LearningGUI 移植简易指南

作者：QQ960747373

2018 年 2 月 7 日

LearningGUI 源码开发发布包名称为：LearninGUI-GPLv3-0-3.tgz。解压后，形成目录如图 1 所示。

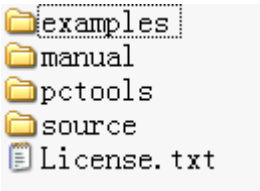

图 1　LearningGUI 开发包目录结构

其中，License.txt 是使用 LearningGUI 源码需要遵循的 GPLv3 协议文本；manual 目录是存放 LearningGUI 开发手册目录；pctools 目录是存放 LearninGUI 图片转换工具目录。

source 目录是 LearningGUI 库源码目录，库源码目录结构如图 2 所示。

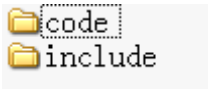

图 2　LearningGUI 库源码目录结构

code 目录主要包括.c 和部分内部使用的.h 文件。

include 目录是用户头文件目录。其中重要的头文件有两个：config_basic.h 和 config_win.h，都是存放 LearningGUI 所定义的宏，如果用户需要进行 LearningGUI 裁剪和修改 LearningGUI 默认配置，则用户需要修改这两个文件。

config_basic.h 快照如图 3 和图 4 所示。

```
#ifndef   __LGUI_CONFIG_BASIC_HEADER__
#define   __LGUI_CONFIG_BASIC_HEADER__

/* Multi-thread support macro */
/* #define  _LG_MULTI_THREAD_   */


/* Window support macro */
#define  _LG_WINDOW_


/* Color support macro */
#define  _LG_COLOR_
    /* For screen color bit deepth */
    #define  _LG_1_BIT_COLOR_
    #define  _LG_2_BIT_COLOR_
    #define  _LG_4_BIT_COLOR_
    #define  _LG_8_BIT_COLOR_
    #define  _LG_15_BIT_COLOR_
    #define  _LG_16_BIT_COLOR_
    #define  _LG_24_BIT_COLOR_
    #define  _LG_32_BIT_COLOR_

    /* Common palette data set */
    #define   _LG_COMMON_PALETTE_
```

图 3　config_basic.h 文件快照 1

```
/* Screen support macro */
#define  _LG_SCREEN_
    /* #define  _LG_SNAPSHOT_ */
    /* #define  _LG_NEED_REFRESH_SCREEN_ */


/* Keyboard support macro */
/* #define   _LG_KEYBOARD_ */

/* MTJT: Mouse-Touchscreen-Joystick-Tablet */
/* MTJT support macro */
/* #define   _LG_MTJT_ */

/* Cursor support macro */
#define  _LG_CURSOR_
    #define  MAX_CURSOR_WIDTH                    16
    #define  MAX_CURSOR_HEIGHT                   16
```

图 4　config_basic.h 文件快照 2

其中，_LG_MULTI_THREAD_表示支持多线程宏，用户需要提供相应的同步接口驱动；_LG_WINDOW_表示支持 windows 版本宏；_LG_KEYBOARD_表示支持键盘驱动宏；_LG_MTJT_表示支持鼠标触摸驱动宏。如果用户不需要相关的功能，则屏蔽掉相关的宏即可。

config_win.h 快照如图 5 所示。

```
/* PushButton widget */
#define  _LG_PUSH_BUTTON_WIDGET_
    /* PushButton window color */
    #define  PBTN_WIN_DISABLED_BCOLOR            GUI_LIGHT_GRAY
    #define  PBTN_WIN_DISABLED_FCOLOR            GUI_DARK

    #define  PBTN_WIN_INACTIVE_BCOLOR            GUI_LIGHT_GRAY
    #define  PBTN_WIN_INACTIVE_FCOLOR            GUI_GRAY

    #define  PBTN_WIN_ACTIVE_BCOLOR              GUI_LIGHT_GRAY
    #define  PBTN_WIN_ACTIVE_FCOLOR              GUI_BLACK

    /* PushButton client color */
    #define  PBTN_CLI_DISABLED_BCOLOR            GUI_LIGHT_GRAY
    #define  PBTN_CLI_DISABLED_FCOLOR            GUI_GRAY
    #define  PBTN_CLI_DISABLED_TBCOLOR           GUI_LIGHT_GRAY
    #define  PBTN_CLI_DISABLED_TFCOLOR           GUI_BLACK

    #define  PBTN_CLI_INACTIVE_BCOLOR            GUI_LIGHT_GRAY
    #define  PBTN_CLI_INACTIVE_FCOLOR            GUI_BLACK
    #define  PBTN_CLI_INACTIVE_TBCOLOR           GUI_LIGHT_GRAY
    #define  PBTN_CLI_INACTIVE_TFCOLOR           GUI_BLACK

    #define  PBTN_CLI_ACTIVE_BCOLOR              0x00E0E0E0
    #define  PBTN_CLI_ACTIVE_FCOLOR              GUI_BLACK
    #define  PBTN_CLI_ACTIVE_TBCOLOR             GUI_LIGHT_GRAY
    #define  PBTN_CLI_ACTIVE_TFCOLOR             GUI_BLACK
```

