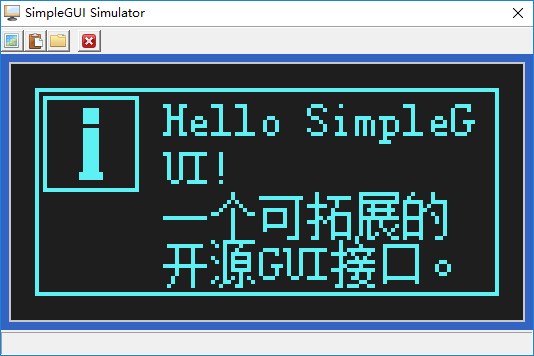
SimpleGUI

一套针对单色显示屏的开源GUI接口

**SimpleGUI API文档**



**开源，是一种态度。**

# 目录

[目录 2](#_Toc496474412)

[文件结构 3](#_Toc496474413)

[环境参数设置 4](#_Toc496474414)

[数据类型定义 5](#_Toc496474415)

[API 7](#_Toc496474416)

# 文件结构

Todo

# 环境参数设置

为适应可能出现的用户环境需求与变化，同时方便移植，SimpleGUI在实现代码中使用了一些宏定义作为机能的开关控制，用户可以通宏定义的不同值来对SimpleGUI的一些全局属性与逻辑操作进行修改或使能控制。以下为这些控制用宏定义的名字与使用方法。

1. \_SIMPLE\_GUI\_ENABLE\_ICONV\_GB2312\_

此宏定义关联文字显示API对非ASCII文字的解码方式。设置值大于0时有效，文字相关的API接口将会视输入的文字为UTF-8格式并转换成GB2312格式进行解析，若此值为0，则视输入文字为GB2312格式进行解析。此宏定义多用于模拟器环境中，依赖iconv库进行转码操作，如果需要在目标平台上使能此宏定义，需要用户自行编写转码函数与转码表。具体操作将会在“文字显示”章节详细阐述。

1. \_SIMPLE\_GUI\_VIRTUAL\_ENVIRONMENT\_SIMULATOR\_

此宏定义关联SimpleGUI的运行环境，设置值大于0时有效。当此宏定义有效，意味着SimpleGUI正运行于模拟环境中，用户在移植到目标平台后，请务必将此宏定义的值修改为0以避免不必要的错误。

1. \_SIMPLE\_GUI\_ENABLE\_DYNAMIC\_MEMORY\_

此宏定义关联SimpleGUI移植目标平台的动态内存操作， 设置值大于0时有效。当此宏定义设置为有效时，意味着目标平台已实现了动态内存管理或支持动态内存管理，且用户已经做好相应的实现或移植，此时SimpleGUI的相关依赖内容也会被使能，如列表项目的动态增减。

1. \_SIMPLE\_GUI\_ENABLE\_BASIC\_FONT\_

此宏定义关联SimpleGUI的内置基础字体， 设置值大于0时有效。SimpleGUI内部存储了一组尺寸为6\*8像素的可见ASCII字符集数据，当此宏定义有效时，此套字体数据将被支持，且文字显示API将可以使用此文字数据显示基本ASCII字符内容。

此功能设计为，如果用户使用外部字库且外部字库出现损坏或数据异常时，可以使用此字库数据输出一些警告或调试信息。

这些宏定义推荐在开发环境或Makefile中进行过全局定义，如有特殊需求需要在文件中定义，请用户自行创建文件并在必要的处理中包含此文件。

# 数据类型定义

1. **基本数据类型定义。**

为避免因平台和编译器的差异造成的对基本数据类型的定义不同，进而导致代码产生不可预知的错误，SimpleGUI在SGUI\_Typedef.h文件中重新定义了包括整数、字符、字符指针在内的一系列基本数据类型，在SimpleGUI内部的代码实现中将一致性的使用重新定义过的数据类型，用户需要在使用前明确所在平台的数据类型定义并作出相应修改，以避免因溢出或其他异常导致的错误。

具体内容请参照SGUI\_Typedef.h文件。

1. **特殊数据类型定义。**

为方便API的实现、表达与使用，SimpleGUI在基本数据类型的基础上，定义了一些特殊的数据结构。

* 1. **矩形区域**

此定义主要用于位图绘制时，标示显示区域、实际位图数据与位图数据偏移，其原型定义如下：

typedef struct \_st\_rectangle\_

{

SGUI\_INT PosX;

