RTOS_Melis 图形界面

开发指南

版本号: 1.0

发布日期: 2021.05.21

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.05.21	KPA0511	建立初始版本
			“”

版权所有 © 珠海全志科技股份有限公司。保留一切权利

目　录

ALLWINNER®

版权所有 © 珠海全志科技股份有限公司。保留一切权利

版权所有 © 珠海全志科技股份有限公司。保留一切权利

ALLWINNER®

版权所有 © 珠海全志科技股份有限公司。保留一切权利

版权所有 © 珠海全志科技股份有限公司。保留一切权利

版权所有 © 珠海全志科技股份有限公司。保留一切权利

1 前言

1.1 文档简介

本文档主要介绍在 Melis 上开发应用程序需具备的基本知识和应用程序开发的流程，让开发者可以快速掌握在 Melis 进行应用开发的基本技能。

1.2 目标读者

- UI 中间件开发人员/维护人员
- 图形界面应用开发者

1.3 适用范围

表 1-1：适用产品列表

产品名称	内核版本	相关文件
F133	Melis	UI 中间件：melis-v3.0/source/emodules/mod_orange 应用：melis-v3.0/source/livedesk

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2 应用组织框架介绍

2.1 应用启动流程介绍

图 2-1: startup 流程

版权所有 © 珠海全志科技股份有限公司。保留一切权利

应用启动是由安装 startup.sh 中指定的文件开始的，命令行程序会对该脚本中指定的文件，按顺序对其进行 insmod 指令操作。首先安装 orange.mod 图形界面库，然后安装应用程序 desktop.mod。

```
source\projects\f133-prototype-machine\data\UDISK\startup.sh

#echo "Execute startup script begin............."
insmod d:\mod\orange.mod
insmod d:\apps\desktop.mod
#echo "..............Execute startup script end"
```

创建 desktop 时，首先初始化消息机制，包含了消息的收集任务和转发任务的创建。然后创建应用主线程，包含了应用相关资源初始化，和主界面 DESKTOP 创建，最后启动消息循环。

```
__mp *INIT_MOpen(__u32 mid, __u32 mod)
{
    msg_emit_init();//初始化消息收集转发函数
    init_data.mid      = mid;
    init_data.init_tid = esKRNL_TCreate(application_init_process, NULL, 0x10000,
    KRNL_priolevel3);//应用主线程
    esKRNL_TaskNameSet(init_data.init_tid, "initprocess");
    if (init_data.init_tid == 0)
    {
        return NULL;
    }
    return (__mp *)&init_data;
}
```

2.2 主界面介绍

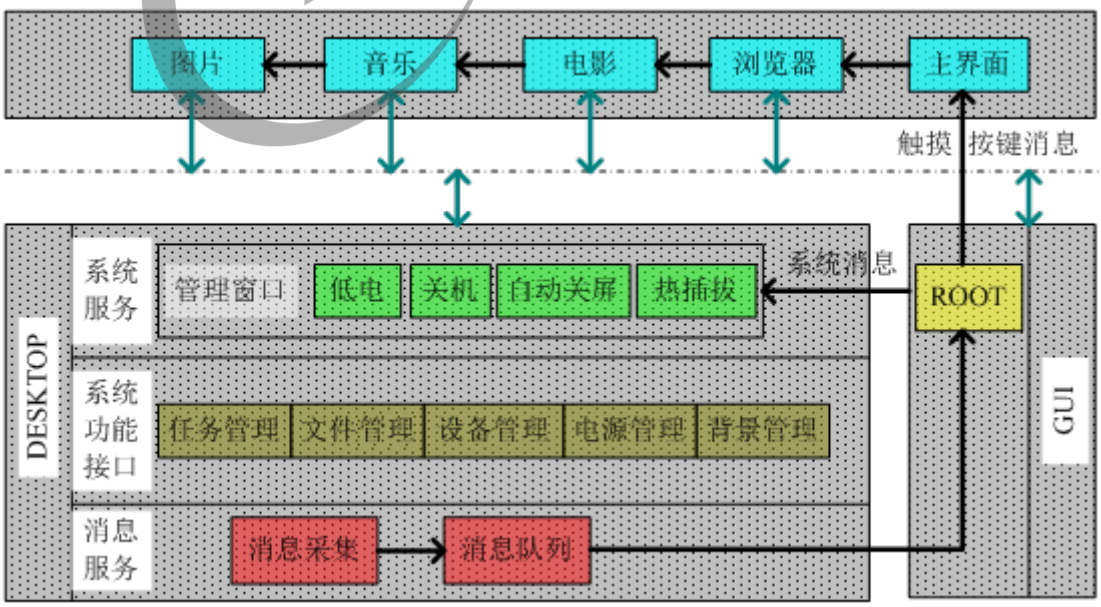

图 2-2: 主界面架构

ALLWINNER®

桌面系统主要包含如下三个部分：

1. 提供消息采集和消息发送服务，发送对象是 GUI 系统，如触摸，按键等消息会汇集，直接处理或者转发到焦点应用的对应回调函数中处理；
2. 提供系统接口（任务管理，文件管理，设备管理，电源管理，背景管理等）；
3. 提供系统服务（低电、关机、自动关屏、自动关机、热插拔消息处理等）；

2.3 GUI 窗口体系和消息机制

Melis 应用程序由窗口组成，所有用户的操作（触摸、按键）都通过消息发送到窗口，在窗口的消息处理函数中进行处理。窗口体系和消息传递都在 GUI 实现。

2.3.1 窗口类型

- **Manage window(管理窗口)**

管理窗口是虚拟窗口，负责消息分发和处理，支持跨图层管理，它有两种用途：1. 应用程序的入口窗口，有自己的消息队列，负责消息接收和分发。2. 统筹多图层下的窗口消息处理，形成一个统一的整体。

- **Layer window(图层窗口)**

图层窗口对应一个图层，是屏幕管理的基本单位，是屏幕中的一个实体窗口，有自己的显示区域和 Z 序，也是其他实体窗口（frmwin 和控件窗口）的载体。

- **Frame window**

寄生在 framebuffer 上，实体窗口，有自己的矩形区域，可以再上面写字，画图。

- **Ctrl window(控件窗口)**

封装好的具有特定属性和操作的特殊窗口，如 button、slider 等。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2.4 窗口关系

父子关系

窗口可以指定父窗口，以此来建立窗口树，管理窗口的父窗口只能是管理窗口或者空窗口 (系统默认根窗口为其父窗口)，frame window 的父窗口只能是管理窗口，控件窗口的父窗口可以是 frame window，也可以是控件。

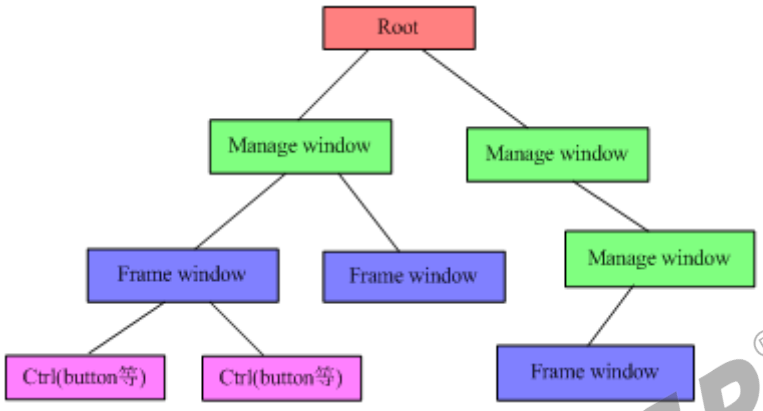

图 2-3: 典型窗口树

寄生关系

每个 frame window 都必需有一个 layer window 作为载体，即需要先创建一个 layer window，然后在创建 frame window 时将这个 layer window 作为参数传给它，我们就说这个 frame window 寄生在这个 layer window 上。

2.5 消息机制

外部事件消息

主要指由外设触发而由输入子系统传递给 GUI 消息接收器的消息，如触摸、按键等消息。

GUI 系统消息

主要指由外部事件或相应的函数触发的消息，如绘制、大小改变、设焦等相关的消息。

消息优先级

根据消息的优先级的不同又分为同步消息，异步消息（通知消息）。对同步消息而言，发送线程需要等消息执行完成之后再返回，同步消息往往用来处理高优先级的消息。异步消息往往用于非高优先级的消息，对异步消息而言，发送线程只需要将消息投递到指定的消息队列中之后然后返回，并不等消息完全执行完毕。需要注意的是，区分同步消息和异步消息并不在消息本身，而在选择发送消息的方式，`GUI_SendMessage()`发送同步消息，`GUI_SendNotifyMessage()`发送异步消息。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

消息路由

创建 Manage window 和 Frame window 时需要指定他们的窗口回调函数，即消息处理函数，所有消息的响应均在这个回调函数里实现。通过`GUI_SetActiveManWin()`可以指定哪个 Root 下面第一级管理窗口为 Active window，触摸消息会根据触摸区域选定焦点 Frame window，触摸消息通过 Active window 直接传递给焦点 Frame window；通过`GUI_WinSetFocusChild`可以指定按键消息焦点窗口（Manage window 或 Frame window），按键消息会从 Active window 逐级传递到焦点窗口。

