色选项）。

图17. 禁用/启用字体字符



然后仅启用您想要保留的字符块（上图的蓝色选项），您只需输入您选择的范围即可。

图18. 字符范围选择



然后另存为“C”文件。

例如，如果此字体仅需要字符‘A’，则生成的字体文件将如下所示：

```
#include "GUI.h"
```

```
#ifndef GUI_CONST_STORAGE
#define GUI_CONST_STORAGE const
#endif
```

```

/* The following line needs to be included in any file selecting the
font.
*/
extern GUI_CONST_STORAGE GUI_FONT GUI_FontLucida_Console20;

/* Start of unicode area <Basic Latin> */
GUI_CONST_STORAGE unsigned char acGUI_FontLucida_Console20_0041[ 26] = { /* code 0041,
LATIN CAPITAL LETTER A */
    _XX_,_,_,_,
    _XXXX,_,_,_,
    _XXXX,_,_,_,
    _X_XX,_,_,_,
    _XX_X,X_,_,_,
    _XX_X,X_,_,_,
    _XX_X,XX_,_,_,
    _XX_,XX_,_,_,
    _XXXXXX,XX_,_,_,
    _XXXXXX,XXX_,_,_,
    _XX_,_,XX_,_,_,
    _XX_,_,XX_,_,_,
    XX_,_,XX_};

GUI_CONST_STORAGE GUI_CHARINFO_EXT GUI_FontLucida_Console20_CharInfo[1] = {
    { 12, 13, 0, 3, 12, acGUI_FontLucida_Console20_0041 } /* code 0041, LATIN CAPITAL LETTER
A */
};

GUI_CONST_STORAGE GUI_FONT_PROP_EXT GUI_FontLucida_Console20_Prop1 = {
    0x0041 /* first character */
    ,0x0041 /* last character */
    ,&GUI_FontLucida_Console20_CharInfo[ 0] /* address of first character */
    ,(GUI_CONST_STORAGE GUI_FONT_PROP_EXT *)0 /* pointer to next GUI_FONT_PROP_EXT */
};

GUI_CONST_STORAGE GUI_FONT GUI_FontLucida_Console20 = {
    GUI_FONTTYPE_PROP_EXT /* type of font */
    ,20 /* height of font */
    ,20 /* space of font y */
    ,1 /* magnification x */
    ,1 /* magnification y */
    ,{&GUI_FontLucida_Console20_Prop1}
    ,16 /* Baseline */
    ,11 /* Height of lowercase characters */
    ,13 /* Height of capital characters */
};