图 5　config_win.h 文件快照

如果用户希望修改控件的默认颜色，则需要修改 config_win.h 文件。

Examples 目录是 LearningGUI 演示例子目录，如图 6 所示。

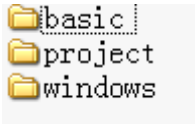

图 6　LearningGUI 演示例子目录结构

basic 目录存放 LearningGUI basic 版本演示例子；project 存放 basic 版综合演示例子；windows 目录存放 LearningGUI windows 版本演示例子，windows 演示例子如图 7 所示。

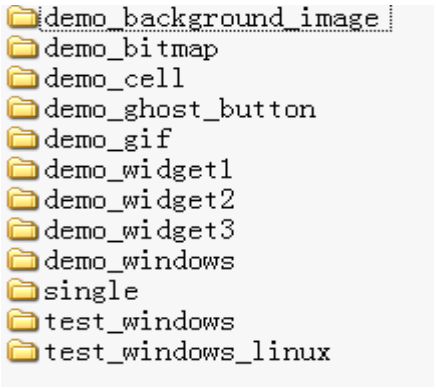

图 7　Windows 演示例子目录结构

其中，test_windows 是 windows 版本最基本的演示例子，demo_windows 是多窗口演示例子。

第一部分：MCU 移植

MCU 移植，就是如何将 LearningGUI 源码嵌入到你的应用中。

第一步：用户建立和调通自己的 LCD 显示工程。

这一步与 LearningGUI 无关。需要用户编写和调通 LCD 驱动接口函数。

LCD 驱动接口需要如下功能：

1.1 打点输出函数；

1.2 读点输入函数；

如果需要优化的话，需要编写和调通如下功能：

1.3 输出水平线函数；

1.4 输出垂直线函数；

1.5 填充矩形函数；

第二步：将 LearningGUI 源码嵌入到自己的 LCD 显示工程中。

2.1 在 LCD 显示工程内建立 LearningGUI 头文件目录，如 LearningGUI_include 目录。并将 LearninGUI 库源码 include 目录下所有文件拷贝并增加到该目录中。

2.2 在 LCD 显示工程内建立 LearningGUI 代码目录，如 LearningGUI_code 目录。并将 LearningGUI 库源码 code 目录下所有文件（除 common.mak 和 makefile 两个文件外）拷贝并增加到该目录中。

提示：code 目录中主要是.c 文件，同时包括少量的 LearningGUI 内部使用的.h 文件。

第三步：设置 LCD 显示工程头文件搜索路径。

在 IDE 中，设置头文件搜索路径为如 LearningGUI_include。

第四步：全部重新编译工程，并且确保通过编译，无编译错误发生。

第五步：将 LearningGUI 演示例子源码嵌入到自己的 LCD 显示工程中。

如 test_window 演示例子，文件目录结构如图 8 所示。

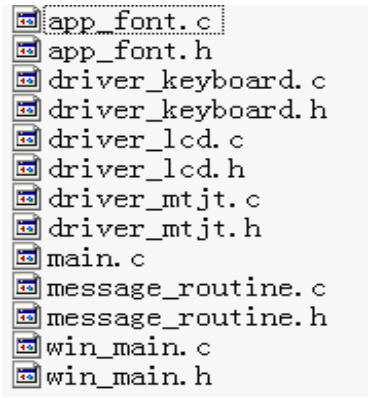

图 8 test_window 演示例子目录结构

5.1 将上述所有文件（除 main.c 文件外）拷贝并增加自己的 LCD 显示工程中。

5.2 修改自己的 main.c 文件。

按照演示工程（test_window）main.c 的式样，修改自己的 main.c 文件。

演示工程的 main.c 文件内容主要如下：

```
#include   <stdio.h>
#include   <string.h>

#include   <lgui.h>

#include   "driver_lcd.h"
#include   "driver_keyboard.h"
#include   "driver_mtjt.h"

#include   "message_routine.h"
#include   "win_main.h"
#include   "app_font.h"

int   main(void)
{
      HWND              p       = NULL;
      GUI_MESSAGE    msg;
      int                  ret;

      /* Step 1: register driver(s) */
      register_screen();
      /* User keyboard */
      #ifdef       _LG_KEYBOARD_
      register_keyboard();
      #endif
      /* User mtjt */
      #ifdef       _LG_MTJT_
      register_mtjt();
      #endif

      /* Step 2: call gui_open */
      ret = gui_open( );
      if ( ret < 0 )
            return    -1;

      /* Step 3: init system */
      /* Set window default font */
      win_set_window_default_font((GUI_FONT *)(&app_ascltn));
      /* Set client default_font */
      win_set_client_default_font((GUI_FONT *)(&app_ascltn));

      /* Step 4: call message_set_routine */
      ret = message_set_routine(message_main_routine);
      if ( ret < 0 )
            return    -1;

      /* Step 5: create and show user GUI */
      /* Create user main window */
      p = create_user_main_window_1();
      if ( p == NULL )
            return    -1;

      win_show(ghmai1);

      /* Step 6: message loop */
      while( 1 )
      {
            ret = message_get(&msg);
            if (ret > 0)
            {
                  if ( MESSAGE_IS_QUIT(&msg) )
                        break;

                  message_dispatch(&msg);
            }

            /* User code ... */
            /*
              * ......
              */

      }

      /* Step 7: user clean    */

      /* Step 8: call gui_open */
      gui_close( );

      return    1;
}
```

