GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

应用笔记

AN042

目录

图索引

表索引

1. 简介

LVGL 是一种开源免费的图形库，它提供创建嵌入式 GUI 所需的一切，它具有易于使用的图形元素，精美的视觉效果和低内存占用。具体介绍可参考 https://github.com/littlevgl/lvgl。

2. 开发环境

开发环境介绍主要如下：

- 硬件开发板：GD32F450i-EVAL-V1.1 开发板
- Cortex-M4：GD32F450IKH6
- 操作系统：Win10-64 位
- 开发环境：KEIL 5.27
- 固件库：GD32F4xx_Firmware_Library V2.1.3
- GUI：LVGL 8.1.0

3. 移植

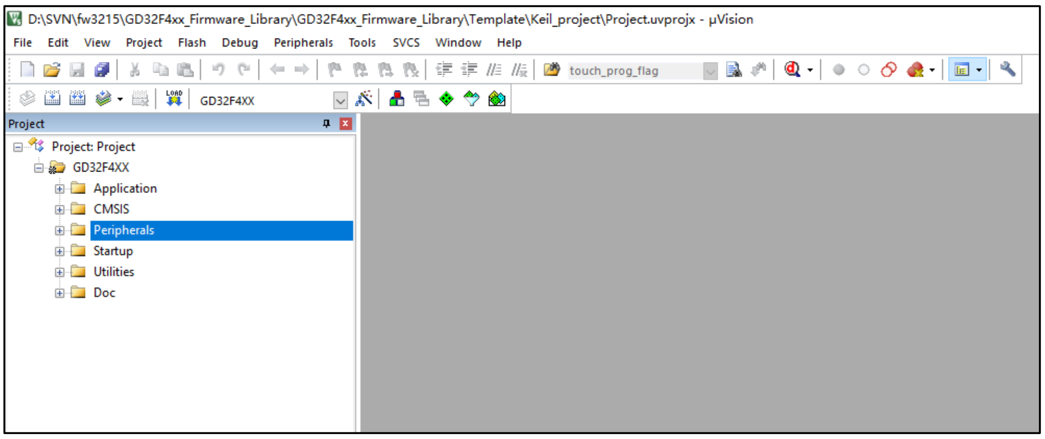
3.1. 素材准备

移植过程中需要用到 LVGL 图形库，LVGL 图形库的下载地址为：https://github.com/lvgl/lvgl 本手册使用的是 LVGL8.1.0 版本。下载 GD32F4xx SDK 官方固件开发包 GD32F4xx Firmware Library，下载地址为：http://gd32mcu.com/cn/download/7?kw=GD32F4。本手册采用固件版本为 V2.1.3。

3.2. 建立工程

将下载的 GD32F4xx Firmware Library 固件包解压后，进入目录“GD32F4xx_Firmware_Library\Template\Keil_project”，该默认工程使用 KEIL4 打开，将工程名称后缀“Project.uvproj”修改为“Project.uvprojx”，即可使用 KEIL5 打开，打开后界面如*图 3-1. 打开工程*所示。

图 3-1. 打开工程



本手册移植 LVGL 会使用到开发板上的 SDRAM 硬件，因此在 Utilities 文件夹下加入 gd32f450i_eval_exmc_sdram.c，修改部分代码，将第 38 行“#include "drv_usb_hw.h"”修改为“#include "systick.h"”，第 188 行“usb_mdelay (10);”修改为“delay_1ms(10);”，修改完成后，即可编译成功。

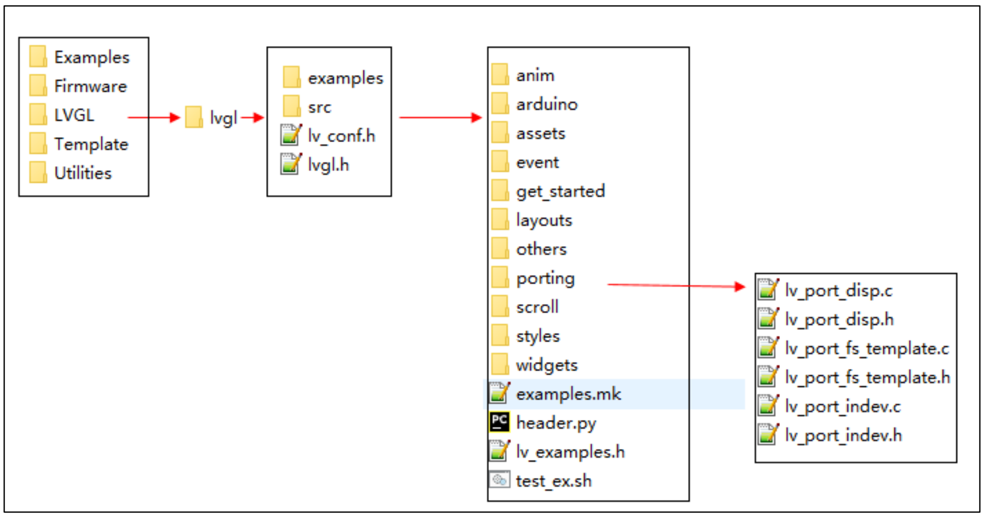
3.3. 导入 LVGL 到 KEIL 中

导入 LVGL 到 KEIL 中，流程如下：

1. 在 GD32F4xx_Firmware_Library 目录下新建 LVGL 文件夹；
2. 将下载的 lvgl 源码解压后提取 example 和 src 文件夹，“lv_conf_template.h”和“lvgl.h”文件放到 LVGL\lvgl 文件夹目录下，并将“lv_conf_template.h”名称修改为“lv_conf.h”；

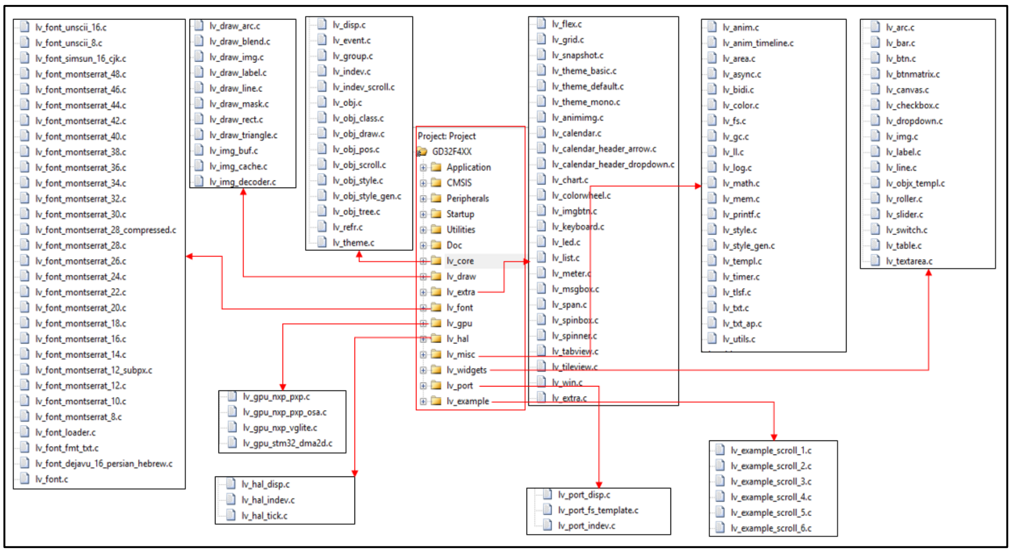
3. 在“LVGL\lvgl\examples\porting”文件夹目录下，将“lv_port_disp_template.c”和“lv_port_disp_template.h”文件名称修改为“lv_port_disp.c”和“lv_port_disp.h”,将“lv_port_indev_template.c”和“lv_port_indev_template.h”文件名称修改为“lv_port_indev.c”和“lv_port_indev.h”,工程目录结构如*图 3-2. 工程目录结构*所示。