```



### 4.9.3 语言支持

使用阿拉伯语、希伯来语、泰语或中文等语言编写的文本包含字符，这些字符通常不是 STemWin 附带字体的一部分。

对于具有不同读/写方向的语言以及所有已知文化的每个有意义的字符或文本元素，Unicode 标准包含唯一的数字代码点。

为了能够使用 Unicode 字符解码文本，STemWin 使用 UTF-8 编码方法。（有关更多详细信息，请参见用户手册）。

以下部分说明如何使用上述语言支持字体的使用。

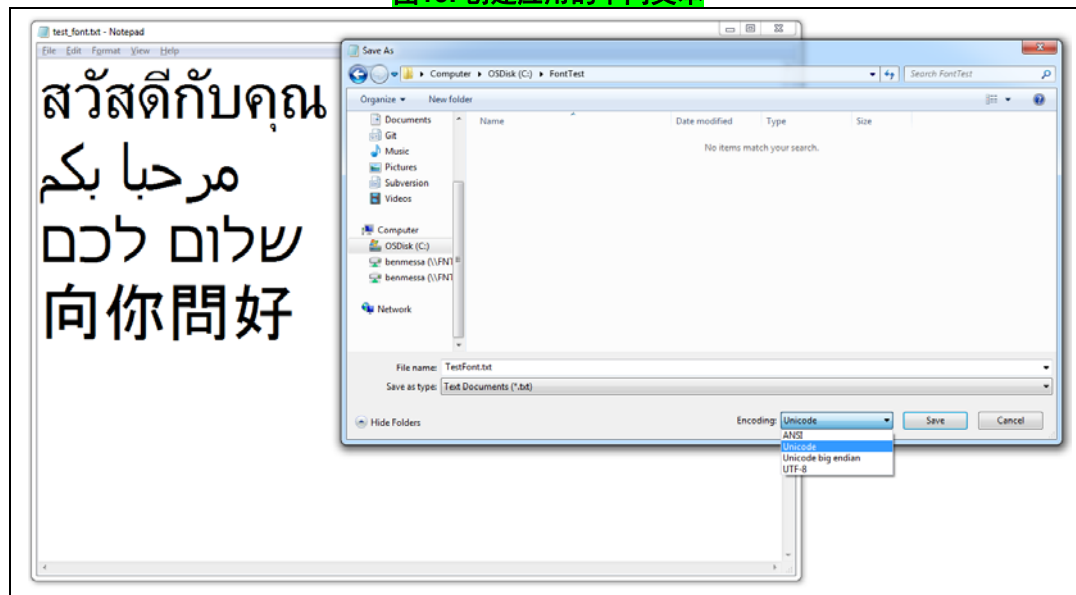
对于上述四种语言的任意一种，需要完成两项任务：

- 生成待使用的“C”文件
- 在 STemWin 应用程序上添加所需的软件，以便能够显示所需的字符

假设用户正在尝试显示翻译成泰语、阿拉伯语、希伯来语和繁体中文的“欢迎”字符串。

首先，用户使用这些语言编写所需的文本，并使用“Unicode”格式保存文本文件。用户可以在此任务中使用“Notepad.exe”，如图 19 所示。

图19. 创建应用的不同文本



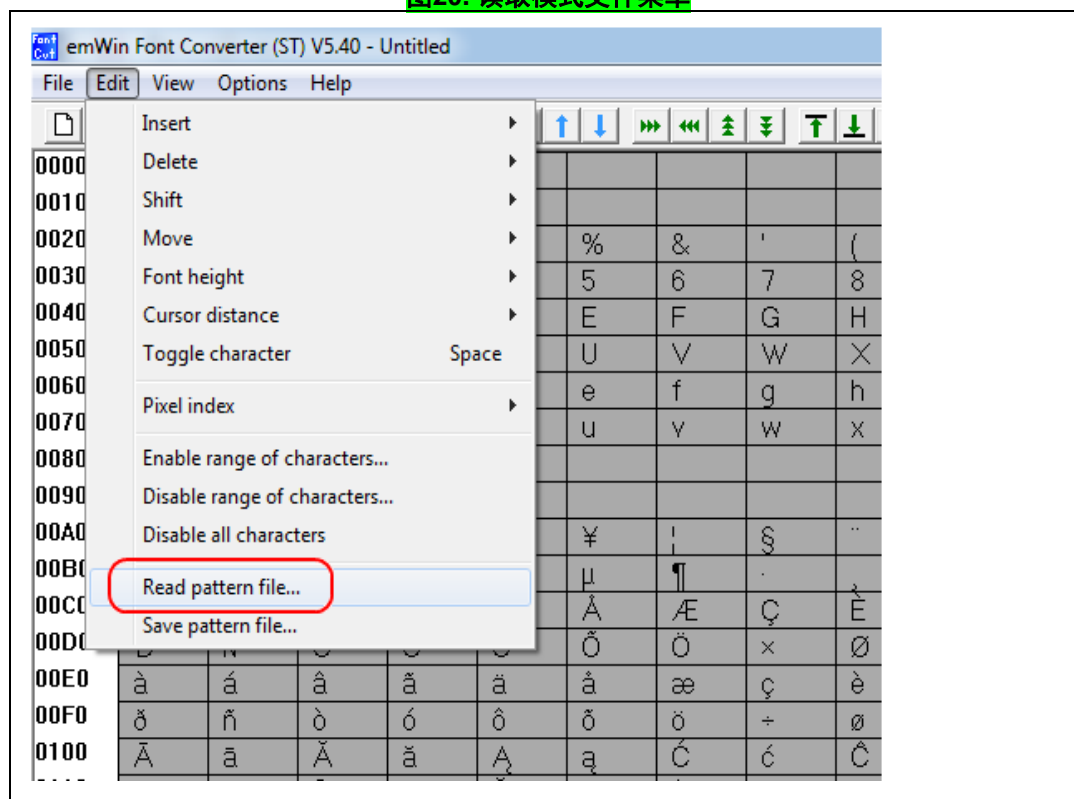
#### “C”文件生成

接下来，用户使用“FontConverter”，选择支持所有这些语言的字体。在所给的案例中，用户生成两个字体文件，第一个用于阿拉伯语、希伯来语和泰语；第二个用于繁体中文。

