

**计算机科学与工程学院**

**“嵌入式系统”实验报告书**

**题目：嵌入式系统作业3——LCD**

**学号：922106840127**

**姓名：刘宇翔**

**成绩**

**日期： 2025年 3月 14日**

# 1 题目要求

1. **题目设计要求**

【1】作业目标：

（1）熟悉FSMC工作原理；

（2）能够熟练掌握FSMC嵌入式程序开发；

（3）独自完成LCD、字库、图形、字模使用方法。

【2】作业内容：

设计程序将自己的中文姓名、学号和证据照显示在LCD屏上。

【3】完成要求：

（1）工程名称命名为：学号\_lcd;

（2）学号 每隔1分钟自动换一个颜色

（3）中文姓名显示采用跑马形式形式，从左向右移动，正好5秒移动完后，从左再此循环显示；

（4）中文姓名不得用图片形式，须由字模生成；

（5）程序代码附有必要的注释，在开发板上完成验证。

1. **拟实现的具体功能**

完成基于FSMC开发的，通过使用相关工具独自完成LCD、字库、图形、字模使用方法，编程实现本实验任务，编译通过后下载程序到开发板上，能够成功在LCD上展现个人的学号、姓名以及证件照的内容，完成实验要求。

# 2 总体设计

## 2.1硬件设计

**1. 硬件设计思路**

本次实验使用的是RT-Tread Studio为基础的工程模板，主要使用了LVGL库的头文件以及相关函数用于完成图像的显示功能。

MCU的LCD驱动程序与LVGL库接口对接，实现图形缓冲区数据传输。通过RT-Thread提供的多线程机制，分配独立任务处理跑马灯动画与学号颜色切换，保证实时性和响应性。

通过GPIO和定时器确保各显示模块与MCU之间的信号控制，避免因数据传输延时造成显示异常。

在硬件设计中，我也充分考虑了调试信号和测试点布局，便于在开发过程中进行电路和信号质量检查，确保系统各模块正常工作。

本硬件设计思路旨在搭建一个以MCU为核心、LCD为显示终端的嵌入式系统，通过RT-Thread操作系统和LVGL库，实现高效、实时的图形显示效果。设计过程中需兼顾性能、稳定性以及扩展性，确保系统在运行动画和颜色变换等动态效果时保持流畅、可靠的显示效果。该设计不仅满足实验要求，还为后续功能扩展预留了充足的接口和硬件资源。