图 3-2. 工程目录结构



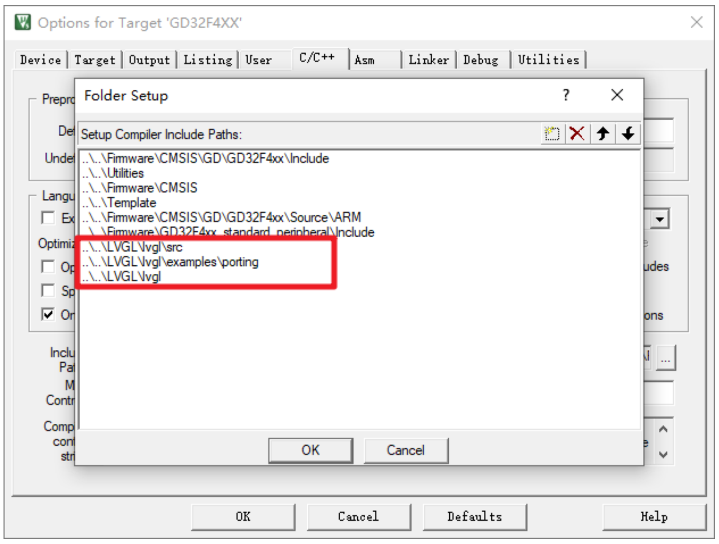
4. 在 KEIL 工程中添加 LVGL 相关文件的项目管理项，并在对应管理项添加对应 LVGL 中的.c 文件，其中 lv_example 文件夹下包含官方提供的各种控件的使用例程，本手册以 scroll 为例导入该 example 作为演示。添加的 KEIL 工程结构如*图 3-3. KEIL 工程结构*所示。

图 3-3. KEIL 工程结构



5. 在工程中添加相应的头文件，如*图 3-4. KEIL Folder Setup 设置*所示。

图 3-4. KEIL Folder Setup 设置


Options for Target 'GD32F4XX'

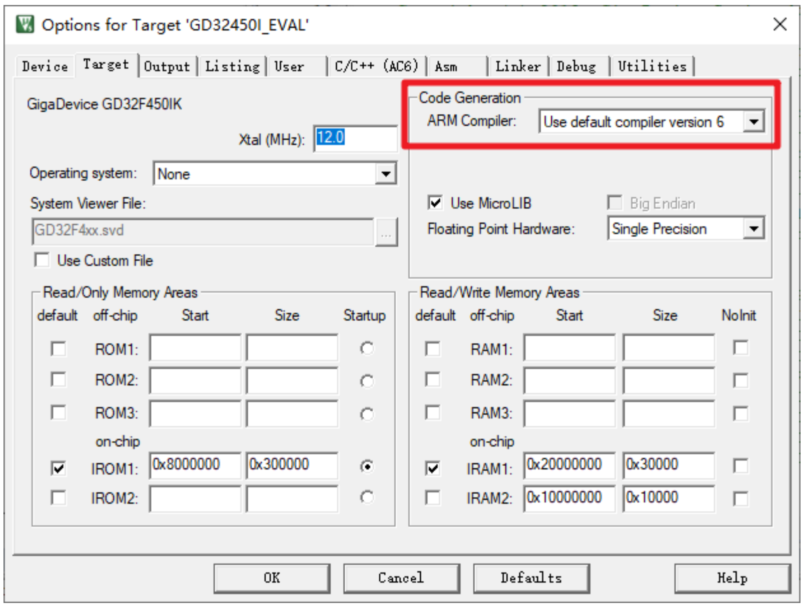
Folder Setup

Setup Compiler Include Paths:

```
..\..\Firmware\CMSIS\GD\GD32F4xx\Include
..\..\Utilities
..\..\Firmware\CMSIS
..\..\Template
..\..\Firmware\CMSIS\GD\GD32F4xx\Source\ARM
..\..\Firmware\GD32F4xx_standard_peripheral\Include
..\..\LVGL\lvgl\src
..\..\LVGL\lvgl\examples\porting
..\..\LVGL\lvgl
```

6. 将编译器版本由版本 5 修改为版本 6，如*图3-5. KEIL 编译器版本设置*所示。

图 3-5. KEIL 编译器版本设置


Options for Target 'GD32450I_EVAL'

GigaDevice GD32F450IK

Xtal (MHz): 12.0

Operating system: None

System Viewer File: GD32F4xx.svd

Use Custom File: ☐

Code Generation — ARM Compiler: Use default compiler version 6

Use MicroLIB: ✓ Big Endian: ☐

Floating Point Hardware: Single Precision

Read/Only Memory Areas default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
✓	IROM1:	0x8000000	0x300000	●
☐	IROM2:			○

Read/Write Memory Areas default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
✓	IRAM1:	0x20000000	0x30000	☐
☐	IRAM2:	0x10000000	0x10000	☐

3.4. 修改配置文件

在添加完相应的 LVGL 文件后，需要修改 lv_conf.h 文件，修改内容如下：

1. 将第 10 行“#if 0”修改为“#if 1”，使能该配置文件；
2. 修改第 24 行，将“LV_COLOR_DEPTH 32”修改为“LV_COLOR_DEPTH 16”，即配置色彩的显示模式为 RGB565；
3. GD32F450i-EVAL-V1.1 开发板配备一块分辨率为 480*272 的屏幕，因此添加宏定义：“#define LV_HOR_RES_MAX (480)”和“#define LV_VER_RES_MAX (272)”，其中

LV_HOR_RES_MAX 为显示器的水平分辨率，LV_VER_RES_MAX 为显示器的垂直分辨率。

3.5. 添加定时器为 LVGL 提供心跳

配置 TIMER6 定时 1ms 进入中断，在中断中为 LVGL 提供 1ms 的心跳节拍。TIMER6 相关配置及中断服务函数如*表 3-1. TIMER 配置函数*所示。