当然用户业务代码，根据需要放在合适的地方。

5.3 填写驱动接口

按照 driver_lcd.c 文件模板格式，接上显示驱动代码和注册显示驱动代码。

注册显示驱动代码如下所示：

```
int   register_screen(void)
{
    GUI_SCREEN   screen = {0};


    memset(&screen, 0, sizeof(screen));

    screen.width                        = 320;
    screen.height                       = 240;

    screen.is_hline_accelerate          = 0;
    screen.is_vline_accelerate          = 0;
    screen.is_rect_fill_accelerate      = 0;

    screen.open                         = driver_lcd_open;
    screen.close                        = driver_lcd_close;

    screen.gui_to_screen_color          = driver_lcd_gui_to_r5g6b5;
    screen.screen_to_gui_color          = driver_lcd_r5g6b5_to_gui;

    screen.output_sequence_start        = driver_lcd_output_sequence_start;
    screen.output_pixel                 = driver_lcd_output_pixel;
    screen.output_hline                 = driver_lcd_output_hline;
    screen.output_vline                 = driver_lcd_output_vline;
    screen.output_rect_fill             = driver_lcd_output_rect_fill;
    screen.output_sequence_end          = driver_lcd_output_sequence_end;

    screen.input_sequence_start         = driver_lcd_input_sequence_start;
    screen.input_pixel                  = driver_lcd_input_pixel;
    screen.input_sequence_end           = driver_lcd_input_sequence_end;

    #ifndef   _LG_WINDOW_
    screen.clear                        = driver_lcd_clear;
    #endif

    screen.control                      = driver_lcd_control;
    screen.on                           = driver_lcd_on;
    screen.off                          = driver_lcd_off;
    screen.reinit                       = driver_lcd_reinit;

    in_driver_register(DRIVER_SCREEN, &screen);

    return   1;
}
```

其中：

screen.is_hline_accelerate 表示使用使用画水平线加速接口；

screen.is_vline_accelerate 表示使用使用画垂直线加速接口；

screen.is_rect_fill_accelerate 表示使用使用填充矩形加速接口；

如果用户并不是 RGB565 接口屏，则需要使用正确的颜色转换接口函数；

screen.output_sequence_start 表示优化接口，对于 SPI 等屏可以进行片选拉低操作，等 N 次打点输出结束后，可以使用 screen.output_sequence_end 进行片选拉高操作。用户如果不需要这个优化操作，则在函数中直接 return 1 即可。

screen_input_sequence_start 和 screen_input_sequence_end 也是类似操作。

screen.control、screen.on、screen.off、screen.reinit 对于应用操作接口。

第六步：全部重新编译工程，确保编译通过并且无编译错误发生。

键盘驱动和鼠标驱动机制类似于 LCD 显示驱动。

第二部分：Linux 移植

LearningGUI 在 Linux 下运行需要满足几个条件:

1. 不能启动任何图形系统。
2. 以 Framebuffer 显示方式启动 Linux。
3. 需要以 root 用户登录。

第一步：编译 LearningGUI 库。

```
cd  code/
make  clean
make
```

第二步：编译 LearningGUI 演示程序。

进入演示例子目录。

如： cd test_window/

确定设备名称正确

在 driver_lcd.c 文件中，确保以下显示驱动名称正确：

```
#ifndef   GUI_FRAME_BUFFER_DEVICE_NAME
#define   GUI_FRAME_BUFFER_DEVICE_NAME    "/dev/fb0"
#endif
```

在 driver_keyboard.c 文件中，确保以下键盘驱动名称正确：

```
#ifndef  GUI_KEYBOARD_DEVICE_NAME
#define  GUI_KEYBOARD_DEVICE_NAME          "/dev/input/event4"
#endif
```

在 driver_mtjt.c 文件中，确保以下鼠标驱动名称正确：

```
#ifndef  GUI_MTJT_DEVICE_NAME
#define  GUI_MTJT_DEVICE_NAME              "/dev/input/event3"
#endif
```

编译演示程序：

```
make  clean
make
```