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3 图形界面库 Orange

3.1 Orange 简介

Orange 是基于 Melis 操作系统之上的一套 GUI 系统，该系统支持多任务和多图层操作，允许在多个图层上面创建窗口，提供完善的异步和同步窗口消息通讯机制，提供点、线、矩形、扇形、椭圆、圆等基本形状的绘制和填充，支持 TTF 矢量字体、SFT 点阵字体及字体的加边框和加阴影特效，支持文本的显示输出和 BMP 位图的绘制，支持 button、static、listmenu、slider...等控件，支持多国语言和内存设备等。在 Orange 中，窗口是 Orange 管理的基本单位，Orange 采用事件驱动编程，窗口是接收事件并分发处理事件的最小单元，在 Orange 中所有的消息响应基于消息循环，窗口不断从消息队列中获取消息并分发给响应的窗口过程处理。

3.2 接口介绍

3.2.1 GUI_FrmWinCreate

```
PROTOTYPE
H_WIN  GUI_FrmWinCreate (pframewincreate create_info);

ARGUMENTS
create_info frmwin 创建信息结构

RETURNS
Frmwin 句柄

DESCRIPTION
创建 frmwin
```

3.2.2 GUI_ManWinCreate

```
PROTOTYPE
H_WIN  GUI_ManWinCreate (pmanwincreate create_info);

ARGUMENTS
create_info 管理窗口创建信息结构

RETURNS
管理窗口句柄
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
DESCRIPTION
创建管理窗口
```

3.2.3 GUI_FrmWinDelete

```
PROTOTYPE
__s32  GUI_FrmWinDelete(H_WIN  hframewin);

ARGUMENTS
hframewin   frmwin窗口句柄

RETURNS
ORANGE_OK   删除成功
ORANGE_FAIL 删除失败

DESCRIPTION
删除frmwin
```

3.2.4 GUI_ManWinDelete

```
PROTOTYPE
__s32  GUI_ManWinDelete (H_WIN  hmanwin);

ARGUMENTS
hmanwin          管理窗口的句柄

RETURNS
ORANGE_OK   删除成功
ORANGE_FAIL 删除失败

DESCRIPTION
删除管理窗口
```

3.2.5 GUI_CtrlWinCreate

```
PROTOTYPE
H_WIN  GUI_CtrlWinCreate (__gui_ctlwincreate_para_t*create_info);

ARGUMENTS
create_info 控件窗口创建信息结构

RETURNS
控件窗口句柄

DESCRIPTION
创建控件窗口
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.6 GUI_CtrlWinDelete

```
PROTOTYPE
___s32  GUI_CtrlWinDelete (H_WIN  hWnd);

ARGUMENTS
hWnd     窗口句柄

RETURNS
ORANGE_OK    删除成功
ORANGE_FAIL 删除失败

DESCRIPTION
删除控件窗口
```

3.2.7 GUI_WinThreadCleanup

```
PROTOTYPE
Void    GUI_WinThreadCleanup (H_WIN hManWnd);

ARGUMENTS
hManWnd 窗口句柄

RETURNS
None

DESCRIPTION
gui窗口线程相关的信息删除，用来在主窗口或者framewin窗口结束后，清楚与线程相关的信息。
```

3.2.8 GUI_SetActiveManWin

```
PROTOTYPE
H_WIN  GUI_SetActiveManWin (H_WIN  hWnd);

ARGUMENTS
hWnd     系统全局焦点管理窗口句柄

RETURNS
上一次的焦点管理窗口的句柄

DESCRIPTION
通过这个函数设置系统全局的焦点管理窗口句柄。该窗口通常为某个应用程序的入口窗口
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.9 GUI_SetActiveManWin

```
PROTOTYPE
H_WIN  GUI_SetActiveManWin (H_WIN  hWnd);

ARGUMENTS
hWnd     系统全局焦点管理窗口句柄

RETURNS
上一次的焦点管理窗口的句柄

DESCRIPTION
通过这个函数设置系统全局的焦点管理窗口句柄。该窗口通常为某个应用程序的入口窗口
```

3.2.10 GUI_WinGetFocusChild

```
PROTOTYPE
H_WIN  GUI_WinGetFocusChild (H_WIN h_win);

ARGUMENTS
H_WIN    父窗口句柄

RETURNS
H_WIN    焦点子窗口句柄

DESCRIPTION
获取焦点子窗口句柄
```

3.2.11 GUI_WinGetFocusChild

```
PROTOTYPE
__s32   GUI_WinSetFocusChild (H_WIN h_win);

ARGUMENTS
H_WIN    需要设置的焦点子窗口句柄

RETURNS
ORANGE_OK    设置成功
ORANGE_FAIL  设置失败

DESCRIPTION
设置焦点子窗口
```

ALLWINNER®

3.2.12 GUI_WinGetFocusChild

```
PROTOTYPE
__s32   GUI_WinSetFocusChild (H_WIN h_win);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口类型

DESCRIPTION
获取窗口类型
```

3.2.13 GUI_WinGetMainManWin

```
PROTOTYPE
H_WIN   GUI_WinGetMainManWin (H_WIN hWnd)

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的入口管理窗口句柄

DESCRIPTION
通过该窗口获取该应用程序的全局管理窗口的句柄
```

3.2.14 GUI_WinGetManWin

```
PROTOTYPE
H_WIN   GUI_WinGetManWin(H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的父管理窗口句柄

DESCRIPTION
通过该窗口获取该窗口的局部管理窗口的句柄。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.15 GUI_WinGetParent

```
PROTOTYPE
H_WIN   GUI_WinGetParent(H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的父窗口句柄

DESCRIPTION
通过该窗口获取该窗口的父窗口的句柄。
```

3.2.16 GUI_WinGetFirstChild

```
PROTOTYPE
H_WIN   GUI_WinGetFirstChild(H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的第一个子窗口句柄

DESCRIPTION
通过该窗口获取该窗口的第一个子窗口的句柄。
```

3.2.17 GUI_WinGetNextBro

```
PROTOTYPE
H_WIN   GUI_WinGetNextBro(H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的下一个兄弟窗口句柄

DESCRIPTION
通过该窗口获取该窗口的下一个兄弟窗口的句柄。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.18 GUI_WinGetNextHostedWin

```
PROTOTYPE
H_WIN   GUI_WinGetNextHostedWin(H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的下一个Hosted窗口句柄

DESCRIPTION
通过该窗口获取该窗口的下一个Hosted窗口的句柄。
```

3.2.19 GUI_WinGetFirstHostedWin

```
PROTOTYPE
H_WIN  GUI_WinGetFirstHostedWin(H_WIN hWnd);

ARGUMENTS
hWnd   窗口句柄

RETURNS
窗口的第一个Hosted窗口句柄

DESCRIPTION
通过该窗口获取该窗口的第一个Hosted窗口的句柄
```

3.2.20 GUI_WinGetOwnerWin

```
PROTOTYPE
H_WIN  GUI_WinGetOwnerWin(H_WIN hWnd);

ARGUMENTS
hWnd   窗口句柄

RETURNS
窗口的Owner窗口句柄

DESCRIPTION
通过该窗口获取该窗口的Owner窗口的句柄。
```

ALLWINNER®

3.2.21 GUI_WinGetRootWin

```
PROTOTYPE
H_WIN   GUI_WinGetRootWin (void);

ARGUMENTS
void

RETURNS
返回根窗口句柄

DESCRIPTION
获取根窗口句柄
```

3.2.22 GUI_WinIsAncestor

```
PROTOTYPE
__bool  GUI_WinIsAncestor (H_WIN hWnd, H_WIN hChild);

ARGUMENTS
hWnd    祖先窗口句柄
hChild   子孙窗口句柄

RETURNS
满足条件返回ORANGE_TRUE
否则返回ORANGE_FALSE

DESCRIPTION
判断hWnd窗口是否是hChild窗口的祖先窗口。
```

3.2.23 GUI_WinIsChild

```
PROTOTYPE
__bool  GUI_WinIsChild (H_WIN hWnd,H_WIN hPraent);

ARGUMENTS
hWnd     窗口句柄
hPraent  父窗口句柄

RETURNS
满足条件返回ORANGE_TRUE
否则返回ORANGE_FALSE

DESCRIPTION
判断hPraent窗口是否是hWnd窗口的父窗口
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.24 GUI_WinGetDlgItem

```
PROTOTYPE
H_WIN   GUI_WinGetDlgItem (H_WIN hDlg, __s32 nIDDlgItem);

ARGUMENTS
hDlg        父窗口句柄
nIDDlgItem  子窗口ID

RETURNS
子窗口句柄

DESCRIPTION
通过子窗口id来获取子窗口句柄。
```

3.2.25 GUI_WinGetItemId

```
PROTOTYPE
__u32   GUI_WinGetItemId (H_WIN hItem);

ARGUMENTS
hItem   控件窗口句柄

RETURNS
控件窗口ID

DESCRIPTION
获取控件窗口id。
```

3.2.26 GUI_WinGetHandFromName

```
PROTOTYPE
H_WIN   GUI_WinGetHandFromName (char * name);

ARGUMENTS
name    窗口名称

RETURNS
窗口的窗口句柄

DESCRIPTION
通过该窗口窗口名称该窗口的句柄。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.27 GUI_WinGetAddData

```
PROTOTYPE
__u32   GUI_WinGetAddData (H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的额外控制信息结构句柄

DESCRIPTION
获取该窗口的额外控制信息结构句柄。
```