表 3-1. TIMER 配置函数

```
/* this function handles Timer6 Handler exception */
void TIMER6_IRQHandler(void)
{
    if(SET == timer_interrupt_flag_get(TIMER6, TIMER_INT_FLAG_UP)){
        /* clear channel 0 interrupt bit */
        timer_interrupt_flag_clear(TIMER6, TIMER_INT_FLAG_UP);
        lv_tick_inc(1);
    }
}

void timer6_config()
{
    /* ----------------------------------------------------------------------------
    TIMER6 Configuration:
    TIMER6CLK = SystemCoreClock/20000 = 10kHz, the period is 1s(0.1ms).
    ---------------------------------------------------------------------------- */
    timer_parameter_struct timer_initpara;
    rcu_periph_clock_enable(RCU_TIMER6);
    timer_deinit(TIMER6);
    /* initialize TIMER init parameter struct */
    timer_struct_para_init(&timer_initpara);
    /* TIMER6 configuration */
    timer_initpara.prescaler            = 19999;
    timer_initpara.alignedmode          = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection     = TIMER_COUNTER_UP;
    timer_initpara.period               = 9;
    timer_initpara.clockdivision        = TIMER_CKDIV_DIV1;
    timer_initpara.repetitioncounter = 0;
    timer_init(TIMER6, &timer_initpara);
    timer_enable(TIMER6);
    timer_interrupt_flag_clear(TIMER6, TIMER_INT_FLAG_UP);
    timer_interrupt_enable(TIMER6, TIMER_INT_UP);
}
    nvic_irq_enable(TIMER6_IRQn, 1, 1);
```

3.6. 移植显示驱动

LVGL 的显示驱动文件主要修改 lv_prot_disp.h 和 lv_prot_disp.c 文件，因为开发板板载一颗 MT48LC16M16A2 SDRAM 芯片，大小为 32MB，具有较大的存储空间，所以在代码中将 TLI 中 frame buffer 定义到地址 0x0c000000，参考*表 3-8. main.c 源码*。在 LVGL 显示驱动接口定义 COLOR_BUF_SIZE 大小的缓冲区作为 LVGL 的 buffer，通过 DMA 的方式对屏幕像素数据进行刷新，修改后代码如下*表 3-2. lv_prot_disp.h 源码*和*表 3-3. lv_prot_disp.c 源码*所示：

表 3-2. lv_prot_disp.h 源码

```
/* copy this file as "lv_port_disp.h" and set this value to "1" to enable content */
#if 1
#ifndef LV_PORT_DISP_TEMPL_H
#define LV_PORT_DISP_TEMPL_H
#ifdef __cplusplus
extern "C" {
#endif
#include "lvgl.h"
void lv_port_disp_init(void);
#ifdef __cplusplus
}
#endif
#endif /*LV_PORT_DISP_TEMPL_H*/

#endif /*Disable/Enable content*/
```

表 3-3. lv_prot_disp.c 源码

```
#include "lv_port_disp_template.h"
#include "../../lvgl.h"
#include "gd32f4xx.h"

static void disp_flush(lv_disp_drv_t *disp_drv, const lv_area_t *area, lv_color_t *color_p);
static int32_t              x1_flush;
static int32_t              y1_flush;
static int32_t              x2_flush;
static int32_t              y2_fill;
static int32_t              y_fill_act;
static const lv_color_t *buf_to_flush;
static lv_disp_t *our_disp = NULL;
static lv_disp_drv_t disp_drv;
extern uint16_t *my_fb;
#define LCD_FRAME_BUF_ADDR              0XC0000000
#define TLI_LCD_FRAMEBUF_SIZE         (480*272*2)
#define COLOR_BUF_SIZE                (LV_HOR_RES_MAX*LV_VER_RES_MAX)
```

```
static lv_color_t
color_buf2[COLOR_BUF_SIZE]__attribute__(( section(".ARM.__at_0xC007E900")));

/*!
    \brief          dma transfer data
    \param[in]   src_addr: source address
    \param[in]   dst_addr: destination address
    \param[in]   datalength: data length
    \param[out] none
    \retval         none
*/
void dma_transfer(uint32_t src_addr, uint32_t dst_addr, uint32_t datalength)
{
    dma_periph_address_config(DMA1, DMA_CH0, src_addr);
    dma_memory_address_config(DMA1, DMA_CH0, DMA_MEMORY_0, dst_addr);
    dma_transfer_number_config(DMA1, DMA_CH0, datalength);
    dma_interrupt_disable(DMA1, DMA_CH0, DMA_CHXCTL_HTFIE);
    dma_interrupt_enable(DMA1, DMA_CH0, DMA_CHXCTL_FTFIE);
    dma_channel_enable(DMA1, DMA_CH0);
}

void lv_port_disp_init(void)
{
    static lv_disp_draw_buf_t buf;
    lv_disp_draw_buf_init(&buf, color_buf2, NULL, LV_HOR_RES_MAX * LV_VER_RES_MAX);
    lv_disp_drv_init(&disp_drv);
    disp_drv.draw_buf = &buf;
    disp_drv.flush_cb = disp_flush;
    disp_drv.hor_res = LV_HOR_RES_MAX;
    disp_drv.ver_res = LV_VER_RES_MAX;
    lv_disp_drv_register(&disp_drv);
}

/*Flush the content of the internal buffer the specific area on the display
 *You can use DMA or any hardware acceleration to do this operation in the background but
 *'lv_disp_flush_ready()' has to be called when finished.*/
static void disp_flush(lv_disp_drv_t *disp_drv, const lv_area_t *area, lv_color_t *color_p)
{
    /*Return if the area is out the screen*/
    if(area->x2 < 0) {
        return;
    }
    if(area->y2 < 0) {
        return;
    }
    if(area->x1 > LV_HOR_RES_MAX - 1) {
        return;
    }
    if(area->y1 > LV_VER_RES_MAX - 1) {
        return;
    }

    /*Truncate the area to the screen*/
    int32_t act_x1 = area->x1 < 0 ? 0 : area->x1;
    int32_t act_y1 = area->y1 < 0 ? 0 : area->y1;
    int32_t act_x2 = area->x2 > LV_HOR_RES_MAX - 1 ? LV_HOR_RES_MAX - 1 : area->x2;
    int32_t act_y2 = area->y2 > LV_VER_RES_MAX - 1 ? LV_VER_RES_MAX - 1 : area->y2;

    x1_flush = act_x1;
    y1_flush = act_y1;
    x2_flush = act_x2;
    y2_fill = act_y2;
    y_fill_act = act_y1;
    buf_to_flush = color_p;
    dma_transfer((uint32_t)buf_to_flush, (uint32_t)&my_fb[y_fill_act * LV_HOR_RES_MAX +
x1_flush],(x2_flush - x1_flush + 1));
}

/* this function handles DMA1 Handler exception */
void DMA1_Channel0_IRQHandler(void)
{
    if(dma_interrupt_flag_get(DMA1, DMA_CH0, DMA_INT_FLAG_FTF)) {
        dma_interrupt_flag_clear(DMA1, DMA_CH0, DMA_INT_FLAG_FTF);
        y_fill_act ++;

        if(y_fill_act > y2_fill) {
            lv_disp_flush_ready(&disp_drv);
        } else {
            buf_to_flush += x2_flush - x1_flush + 1;
            dma_transfer((uint32_t)buf_to_flush, (uint32_t)&my_fb[y_fill_act *
LV_HOR_RES_MAX + x1_flush],
                                    (x2_flush - x1_flush + 1));
        }
    }

}
```

3.7. 移植触摸驱动

LVGL 的触摸驱动文件主要修改 lv_prot_indev.h 和 lv_prot_indev.c 文件，修改后内容如*表 3-4. lv_prot_indev.h 源码*和*表 3-5. lv_prot_indev.c 源码*所示，其中所包含的 touch.c 和 touch.h 文件参考*表 3-6. touch.h 源码*和*表 3-7. touch.c 源码*。