SGUI\_INT PosY;

SGUI\_INT Width;

SGUI\_INT Height;

}SGUI\_RECT\_AREA;

PosX为矩形左上角的X坐标。

PosY为矩形左上角的Y坐标。

Width为矩形宽度。

Height为矩形高度。

由于此数据结构是为绘图设计，在实际使用中，除了以上四项参数外，可能还需要其他判断，为此，此数据类型还有相应的运算宏定义：

#define RECTANGLE\_X\_START(ST) ((ST).PosX)

#define RECTANGLE\_X\_END(ST) (((ST).PosX + (ST).Width - 1))

#define RECTANGLE\_Y\_START(ST) ((ST).PosY)

#define RECTANGLE\_Y\_END(ST) (((ST).PosY + (ST).Height - 1))

#define RECTANGLE\_WIDTH(ST) ((ST).Width)

#define RECTANGLE\_HEIGHT(ST) ((ST).Height)

#define RECTANGLE\_VALID\_WIDTH(ST)

((RECTANGLE\_X\_START(ST)>0)?RECTANGLE\_WIDTH(ST):(RECTANGLE\_WIDTH(ST)+RECTANGLE\_X\_START(ST)))

#define RECTANGLE\_VALID\_HEIGHT(ST)

((RECTANGLE\_Y\_START(ST)>0)?RECTANGLE\_HEIGHT(ST):(RECTANGLE\_HEIGHT(ST)+RECTANGLE\_Y\_START(ST)))

RECTANGLE\_X\_START意为矩形左边界X轴坐标。

RECTANGLE\_X\_END意为矩形右边界X坐标。

RECTANGLE\_Y\_START意为矩形上边界Y坐标。

RECTANGLE\_Y\_END意为矩形下边界Y坐标。

RECTANGLE\_WIDTH意为矩形宽度。

RECTANGLE\_HEIGHT意为矩形高度。

RECTANGLE\_VALID\_WIDTH意为矩形的可见宽度。

RECTANGLE\_VALID\_HEIGHT意为矩形的可见高度。

* 1. **实时时钟**

此定义主要用于在HMI引擎中调用和显示实时时钟，与共通函数中的GetNowTime函数搭配使用，原型定义如下：

typedef struct

{

SGUI\_UINT16 Year;

SGUI\_UINT16 Month;

SGUI\_UINT16 Day;

SGUI\_UINT16 Hour;

SGUI\_UINT16 Minute;

SGUI\_UINT16 Second;

}SGUI\_TIME;

其中Year、Month、Day、Hour、Minute、Second分别对应年、月、日时、分、秒。

# API

1. **绘图API**

绘图API的实现文件位于GUI文件夹中，主要负责显示屏硬件的控制与屏幕绘图。

SimpleGUI的绘图API名称全部遵从以下格式:

SGUI\_分类\_函数名(参数…)

所有绘图API均以SGUI开头，分类表示这个函数的用途类型，后面的函数名则是标识函数的具体用途。

* 1. **共通处理**

共通处理函数实现在SGUI\_Common.c文件中，主要负责SimpleGUI的全局共通处理和SimpleGUI与外部的数据交互，函数名全部以SGUI\_Common开始。

* + 1. **SGUI\_Common\_IntegerToStringWithDecimalPoint**

**功能描述：**将一个有符号整数转化为字符串并在指定位置插入小数点。

**原型声明：**SGUI\_SIZE SGUI\_Common\_IntegerToStringWithDecimalPoint

(SGUI\_INT iInteger, SGUI\_UINT uiDecimalPlaces, SGUI\_PSZSTR pszStringBuffer, SGUI\_INT iAlignment, SGUI\_CHAR cFillCharacter)