3.2.28 GUI_WinSetAddData

```
PROTOTYPE
__s32   GUI_WinSetAddData(H_WIN hWnd, __u32 dwAddData);

ARGUMENTS
hWnd            窗口句柄
dwAddData       窗口额外控制信息结构地址

RETURNS
ORANGE_OK   设置成功
ORANGE_FAIL 设置失败

DESCRIPTION
设置该窗口的额外控制信息结构句柄。
```

3.2.29 GUI_WinGetStyle

```
PROTOTYPE
__u32   GUI_WinGetStyle (H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的类型信息

DESCRIPTION
获取窗口的类型信息。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.30 GUI_WinGetFrmWin

```
PROTOTYPE
H_WIN   GUI_WinGetFrmWin (H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
窗口的frmwin窗口句柄

DESCRIPTION
通过该窗口获取该窗口的frmwin窗口的句柄。
```

3.2.31 GUI_WinGetAttr

```
PROTOTYPE
void*   GUI_WinGetAttr (H_WIN h_win);

ARGUMENTS
h_win   窗口句柄

RETURNS
void*   窗口属性地址

DESCRIPTION
获取窗口属性(用户传递窗口时需要传递的数据)。
```

3.2.32 GUI_WinSetAttr

```
PROTOTYPE
__s32   GUI_WinSetAttr (H_WIN hWnd, void *attr);

ARGUMENTS
hWnd            窗口句柄
attr                窗口私有属性信息

RETURNS
ORANGE_OK   设置成功
ORANGE_FAIL 设置失败

DESCRIPTION
设置该窗口的窗口私有属性信息。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.33 GUI_WinGetLyrWin

```
PROTOTYPE
H_LYR  GUI_WinGetLyrWin(H_WIN h_win);

ARGUMENTS
h_win    窗口句柄

RETURNS
H_LYR    图层窗口句柄

DESCRIPTION
获取窗口所在图层的句柄
```

3.2.34 GUI_WinGetName

```
PROTOTYPE
__s32  GUI_WinGetName(H_WIN h_win, char * name);

ARGUMENTS
h_win    窗口句柄
name     窗口名字

RETURNS
ORANGE_OK    成功
ORANGE_FAIL 失败

DESCRIPTION
获取窗口名字
```

3.2.35 GUI_ManWinDefaultProc

```
PROTOTYPE
void    GUI_ManWinDefaultProc(__gui_msg_t * msg);

ARGUMENTS
msg      输入消息指针

RETURNS
void

DESCRIPTION
管理窗口默认处理函数
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.36 GUI_CtrlWinDefaultProc

```
PROTOTYPE
void   GUI_CtrlWinDefaultProc(__gui_msg_t * msg);

ARGUMENTS
msg     输入消息指针

RETURNS
void

DESCRIPTION
控件窗口默认处理函数
```

3.2.37 GUI_FrmWinDefaultProc

```
PROTOTYPE
void   GUI_FrmWinDefaultProc(__gui_msg_t * msg);

ARGUMENTS
msg     输入消息指针

RETURNS
void

DESCRIPTION
Frmwin窗口默认处理函数
```

3.2.38 GUI_WinSetCallback

```
PROTOTYPE
__pGUI_WIN_CB GUI_WinSetCallback (H_WIN h_win, __pGUI_WIN_CB cb);

ARGUMENTS
h_win   窗口句柄
cb      窗口回调函数

RETURNS
调整后的回调函数入口地址

DESCRIPTION
设置窗口回调函数
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.39 GUI_WinGetCallback

```
PROTOTYPE
__pGUI_WIN_CB GUI_WinGetCallback (H_WIN hWnd);

ARGUMENTS
hWnd     窗口句柄

RETURNS
调整后的回调函数入口地址

DESCRIPTION
获取窗口回调函数
```

3.2.40 GUI_WinSetNotifyCallback

```
PROTOTYPE
NOTIFPROC GUI_WinSetNotifyCallback(H_WIN hwnd, NOTIFPROC notif_proc);

ARGUMENTS
hwnd     窗口句柄
notif_proc   窗口通知回调函数

RETURNS
调整后的窗口通知回调函数入口地址

DESCRIPTION
设置窗口后处理回调函数
```

3.2.41 GUI_WinGetNotifyCallback

```
PROTOTYPE
NOTIFPROC GUI_WinGetNotifyCallback(H_WIN hwnd);

ARGUMENTS
hwnd     窗口句柄

RETURNS
调整后的窗口通知回调函数入口地址

DESCRIPTION
获取窗口后处理回调函数
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.42 GUI_LyrWinWinCreate

```
PROTOTYPE
H_LYR  GUI_LyrWinWinCreate(__gui_lyrwincreate_info_t * create_info);

ARGUMENTS
create_info     图层创建信息结构指针

RETURNS
图层句柄

DESCRIPTION
创建图层
```

3.2.43 GUI_LyrWinWinDelete

```
PROTOTYPE
__s32  GUI_LyrWinWinDelete (H_LYR h_lyr);

ARGUMENTS
h_lyr   图层句柄

RETURNS
ORANGE_OK       删除成功
ORANGE_FAIL     删除失败

DESCRIPTION
删除图层
```

3.2.44 GUI_LyrWinSetSrcWindow

```
PROTOTYPE
__s32  GUI_LyrWinSetSrcWindow(H_LYR h_lyr, const RECT * rect);

ARGUMENTS
h_lyr   图层句柄
rect        源区域矩形指针

RETURNS
ORANGE_OK       设置成功
ORANGE_FAIL     设置失败

DESCRIPTION
设置图层显示区域在图层中的位置窗口
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.45 GUI_LyrWinSetScnWindow

```
PROTOTYPE
__s32  GUI_LyrWinSetScnWindow(H_LYR h_lyr, const RECT * rect);

ARGUMENTS
h_lyr   图层句柄
rect        屏幕区域矩形指针

RETURNS
ORANGE_OK       设置成功
ORANGE_FAIL     设置失败

DESCRIPTION
设置图层显示区域在屏幕中的位置窗口
```

3.2.46 GUI_LyrWinGetSrcWindow

```
PROTOTYPE
__s32  GUI_LyrWinGetSrcWindow(H_LYR h_lyr, RECT * rect);

ARGUMENTS
h_lyr   图层句柄
rect        源区域矩形指针

RETURNS
ORANGE_OK       获取成功
ORANGE_FAIL     获取失败

DESCRIPTION
获取图层显示区域在图层中的位置窗口
```

3.2.47 GUI_LyrWinGetScnWindow

```
PROTOTYPE
__s32  GUI_LyrWinGetScnWindow(H_LYR h_lyr, RECT * rect);

ARGUMENTS
h_lyr   图层句柄
rect        屏幕区域矩形指针

RETURNS
ORANGE_OK       获取成功
ORANGE_FAIL     获取失败

DESCRIPTION
获取图层显示区域在屏幕中的位置窗口。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.48 GUI_LyrMove

```
PROTOTYPE
__s32  GUI_LyrMove(H_LYR h_lyr, __s32 x, __s32 y);

ARGUMENTS
h_lyr    图层句柄
x        x方向移动的相对坐标
y        y方向移动的相对坐标

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
在屏幕中移动图层。此时需要向窗口发送消息
```

3.2.49 GUI_LyrWinSetFB

```
PROTOTYPE
__s32  GUI_LyrWinSetFB(H_LYR h_lyr, const FB * fb);

ARGUMENTS
h_lyr    图层句柄
fb       指向需设定的FB结构体

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
设置图层的fb，此函数和下面的get函数作为特定场合使用的，一般应用程序不会使用这组接口。
```

3.2.50 GUI_LyrWinSel

```
PROTOTYPE
__s32  GUI_LyrWinSel(H_LYR h_lyr);

ARGUMENTS
h_lyr    图层句柄

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
选择图层为当前操作图层
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.51 GUI_LyrWinSetFocus

```
PROTOTYPE
__s32  GUI_LyrWinSetFocus(H_LYR h_lyr);

ARGUMENTS
h_lyr    图层句柄

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
将图层设置为焦点图层
```

3.2.52 GUI_LyrWinCacheOn

```
PROTOTYPE
__s32  GUI_LyrWinCacheOn(void);

ARGUMENTS
void

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
打开命令cache
```

3.2.53 GUI_LyrWinCacheOff

```
PROTOTYPE
__s32  GUI_LyrWinCacheOff(void);

ARGUMENTS
void

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
执行cache里的所有的命令，并停止cache。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.54 GUI_LyrWinGetSta

```
PROTOTYPE
__gui_lyr_sta_t GUI_LyrWinGetSta(H_LYR h_lyr);

ARGUMENTS
h_lyr    图层句柄

RETURNS
图层状态信息

DESCRIPTION
获取图层的状态
```

3.2.55 GUI_LyrWinSetSta

```
PROTOTYPE
__s32   GUI_LyrWinSetSta(H_LYR h_lyr, __gui_lyr_sta_t status);

ARGUMENTS
h_lyr    图层句柄
status   图层状态

RETURNS
ORANGE_OK        设置成功
ORANGE_FAIL      设置失败

DESCRIPTION
设置图层的状态
```

3.2.56 GUI_NotifyMSGQ

```
PROTOTYPE
void GUI_NotifyMSGQ(__win_msgqueue_t *qmsg);