表 3-4. lv_prot_indev.h 源码

```
/* copy this file as "lv_port_indev.h" and set this value to "1" to enable content */
#if 1
#ifndef LV_PORT_INDEV_TEMPL_H
#define LV_PORT_INDEV_TEMPL_H
#ifdef __cplusplus
extern "C" {
#endif
#include "lvgl.h"
void lv_port_indev_init(void);
#ifdef __cplusplus
} /*extern "C"*/
#endif
#endif /*LV_PORT_INDEV_TEMPL_H*/

#endif /*Disable/Enable content*/
```

表 3-5. lv_prot_indev.c 源码

```
#include "lv_port_indev_template.h"
#include "../../lvgl.h"
#include "touch_panel.h"

extern uint16_t touch_ad_x,touch_ad_y;
static void touchpad_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data);
static void touchpad_get_xy(lv_coord_t * x, lv_coord_t * y);

lv_indev_t * indev_touchpad;
lv_indev_t * indev_mouse;
lv_indev_t * indev_keypad;
lv_indev_t * indev_encoder;
lv_indev_t * indev_button;

static int32_t encoder_diff;
static lv_indev_state_t encoder_state;

void lv_port_indev_init(void)
{
    static lv_indev_drv_t indev_drv;
    /*Register a touchpad input device*/
    lv_indev_drv_init(&indev_drv);
    indev_drv.type = LV_INDEV_TYPE_POINTER;
    indev_drv.read_cb = touchpad_read;
    indev_touchpad = lv_indev_drv_register(&indev_drv);
}

/* will be called by the library to read the touchpad */
static void touchpad_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data)
{
    static lv_coord_t last_x = 0;
    static lv_coord_t last_y = 0;

    /* Save the pressed coordinates and the state */
    if(touch_scan()) {
        touchpad_get_xy(&last_x, &last_y);
        data->state = LV_INDEV_STATE_PR;
    } else {
        data->state = LV_INDEV_STATE_REL;
    }

    /* set the last pressed coordinates */
    data->point.x = last_x;
    data->point.y = last_y;
}

/*Get the x and y coordinates if the touchpad is pressed*/
static void touchpad_get_xy(lv_coord_t * x, lv_coord_t * y)
{
    (*x) = touch_coordinate_x_get(touch_ad_x);
    (*y) = LCD_Y - touch_coordinate_y_get(touch_ad_y);
}
/*Get the x and y coordinates if the touchpad is pressed*/
static void touchpad_get_xy(lv_coord_t * x, lv_coord_t * y)
{
    /*Your code comes here*/

    (*x) = touch_coordinate_x_get(touch_ad_x);
    (*y) = LCD_Y - touch_coordinate_y_get(touch_ad_y);

}
```

表 3-6. touch.h 源码

```
#ifndef TOUCH_PANEL_H
```

```
#define TOUCH_PANEL_H

#include "gd32f4xx.h"

#define AD_Left       300
#define AD_Right      3850
#define AD_Top        220
#define AD_Bottom     3850

#define LCD_X         480
#define LCD_Y         272

#define CH_X          0XD2
#define CH_Y          0X92

/* SPI SCK pin */
#define   SPI_SCK_PIN                   GPIO_PIN_7
#define   SPI_SCK_PORT                   GPIOF
#define   SPI_SCK_LOW()                 gpio_bit_reset(SPI_SCK_PORT, SPI_SCK_PIN)
#define   SPI_SCK_HIGH()                gpio_bit_set(SPI_SCK_PORT, SPI_SCK_PIN)

/* SPI MOSI pin */
#define   SPI_MOSI_PIN                  GPIO_PIN_9
#define   SPI_MOSI_PORT                  GPIOF
#define   SPI_MOSI_LOW()                gpio_bit_reset(SPI_MOSI_PORT, SPI_MOSI_PIN)
#define   SPI_MOSI_HIGH()               gpio_bit_set(SPI_MOSI_PORT, SPI_MOSI_PIN)

/* SPI MISO pin */
#define   SPI_MISO_PIN                  GPIO_PIN_8
#define   SPI_MISO_PORT                  GPIOF
#define   SPI_MISO_READ()               gpio_input_bit_get(SPI_MISO_PORT, SPI_MISO_PIN)

/* SPI Chip select pin */
#define   SPI_TOUCH_CS_PIN               GPIO_PIN_6
#define   SPI_TOUCH_CS_PORT               GPIOF
#define   SPI_TOUCH_CS_LOW()             gpio_bit_reset(SPI_TOUCH_CS_PORT,
SPI_TOUCH_CS_PIN)
#define   SPI_TOUCH_CS_HIGH()            gpio_bit_set(SPI_TOUCH_CS_PORT,
SPI_TOUCH_CS_PIN)

/* LCD touch interrupt request pin */
#define   TOUCH_PEN_INT_PIN              GPIO_PIN_3
#define   TOUCH_PEN_INT_PORT              GPIOI
```

```
#define   TOUCH_PEN_INT_READ()          gpio_input_bit_get(TOUCH_PEN_INT_PORT,
TOUCH_PEN_INT_PIN)
/* touch panel gpio configure */
void touch_panel_gpio_configure(void);
/* touch start */
void touch_start(void);
/* write data to touch screen */
void touch_write(uint8_t d);
/* read the touch AD value */
uint16_t touch_read(void);
/* read the touch pen interrupt request signal */
FlagStatus touch_pen_irq(void);
/* get the AD sample value of touch location at X coordinate */
uint16_t touch_ad_x_get(void);
/* get the AD sample value of touch location at Y coordinate */
uint16_t touch_ad_y_get(void);
/* get channel X+ AD average sample value */
uint16_t touch_average_ad_x_get(void);
/* get channel Y+ AD average sample value */
uint16_t touch_average_ad_y_get(void);
/* get X coordinate value of touch point on LCD screen */
uint16_t touch_coordinate_x_get(uint16_t adx);
/* get Y coordinate value of touch point on LCD screen */
uint16_t touch_coordinate_y_get(uint16_t ady);

uint16_t touch_data_filter(uint8_t channel_select);
ErrStatus touch_ad_xy_get(uint16_t *ad_x, uint16_t *ad_y);

ErrStatus touch_scan(void);
```