一旦在“FontConverter”上打开了字体，则用户需要禁用所有字符，如 优化 所示。

下一步，用户读取包含不同字符串的模式文件。

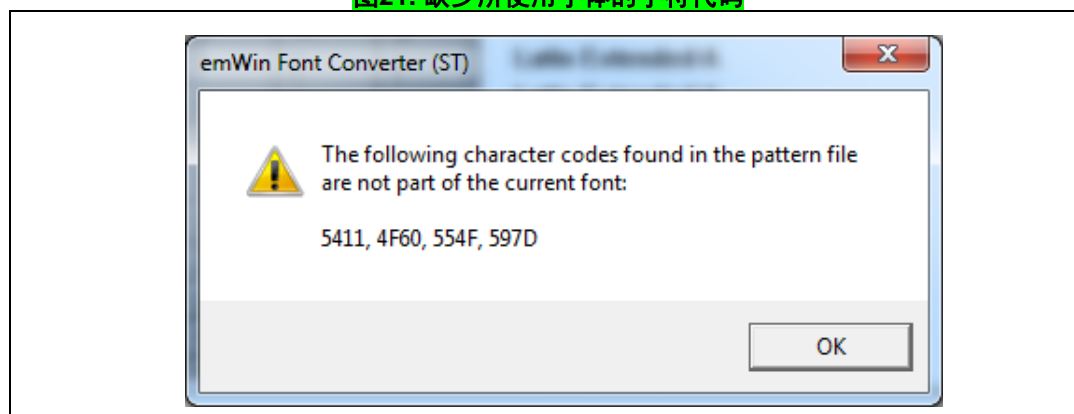
图20. 读取模式文件菜单



浏览和加载已创建的文件。在本例中，即“TestFont.txt”。

如果使用的字体不支持“TestFont.txt”中的任何语言，则会显示警告消息。

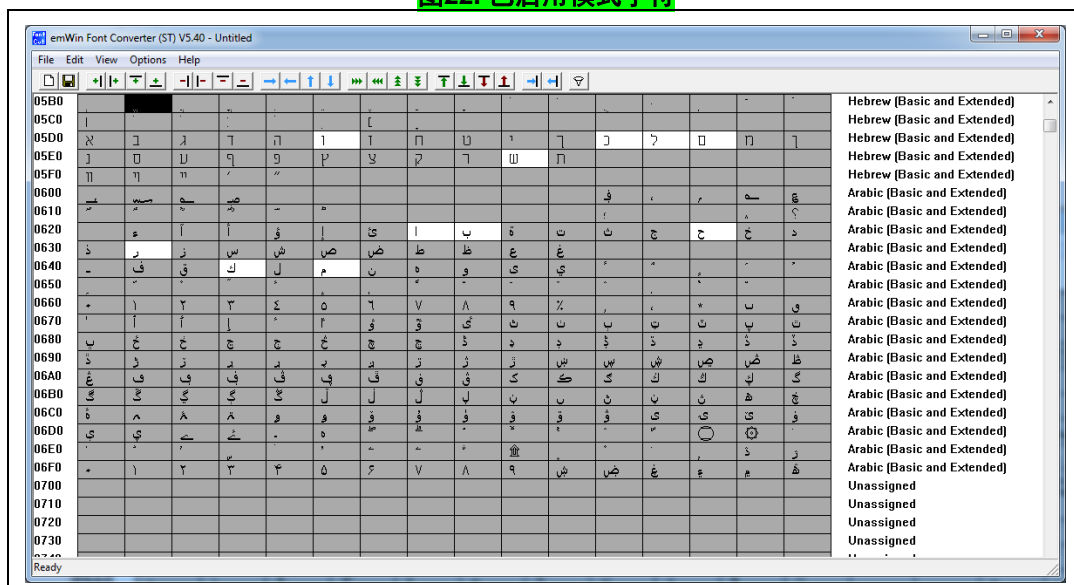
图21. 缺少所使用字体的字符代码



这一点没有关系，只需按“确定”按钮即可。

接下来，将启用写在“FontTest.txt”上的字符。

图22. 已启用模式字符



注意：

在显示确切需要的字符串方面，泰语和阿拉伯语都存在一些使用限制：

- ◆ 对于阿拉伯字体，需要激活以下范围：0x600至0x6FF
- ◆ 对于泰语字体
  - 必须将字体文件设置为“Extended”
  - 需要激活以下范围：0xE00-0xE7F