ARGUMENTS
qmsg     指向需通知的消息队列


RETURNS
void

DESCRIPTION
通知相应的消息队列，有消息来到。
```

ALLWINNER®

3.2.57 GUI_SendMessage

```
PROTOTYPE
__s32 GUI_SendMessage(__gui_msg_t *msg);

ARGUMENTS
msg     指向需发送的消息

RETURNS
ORANGE_FAIL 发送失败；否则返回该消息目标窗口回调函数的返回值。

DESCRIPTION
发送消息。如果该消息的目标窗口所在的线程和当前线程是同一线程，则直接发送该消息到目标窗口的回调函数，否则发送同
    步消息到目标窗口所在线程的消息队列。
```

3.2.58 GUI_PostMessage

```
PROTOTYPE
__s32 GUI_PostMessage(__gui_msg_t *msg);

ARGUMENTS
msg     指向需投递的消息

RETURNS
ORANGE_FAIL 投递失败
ORANGE_OK   投递成功

DESCRIPTION
投递消息到消息的目标窗口所在的消息队列。
```

3.2.59 GUI_SendNotifyMessage

```
PROTOTYPE
__s32 GUI_SendNotifyMessage(__gui_msg_t *msg);

ARGUMENTS
msg     指向需发送的通知消息

RETURNS
ORANGE_FAIL 发送失败
ORANGE_OK   发送成功

DESCRIPTION
发送通知消息到该消息的目标窗口所在的消息队列。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.60 GUI_GetMessageEx

```
PROTOTYPE
__s32 GUI_GetMessageEx(__gui_msg_t *msg, H_WIN  hManWin);

ARGUMENTS
msg      指向取到的消息
hManWin 管理窗口句柄

RETURNS
ORANGE_FAIL 取消息失败
ORANGE_TRUE 取消息成功

DESCRIPTION
从管理窗口hManWin对应的消息队列中取消息。
```

3.2.61 GUI_MsgSetRepeatTimes

```
PROTOTYPE
__s32 GUI_MsgSetRepeatTimes(H_WIN  hManWin,__u32 count);

ARGUMENTS
hManWin 管理窗口句柄
count        长按键计数次数

RETURNS
ORANGE_FAIL 设置失败
ORANGE_OK    设置成功

DESCRIPTION
设置管理窗口hManWin对应的消息队列的长按键计数次数。
```

3.2.62 GUI_DispatchMessage

```
PROTOTYPE
__s32 GUI_DispatchMessage(__gui_msg_t *msg);

ARGUMENTS
msg      指向待派发的消息

RETURNS
HWND_INVALID         失败，否则返回该消息目标窗口回调函数的返回值

DESCRIPTION
派发msg到该消息对应的目标窗口的回调函数。
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.63 GUI_SendAsyncMessage

```
PROTOTYPE
__s32 GUI_SendAsyncMessage(__gui_msg_t *msg);

ARGUMENTS
msg     指向待发送的消息

RETURNS
ORANGE_FAIL 失败，否则返回该消息对应的目标窗口的回调函数的返回值

DESCRIPTION
发送异步消息。
```

3.2.64 GUI_ThrowAwayMessages

```
PROTOTYPE
__s32 GUI_ThrowAwayMessages (H_WIN hWnd);

ARGUMENTS
hWnd    窗口句柄

RETURNS
ORANGE_FAIL 失败，否则返回丢掉的消息个数

DESCRIPTION
丢掉窗口hWnd对应的消息队列中的所有消息。
```

3.2.65 GUI_SetSyncMsgRetVal

```
PROTOTYPE
__s32 GUI_SetSyncMsgRetVal(__gui_msg_t *msg,__s32 ret);

ARGUMENTS
msg     指向需设置返回值的消息
ret     返回值

RETURNS
ORANGE_FAIL 失败
ORANGE_OK   成功

DESCRIPTION
设置同步消息的返回值。
```

ALLWINNER®

3.2.66 GUI_PostSyncSem

```
PROTOTYPE
__s32    GUI_PostSyncSem (__gui_msg_t *msg);

ARGUMENTS
msg      指向同步消息


RETURNS
ORANGE_FAIL 失败
ORANGE_OK   成功

DESCRIPTION
释放同步消息的信号量。
```

3.2.67 GUI_ClearRect

```
PROTOTYPE
void GUI_ClearRect(int x0, int y0, int x1, int y1);

ARGUMENTS
x0  矩形左上角x坐标.
y0  矩形左上角y坐标.
x1  矩形右下角x坐标.
y1  矩形右下角y坐标.

RETURNS
void

DESCRIPTION
用背景色去填充当前图层上指定的矩形.
```

3.2.68 GUI_DrawPixel

```
PROTOTYPE
void GUI_DrawPixel(int x, int y);

ARGUMENTS
x  指定像素的x坐标.
y  指定像素的y坐标.

RETURNS
void

DESCRIPTION
在当前图层的指定位置处渲染一个像素.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.69 GUI_DrawPoint

```
PROTOTYPE
void GUI_DrawPoint(int x, int y);

ARGUMENTS
x   绘制点的x坐标.
y   绘制点的y坐标.

RETURNS
void

DESCRIPTION
用当前的画刷在当前图层的指定位置处绘制一个点.
```

3.2.70 GUI_DrawRect

```
PROTOTYPE
void GUI_DrawRect(int x0, int y0, int x1, int y1);

ARGUMENTS
x0   矩形左上角x坐标.
y0   矩形左上角y坐标.
x1   矩形右下角x坐标.
y1   矩形右下角x坐标.

RETURNS
void

DESCRIPTION
在当前图层的指定位置绘制一个矩形.
```

3.2.71 GUI_DrawRectEx

```
PROTOTYPE
void GUI_DrawRectEx(const GUI_RECT *pRect);

ARGUMENTS
x0   矩形左上角x坐标.
y0   矩形左上角y坐标.
x1   矩形右下角x坐标.
y1   矩形右下角x坐标.

RETURNS
pRect   指向一个GUI_RECT 结构体的矩形，该矩形包括了待绘制矩形在图层上的坐标位置。

DESCRIPTION
在当前图层的指定位置绘制一个矩形.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.72 GUI_DrawHLine

```
PROTOTYPE
void GUI_DrawHLine(int y, int x0, int x1);

ARGUMENTS
y    水平直线在当前图层y方向上的位置.
x0   水平直线在当前图层x方向上的起始位置.
x1   水平直线在当前图层x方向上的终止位置.

RETURNS
void

DESCRIPTION
在当前图层绘制一条水平直线，如果x1<x0，不进行任何绘制。对于大多数LCD控制器，该函数被执行的非常快；该函数画水
     平直线的话比GUI_DrawLine()函数快。
```

3.2.73 GUI_DrawLine

```
PROTOTYPE
void GUI_DrawLine(int x0, int y0, int x1, int y1);

ARGUMENTS
x0   直线在当前图层x方向上的起始位置.
y0   直线在当前图层y方向上的起始位置.
x1   直线在当前图层x方向上的终止位置.
y1   直线在当前图层y方向上的终止位置.

RETURNS
void

DESCRIPTION
在当前图层上从指定的起点到终点绘制一条直线.
```

3.2.74 GUI_DrawLineRel

```
PROTOTYPE
void GUI_DrawLineRel(int dx, int dy);

ARGUMENTS
dx   x方向上的相对坐标.
dy   y方向上的相对坐标.

RETURNS
void

DESCRIPTION
绘制一条直线，该直线的起点是当前位置，终点由相对于起点位置的相对坐标(dx, dy)指定.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.75 GUI_DrawLineTo

```
PROTOTYPE
void GUI_DrawLineTo(int x, int y);

ARGUMENTS
x   x方向上的终点坐标.
y   y方向上的终点坐标.

RETURNS
void

DESCRIPTION
在当前图层绘制一条直线，该直线的起点是当前位置，终点坐标是(x,y).
```

3.2.76 GUI_DrawPolyLine

```
PROTOTYPE
void GUI_DrawPolyLine(const GUI_POINT* pPoint, int NumPoints, int x0, int y0);

ARGUMENTS
pPoint      指向折线的拐点构成的数组.
NumPoints   拐点个数.
x0          指定x方向上的原点坐标.
y0          指定y方向上的原点坐标.

RETURNS
void

DESCRIPTION
在当前图层绘制折线，该折线的拐点由pPoint数组中的point元素指定，原点由(x0, y0)指定，若x0=0， y0 = 0，则折
    线的起始点由pPoint数组中的第一个点开始，否则由pPoint[0].x + x0，和pPoint[0].y + y0作为起始点.
```

3.2.77 GUI_DrawVLine

```
PROTOTYPE
void GUI_DrawVLine(int x, int y0, int y1);

ARGUMENTS
x    垂直直线在当前图层x方向上的位置.
y0   垂直直线在当前图层y方向上的起始位置.
y1   垂直直线在当前图层y方向上的终止位置.

RETURNS
void

DESCRIPTION
在当前图层绘制一条垂直直线，如果y1<y0，不进行任何绘制。对于大多数LCD控制器，该函数被执行的非常快；该函数画垂
    直直线的话比GUI_DrawLine()函数快.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.78 GUI_GetLineStyle

```
PROTOTYPE
U8 GUI_GetLineStyle (void);

ARGUMENTS
void

RETURNS
正在使用的绘制风格.