**参数说明：**

iInteger：将要被转换的数字。

uiDecimalPlaces：小数位数，如果为0则不插入小数点。

pszStringBuffer：转换输出字符串的输出缓存。

iAlignment：对其方式与宽度，单位为半角字符宽度，大于0则右对齐，小于0则左对齐。如果转换完的宽度大于对齐宽度，则以转换完的实际宽度为准。

cFillCharacter：若对其后还有留白位置，则以此字符填充，通常使用空格。

**返回值：**转换的字符串长度。

**注意事项：**请注意转换输出缓冲区的长度，如果发生内存越界，将产生不可预知的错误。

* + 1. **SGUI\_Common\_IntegerToString**

**功能描述：**将一个有符号整数转化为字符串。

**原型声明：**SGUI\_SIZE SGUI\_Common\_IntegerToString

(SGUI\_INT iInteger, SGUI\_PSZSTR pszStringBuffer, SGUI\_UINT uiBase, SGUI\_INT iAlignment, SGUI\_CHAR cFillCharacter)

**参数说明：**

iInteger：将要被转换的数字。

pszStringBuffer：转换输出字符串的输出缓存。

uiBase：转换基数，只允许8、10和16进制。

iAlignment：对其方式与宽度，单位为半角字符宽度，大于0则右对齐，小于0则左对齐。如果转换完的宽度大于对齐宽度，则以转换完的实际宽度为准。

cFillCharacter：若对其后还有留白位置，则以此字符填充，通常使用空格。

**返回值：**转换的字符串长度。

**注意事项：**请注意转换输出缓冲区的长度，如果发生内存越界，将产生不可预知的错误。

* + 1. **SGUI\_Common\_ConvertStringToUnsignedInteger**

**功能描述：**将一个字符串的有效部分转换为一个无符号整数。

**原型声明：**SGUI\_UINT SGUI\_Common\_ConvertStringToUnsignedInteger

(SGUI\_PSZSTR szString, SGUI\_CHAR\*\* ppcEndPointer, SGUI\_UINT uiBase)

**参数说明：**

szString：将要被转换的字符串。

ppcEndPointer：转换结束处的字符指针，如果字符串中出现了非数字字符，则会在该处终止，同时此指针指向该处，若转换至字符串尾，则该指针也指向字符串尾的NULL。

uiBase：转换基数，只允许8、10和16进制。

**返回值：**转换的数字。

**注意事项：**若输入的字符串第一个字符即为非数字，则返回值为0。

* + 1. **SGUI\_Common\_ConvertStringToInteger**

**功能描述：**将一个字符串的有效部分转换为一个有符号整数。

**原型声明：**SGUI\_INT SGUI\_Common\_ConvertStringToInteger

(SGUI\_PSZSTR szString, SGUI\_CHAR\*\* ppcEndPointer, SGUI\_UINT uiBase)

**参数说明：**

szString：将要被转换的字符串。

ppcEndPointer：转换结束处的字符指针，如果字符串中出现了非数字字符，则会在该处终止，同时此指针指向该处，若转换至字符串尾，则该指针也指向字符串尾的NULL。

uiBase：转换基数，只允许8、10和16进制。

**返回值：**转换的数字。

**注意事项：**若输入的字符串第一个字符即为非数字，则返回值为0。

* + 1. **SGUI\_Common\_EncodeConvert**

**功能描述：**字符串编码转换。

**原型声明：**SGUI\_PSZSTR SGUI\_Common\_EncodeConvert

(SGUI\_PCSZSTR szSourceEncode, SGUI\_PSZSTR szDestinationEncode, SGUI\_PSZSTR szSource)

**参数说明：**

szSourceEncode：源字符串编码。

szDestinationEncode：目标编码。

szSource：要转换的字符串。

**返回值：**转换缓冲的指针。

**注意事项：**此函数依赖iconv外部库，用于在模拟器环境中使用UTF-8编码格式的字符串，由于示例的字符解码使用GB2312格式，故需要转码。由于转码库体积庞大，所以通常情况下此函数不需要在目标单片机平台上实现，届时只需要关闭\_SIMPLE\_GUI\_ENABLE\_ICONV\_GB2312\_宏定义即可。

* + 1. **SGUI\_Common\_Allocate**

**功能描述：**申请堆内存空间。

**原型声明：**SGUI\_Common\_Allocate(SGUI\_SIZE uiSize)

**参数说明：**

**uiSize：**要申请的字节数。

**返回值：**申请到的内存空间头指针。

**注意事项：**模拟环境中，此环境是对C标准函数malloc的重新封装，移植到目标平台后如果想使用此函数，请确认平台支持或用户已自行实现MMU后，重写此函数，并将宏\_SIMPLE\_GUI\_ENABLE\_DYNAMIC\_MEMORY\_有效化。