完成后，只需保存至“C”文件。

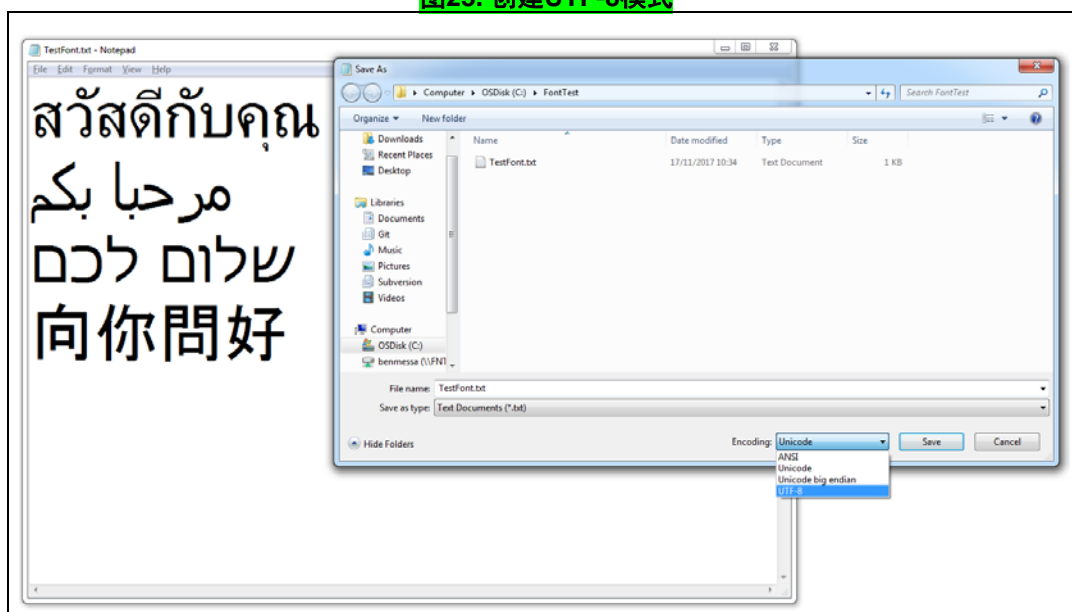
### 需要软件方可添加

由于上述描述，因此，我们正在添加两个字体文件，以便用户要添加到应用程序编译中。

现在，为了能够在应用程序上显示这些字符串，用户调整应用程序软件以解码Unicode。

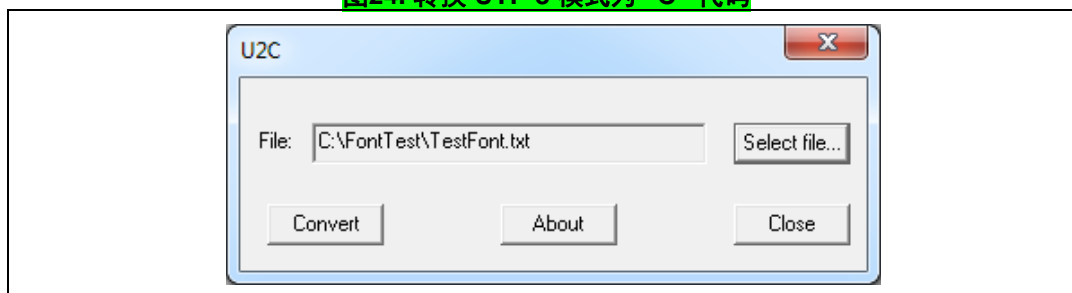
STemWin能够解码UTF-8编码的字符串，因此如要显示文件“FontTest.txt”上添加的Unicode字符，用户首先需要将其保存为UTF-8格式。

图23. 创建UTF-8模式



如要将UTF-8文本转换为“C”代码，请转到以下路径  
“Middlewares\ST\STemWin\Software”上的STM32CubeFW，使用文件“FontTest.txt”作为输入执行“U2C.exe”。

图24. 转换 UTF-8 模式为 “C” 代码



“U2C”转换器的输出将为：

```
"\xe0\xb8\xaa\xe0\xb8\xa7\xe0\xb8\xb1\xe0\xb8\xaa\xe0\xb8\x94\xe0\xb8\xb5\xe0\xb8\x81\xe0\xb8\x
b1\xe0\xb8\x9a\xe0\xb8\x84\xe0\xb8\xb8\xe0\xb8\x93"
"\xd9\x85\xd8\xb1\xd8\xad\xd8\xa8\xd8\xa7 \xd8\xa8\xd9\x83\xd9\x85"
"\xd7\xa9\xd7\x9c\xd7\x95\xd7\x9d \xd7\x9c\xd7\x9b\xd7\x9d"
"\xe5\x90\x91\xe4\xbd\xa0\xe5\x95\x8f\xe5\xa5\xbd"
```