表 3-7. touch.c 源码

```
#include "gd32f4xx.h"
#include "touch_panel.h"
#include "math.h"
#include <stdlib.h>

/* number of filter reads */
#define FILTER_READ_TIMES                5
/* lost value of filter */
#define FILTER_LOST_VAL                 1
/* error range of AD sample value */
#define AD_ERR_RANGE                     6
uint16_t touch_ad_x=0,touch_ad_y=0;
/* set or reset touch screen chip select pin */
```

```
static void spi_cs(uint8_t a);
/* set or reset SPI MOSI pin */
static void spi_mosi(uint8_t a);
/* set or reset touch screen clock SPI SCK pin */
static void spi_clk(uint8_t a);
/* get SPI MISO pin input status */
static FlagStatus spi_miso(void);
/* SPI delay function */
static void spi_delay(uint16_t i);
/*!
    \brief      touch panel gpio configure
    \param[in]  none
    \param[out] none
    \retval     none
*/
void touch_panel_gpio_configure(void)
{
    /* GPIO clock enable */
    rcu_periph_clock_enable(RCU_GPIOI);
    rcu_periph_clock_enable(RCU_GPIOB);
    rcu_periph_clock_enable(RCU_GPIOF);

     gpio_af_set(SPI_SCK_PORT, GPIO_AF_5, SPI_SCK_PIN);
    gpio_mode_set(SPI_SCK_PORT, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
SPI_SCK_PIN);
    gpio_output_options_set(SPI_SCK_PORT, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,
SPI_SCK_PIN);

     gpio_af_set(SPI_MOSI_PORT, GPIO_AF_5, SPI_MOSI_PIN);
    gpio_mode_set(SPI_MOSI_PORT, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
SPI_MOSI_PIN);
    gpio_output_options_set(SPI_MOSI_PORT, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,
SPI_MOSI_PIN);

    gpio_af_set(SPI_MISO_PORT, GPIO_AF_5, SPI_MISO_PIN);
    gpio_mode_set(SPI_MISO_PORT, GPIO_MODE_INPUT, GPIO_PUPD_NONE,
SPI_MISO_PIN);
    gpio_output_options_set(SPI_MISO_PORT, GPIO_OTYPE_PP, GPIO_OSPEED_50MHZ,
SPI_MISO_PIN);
    gpio_mode_set(SPI_TOUCH_CS_PORT, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
SPI_TOUCH_CS_PIN);
    gpio_output_options_set(SPI_TOUCH_CS_PORT, GPIO_OTYPE_PP,
GPIO_OSPEED_50MHZ, SPI_TOUCH_CS_PIN);
```

```
    /* touch pen IRQ pin PI3 configure */
    gpio_mode_set(TOUCH_PEN_INT_PORT, GPIO_MODE_INPUT, GPIO_PUPD_NONE,
TOUCH_PEN_INT_PIN);
    gpio_output_options_set(TOUCH_PEN_INT_PORT, GPIO_OTYPE_PP,
GPIO_OSPEED_50MHZ, TOUCH_PEN_INT_PIN);

    /* set chip select pin high */
    SPI_TOUCH_CS_HIGH();
}
/*!
    \brief      set or reset touch screen chip select pin
    \param[in]  a: specified the low or high level of chip select pin output
    \param[out] none
    \retval     none
*/
static void spi_cs(uint8_t a)
{
    if(a){
        SPI_TOUCH_CS_HIGH();
    }else{
        SPI_TOUCH_CS_LOW();
    }
}

/*!
    \brief      set or reset SPI MOSI pin
    \param[in]  a: specified the low or high level of SPI MOSI pin output
    \param[out] none
    \retval     none
*/
static void spi_mosi(uint8_t a)
{
    if(a){
        SPI_MOSI_HIGH();
    }else{
        SPI_MOSI_LOW();
    }
}

/*!
    \brief      set or reset touch screen clock SPI SCK pin
    \param[in]  a: specified the low or high level of SPI SCK pin output
    \param[out] none
    \retval     none
*/
static void spi_clk(uint8_t a)
{
    if(a){
        SPI_SCK_HIGH();
    }else{
        SPI_SCK_LOW();
    }
}

/*!
    \brief      get SPI MISO pin input status
    \param[in]  none
    \param[out] none
    \retval     input status of gpio pin: SET or RESET
*/
static FlagStatus spi_miso(void)
{
    return SPI_MISO_READ();
}

/*!
    \brief      SPI delay function
    \param[in]  none
    \param[out] none
    \retval     none
*/
static void spi_delay(uint16_t i)
{
    uint16_t k;
    for (k=0;k<i;k++);
}

/*!
    \brief      touch start
    \param[in]  none
    \param[out] none
    \retval     none
*/
void touch_start(void)
{
    spi_clk(0);
    spi_cs(1);
    spi_mosi(1);
    spi_clk(1);
    spi_cs(0);
}

/*!
    \brief        write data to touch screen
    \param[in]    d: the data to be written
    \param[out] none
    \retval       none
*/
void touch_write(uint8_t d)
{
    uint8_t buf, i ;
    spi_clk(0);
    for( i = 0; i < 8; i++){
        buf = ((d >> (7-i)) & 0x1);
        spi_mosi(buf);
        spi_clk(0);
        spi_clk(1);
        spi_clk(0);
    }
}

/*!
    \brief        read the touch AD value
    \param[in]    None
    \param[out] none
    \retval       the value of touch AD
*/
uint16_t touch_read(void)
{
    uint16_t buf ;
    uint8_t i ;

    buf=0;
    for( i = 0; i < 12; i++){
        buf = buf << 1 ;
        spi_clk(1);
        spi_clk(0);
        if(RESET != spi_miso()){
            buf = buf + 1 ;
        }
    }
    return( buf );
}

/*!
    \brief          read the touch pen interrupt request signal
    \param[in]      none
    \param[out]     none
    \retval         the status of touch pen: SET or RESET
      \arg              SET: touch pen is inactive
      \arg              RESET: touch pen is active
*/
FlagStatus touch_pen_irq(void)
{
    return TOUCH_PEN_INT_READ();
}

/*!
    \brief          get the AD sample value of touch location at X coordinate
    \param[in]      none
    \param[out]     none
    \retval         channel X+ AD sample value
*/
uint16_t touch_ad_x_get(void)
{
    if (RESET != touch_pen_irq()){
        /* touch pen is inactive */
        return 0;
    }
    touch_start();
    touch_write(0x00);
    touch_write(CH_X);
    return (touch_read());
}

/*!
    \brief          get the AD sample value of touch location at Y coordinate
    \param[in]      none
    \param[out]     none
    \retval         channel Y+ AD sample value
*/
```

```
uint16_t touch_ad_y_get(void)
{
    if (RESET != touch_pen_irq()){
        /* touch pen is inactive */
        return 0;
    }
    touch_start();
    touch_write(0x00);
    touch_write(CH_Y);
    return (touch_read());
}

/*!
    \brief      get channel X+ AD average sample value
    \param[in]  none
    \param[out] none
    \retval     channel X+ AD average sample value
*/
uint16_t touch_average_ad_x_get(void)
{
    uint8_t i;
    uint16_t temp=0;
    for (i=0;i<8;i++){
        temp+=touch_ad_x_get();
        spi_delay(1000);
    }
    temp>>=3;

    return temp;
}

/*!
    \brief      get channel Y+ AD average sample value
    \param[in]  none
    \param[out] none
    \retval     channel Y+ AD average sample value
*/
uint16_t touch_average_ad_y_get(void)
{
    uint8_t i;
    uint16_t temp=0;
    for (i=0;i<8;i++){
        temp+=touch_ad_y_get();
        spi_delay(1000);
    }
    temp>>=3;

    return temp;
}

/*!
    \brief        get X coordinate value of touch point on LCD screen
    \param[in]    adx : channel X+ AD average sample value
    \param[out] none
    \retval       X coordinate value of touch point
*/
uint16_t touch_coordinate_x_get(uint16_t adx)
{
    uint16_t sx = 0;
    uint32_t
    r = adx - AD_Left;
    r *= LCD_X - 1;
    sx =   r / (AD_Right - AD_Left);
    if (sx <= 0 || sx > LCD_X)
        return 0;
    return sx;
}

/*!
    \brief        get Y coordinate value of touch point on LCD screen
    \param[in]    ady : channel Y+ AD average sample value
    \param[out] none
    \retval       Y coordinate value of touch point
*/
uint16_t touch_coordinate_y_get(uint16_t ady)
{
    uint16_t sy = 0;
    uint32_t
    r = ady - AD_Top;
    r *= LCD_Y - 1;
    sy =   r / (AD_Bottom - AD_Top);
    if (sy <= 0 || sy > LCD_Y)
        return 0;
    return sy;
}
```

```
/*!
    \brief          get a value (X or Y) for several times. Order these values,
                    remove the lowest and highest and obtain the average value
    \param[in]    channel_select: select channel X or Y
      \arg              CH_X: channel X
      \arg              CH_Y: channel Y
    \param[out] none
    \retval         a value(X or Y) of touch point
*/
uint16_t touch_data_filter(uint8_t channel_select)
{
    uint16_t i=0, j=0;
    uint16_t buf[FILTER_READ_TIMES];
    uint16_t sum=0;
    uint16_t temp=0;
    /* Read data in FILTER_READ_TIMES times */
    for(i=0; i < FILTER_READ_TIMES; i++){
        if (CH_X == channel_select){
            buf[i] = touch_ad_x_get();
        }else{      /* CH_Y == channel_select */
            buf[i] = touch_ad_y_get();
        }
    }
    /* Sort in ascending sequence */
    for(i = 0; i < FILTER_READ_TIMES - 1; i++){
        for(j = i + 1; j < FILTER_READ_TIMES; j++){
            if(buf[i] > buf[j]){
                temp = buf[i];
                buf[i] = buf[j];
                buf[j] = temp;
            }
        }
    }
    sum = 0;
    for(i = FILTER_LOST_VAL; i < FILTER_READ_TIMES - FILTER_LOST_VAL; i++){
        sum += buf[i];
    }
    temp = sum / (FILTER_READ_TIMES - 2 * FILTER_LOST_VAL);

    return temp;
}

/*!
```

```
    \brief        get the AD sample value of touch location.
                     get the sample value for several times,order these values,remove the lowest and
highest and obtain the average value
    \param[in]   channel_select: select channel X or Y
    \param[out] none
      \arg           ad_x: channel X AD sample value
      \arg           ad_y: channel Y AD sample value
    \retval        ErrStatus: SUCCESS or ERROR
*/
ErrStatus touch_ad_xy_get(uint16_t *ad_x, uint16_t *ad_y)
{
    uint16_t ad_x1=0, ad_y1=0, ad_x2=0, ad_y2=0;

    ad_x1 = touch_data_filter(CH_X);
    ad_y1 = touch_data_filter(CH_Y);
    ad_x2 = touch_data_filter(CH_X);
    ad_y2 = touch_data_filter(CH_Y);

    if((abs(ad_x1 - ad_x2) > AD_ERR_RANGE) || (abs(ad_y1 - ad_y2) > AD_ERR_RANGE)){
        return ERROR;
    }
    *ad_x = (ad_x1 + ad_x2) / 2;
    *ad_y = (ad_y1 + ad_y2) / 2;

    return SUCCESS;
}

/*!
    \brief        detect the touch event
    \param[in]   none
    \param[out] none
    \retval        ErrStatus: SUCCESS or ERROR
*/
ErrStatus touch_scan(void)
{
    uint8_t invalid_count = 0;
    if (RESET == touch_pen_irq()){ /* touch pen is active */
        while((SUCCESS != touch_ad_xy_get(&touch_ad_x, &touch_ad_y))&& (invalid_count <
20)){
            invalid_count++;
        }

        if(invalid_count >= 20){
                touch_ad_x = 0;
                touch_ad_y = 0;
                return ERROR;
            }
        }else{
            touch_ad_x = 0;
            touch_ad_y = 0;
            return ERROR;
        }
        return SUCCESS;
}
```