* + 1. **SGUI\_Common\_Free**

**功能描述：**释放堆内存空间。

**原型声明：**SGUI\_Common\_Free(void\* pFreePointer)

**参数说明：**

pFreePointer：要释放的内存头指针。

**返回值：无。**

**注意事项：**模拟环境中，此环境是对C标准函数free的重新封装，移植到目标平台后如果想使用此函数，请确认平台支持或用户已自行实现MMU后，重写此函数，并将宏\_SIMPLE\_GUI\_ENABLE\_DYNAMIC\_MEMORY\_有效化。

* + 1. **SGUI\_Common\_MemoryCopy**

**功能描述：**复制内存块。

**原型声明：**void\* SGUI\_Common\_MemoryCopy

(void\* pDest, const void\* pSrc, SGUI\_UINT uiSize)

**参数说明：**

pDest：目标内存块头指针。

pSrc：源内存块头指针。

uiSize：复制内存块的大小，单位字节。

**返回值：**目标内存块头指针。

**注意事项：**模拟环境中，此环境是对C标准函数memcpy的重新封装，用户如果不使用标准库，则需自行实现内存复制过程。

* + 1. **SGUI\_Common\_MemorySet**

**功能描述：**设置内存块**。**

**原型声明：**void SGUI\_Common\_MemorySet

(void\* pMemoryPtr, SGUI\_BYTE iSetValue, SGUI\_UINT uiSize)

**参数说明：**

pMemoryPtr：要设置的内存块头指针。

iSetValue：要设置的每一个字节的值。

uiSize：内存块的大小，单位字节。

**返回值：**目标内存块头指针。

**注意事项：**模拟环境中，此环境是对C标准函数memcpy的重新封装，用户如果不使用标准库，则需自行实现内存复制过程。

* + 1. **SGUI\_Common\_StringLength**

**功能描述：**测量字符串的长度。

**原型声明：**SGUI\_SIZE SGUI\_Common\_StringLength

(SGUI\_PCSZSTR szString)

**参数说明：**

szString：字符串头指针。

**返回值：**字符串长度。

**注意事项：**模拟环境中，此环境是对C标准函数strlen的重新封装，用户如果不使用标准库，则需自行实现字符串长度计算的过程。

* + 1. **SGUI\_Common\_StringCopy**

**功能描述：**复制字符串。

**原型声明：**SGUI\_PSZSTR SGUI\_Common\_StringCopy(SGUI\_PSZSTR szDest, SGUI\_PCSZSTR szSrc)

**参数说明：**

szDest：复制的字符串缓存。

szSrc：被复制的字符串缓存。

**返回值：**复制的字符串缓存头指针。

**注意事项：**模拟环境中，此环境是对C标准函数strcpy的重新封装，用户如果不使用标准库，则需自行实现字符串复制的过程。

* + 1. **SGUI\_Common\_StringLengthCopy**

**功能描述：**复制不超过特定长度的字符串。

**原型声明：**SGUI\_PSZSTR SGUI\_Common\_StringLengthCopy

(SGUI\_PSZSTR szDest, SGUI\_PCSZSTR szSrc, SGUI\_SIZE uiSize)

**参数说明：**

szDest：复制的字符串缓存。

szSrc：被复制的字符串缓存。

uiSize：复制字符串的长度，单位为字节。

**返回值：**复制的字符串缓存头指针。

**注意事项：**模拟环境中，此环境是对C标准函数strncpy的重新封装，用户如果不使用标准库，则需自行实现字符串复制的过程。

* + 1. **SGUI\_Common\_GetNowTime**

**功能描述：**获取当前特定系统时间。

**原型声明：**void SGUI\_Common\_GetNowTime

(SGUI\_TIME\* pstTime)

**参数说明：**

pstTime：保存时间数据的结构体。

**返回值：**复制的字符串缓存头指针。

**注意事项：**此函数需绑定系统RTC的处理，需要用户更具目标平台自行实现，读取RTC时间并赋值到参数指定的RTC结构体中。

* + 1. **SGUI\_Common\_RefreshScreen**

**功能描述：**刷新屏幕显示。

**原型声明：**void SGUI\_Common\_RefreshScreen(void)

**参数说明：**无。

**返回值：**无。

**注意事项：**用于更新屏幕显示的接口，需要用户自行实现，通常用于使用了显示缓存的情况下。

* + 1. **SGUI\_Common\_ReadFlashROM**

**功能描述：**读取外部存储中的数据。

**原型声明：**void SGUI\_Common\_ReadFlashROM(SGUI\_ROM\_ADDRESS uiAddressHead, SGUI\_SIZE uiDataLength, SGUI\_BYTE\* pBuffer)