DESCRIPTION
获得正在使用的绘制风格(line的绘制风格，该绘制风格的设定只有pen size为1时有效).
```

3.2.79 GUI_MoveRel

```
PROTOTYPE
void GUI_MoveRel(int dx, int dy);

ARGUMENTS
dx   x方向上的相对距离.
dy   y方向上的相对距离.

RETURNS
正在使用的绘制风格.

DESCRIPTION
将绘制点的当前位置移动到相对于当前位置的一个坐标点(dx, dy).
```

3.2.80 GUI_MoveTo

```
PROTOTYPE
void GUI_MoveTo(int x, int y);

ARGUMENTS
x    x方向上新的坐标位置.
y    y方向上新的坐标位置.

RETURNS
void

DESCRIPTION
将绘制点的当前位置移动到坐标点(x, y)处.
```

ALLWINNER®

3.2.81 GUI_SetLineStyle

```
PROTOTYPE
U8 GUI_SetLineStyle(U8 LineStyle);

ARGUMENTS
LineStyle  被设定的线的绘制风格.
GUI_LS_SOLID    实线绘制(default).
GUI_LS_DASH 折线绘制.
GUI_LS_DOT  点线绘制.
GUI_LS_DASHDOT  折线和点线交替绘制.
GUI_LS_DASHDOTDOT   折线和连续的两个点线交替绘制.

RETURNS
老的绘制风格

DESCRIPTION
设置当前线的绘制风格，被使用在GUI_DrawLine接口中，该绘制风格的设定只有pen size为1时有效.
```

3.2.82 GUI_DrawPolygon

```
PROTOTYPE
void GUI_DrawPolygon(const GUI_POINT* pPoint, int NumPoints, int x, int y);

ARGUMENTS
pPoint     指向多边形各顶点构成的point数组.
NumPoints  顶点数.
x          原点在x方向上的位置.
y          原点在y方向上的位置.

RETURNS
void

DESCRIPTION
在当前图层绘制多边形.
```

3.2.83 GUI_EnlargePolygon

```
PROTOTYPE
void GUI_EnlargePolygon(GUI_POINT* pDest, const GUI_POINT* pSrc, int NumPoints, int Len);

ARGUMENTS
pDest      指向目标多边形.
pSrc       指向源多边形.
NumPoints  多边形顶点个数.
Len        多边形放大的长度，以像素为单位.

RETURNS
void

DESCRIPTION
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
放大多边形，注意：要保证目标数组等于或大于源数组.
```

3.2.84 GUI_FillPolygon

```
PROTOTYPE
void GUI_FillPolygon(const GUI_POINT* pPoint, int NumPoints, int x, int y);

ARGUMENTS
pPoint     指向被填充的多边形.
NumPoints 多边形顶点数.
x          原点在x方向上的位置.
y          原点在y方向上的位置.

RETURNS
void

DESCRIPTION
在当前图层绘制一个被填充了的多边形，填充颜色由GUI_SetColor接口指定.
```

3.2.85 GUI_DrawCircle

```
PROTOTYPE
void GUI_DrawCircle(int x0, int y0, int r);

ARGUMENTS
x0   圆心在x方向上的位置.
y0   圆心在y方向上的位置.
r    半径. 最小值: 0(一个点); 最大值180.

RETURNS
void

DESCRIPTION
在当前图层绘制圆形.
```

3.2.86 GUI_FillCircle

```
PROTOTYPE
void GUI_FillCircle(int x0, int y0, int r);

ARGUMENTS
x0   圆心在x方向上的位置.
y0   圆心在y方向上的位置.
r    半径. 最小值: 0(一个点); 最大值180.

RETURNS
void
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
DESCRIPTION
在当前图层绘制一个被填充了的圆形，填充颜色由GUI_SetColor接口指定。该函数不能处理半径超过180个像素的圆.
```

3.2.87 GUI_DrawArc

```
PROTOTYPE
void GL_DrawArc (int xCenter, int yCenter, int rx, int ry, int a0, int a1);

ARGUMENTS
xCenter  弧形中心在x方向上的位置.
yCenter  弧形中心在y方向上的位置.
rx  X方向上的半径(pixels).
ry  Y方向上的半径(pixels)，暂未用.
a0  起始角度(degrees).
a1  终止角度 (degrees).

RETURNS
void

DESCRIPTION
在当前图层绘制一个弧形，该弧形的半径不能超过180个像素.
```

3.2.88 GUI_DrawGraph

```
PROTOTYPE
void GUI_DrawGraph(I16 *paY, int NumPoints, int x0, int y0);

ARGUMENTS
paY         指向包含图形Y值的一个数组.
NumPoints   Y值的个数.
x0          X方向上的起始点.
y0          Y方向上的起始点.

RETURNS
void

DESCRIPTION
在当前图层绘制一张图. 该函数首先通过x0、y0和给定数组paY中的第一个Y值设定当前的显示位置，然后按点(x0 + 1,
    y0 + *(paY +1))、点(x0 + 2, y0 + *(paY + 2))依次绘制直线.
```

3.2.89 GUI_RestoreContext

```
PROTOTYPE
void GUI_RestoreContext(const GUI_CONTEXT* pContext);

ARGUMENTS
pContext        指向包含新内容的一个GUI_CONTEXT结构体.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
RETURNS
void

DESCRIPTION
设置pContext为当前使用的GUI Context.
```

3.2.90 GUI_SaveContext

```
PROTOTYPE
void GUI_SaveContext(GUI_CONTEXT* pContext);

ARGUMENTS
pContext        指向保存当前内容的一个GUI_CONTEXT结构体.

RETURNS
void

DESCRIPTION
保存当前GUI Context. (See also GUI_RestoreContext).
```

3.2.91 GUI_SetClipRect

```
PROTOTYPE
void GUI_SetClipRect(const GUI_RECT* pRect);

ARGUMENTS
pRect    指向被用于剪切的矩形，如果是NULL则恢复默认的剪切域.

RETURNS
void

DESCRIPTION
为了限制输出范围设置剪切矩形.
```

3.2.92 GUI_DispChar

```
PROTOTYPE
void GUI_DispChar(U16 c);

ARGUMENTS
c    显示的字符.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的当前显示位置绘制单个字符.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.93 GUI_DispCharAt

```
PROTOTYPE
void GUI_DispCharAt(U16 c, I16P x, I16P y);

ARGUMENTS
c    显示的字符.
x    指定x方向上的显示位置.
y    指定y方向上的显示位置.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的指定位置绘制单个字符.
```

3.2.94 GUI_DispChars

```
PROTOTYPE
void GUI_DispChars(U16 c, int Cnt);

ARGUMENTS
c      显示的字符.
Cnt    重复次数(0 <= Cnt <= 32767).

RETURNS
void

DESCRIPTION
用当前字体在当前图层的指定位置重复绘制单个字符Cnt次.
```

3.2.95 GUI_DispNextLine

```
PROTOTYPE
void GUI_DispNextLine(void);

ARGUMENTS
void

RETURNS
void

DESCRIPTION
移动光标到下一行的开始.
```

ALLWINNER®

3.2.96 GUI_DispString

```
PROTOTYPE
void GUI_DispString(const char GUI_FAR *s);

ARGUMENTS
s   显示的字符串.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的当前文本显示位置显示一串字符串.
```

3.2.97 GUI_DispStringAt

```
PROTOTYPE
void GUI_DispStringAt(const char GUI_FAR *s, int x, int y);

ARGUMENTS
s   显示的字符串.
x   指定x方向上的显示位置.
y   指定y方向上的显示位置.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的指定位置显示一串字符串.
```

3.2.98 GUI_DispStringAtCEOL

```
PROTOTYPE
void GUI_DispStringAtCEOL(const char GUI_UNI_PTR *s, int x, int y);

ARGUMENTS
s    显示的字符串.
x    指定x方向上的显示位置.
y    指定y方向上的显示位置.

RETURNS
void

DESCRIPTION
该函数与GUI_DispStringAt()接口的参数非常相近，它们都是用当前字体在当前图层的指定位置显示一串字符串，不同的
     是该函数会清除到行尾的剩余部分。例如某字符串覆盖原先的一行字符串，但该字符串的长度小于原先字符串的长度，
     这时就可以用该函数将该行剩余部分清除。
```

ALLWINNER®

3.2.99 GUI_DispStringHCenterAt

```
PROTOTYPE
void GUI_DispStringHCenterAt(const char GUI_FAR *s, int x, int y);

ARGUMENTS
s    显示的字符串.
x    指定x方向上的显示位置.
y    指定y方向上的显示位置.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的指定位置的水平中心处显示一串字符串.
```

3.2.100 GUI_DispStringInRect

```
PROTOTYPE
void GUI_DispStringInRect(const char GUI_FAR *s, const GUI_RECT *pRect, int Align);

ARGUMENTS
s       显示的字符串.
pRect   指定的矩形区域.
Align   对齐模式.
align:
GUI_TA_LEFT 水平方向靠左
GUI_TA_RIGHT    水平方向靠右
GUI_TA_HCENTER  水平方向中间
GUI_TA_TOP  垂直方向靠上
GUI_TA_BOTTOM   垂直方向靠下
GUI_TA_VCENTER  垂直方向中间

RETURNS
void

DESCRIPTION
用当前字体在当前图层的指定矩形中按指定的对齐模式显示一串字符串.
```