3.8. 主函数代码编写

在主函数中先后完成 EXMC 接口对 SDRAM 初始化，TLI 接口对 LCD 初始化，SPI 接口对触控芯片初始化，DMA 初始化和 LVGL 初始化，在初始化完成后调用官方 example 例程，手册调用 lv_example_scroll_6()函数，在 while(1)中不断调用 lv_task_handler()函数，具体代码如*表 3-8. main.c 源码*。

表 3-8. main.c 源码

```
#include "gd32f4xx.h"
#include "gd32f450i_eval.h"
#include "systick.h"
#include <stdio.h>
#include "exmc_sdram.h"
#include "../../lvgl.h"
#include "lv_port_disp_template.h"
#include "lv_port_indev_template.h"
#include "../lv_examples.h"
#include "lv_demo.h"
#include "touch_panel.h"

#define HORIZONTAL_SYNCHRONOUS_PULSE   41
#define HORIZONTAL_BACK_PORCH          2
#define ACTIVE_WIDTH                   480
#define HORIZONTAL_FRONT_PORCH         2

#define VERTICAL_SYNCHRONOUS_PULSE     10
#define VERTICAL_BACK_PORCH            2
#define ACTIVE_HEIGHT                  272
#define VERTICAL_FRONT_PORCH           2

__IO uint16_t * my_fb = (__IO uint16_t*) (0xc0000000);
```

```
static void tli_config(void);
static void tli_blend_config(void);
static void tli_gpio_config(void);
static void lcd_config(void);
static void lcd_init(void);
static void dma_config(void);
/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    /* configure systick */
    systick_config();
    nvic_priority_group_set(NVIC_PRIGROUP_PRE1_SUB3);

    /* config the EXMC access mode */
    exmc_synchronous_dynamic_ram_init(EXMC_SDRAM_DEVICE0);
    lcd_config();
    lcd_init();
    /* configure the GPIO of SPI touch panel */
    touch_panel_gpio_configure();
    delay_1ms(50);
    dma_config();
    delay_1ms(1000);

    lv_init();
    lv_port_disp_init();
    lv_port_indev_init();
    lv_example_scroll_6();
    while(1){
        lv_task_handler();
    }
}

static void dma_config(void)
{
    dma_multi_data_parameter_struct  dma_init_parameter;
    /* peripheral clock enable */
    rcu_periph_clock_enable(RCU_DMA1);
    /* DMA peripheral configure */
    dma_deinit(DMA1,DMA_CH0);

    dma_init_parameter.periph_width = DMA_PERIPH_WIDTH_16BIT;
    dma_init_parameter.periph_inc = DMA_PERIPH_INCREASE_ENABLE;
    dma_init_parameter.memory_width = DMA_MEMORY_WIDTH_16BIT;
    dma_init_parameter.memory_inc = DMA_MEMORY_INCREASE_ENABLE;
    dma_init_parameter.memory_burst_width = DMA_MEMORY_BURST_4_BEAT;
    dma_init_parameter.periph_burst_width = DMA_PERIPH_BURST_4_BEAT;
    dma_init_parameter.critical_value = DMA_FIFO_4_WORD;
    dma_init_parameter.circular_mode = DMA_CIRCULAR_MODE_DISABLE;
    dma_init_parameter.direction = DMA_MEMORY_TO_MEMORY;
    dma_init_parameter.priority = DMA_PRIORITY_ULTRA_HIGH;
    dma_multi_data_mode_init(DMA1,DMA_CH0,&dma_init_parameter);

    nvic_irq_enable(DMA1_Channel0_IRQn, 0, 0);
}

/*!
    \brief      LCD Configuration
    \param[in]  none
    \param[out] none
    \retval     none
*/
static void lcd_config(void)
{
    /* configure the GPIO of TLI */
    tli_gpio_config();
    /* configure TLI peripheral */
    tli_config();
}

static void lcd_init(void)
{
    tli_layer_enable(LAYER0);
    tli_layer_disable(LAYER1);
    tli_reload_config(TLI_RL_FBR);
    /* enable TLI */
    tli_enable();
}
/*!
    \brief      configure TLI peripheral
    \param[in]    none
    \param[out] none
    \retval          none
*/
static void tli_config(void)
{
    tli_parameter_struct                    tli_init_struct;
    tli_layer_parameter_struct              tli_layer_init_struct;

    rcu_periph_clock_enable(RCU_TLI);
    tli_gpio_config();

    /* configure the PLLSAI clock to generate lcd clock */
    if(ERROR == rcu_pllsai_config(192, 2, 3)){
        while(1);
    }

    rcu_tli_clock_div_config(RCU_PLLSAIR_DIV8);
    rcu_osci_on(RCU_PLLSAI_CK);
    if(ERROR == rcu_osci_stab_wait(RCU_PLLSAI_CK)){
        while(1);
    }
    /* configure TLI parameter struct */
    tli_init_struct.signalpolarity_hs = TLI_HSYN_ACTLIVE_LOW;
    tli_init_struct.signalpolarity_vs = TLI_VSYN_ACTLIVE_LOW;
    tli_init_struct.signalpolarity_de = TLI_DE_ACTLIVE_LOW;
    tli_init_struct.signalpolarity_pixelck = TLI_PIXEL_CLOCK_TLI;
    /* LCD display timing configuration */
    tli_init_struct.synpsz_hpsz = HORIZONTAL_SYNCHRONOUS_PULSE - 1;
    tli_init_struct.synpsz_vpsz = VERTICAL_SYNCHRONOUS_PULSE - 1;
    tli_init_struct.backpsz_hbpsz        =        HORIZONTAL_SYNCHRONOUS_PULSE        +
HORIZONTAL_BACK_PORCH - 1;
    tli_init_struct.backpsz_vbpsz          =          VERTICAL_SYNCHRONOUS_PULSE          +
VERTICAL_BACK_PORCH - 1;
    tli_init_struct.activesz_hasz          =          HORIZONTAL_SYNCHRONOUS_PULSE          +
HORIZONTAL_BACK_PORCH + ACTIVE_WIDTH - 1;
    tli_init_struct.activesz_vasz            =            VERTICAL_SYNCHRONOUS_PULSE            +
VERTICAL_BACK_PORCH + ACTIVE_HEIGHT - 1;
    tli_init_struct.totalsz_htsz            =            HORIZONTAL_SYNCHRONOUS_PULSE            +
HORIZONTAL_BACK_PORCH + ACTIVE_WIDTH + HORIZONTAL_FRONT_PORCH - 1;
    tli_init_struct.totalsz_vtsz              =              VERTICAL_SYNCHRONOUS_PULSE              +
VERTICAL_BACK_PORCH + ACTIVE_HEIGHT + VERTICAL_FRONT_PORCH - 1;
    tli_init_struct.backcolor_red = 0;
    tli_init_struct.backcolor_green = 0;
    tli_init_struct.backcolor_blue = 0;
    tli_init(&tli_init_struct);
    memset((void*)my_fb, 0x0000, LV_HOR_RES_MAX * LV_VER_RES_MAX*16);
    /* TLI layer0 configuration */
    /* TLI window size configuration */
    tli_layer_init_struct.layer_window_leftpos = HORIZONTAL_SYNCHRONOUS_PULSE +
HORIZONTAL_BACK_PORCH;