**参数说明：**

uiAddressHead：读取的首地址。

uiDataLength：读取数据的长度。

pBuffer：存放读取数据的缓冲区的首地址。

**返回值：**无。

**注意事项：**此函数需根据实际系统平台，实现从内部或外部Flash中读取数据的操作。在SimpleGUI中，此函数通常用于读取字模、图片或图标数据等。

* + 1. **SGUI\_Common\_Delay**

**功能描述：**延时等待。

**原型声明：**void SGUI\_Common\_Delay(SGUI\_UINT32 uiTimeMs)

**参数说明：**

uiTimeMs：等待的毫秒数。

**返回值：**无。

**注意事项：**根据实际系统平台实现的延时函数，用于HMI引擎中，绘图引擎不使用此函数。

* 1. **坐标系定义**

在SimpleGUI的绘图操作中，以屏幕有效显示区域的左上角为坐标原点，X轴向右为正方向，Y轴向下为正方向，起始坐标为(0,0)，以屏幕像素为单位向正方向增长。

* 1. **基础绘图**

基础绘图函数的实现位于SGUI\_Basic.c文件中，主要负责SimpleGUI中点、线、面和位图等基本图形的绘制功能，函数名全部以SGUI\_Basic开始。

* + 1. **数据类型定义**

为方便基础绘图API的实现与使用，SimpleGUI在此部分定义了两种数据类型分别用于像素点颜色的说明和绘图方式的说明。

* 像素点颜色类型为SGUI\_COLOR，其原型定义如下：

typedef enum

{

SGUI\_COLOR\_BKGCLR = 0,

SGUI\_COLOR\_FRGCLR = 1,

SGUI\_COLOR\_TRANS = 2,

}SGUI\_COLOR;

SGUI\_COLOR\_BKGCLR为背景色，通常指显示屏幕上未被点亮（或未被有效化）的点的颜色。

SGUI\_COLOR\_FRGCLR为前景色，通常指显示屏幕上被点亮（或被有效化）的点的颜色。

SGUI\_COLOR\_TRANS为透明。指忽略当前点的绘制，保持当前点（或区域）的显示状态不变。

* 绘图模式型为SGUI\_DRAW\_MODE，其原型定义如下：

typedef enum

{

SGUI\_DRAW\_NORMAL = 0,

SGUI\_DRAW\_REVERSE = 1,

}SGUI\_DRAW\_MODE;

SGUI\_DRAW\_NORMAL为正常绘制，指根据上述SGUI\_COLOR类型中对点颜色的定义进行绘制。

SGUI\_DRAW\_REVERSE为反色绘制，指在绘制是将上述SGUI\_COLOR类型中的SGUI\_COLOR\_BKGCLR与SGUI\_COLOR\_FRGCLR两种定义互换，绘制反色图形，此定义通常用于绘制反色位图。

* + 1. **SGUI\_Basic\_ClearScreen**

**功能描述：**清空屏幕显示。

**原型声明：**void SGUI\_Basic\_ClearScreen(void)

**参数说明：**无。

**返回值：**无。

**注意事项：**因需要适配的硬件平台不同，有的屏幕设备支持清空操作，有的则必须手动填充，所以此函数需要用户根据实际使用的设备自行实现。

* + 1. **SGUI\_Basic\_DrawPoint**

**功能描述：**在屏幕上绘制点。

**原型声明：**void SGUI\_Basic\_DrawPoint(SGUI\_UINT uiPosX, SGUI\_UINT uiPosY, SGUI\_COLOR eColor)

**参数说明：**

uiPosX：要绘制点的X坐标。

uiPosY：要绘制点的Y坐标。

eColor：绘制点的颜色。

**返回值：**无。

**注意事项：**SimpleGUI只针对单色屏幕设计，绘制点的颜色只有“黑”和“白”两种，分别对应像素的设置状态和清空状态，详情请参考SGUI\_COLOR\_BKGCLR数据类型的定义。

* + 1. **SGUI\_Basic\_DrawLine**

**功能描述：**在屏幕上绘制直线段。

**原型声明：**void SGUI\_Basic\_DrawLine

(SGUI\_INT uiStartX, SGUI\_INT uiStartY, SGUI\_INT uiEndX, SGUI\_INT uiEndY, SGUI\_COLOR eColor)