3.2.101 GUI_DispStringInRectWrap

```
PROTOTYPE
void GUI_DispStringInRectWrap(const char GUI_UNI_PTR * s, GUI_RECT * pRect, int TextAlign,
    GUI_WRAPMODE WrapMode);

ARGUMENTS
s            显示的字符串.
pRect        指定的矩形区域.
Align       对齐模式:
GUI_TA_LEFT     水平方向靠左
GUI_TA_RIGHT    水平方向靠右
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
GUI_TA_HCENTER  水平方向中间
GUI_TA_TOP      垂直方向靠上
GUI_TA_BOTTOM   垂直方向靠下
GUI_TA_VCENTER  垂直方向中间
WrapMode   限制方式:
GUI_WRAPMODE_NONE   无限制.
GUI_WRAPMODE_WORD   Word方式限制.
GUI_WRAPMODE_CHAR   Char方式限制.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的指定矩形中按指定的对齐模式和限制方式(可选的)显示一串字符串.
```

3.2.102 GUI_DispStringLen

```
PROTOTYPE
void GUI_DispStringLen(const char GUI_UNI_PTR *s, int MaxNumChars);

ARGUMENTS
s    显示的字符串. 该字符串必须是以\0结束的8bit字符数组，允许是NULL.
MaxNumChars 显示的字符个数.

RETURNS
void

DESCRIPTION
用当前字体在当前图层的当前显示位置显示指定数目的字符. 如果给定字符串的字符个数少于给定的数目则不足的数目由空格
    补齐; 如果多于给定的数目则只有给定数目的字符被显示.
```

3.2.103 GUI_GetTextMode

```
PROTOTYPE
int GUI_GetTextMode(void);

ARGUMENTS
void

RETURNS
文本模式

DESCRIPTION
获取当前文本模式.
```

ALLWINNER®

3.2.104 GUI_SetTextMode

```
PROTOTYPE
int GUI_SetTextMode(int TextMode);

ARGUMENTS
TextMode   文本模式:
GUI_TEXTMODE_NORMAL     正常模式.
GUI_TEXTMODE_REVERSE    相反模式.
GUI_TEXTMODE_TRANSPARENT    透明模式.
GUI_TEXTMODE_XOR    异或模式.

RETURNS
先前被选中的文本模式.

DESCRIPTION
设置当前文本模式.
```

3.2.105 GUI_SetTextStyle

```
PROTOTYPE
char GUI_SetTextStyle(char Style);

ARGUMENTS
Style       文本样式:
GUI_TS_NORMAL   普通样式 (default).
GUI_TS_UNDERLINE    下划线样式.
GUI_TS_STRIKETHRU   中间穿透样式.
GUI_TS_OVERLINE 上划线样式

RETURNS
先前被选中的文本样式.

DESCRIPTION
设置当前文本样式.
```

3.2.106 GUII_SetFontMode

```
PROTOTYPE
GUI_FONTMODE GUI_SetFontMode(GUI_FONTMODE fm);

ARGUMENTS
fm      字体模式:
GUI_FONTMODE_8BPP32     用8bpp调色板的最后32个颜色值作为字体颜色
GUI_FONTMODE_8BPP256    用8bpp调色板的256个颜色值作为字体颜色
GUI_FONTMODE_8BPP128_1  用8bpp调色板的前128个颜色值作为字体颜色
GUI_FONTMODE_8BPP128_2  用8bpp调色板的后128个颜色值作为字体颜色.

RETURNS
先前选中的字体模式.
```

ALLWINNER®

```
DESCRIPTION
仅用在8bpp颜色模式的图层上，来为字体设置调色板的颜色索引值.
```

3.2.107 GUI_GetTextAlign

```
PROTOTYPE
int GUI_GetTextAlign(void);

ARGUMENTS
void

RETURNS
返回当前文本的对齐方式.

DESCRIPTION
获取当前文本的对齐方式.
```

3.2.108 GUI_SetLBorder

```
PROTOTYPE
void GUI_SetLBorder(int x);

ARGUMENTS
X  新的左边界位置.

RETURNS
void

DESCRIPTION
设置当前图层换行时的左边界起始位置.
```

3.2.109 GUI_SetTextAlign

```
PROTOTYPE
int GUI_SetTextAlign:

ARGUMENTS
TextAlign       文本对齐方式(see table below).
GUI_TA_LEFT     水平方向靠左
GUI_TA_RIGHT    水平方向靠右
GUI_TA_HCENTER  水平方向中间
GUI_TA_TOP      垂直方向靠上
GUI_TA_BOTTOM   垂直方向靠下
GUI_TA_VCENTER  垂直方向中间

RETURNS
先前选中的文本对齐方式.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
DESCRIPTION
为字符串的输出设置文本对齐方式.
```

3.2.110 GUI_GotoXY, GUI_GotoX, GUI_GotoY

```
PROTOTYPE
char GUI_GotoXY(int x, int y);
char GUI_GotoX(int x);
char GUI_GotoY(int y);

ARGUMENTS
x    x方向上新的位置(以像素为单位，如果是0则新位置为左边界).
y    y方向上新的位置(以像素为单位，如果是0则新位置为上边界).

RETURNS
通常返回0，如果非0，则当前显示位置超出了图层范围，那么接下来的显示操作会被忽略.

DESCRIPTION
设置当前文本的显示位置.
GUI_GotoXY()     设置(x,y)的值.
GUI_GotoX()      只设置x的值，y值保持不变.
GUI_GotoY()      只设置y的值，x值保持不变.
```

3.2.111 GUI_GetDispPosX

```
PROTOTYPE
int GUI_GetDispPosX(void);

ARGUMENTS
void

RETURNS
当前x方向的显示位置.

DESCRIPTION
获取当前x方向的显示位置.
```

3.2.112 GUI_GetDispPosY

```
PROTOTYPE
int GUI_GetDispPosY(void);

ARGUMENTS
void

RETURNS
当前y方向的显示位置.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
DESCRIPTION
获取当前y方向的显示位置.
```

3.2.113 GUI_Clear

```
PROTOTYPE
void GUI_Clear(void);

ARGUMENTS
void

RETURNS
void

DESCRIPTION
用背景色清除当前图层.
```

3.2.114 GUI_DispCEOL

```
PROTOTYPE
void GUI_DispCEOL(void);

ARGUMENTS
void

RETURNS
void

DESCRIPTION
用背景色清除从当前显示位置到该行行尾的一块矩形，该矩形的高是当前字体的高度.
```

3.2.115 GUI_GetDrawMode

```
PROTOTYPE
GUI_DRAWMODE GUI_GetDrawMode(void);

ARGUMENTS
void

RETURNS
当前选择的绘制模式

DESCRIPTION
返回当前选择的绘制模式.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.116 GUI_SetDrawMode

```
PROTOTYPE
GUI_DRAWMODE GUI_SetDrawMode(GUI_DRAWMODE mode);

ARGUMENTS
mode  绘制模式(see table below).
GUI_DM_NORMAL   正常绘制模式
GUI_DM_XOR  异或绘制模式
GUI_DM_TRANS    透明绘制模式
GUI_DM_REV  反转绘制模式

RETURNS
先前被选中的绘制模式.

DESCRIPTION
设置绘制模式.
```

3.2.117 GUI_ClearRect

```
PROTOTYPE
void GUI_ClearRect(int x0, int y0, int x1, int y1);

ARGUMENTS
x0  矩形左上角x坐标.
y0  矩形左上角y坐标.
x1  矩形右下角x坐标.
y1  矩形右下角y坐标.

RETURNS
void

DESCRIPTION
用背景色清除当前图层上的一块区域，该区域由左上角坐标(x0, y0)和右下角坐标(x1, y1)指定.
```

3.2.118 GUI_BMP_Draw

```
PROTOTYPE
int GUI_BMP_Draw(const void* pBMP, int x0, int y0);

ARGUMENTS
pBMP    指向存放bmp文件的buffer的起始地址.
x0      当前图层上绘制bmp图片的起始位置的x坐标.
y0      当前图层上绘制bmp图片的起始位置的y坐标.

RETURNS
ORANGE_OK   成功，否则失败.

DESCRIPTION
在当前图层的指定位置处绘制BMP文件.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.119 GUI_BitString_Draw

```
PROTOTYPE
int GUI_BitString_Draw(FB* fb, int x0, int y0);

ARGUMENTS
fb     指向FB结构体.
x0     当前图层上绘制bmp图片的起始位置的x坐标.
y0     当前图层上绘制bmp图片的起始位置的y坐标.

RETURNS
ORANGE_OK    成功，否则失败.

DESCRIPTION
在当前图层的指定位置上绘制一张图片，该图片的数据buffer由结构体FB中的addr[0]指定.
```

3.2.120 GUI_BitString_DrawEx

```
PROTOTYPE
int GUI_BitString_DrawEx(FB* fb, int x0, int y0);

ARGUMENTS
fb     指向FB结构体.
x0     当前图层上绘制bmp图片的起始位置的x坐标.
y0     当前图层上绘制bmp图片的起始位置的y坐标.

RETURNS
ORANGE_OK    成功，否则失败.

DESCRIPTION
同GUI_BitString_Draw()功能相同，不同处是该函数是对GUI_BitString_Draw()接口的加速，但是该函数没有进行边
    界检查，因此用户必须确保图片buffer不超出当前图层的buffer.
```

3.2.121 GUI_ARGB_Draw

```
PROTOTYPE
int GUI_ARGB_Draw(const void * pBMP, int x0, int y0);

ARGUMENTS
pBMP    指向存放bmp文件的buffer的起始地址.
x0       当前图层上绘制bmp图片的起始位置的x坐标.
y0       当前图层上绘制bmp图片的起始位置的y坐标.

RETURNS
ORANGE_OK    成功，否则失败.

DESCRIPTION
在当前图层的指定位置处绘制BMP文件.与GUI_BMP_Draw()接口相比该函数在横屏绘制时更快.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.122 GUI_SFT_CreateFont

```
PROTOTYPE
GUI_FONT * GUI_SFT_CreateFont( unsigned int  pixelSize,const char  *font_file );

ARGUMENTS
pixelSize   字体大小(像素为单位)
font_file    字库文件

RETURNS
创建成功    创建好的字体句柄.
创建失败    NULL.