每行（“”之间的文本）对应于“txt”文件里同一行上的字符串。

现在，用户可以使用此输出创建应用程序：

```
#include "GUI.h"
/* Font for Thai, Arabic and Hebrew languages */
#include "MicrosoftSansSerif20.c"
/* Font for Chinese language */
#include "Microsoft_YaHeiUI20.c"
```

```

static const char * _apStrings[] =
{
    "\xe0\xb8\xaa\xe0\xb8\xa7\xe0\xb8\xb1\xe0\xb8\xaa\xe0\xb8\x94\xe0\xb8\xb5\xe0\xb8\x81\xe0\xb8\x
b1\xe0\xb8\x9a\xe0\xb8\x84\xe0\xb8\xb8\xe0\xb8\x93",
    "\xd9\x85\xd8\xb1\xd8\xad\xd8\xa8\xd8\xa7 \xd8\xa8\xd9\x83\xd9\x85",
    "\xd7\xa9\xd7\x9c\xd7\x95\xd7\x9d \xd7\x9c\xd7\x9b\xd7\x9d",
    "\xe5\x90\x91\xe4\xbd\xa0\xe5\x95\x8f\xe5\xa5\xbd",
};

void MainTask(void)
{
    int i;
    GUI_Init();
    GUI_SetFont(&GUI_FontMicrosoftSansSerif20);
    GUI_UC_SetEncodeUTF8();
    /* Thai string */
    GUI_DispString(_apStrings[0]);
    GUI_DispNextLine();

    /* Arabic and Hebrew strings:
       In this case we need to enable bidirectional
       (right to left) text support */
    GUI_UC_EnableBIDI(1);
    GUI_DispString(_apStrings[1]);
    GUI_DispNextLine();
    GUI_DispString(_apStrings[2]);
    GUI_DispNextLine();

    /* Chinese string, bidirectional no more needed */
    GUI_UC_EnableBIDI(0);
    GUI_DispString(_apStrings[3]);

    while(1)
    {
        GUI_Delay(500);
    }
}

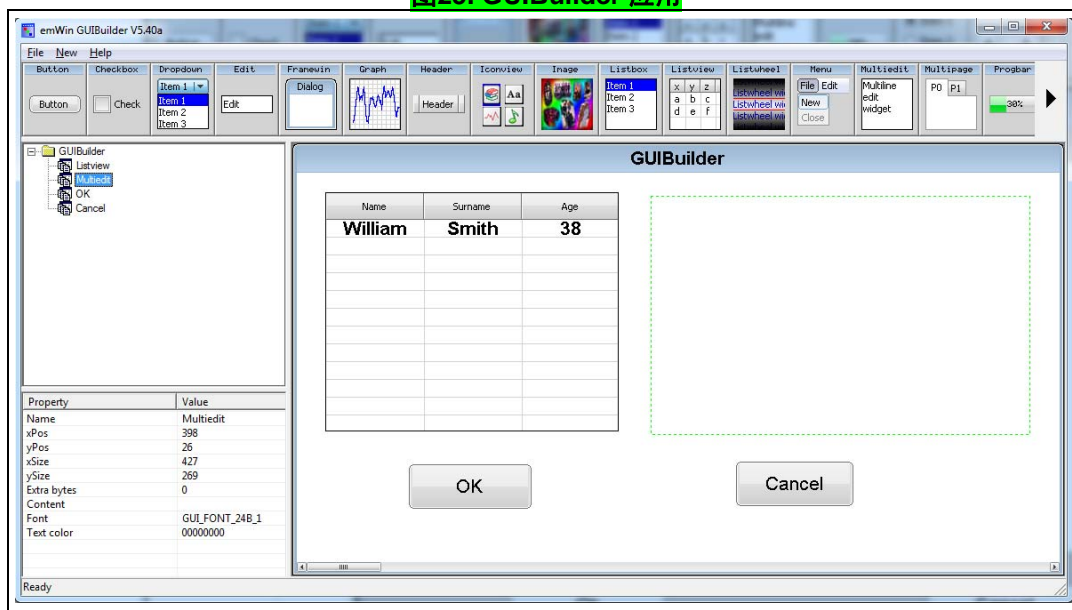
```

## 4.10 GUIBuilder

GUIBuilder 是一个能够轻松创建对话的工具；与编写源代码不同，用户可以通过鼠标拖放来放置小工具以及设置小工具大小。可通过弹出菜单添加的其它属性。编辑小工具的属性可以进行微调。