**参数说明：**

uiStartX：线段起始点的X坐标。

uiStartY：线段起始点的Y坐标。

uiStartX：线段终止点的X坐标。

uiStartY：线段终止点的Y坐标。

eColor：绘制线段的颜色。

**返回值：**无。

**注意事项：**线段起止点的坐标值可以为负值，当为负值时意为坐标位于屏幕显示区域上侧或左侧以外的区域。超出屏幕显示区域部分的线段将不会被绘制。

* + 1. **SGUI\_Basic\_DrawRectangle**

**功能描述：**在屏幕上绘制封闭矩形。

**原型声明：**void SGUI\_Basic\_DrawRectangle

(SGUI\_UINT uiStartX, SGUI\_UINT uiStartY, SGUI\_UINT uiWidth, SGUI\_UINT uiHeight, SGUI\_COLOR eEdgeColor, SGUI\_COLOR eFillColor)

**参数说明：**

uiStartX：矩形左上角点的X坐标。

uiStartY：矩形左上角点的Y坐标。

uiWidth：矩形的宽度，以像素为单位。

uiStartY：矩形的高度，以像素为单位。

eEdgeColor：矩形边框的颜色。

eFillColor：矩形内部的填充颜色。

**返回值：**无。

**注意事项：**如果只需要绘制矩形边框，不希望矩形内部被填充，则eFillColor参数传入SGUI\_COLOR\_TRANS即可。超出屏幕显示区域的部分将不会被绘制。

* + 1. **SGUI\_Basic\_DrawCircle**

**功能描述：在屏幕上绘制封闭圆形。**

**原型声明：**void SGUI\_Basic\_DrawCircle(SGUI\_UINT uiCx, SGUI\_UINT uiCy, SGUI\_UINT uiRadius, SGUI\_COLOR eEdgeColor, SGUI\_COLOR eFillColor)

**参数说明：**

uiCx：圆形圆心的X坐标。

uiCy：圆形圆心的Y坐标。

uiRadius：圆的半径，以像素为单位。

eEdgeColor：圆的边框的颜色。

eFillColor：圆的内部的填充颜色。

**返回值：**无。

**注意事项：**如果只需要绘制圆形边框，不希望矩形内部被填充，则eFillColor参数传入SGUI\_COLOR\_TRANS即可。

* + 1. **SGUI\_Basic\_ReverseBlockColor**

**功能描述：**反色矩形区域。

**原型声明：**void SGUI\_Basic\_ReverseBlockColor(SGUI\_UINT uiStartX, SGUI\_UINT uiStartY, SGUI\_UINT uiWidth, SGUI\_UINT uiHeight)

**参数说明：**

uiStartX：矩形左上角点的X坐标。

uiStartY：矩形左上角点的Y坐标。

uiWidth：矩形的宽度，以像素为单位。

uiStartY：矩形的高度，以像素为单位。

**返回值：**无。

**注意事项：**无。

* + 1. **SGUI\_Basic\_DrawBitMap**

**功能描述：**绘制位图。

**原型声明：**void SGUI\_Basic\_DrawBitMap(SGUI\_RECT\_AREA\* pstDisplayArea, SGUI\_RECT\_AREA\* pstDataArea, SGUI\_BYTE\* pDataBuffer, SGUI\_DRAW\_MODE eDrawMode)