## 2.2 软件设计

**1. 系统总体设计**

本程序基于 RT-Thread 实时操作系统和 LVGL 图形库，主要运行在嵌入式开发板上。程序的目标功能有两个：

**（1）学号颜色自动切换**：学号文本每隔 1 分钟更换一次显示颜色。

**（2）中文姓名跑马灯效果**：姓名文本以跑马灯的形式从屏幕左侧平滑移动至右侧，完成一次移动周期为 5 秒，移动结束后立即从左侧重新开始。

**（3）在LCD屏幕上展示我的证件照图像。**

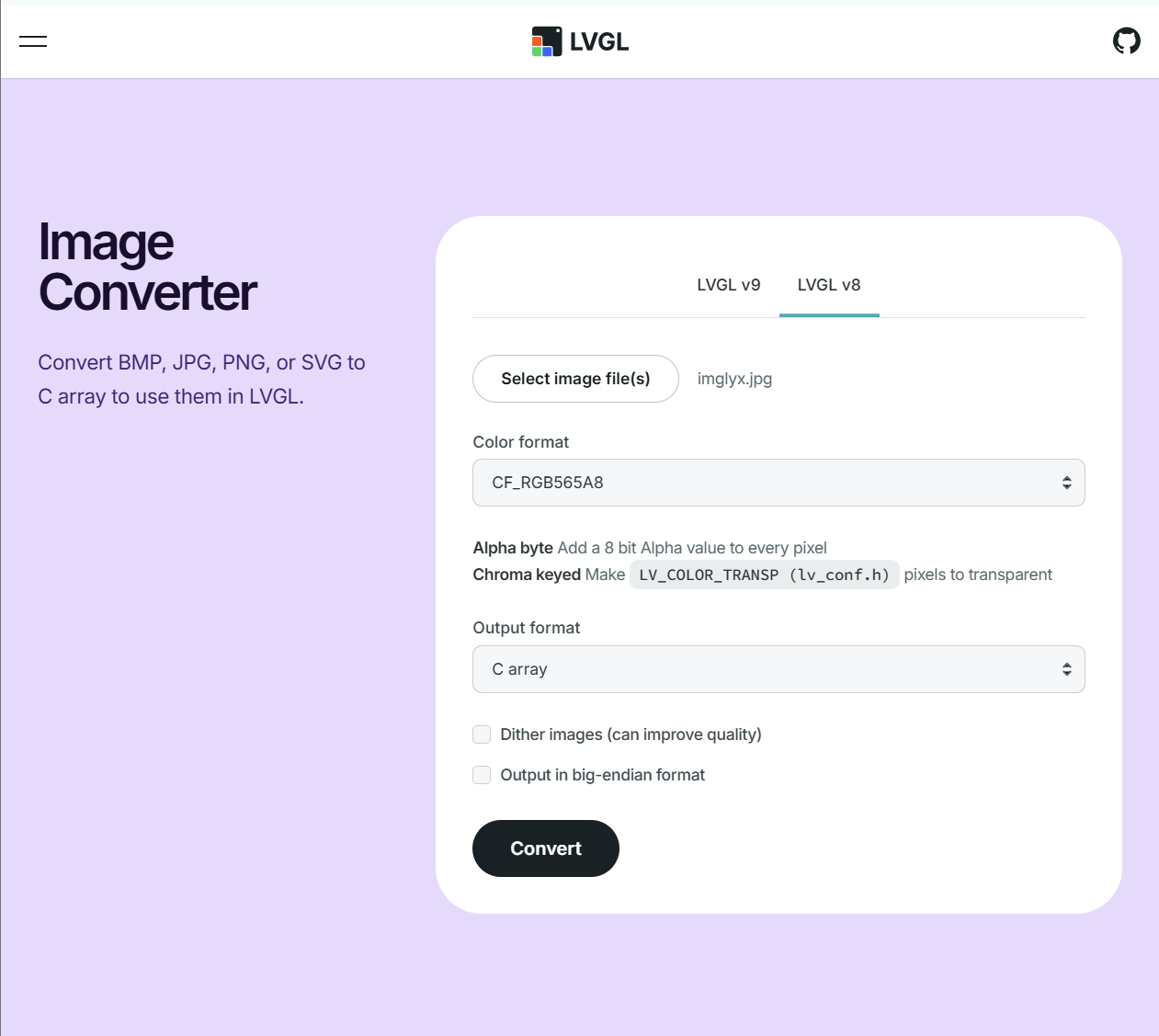
整个程序采用多线程设计，使用两个独立的线程分别处理这两个功能，从而互不干扰，保证界面动态效果流畅运行。