GUIBuilder 可以生成对话的 C 代码，既可定制化设置，也可以集成到项目当中。

图25. GUIBuilder 应用



#### 4.10.1 GUIBuilder 的基本使用

- 首先从 FRAMEWIN 或 WINDOW 小工具开始：只有这些小工具才能作为对话的父窗口。
- 将小工具放置在父窗口中：通过移动鼠标和/或编辑属性窗口中的属性，可以放置小工具以及调整大小。
- 配置小工具：弹出菜单显示了可用的选项。
- 保存对话：每个对话会保存在单独文件中。文件名会根据父窗口的名称而自动生成。

#### 4.10.2 创建程序

利用 GUIBuilder 创建的文件中包含对话的创建程序。程序名称中包含父窗口的名称：  
WM\_HWIN Create<WindowName>(void);

只须简单地调用以下程序即可创建对话：

```
hWin = CreateFramewin();
```

#### 4.10.3 用户自定义代码

所生成的代码包含一对注释，用户代码可添加到注释之间。为了让 GUIBuilder 能够回读该文件，必须将代码添加到这些注释之间。

**注：** 把代码添加到用户代码注释以外的区域，会使得 GUIBuilder 无法读出文件中的内容。

#### 4.10.4 回调程序

回调程序是所生成文件的主要组成部分。回调函数中一般包含以下消息句柄：

- WM\_INIT\_DIALOG

创建对话的所有小工具之后，立即在此完成小工具的初始化。用户代码区域可用于添加更多初始化代码。

• WM\_NOTIFY\_PARENT

含有待添加用户代码的（空）消息句柄。小工具的每一条通知消息，都对应一个消息句柄。可为通知消息添加更多反应动作。

## 4.11 以不同的分辨率移植 GUI

可以在“LCDConf.c”上完成GUI分辨率的全部设置，从一个分辨率切换至另一个分辨率非常简单。

该设置取决于将使用的板的屏幕分辨率。此时，已知屏幕的物理尺寸。

变量“XSIZE\_PHYS”和“YSIZE\_PHYS”将在“LCDConf.c”上设置，以提供物理屏幕尺寸。

假设我们的屏幕分辨率为（800x480），“LCDConf.c”具有以下内容：

```
/* Define physical screen sizes */
```

```
#define XSIZE_PHYS 800
```

```
#define YSIZE_PHYS 480
```

这些变量将用于初始化链接到屏幕的硬件（例如：LTDC/DSI）。

然后使用“XSIZE\_0”和“YSIZE\_0”变量设置GUI分辨率。

“0”对应于层“0”，因此如果使用层“1”，则还存在“XSIZE\_1”和“YSIZE\_1”。

当然，GUI分辨率必须适合物理分辨率。

```
#define XSIZE_0    XSIZE_PHYS
```

```
#define YSIZE_0    YSIZE_PHYS
```

如果现在必须修改屏幕（或GUI分辨率），则必须如上所述修改变量。

## 4.12 如何添加新效果

STemWin可以使用内存设备API，添加模糊、混合和抖动等效果。

用户还可以使用内存设备或所使用的帧缓冲器的任何矩形部分来创建一些其他效果。

请注意，所有添加的效果和数据处理都将以ARGB8888格式完成。

如果需要，在LCD上显示时将进行颜色转换。

我们的设想是直接操纵数据进行修改。因此，必须知道待处理数据的基址、宽度和高度。

有两种方法可以达成这一目标：

- ◆ 使用空闲的内存基址
- ◆ 使用该内存设备，这将动态分配给定的内存

了解STemWin提供的众多API，这样会使得内存设备的使用更简单、更安全。

在软件中待添加的新效果将由滤波器定义，该滤波器将数据指针、数据宽度和高度作为输入，并将修改后的数据作为输出提供给新地址。

这将包括以下步骤：

- ◆ 创建源内存设备，每个像素32位（Bpp）
- ◆ 创建具有相同大小的32 Bpp目标内存设备
- ◆ 使用“GUI\_MEMDEV\_Select（hSrc）”选择源内存设备，在其上绘制数据，当应用新效果时，数据将修改
- ◆ 获取源内存设备和目标内存设备的指针，以便进行直接操作
  - 将使用API“void \* GUI\_MEMDEV\_GetDataPtr（GUI\_MEMDEV\_Handle hSrc）”。
  - 请注意，如果位图是32 Bpp，则其数据指针可以直接用作源指针
- ◆ 使用这些指针、宽度和高度，将效果滤波器应用于源数据并将其写入目标内存设备。
- ◆ 一旦修改，目标内存设备就可以显示在LCD屏幕上
  - GUI\_MEMDEV\_Select(0);
  - GUI\_MEMDEV\_CopyToLCD(hDest);