**参数说明：**

pstDisplayArea：指定位图的显示区域。

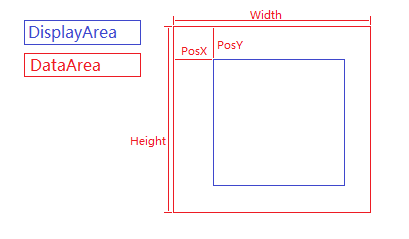
pstDataArea：位图大小以及显示的偏移量，以像素为单位。

pDataBuffer：位图数据。

eDrawMode：绘制方式（正常或反色）。

**返回值：**无。

**注意事项：**传入参数中pstDisplayArea指向的数据为显示位图的矩形区域，超出区域的部分将不被显示，pstDataArea指向的数据标明了位图的实际大小以及在显示区域内的偏移量，具体逻辑关系如下图所示：

****

图中红色代表位图的实际尺寸，蓝色代表指定的显示区域尺寸，则实际显示时，只有蓝色区域的中的位图被显示出来。

* 1. **文字显示**

文字显示函数的实现位于SGUI\_Text.c文件中，主要负责SimpleGUI中文字的绘制，包括单字、单行文字、多行文字等，函数名全部以SGUI\_Text开始。

文字显示函数依赖基础绘图中的位图绘制函数，实现上，所有文字均作为位图处理。

* + 1. **数据类型定义**

为方便文字显示API的实现与使用，SimpleGUI在此部分定义了用于文字处理使用的特殊数据类型。

* 字模数据源类型定义为SGUI\_FONT\_MODULE\_SOURCE，其具体定义如下：

typedef enum

{

SGUI\_FONT\_MODULE\_SOURCE\_BASIC = 0,

SGUI\_FONT\_MODULE\_SOURCE\_NORMAL,

}SGUI\_FONT\_MODULE\_SOURCE;

SGUI\_FONT\_MODULE\_SOURCE\_BASIC意为使用内置基础字模数据。

SGUI\_FONT\_MODULE\_SOURCE\_NORMAL意为使用用户定义的字模数据。

* 字模的字形尺寸数据类型定义为SGUI\_FONT\_SIZE\_STRUCT，其具体定义如下：

typedef enum

{

const SGUI\_UINT16 Size;

const SGUI\_UINT16 Width;

const SGUI\_UINT16 Height;

}SGUI\_FONT\_SIZE\_STRUCT;

Size意为该型文字的字号，通常指文字的高度，以像素为单位。

Width意为该字形文字中半角字符的宽度，一般为高度的一半。

Height意为该字形文字中文字的高度，以像素为单位。

* 字模字号类型定义为SGUI\_FONT\_SIZE，具体定义可根据用户实际使用的字体数据任意指定。
  + 1. **SGUI\_Text\_ReadFontData**

**功能描述：**读取字模数据。

**原型声明：**void SGUI\_Text\_ReadFontData

(SGUI\_FONT\_SIZE eFontSize, SGUI\_UINT16 uiCharacterCode, SGUI\_BYTE\* pFontDataBuffer, SGUI\_UINT32 uiFontBufferSize)

**参数说明：**

eFontSize：使用的文字字号。

uiCharacterCode：字符编码。

pFontDataBuffer：读取的字模数据缓冲区头指针。

uiFontBufferSize：字模数据缓冲区大小。

**返回值：**无。

**注意事项：**此函数用于根据传入的字号和字码取得相应的字模数据，每一个文字的字模数据都相当于一组位图数据，在运行时，此函数依赖SGUI\_Text\_GetCharacterTableIndex函数获取文字在字模数据表中的索引以计算偏移量，然后根据字号数据计算单个字符数据的大小，最后再调用位图绘制接口SGUI\_Basic\_DrawBitMap将文字以位图的形式绘制在屏幕上。

如果用户的字模数据保存在片外Flash存储器上，则应实现共同函数中的SGUI\_Common\_ReadFlashROM函数，并重写此函数以用于从Flash中读取字模数据。

* + 1. **SGUI\_Text\_GetCharacterTableIndex**

**功能描述：**读取字模数据。

**原型声明：**SGUI\_SIZE SGUI\_Text\_GetCharacterTableIndex

(SGUI\_UINT16 uiCharacterCode)

**参数说明：**

uiCharacterCode：字符编码。

**返回值：**字符数据索引。

**注意事项：**此函数用于根据传入的字符编码获取字符在对应的字模数据中的索引号，用户应根据自己使用的字符编码以及使用的字符表重新编写此函数，用以计算字符与索引的相对关系。

SimpleGUI模拟环境中，提供了8像素、32像素的可见ASCII字符的字模数据以及12像素和16像素的GB2312一级编码的中英文混合字模数据，但GB2312编码数据中删除了日语假名、俄文字母等不常用字符，所以索引数和标准GB2312有所差异，使用者可根据实际情况，自行重新制作字模数据。