    tli_layer_init_struct.layer_window_rightpos = (480 + HORIZONTAL_SYNCHRONOUS_PULSE
+ HORIZONTAL_BACK_PORCH - 1);
    tli_layer_init_struct.layer_window_toppos = VERTICAL_SYNCHRONOUS_PULSE +
VERTICAL_BACK_PORCH;
    tli_layer_init_struct.layer_window_bottompos = (272 + VERTICAL_SYNCHRONOUS_PULSE +
VERTICAL_BACK_PORCH - 1);
    /* TLI window pixel format configuration */
    tli_layer_init_struct.layer_ppf = LAYER_PPF_RGB565;
    /* TLI window specified alpha configuration */
    tli_layer_init_struct.layer_sa = 255;
    /* TLI layer default alpha R,G,B value configuration */
    tli_layer_init_struct.layer_default_blue = 0;
    tli_layer_init_struct.layer_default_green = 0;
    tli_layer_init_struct.layer_default_red = 0;
    tli_layer_init_struct.layer_default_alpha = 0;
    /* TLI window blend configuration */
    tli_layer_init_struct.layer_acf1 = LAYER_ACF1_SA;
    tli_layer_init_struct.layer_acf2 = LAYER_ACF2_SA;
    /* TLI layer frame buffer base address configuration */
    tli_layer_init_struct.layer_frame_bufaddr = (uint32_t)my_fb;
    tli_layer_init_struct.layer_frame_line_length = ((480 * 2) + 3);
    tli_layer_init_struct.layer_frame_buf_stride_offset = (480 * 2);
    tli_layer_init_struct.layer_frame_total_line_number = 272;
    tli_layer_init(LAYER0, &tli_layer_init_struct);
}

/*!
    \brief      configure TLI GPIO
    \param[in]  none
    \param[out] none
    \retval     none
*/
static void tli_gpio_config(void)
{
    /* GPIO clock enable */
    rcu_periph_clock_enable(RCU_GPIOB);
    rcu_periph_clock_enable(RCU_GPIOE);
    rcu_periph_clock_enable(RCU_GPIOH);
    rcu_periph_clock_enable(RCU_GPIOI);
    rcu_periph_clock_enable(RCU_GPIOG);
    rcu_periph_clock_enable(RCU_GPIOF);

    /* configure HSYNC(PI10), VSYNC(PI9), PCLK(PG7) */
    /* configure LCD_R7(PG6), LCD_R6(PH12), LCD_R5(PH11), LCD_R4(PH10), 
LCD_R3(PH9),LCD_R2(PH8),
                 LCD_R1(PH3), LCD_R0(PH2), LCD_G7(PI2), LCD_G6(PI1), LCD_G5(PI0), 
LCD_G4(PH15),
                 LCD_G3(PH14), LCD_G2(PH13),LCD_G1(PE6), LCD_G0(PE5),LCD_B7(PI7), 
LCD_B6(PI6),
                 LCD_B5(PI5), LCD_B4(PI4), LCD_B3(PG11),LCD_B2(PG10), LCD_B1(PG12), 
LCD_B0(PE4) */
    /* TLI pins AF configure */
    gpio_af_set(GPIOE,GPIO_AF_14,GPIO_PIN_5);
    gpio_af_set(GPIOE,GPIO_AF_14,GPIO_PIN_6);
    gpio_af_set(GPIOE,GPIO_AF_14,GPIO_PIN_4);

    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_2);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_3);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_8);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_9);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_10);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_11);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_12);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_13);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_14);
    gpio_af_set(GPIOH,GPIO_AF_14,GPIO_PIN_15);

    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_0);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_1);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_2);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_4);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_5);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_6);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_7);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_9);
    gpio_af_set(GPIOI,GPIO_AF_14,GPIO_PIN_10);

    gpio_af_set(GPIOG,GPIO_AF_14,GPIO_PIN_6);
    gpio_af_set(GPIOG,GPIO_AF_14,GPIO_PIN_7);
    gpio_af_set(GPIOG,GPIO_AF_14,GPIO_PIN_10);
    gpio_af_set(GPIOG,GPIO_AF_14,GPIO_PIN_11);
    gpio_af_set(GPIOG,GPIO_AF_14,GPIO_PIN_12);

    /* configure TLI GPIO */
    gpio_mode_set(GPIOE,              GPIO_MODE_AF,              GPIO_PUPD_NONE,
GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6);
    gpio_output_options_set(GPIOE,                                GPIO_OTYPE_PP,
GPIO_OSPEED_200MHZ,GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6);

    gpio_mode_set(GPIOH,              GPIO_MODE_AF,              GPIO_PUPD_NONE,
GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10
                 |GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15);
    gpio_output_options_set(GPIOH,                                GPIO_OTYPE_PP,
GPIO_OSPEED_200MHZ,GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_8|GPIO_PIN_9

|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15);

    gpio_mode_set(GPIOI,              GPIO_MODE_AF,              GPIO_PUPD_NONE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_4
                 |GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_9|GPIO_PIN_10);
    gpio_output_options_set(GPIOI,                                GPIO_OTYPE_PP,
GPIO_OSPEED_200MHZ,GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_4
                 |GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_9|GPIO_PIN_10);

    gpio_mode_set(GPIOG,              GPIO_MODE_AF,              GPIO_PUPD_NONE,
GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12);
    gpio_output_options_set(GPIOG,           GPIO_OTYPE_PP,GPIO_OSPEED_200MHZ,
GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12);

    /* LCD PWM BackLight(PB15) */
    gpio_mode_set(GPIOB, GPIO_MODE_OUTPUT, GPIO_PUPD_PULLUP, GPIO_PIN_15);
    gpio_output_options_set(GPIOB,                                GPIO_OTYPE_PP,
GPIO_OSPEED_50MHZ,GPIO_PIN_15);
    gpio_bit_set(GPIOB,GPIO_PIN_15);
}
```