DESCRIPTION
创建一种点阵字体.
```

3.2.123 GUI_SFT_ReleaseFont

```
PROTOTYPE
void GUI_SFT_ReleaseFont(GUI_FONT *pFont);

ARGUMENTS
pFont   字体句柄

RETURNS
ORANGE_OK   成功.
ORANGE_FAIL 失败.

DESCRIPTION
释放某种点阵字体.
```

3.2.124 GUI_TTF_Done

```
PROTOTYPE
void GUI_TTF_Done(GUI_FONT *pFont);

ARGUMENTS
pFont   字体句柄

RETURNS
void

DESCRIPTION
释放矢量字体.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.125 GUI_GetFont

```
PROTOTYPE
const GUI_FONT * GUI_GetFont(void);

ARGUMENTS
void

RETURNS
当前被选中的字体句柄.

DESCRIPTION
获取当前被选中的字体句柄.
```

3.2.126 GUI_SetFont

```
PROTOTYPE
const GUI_FONT * GUI_SetFont(const GUI_FONT * pNewFont);

ARGUMENTS
pNewFont 字体句柄.

RETURNS
先前被选中的字体句柄.

DESCRIPTION
设置字体，一般用在文本显示输出的时候.
```

3.2.127 GUI_GetCharDistX

```
PROTOTYPE
int GUI_GetCharDistX(U16 c);

ARGUMENTS
c  指定的字符.

RETURNS
字符宽度.

DESCRIPTION
获取指定字符的宽度，一般用于在当前选定的字体中显示一个指定的字符.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.128 GUI_GetFontDistY

```
PROTOTYPE
int GUI_GetFontDistY(void);

ARGUMENTS
void

RETURNS
当前选中字体的Y-spacing.

DESCRIPTION
获取当前选中字体的Y-spacing (Y-spacing指文本中相邻两行之间的垂直距离)
```

3.2.129 GUI_GetFontSizeY

```
PROTOTYPE
int GUI_GetFontSizeY(void);

ARGUMENTS
void

RETURNS
当前选中字体的高度.

DESCRIPTION
当前选中字体的高度.该高度小于或等于Y-spacing.
```

3.2.130 GUI_GetStringDistX

```
PROTOTYPE
int GUI_GetStringDistX(const char GUI_FAR *s);

ARGUMENTS
s    字符串.

RETURNS
字符串的宽度.

DESCRIPTION
在当前字体下，获取字符串的宽度.
```

ALLWINNER®

3.2.131 GUI_GetTextExtend

```
PROTOTYPE
void GUI_GetTextExtend(GUI_RECT* pRect, const char* s, int Len);

ARGUMENTS
pRect    指向存放显示区域size的rect.
s        给定的字符串.
Len      字符串中字符的个数.

RETURNS
void

DESCRIPTION
在当前字体下，获取给定字符串显示区域的大小.
```

3.2.132 GUI_GetYDistOfFont

```
PROTOTYPE
int GUI_GetYDistOfFont(const GUI_FONT* pFont);

ARGUMENTS
pFont   字体句柄.

RETURNS
指定字体的Y- spacing.

DESCRIPTION
获取指定字体的Y- spacing.
```

3.2.133 GUI_GetYSizeOfFont

```
PROTOTYPE
int GUI_GetYSizeOfFont(const GUI_FONT* pFont);

ARGUMENTS
pFont   字体句柄.

RETURNS
指定字体的高度.

DESCRIPTION
获取指定字体的高度.
```

ALLWINNER®

3.2.134 GUI_SetFrameColor8bpp32

```
PROTOTYPE
U8 GUI_SetFrameColor8bpp32(U8 frameColor);

ARGUMENTS
frameColor  字体边框颜色.

RETURNS
先前的字体边框颜色.

DESCRIPTION
设置字体边框颜色.
```

3.2.135 GUI_GetBkColor

```
PROTOTYPE
GUI_COLOR GUI_GetBkColor(void);

ARGUMENTS
void

RETURNS
当前的背景色.

DESCRIPTION
获取当前的背景色.
```

3.2.136 GUI_GetBkColorIndex

```
PROTOTYPE
int GUI_GetBkColorIndex(void);

ARGUMENTS
void

RETURNS
当前背景色的索引值.

DESCRIPTION
获取当前背景色的索引值.
```

ALLWINNER®

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.137 GUI_GetColor

```
PROTOTYPE
GUI_COLOR GUI_GetColor(void);

ARGUMENTS
void

RETURNS
当前的前景色.

DESCRIPTION
获取当前的前景色.
```

3.2.138 GUI_GetColorIndex

```
PROTOTYPE
int GUI_GetColorIndex(void);

ARGUMENTS
void

RETURNS
当前前景色的索引值.

DESCRIPTION
获取当前前景色的索引值.
```

3.2.139 GUI_SetBkColor

```
PROTOTYPE
void GUI_SetBkColor(GUI_COLOR color);

ARGUMENTS
Color  背景色.

RETURNS
void

DESCRIPTION
设置当前的背景色.
```

ALLWINNER®

3.2.140 GUI_SetBkColorIndex

```
PROTOTYPE
void GUI_SetBkColorIndex(int Index);

ARGUMENTS
Index   颜色索引值.

RETURNS
void

DESCRIPTION
设置当前的背景色索引值
```

3.2.141 GUI_SetColor

```
PROTOTYPE
void GUI_SetColor(GUI_COLOR Color);

ARGUMENTS
Color   前景色

RETURNS
void

DESCRIPTION
设置当前的前景色.
```

3.2.142 GUI_SetColorIndex

```
PROTOTYPE
void GUI_SetColorIndex(int Index);

ARGUMENTS
Index   颜色索引值.

RETURNS
void

DESCRIPTION
设置当前的前景色索引值.
```

ALLWINNER®

3.2.143 GUI_Color2Index

```
PROTOTYPE
int GUI_Color2Index(GUI_COLOR Color);

ARGUMENTS
Color   颜色值.

RETURNS
颜色索引值.

DESCRIPTION
颜色值到颜色索引值的转换.
```

3.2.144 GUI_Index2Color

```
PROTOTYPE
int GUI_Index2Color(int  index);

ARGUMENTS
Index  颜色索引值

RETURNS
颜色值.

DESCRIPTION
颜色索引值到颜色值的转换.
```

3.2.145 GUI_Index2Color

```
PROTOTYPE
int GUI_Index2Color(int  index);

ARGUMENTS
Index  颜色索引值

RETURNS
颜色值.

DESCRIPTION
颜色索引值到颜色值的转换.
```

3.2.146 Memory Device

内存设备的主要用途是用来防止绘制时的闪烁现象, 内存设备的基本使用方法非常简单:

1. 创建内存设备 (用 GUI_MEMDEV_Create()).

版权所有 © 珠海全志科技股份有限公司。保留一切权利

2. 激活内存设备 (用 GUI_MEMDEV_Select()).

3. 执行绘制操作.

4. 复制绘制内容到显示设备 (用 GUI_MEMDEV_CopyToLCD()).

5. 如果不再需要改内存设备则删除之 (用 GUI_MEMDEV_Delete()).

3.2.147 GUI_MEMDEV_CopyToLCD

```
PROTOTYPE
void GUI_MEMDEV_CopyToLCD(GUI_MEMDEV_Handle hMem);

ARGUMENTS
hMem    内存设备句柄.

RETURNS
void

DESCRIPTION
复制内存设备内容到显示设备.
```

3.2.148 GUI_MEMDEV_CopyToLCDAt

```
PROTOTYPE
void GUI_MEMDEV_CopyToLCDAt(GUI_MEMDEV_Handle hMem, int x, int y);

ARGUMENTS
hMem    内存设备句柄.
x       显示设备x方向位置
y       显示设备y方向位置

RETURNS
void

DESCRIPTION
复制内存设备内容到显示设备的指定位置.
```

3.2.149 GUI_MEMDEV_Create

```
PROTOTYPE
GUI_MEMDEV_Handle GUI_MEMDEV_Create(int x0, int y0, int XSize, int YSize);

ARGUMENTS
x0    内存设备起始点x坐标.
y0    内存设备起始点y坐标.
xsize  内存设备起宽度.
ysize  内存设备起高度.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
RETURNS
内存设备句柄，如果为0则创建失败.

DESCRIPTION
创建内存设备.
```

3.2.150 GUI_MEMDEV_Delete

```
PROTOTYPE
void GUI_MEMDEV_Delete(GUI_MEMDEV_Handle MemDev);

ARGUMENTS
hMem    内存设备句柄.