* + 1. **SGUI\_Text\_DrawSingleLineText**

**功能描述：**绘制单行文字。

**原型声明：**void SGUI\_Text\_DrawSingleLineText

(SGUI\_PCSZSTR szTextBuffer, SGUI\_FONT\_SIZE eFontSize, SGUI\_RECT\_AREA\* pstDisplayArea, SGUI\_RECT\_AREA\* pstStringDataArea, SGUI\_DRAW\_MODE eFontMode)

**参数说明：**

szTextBuffer：要显示的文字。

eFontSize：使用的字号。

pstDisplayArea：文字的显示区域。

pstStringDataArea：文字的数据区域位置偏移量。

eFontMode：绘制方式（正常或反色）。

**返回值：**无。

**注意事项：**此函数用于根据传入的字符编码获取字符在对应的字模数据中的索引号，用户应根据自己使用的字符编码以及使用的字符表重新编写此函数，用以计算字符与索引的相对关系。

SimpleGUI模拟环境中，提供了8像素、32像素的可见ASCII字符的字模数据以及12像素和16像素的GB2312一级编码的中英文混合字模数据，但GB2312编码数据中删除了日语假名、俄文字母等不常用字符，所以索引数和标准GB2312有所差异，使用者可根据实际情况，自行重新制作字模数据。

* + 1. **SGUI\_Text\_DrawMultipleLinesText**

**功能描述：**绘制多行文字。

**原型声明：**SGUI\_SIZE SGUI\_Text\_DrawMultipleLinesText

(SGUI\_PCSZSTR szTextBuffer, SGUI\_FONT\_SIZE eFontSize, SGUI\_RECT\_AREA\* pstDisplayArea, SGUI\_INT iTopOffset, SGUI\_DRAW\_MODE eFontMode)

**参数说明：**

szTextBuffer：要显示的文字。

eFontSize：使用的字号。

pstDisplayArea：文字的显示区域。

iTopOffset：文字绘制顶部的偏移量，以像素为单位。

eFontMode：绘制方式（正常或反色）。

**返回值：**文字在指定显示区域内所占的实际行数。

**注意事项：**此函在逻辑上，将用户传入的字符串在用户的指定区域内，以用户传入的iTopOffset参数为顶端偏移量，逐个尝试绘制，但实际只绘制用于指定区域内可见的文字（包括不完整的文字），并计算在用户指定的显示区域内，以用户指定宽度为最大宽度的前提下文字将可能被显示为多少行，并将此数字作为返回值返回（具体逻辑请参考基础绘图1.3.8章节），SimpleGUI模拟环境演示例程第一页中的滚动文字就是使用此函数实现。

* + 1. **SGUI\_Text\_GetTextGraphicsWidth**

**功能描述：**获取单行文字图形的宽度。

**原型声明：**SGUI\_SIZE SGUI\_Text\_GetTextGraphicsWidth

(SGUI\_PCSZSTR szText, SGUI\_FONT\_SIZE eFontSize)

**参数说明：**

szText：待分析的文字。

eFontSize：使用的字号。

**返回值：**单行文字图形所占的实际宽度。

**注意事项：**此函数通过测量以字节为单位的字符串长度并与使用字体的半角字符宽度相乘，计算绘制单行文字时可能占用的最大宽度，由于演示程序使用GB2312编码，基本ASCII字符占用1字节，其他字符占用两字节，故此可以使用此方式计算，如果用户使用其他编码格式，请根据实际使用情况重写此函数。

* + 1. **SGUI\_Text\_GetMultiLineTextLines**

**功能描述：**获取多行文字可能被拆分为的行数。

**原型声明：**SGUI\_SIZE SGUI\_Text\_GetMultiLineTextLines

(SGUI\_PCSZSTR szNoticeText, SGUI\_SIZE uiHalfWidthCharInLine)

**参数说明：**

szNoticeText：待分析的文字。

uiHalfWidthCharInLine：每行容纳的最多的半角字符数。

**返回值：**文字可能被拆分为的行数。

**注意事项：**此函数将计算在单行容许的最大半角字符数时，字符串可能被拆分的行数。

* 1. **消息框**
  2. **输入框**
  3. **进度条**
  4. **滚动条**
  5. **列表框**
  6. **实时曲线**

1. **HMI引擎**