4. LVGL 演示效果

本手册演示官方提供的 scroll_1 和 scroll_6 例程演示，用户可根据需要进行不同例程的演示。

图 4-1. lv_example_scroll_1 演示效果

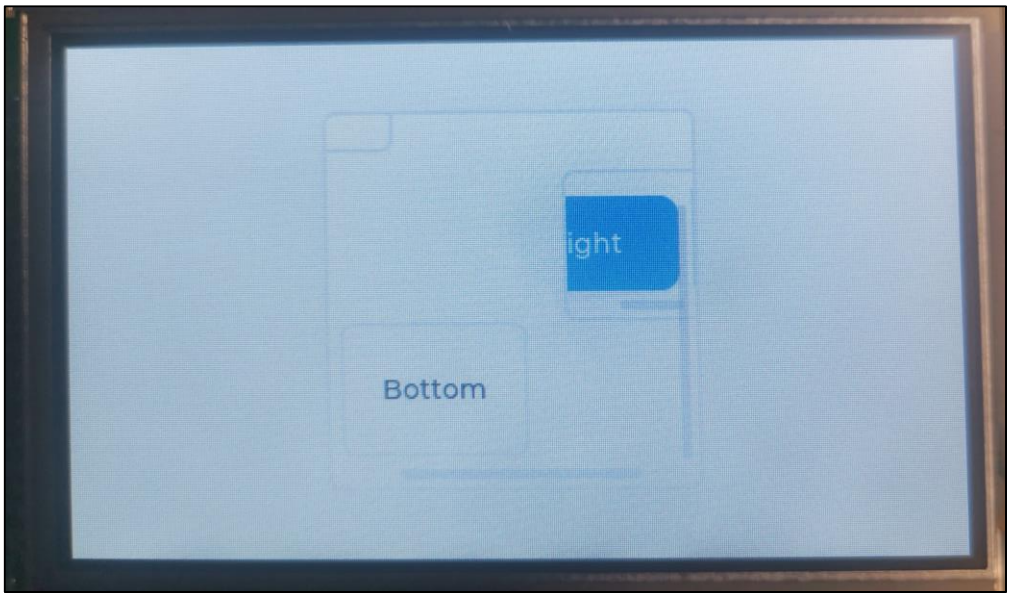
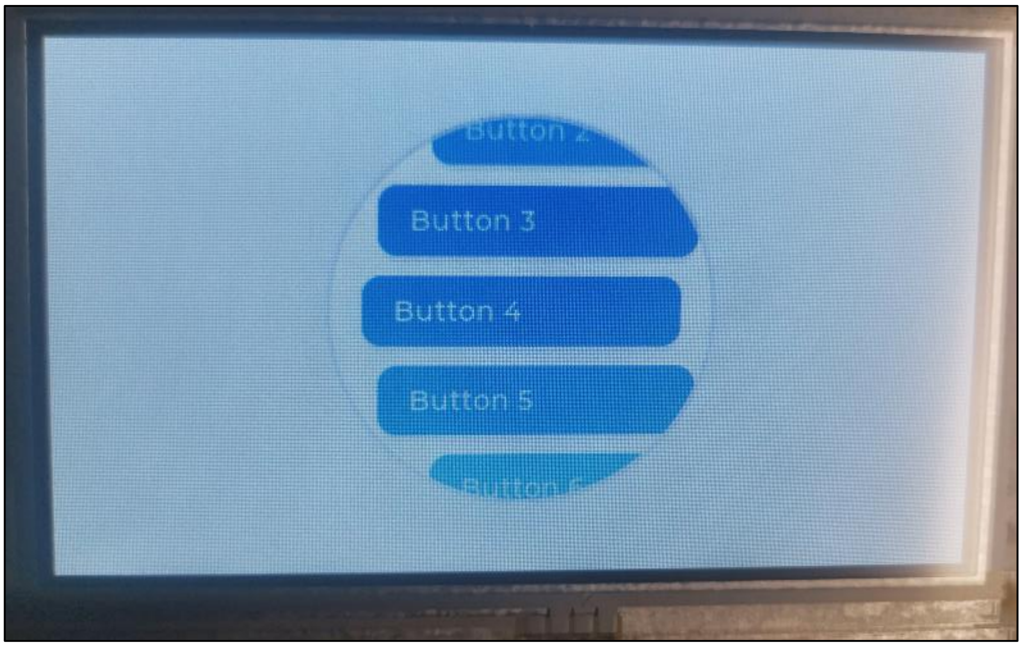


图 4-2. lv_example_scroll_6 演示效果



5. 版本历史

表 5-1. 版本历史

版本号.	描述	日期
1.0	首次发布	2021 年 11 月 30 日