RETURNS
内存设备句柄，如果为0则创建失败.

DESCRIPTION
删除内存设备.
```

3.2.151 GUI_MEMDEV_GetXSize

```
PROTOTYPE
int GUI_MEMDEV_GetXSize(GUI_MEMDEV_Handle hMem);

ARGUMENTS
hMem    内存设备句柄.

RETURNS
内存设备宽度.

DESCRIPTION
获取内存设备宽度.
```

3.2.152 GUI_MEMDEV_GetYSize

```
PROTOTYPE
int GUI_MEMDEV_GetYSize(GUI_MEMDEV_Handle hMem);

ARGUMENTS
hMem    内存设备句柄.

RETURNS
内存设备高度.

DESCRIPTION
获取内存设备高度.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.153 GUI_MEMDEV_Select

```
PROTOTYPE
void GUI_MEMDEV_Select(GUI_MEMDEV_Handle hMem);

ARGUMENTS
hMem    内存设备句柄.

RETURNS
void

DESCRIPTION
激活一个内存设备，如果hMem = 0，则激活显示设备.
```

3.2.154 GUI_MEMDEV_SetOrg

```
PROTOTYPE
void GUI_MEMDEV_SetOrg(GUI_MEMDEV_Handle hMem, int x0, int y0);

ARGUMENTS
hMem    内存设备句柄
x0      左上角像素水平位置
y0      左上角像素垂直位置.

RETURNS
void

DESCRIPTION
改变内存设备在显示设备的原点位置. 当同样的内存设别内容被复制到显示设备的不同区域或者相同的内存设备被用于显示设
    备的不同区域时用该函数比删除当前内存设备再创建更有效.
```

3.2.155 GUI_OpenAlphaBlend

```
PROTOTYPE
void GUI_OpenAlphaBlend();

ARGUMENTS
void

RETURNS
void

DESCRIPTION
打开alpha操作.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.156 GUI_CloseAlphaBlend

```
PROTOTYPE
void GUI_CloseAlphaBlend();

ARGUMENTS
void

RETURNS
void

DESCRIPTION
关闭alpha操作.
```

3.2.157 GUI_CharSetToEncode

```
PROTOTYPE
__s32  GUI_CharSetToEncode( __s32 charset_enm );

ARGUMENTS
charset_enm 字符集:
EPDK_CHARSET_ENM_GB2312 简体中文
EPDK_CHARSET_ENM_UTF8   utf8
EPDK_CHARSET_ENM_UTF16BE    utf16be
EPDK_CHARSET_ENM_UTF16LE    utf16le
EPDK_CHARSET_ENM_BIG5   繁体中文
EPDK_CHARSET_ENM_GBK    中文
EPDK_CHARSET_ENM_SJIS   日文
EPDK_CHARSET_ENM_EUC_JP 日文
EPDK_CHARSET_ENM_EUC_KR 韩文
EPDK_CHARSET_ENM_KI08_R 俄文
EPDK_CHARSET_ENM_ISO_8859_1 西欧语言
EPDK_CHARSET_ENM_ISO_8859_2 中欧语言
EPDK_CHARSET_ENM_ISO_8859_3 南欧语言
EPDK_CHARSET_ENM_ISO_8859_4 北欧语言
EPDK_CHARSET_ENM_ISO_8859_5 西里尔字母
EPDK_CHARSET_ENM_ISO_8859_6 阿拉伯语
EPDK_CHARSET_ENM_ISO_8859_7 希腊语
EPDK_CHARSET_ENM_ISO_8859_8 希伯来语
EPDK_CHARSET_ENM_ISO_8859_9 土耳其语
EPDK_CHARSET_ENM_ISO_8859_10    北欧斯堪的纳维亚语系
EPDK_CHARSET_ENM_ISO_8859_11    泰语
EPDK_CHARSET_ENM_ISO_8859_12    梵文
EPDK_CHARSET_ENM_ISO_8859_13    波罗的海语系
EPDK_CHARSET_ENM_ISO_8859_14    凯尔特人语系
EPDK_CHARSET_ENM_ISO_8859_15    扩展了法语和芬兰语的西欧语系
EPDK_CHARSET_ENM_ISO_8859_16    扩展的东南欧语系
EPDK_CHARSET_ENM_CP874  泰文
EPDK_CHARSET_ENM_CP1250 中欧
EPDK_CHARSET_ENM_CP1251 西里尔文
EPDK_CHARSET_ENM_CP1253 希腊文
EPDK_CHARSET_ENM_CP1255 希伯来文
EPDK_CHARSET_ENM_CP1256 阿拉伯文
EPDK_CHARSET_ENM_CP1257 波罗的海文
EPDK_CHARSET_ENM_CP1258 越南
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
RETURNS
ORANGE_OK    成功
ORANGE_FAIL 失败

DESCRIPTION
设定字符集. 通过设定不同的字符集可以实现多国语言.
```

3.2.158 Lang_Open

```
PROTOTYPE
HLANG   Lang_Open(char *szAppFile, __u32 mode);

ARGUMENTS
szAppFile           文件路径名.
mode                打开模式，目前版本此参数没有意义

RETURNS
HLANG句柄，如果为NULL则打开失败.

DESCRIPTION
打开指定路径语言资源文件，返回文件句柄.
```

3.2.159 Lang_Read

```
PROTOTYPE
int Lang_Read(HLANG hLang, int address, int length, char *buffer);

ARGUMENTS
hLang           文件句柄.
address         相对文件起点处的偏移地址
length          读取的数据长度
buffer          输出读取的数据

RETURNS
实际读取的字节数

DESCRIPTION
读取语言资源文件指定地址处的数据.
```

3.2.160 Lang_GetStringAddress

```
PROTOTYPE
int Lang_GetStringAddress(HLANG hLang, short LangID, short StringID);

ARGUMENTS
hLang           文件句柄.
LangID          语言ID
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

```
StringID            SringID, 在Lang.bat生成的Lang.h文件中定义

RETURNS
地址(相对文件起始偏移量)

DESCRIPTION
查询数据地址.
```

3.2.161 Lang_GetStringSize

```
PROTOTYPE
int Lang_GetStringSize(HLANG hLang, short LangID, short StringID);

ARGUMENTS
hLang           文件句柄.
LangID          语言ID
StringID            SringID, 在Lang.bat生成的Lang.h文件中定义

RETURNS
查询数据长度(byte)

DESCRIPTION
查询数据长度.
```

3.2.162 Lang_GetString

```
PROTOTYPE
int Lang_GetString(HLANG hLang, short LangID, short StringID, char *buffer, int length);

ARGUMENTS
hLang           文件句柄.
LangID          语言ID
StringID            SringID, 在Lang.bat生成的Lang.h文件中定义
buffer          数据输出缓冲区
length          数据输出缓冲区长度

RETURNS
数据实际长度(byte)

DESCRIPTION
查询数据.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.163 Lang_Close

```
PROTOTYPE
int Lang_Close(HLANG hLang);

ARGUMENTS
hLang           文件句柄.

RETURNS
0

DESCRIPTION
关闭句柄，释放内存.
```

3.2.164 OpenRes

```
PROTOTYPE
HRES OpenRes( char * szAppFile, __u32 mode);

ARGUMENTS
szAppFile           文件路径名.
mode                打开模式，目前版本此参数没有意义.

RETURNS
HLANG句柄，如果为NULL则打开失败

DESCRIPTION
打开指定路径图片文件，返回文件句柄.
```

3.2.165 CloseRes

```
PROTOTYPE
int32 CloseRes(HRES hRes);

ARGUMENTS
hRes            文件句柄.

RETURNS
ORANGE_OK    成功
ORANGE_FAIL 失败

DESCRIPTION
关闭句柄，释放内存.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

3.2.166 ReadRes

```
PROTOTYPE
uint32 ReadRes(HRES hRes, uint32 address, uint32 length, void * buffer);

ARGUMENTS
hRes            文件句柄.
address         相对文件起点处的偏移地址
length          读取的数据长度
buffer          输出读取的数据

RETURNS
实际读取的字节数

DESCRIPTION
读取图片文件指定地址处的数据.
```

3.2.167 GetResSize

```
PROTOTYPE
uint32 GetResSize(HRES hRes, uint16 StyleID, uint16 ID);

ARGUMENTS
hRes            文件句柄.
StyleID         图片样式ID
ID              某张图片ID.

RETURNS
资源长度(byte)

DESCRIPTION
获取资源长度.
```

3.2.168 GetResAddr

```
PROTOTYPE
uint32 GetResAddr(HRES hRes, uint16 StyleID, uint16 ID);

ARGUMENTS
hRes            文件句柄.
StyleID         图片样式ID
ID              某张图片ID.

RETURNS
地址(相对文件起始偏移量)

DESCRIPTION
获取资源地址.
```

ALLWINNER®

3.2.169 GetRes

```
PROTOTYPE
int32 GetRes(HRES hRes, uint16 StyleID, uint16 ID, void * Buffer, uint32 Length);

ARGUMENTS
hRes            文件句柄.
StyleID         图片样式ID
ID              某张图片ID.
buffer          数据输出缓冲区
length          数据输出缓冲区长度.

RETURNS
ORANGE_OK   成功
ORANGE_FAIL 失败

DESCRIPTION
获取资源.
```

版权所有 © 珠海全志科技股份有限公司。保留一切权利

著作权声明

版权所有 ©2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、、全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。