## 4.13 如何添加新的小部件

在STemWin上，小部件是指具有增强功能的窗口。

因此，创建新的小部件将基于窗口的创建，但同时应具有存储其他数据的功能。

其中主要包括添加以下功能：

- ◆ “创建”功能，返回创建的小部件的句柄，允许设置回调函数。
- ◆ “设置”和“获取”功能，可请求和设置小部件的特定属性。
- ◆ “回调”功能，用于处理发送到自定义小部件的消息

详细的分步骤说明，请参阅此路径：<https://www.segger.com/downloads/emwin/AN03002>



5 性能与内存开销

5.1 LCD 驱动程序性能

表 6 列出一组用于测试显示驱动程序性能的测试。

表6. 速度测试列表

测试名称	说明
测试1：填充	测量填充速度。在 64 * 64 像素的区域内填充不同颜色。
测试2：小字体	测量小字符的输出速度。在 60 * 64 像素的区域内填充小字符文本。
测试3：大字体	测量大字符的输出速度。在 65 * 48 像素的区域内填充大字符文本。
测试4：1 bpp 位图	测量 1 bpp 位图的填充速度。在 58 * 8 像素的区域内填充 1 bpp 位图。
测试5：2 bpp 位图	测量 2 bpp 位图的填充速度。在 32 * 11 像素的区域内填充 2 bpp 位图。
测试6：4 bpp 位图	测量 4 bpp 位图的填充速度。在 32 * 11 像素的区域内填充 4 bpp 位图。
测试7：8 bpp 位图	测量 8 bpp 位图的填充速度。在 32 * 11 像素的区域内填充 8 bpp 位图。
测试8：16 bpp 位图	测量 16 bpp 位图的填充速度。在 64 * 8 像素的区域内填充 8 bpp 位图。

分别采用FlexColor和Lin驱动程序在STM324xG-EVAL和STM324x9I-EVAL板卡上进行测试。

测试结果如 表 7所示。

表7. FlexColor 和 Lin 驱动程序的速度测试

测试名称	FlexColor	Lin
测试1：填充	7.48 M	73.47 M
测试2：小字体	1.57 M	4.16 M
测试3：大字体	2.35 M	5.96 M
测试4：1bpp 位图	3.23 M	8.81 M
测试5：2bpp 位图	2.28 M	6.29 M
测试6：4bpp 位图	2.22 M	6.13 M
测试7：8bpp 位图	1.17 M	9.71 M
测试8：16bpp 位图	5.57 M	4.55 M

M=百万像素/秒

5.2 STemWin 内存开销

STemWin 的适用范围非常广泛，主要取决于应用程序以及所使用的功能。下述小节将列出各个模块的内存要求，以及示例应用程序的内存要求。

下表显示STemWin主要部件的内存要求。这些数值很大程度上依赖于编译器选项、编译器版本以及所采用的 CPU。请注意，所列数值是各个模块基本功能的要求。

表8. 模块的内存开销

元件	ROM	RAM	说明
窗口管理器	6.2 KB	2.5 KB	使用窗口管理器时，基本应用的附加内存要求。
内存设备	4.7 KB	7 KB	使用内存设备时，基本应用的附加内存要求。
抗锯齿	4.5 KB	2 * LCD_XSIZE	抗锯齿软件项的附加内存要求。
驱动	2 – 8 KB	20 字节	驱动程序的内存要求取决于已配置的驱动程序以及是否使用数据缓存。使用数据缓存时，驱动程序需要更多的RAM空间。
多层	2 – 8 KB	·	对于多层或多显配置，每增加一层都需要消耗额外的内存空间，因为每一层都需要专属的驱动程序。
内核	5.2 KB	80 字节	不使用其它软件项时，典型应用的内存要求。
JPEG	12 KB	38 KB	绘制 JPEG 文件的基本程序。
GIF	3.3 KB	17 KB	绘制 GIF 文件的基本程序。
子图	4.7 KB	16 字节	绘制子图和光标的程序。
字体	1 – 4 KB	·	取决于所使用的字体大小。



表9. 小工具内存开销

元件	ROM	RAM	说明
按钮	1 KB	40 字节	*1
复选框	1 KB	52 字节	*1
下拉菜单	1.8 KB	52 字节	*1
编辑控件	2.2 KB	28 字节	*1
窗口框架	2.2 KB	12 字节	*1
图形	2.9 KB	48 字节	*1
GRAPH_DATA_XY	0.7 KB	1	*1
GRAPH_DATA_YT	0.6 KB	1	*1
标头	2.8 KB	32 字节	*1
列表框	3.7 KB	56 字节	*1
列表视图	3.6 KB	44 字节	*1
菜单	5.7 KB	52 字节	*1
多编辑控件	7.1 KB	16 字节	*1
多页	3.9 KB	32 字节	*1
进度条	1.3 KB	20 字节	*1
单选按钮	1.4 KB	32 字节	*1
滚动条	2 KB	14 字节	*1
滑动条	1.3 KB	16 字节	*1
文字	1 KB	16 字节	*1
日历	0.6 KB	32 字节	*1

# 6 FAQ（常见问题）

本章节收集了关于 STemWin 库包最常见的用户问题，并提供一些解决方案和参考信息。

表10. FAQs

不允许。	问题	回答/解决方案
1	包内含有所有 STemWin 功能吗？	是的。所交付的锁定的二进制文件在编译阶段已经使能了所有的功能。
2	STemWin 库该如何配置（在生成二进制文件过程中）？	<p>用于生成STemWin二进制文件的“GUIConf.h”文件的内容如下：</p> <pre> #define GUI_DEFAULT_FONT      &amp;GUI_Font6x8 #define GUI_NUM_LAYERS        2 #define GUI_SUPPORT_TOUCH      (1) #define GUI_SUPPORT_MOUSE      (1) #define GUI_WINSUPPORT         (1) #define GUI_SUPPORT_MEMDEV      (1) #define GUI_SUPPORT_DEVICES    (1) #define BUTTON_REACT_ON_LEVEL  (1) #define GUI_MEMDEV_SUPPORT_CUSTOMDRAW (1) #define GUI_USE_ARGB           (1)                     </pre>
3	交付的二进制文件会不会太庞大？	不会。取决于具体应用。编译器仅考虑从外部函数所调用的部分；因此，未使用的资源不会包含在最终应用程序当中。
4	如何支持新的 LCD 控制器？	<p>为了支持任意类型的LCD控制器，用户需要实现两个配置文件：</p> <p>LCDConf.c/.h</p> <p>GUIConf.c/.h</p> <p>第 4.1 节详细描述了这些文件的内容。</p>
5	强制使用FreeRTOS操作系统吗？	不。可以使用任何其它操作系统。但需要对应的 GUI_X_OS.c 文件（参见 第 4.1.4 节）。
6	项目编译时没有报错，但运行应用程序时没有显示输出。	<p>这个问题可能由以下原因产生：</p> <ul style="list-style-type: none"> <li>堆栈过小。</li> <li>显示控制器初始化出错。</li> <li>显示接口配置出错。</li> </ul>



7

版本历史

表11. 文档版本历史

日期	版本	变更
2013 年 7 月 19 日	1	初始版本。
07-Feb-2014	2	使用 STemWin 通用版本(XYZ)。 支持 STM324x9I- EVAL 板卡。
2014年3月20日	3	添加对 STM32CubeF2 和 STM32CubeF4 的引用
2014年6月25日	4	在可用软件中添加 STM32CubeF3。
2018年4月4日	5	更新了表 1: 实用软件, 图 1: STemWin 结构层级, 第 2.2 节: 库的说明, 表 2: 所支持的 LCD 控制器, 图 2: 项目树, 第 2.4 节: 交付的二进制文件, 表 3: 支持的板卡和示例, 图 3: STemWin 的应用结构, 第 3 节: 支持的板卡和示例, 第 4.1 节: 配置, 第 4.1.1 节: GUIConf.c, 第 4.1.2 节: LCDConf.c, 第 4.1.4 节: GUI_X.c 或 GUI_X_OS.c, 第 4.2 节: GUI 初始化, 图 25, 表 10: FAQs 增加了 图 4: STemWin 初始化, 第 4.1.3 节: 硬件加速, 第 4.9.2 节: 如何添加新字体, 第 4.9.3 节: 语言支持, 第 4.11 节: 以不同的分辨率移植 GUI, 第 4.12 节: 如何添加新效果, 第 4.13 节: 如何添加新的小部件

表12. 中文文档版本历史

日期	版本	变更
2018年12 月11日	1	中文初始版本。



**重要通知 - 请仔细阅读**

意法半导体公司及其子公司（“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用，ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。本文档的中文版本为英文版本的翻译件，仅供参考之用；若中文版本与英文版本有任何冲突或不一致，则以英文版本为准。

© 2018 STMicroelectronics - 保留所有权利