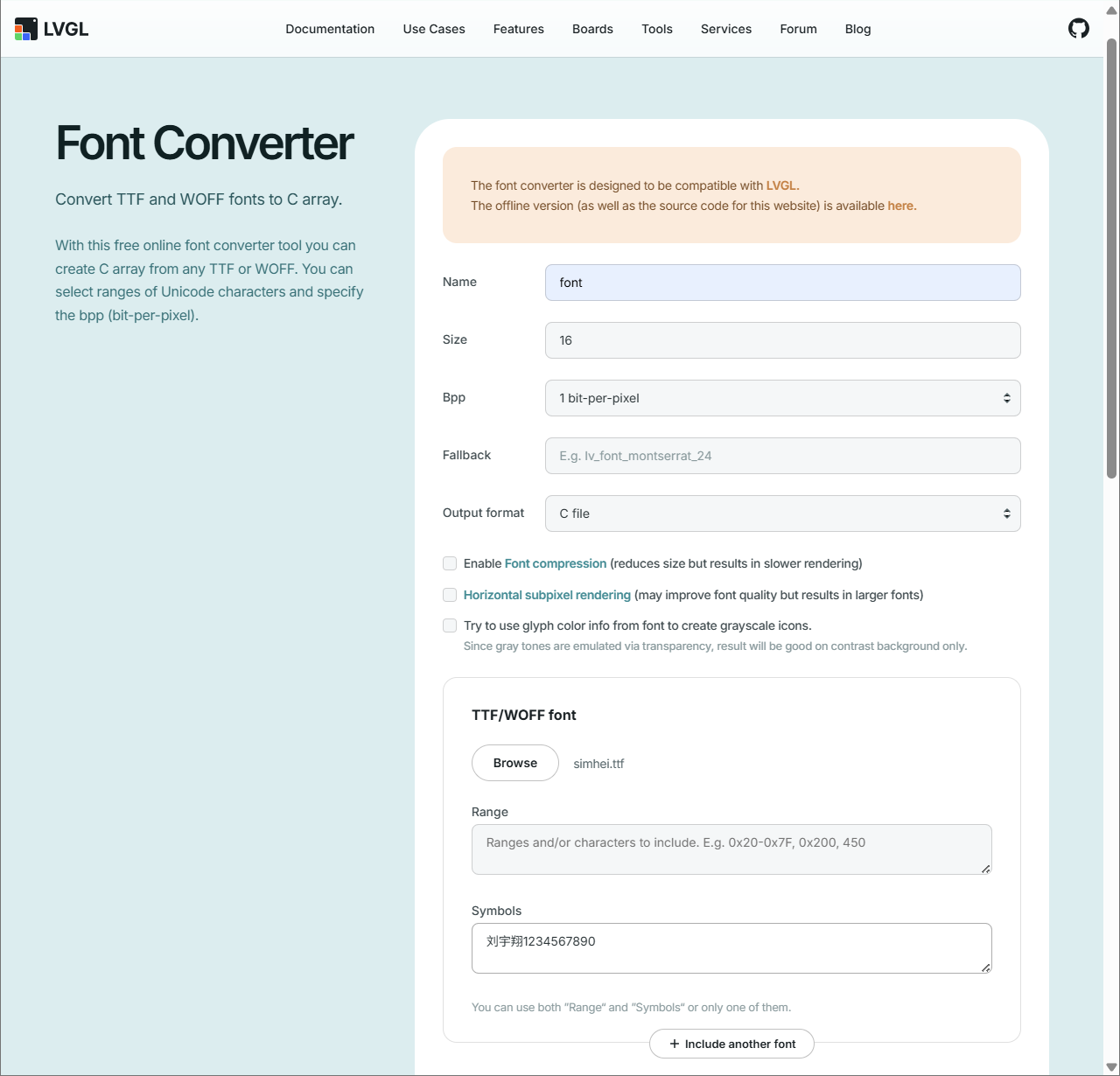
**2. GUI 初始化与资源配置**

**（1）字体和图片资源加载**

使用 LV\_FONT\_DECLARE(font) 和 LV\_IMG\_DECLARE(imglyx) 导入字体和图片资源，其中中文姓名和学号文本均依赖于该字体生成，不采用图片显示，从而满足作业要求。

我将我需要使用的证件照图片以及字体“simhei.tff”上传至LVGL Image Converter以及LVGL Font Converter生成对应的图模与字模文件，相关条件配置如下图所示：





**（2）创建界面元素**

在 lv\_user\_gui\_init 函数中，利用 LVGL 库在当前活动屏幕上创建了三个 UI 对象：

**姓名标签**：通过 lv\_label\_create 创建标签，并用 lv\_obj\_set\_style\_text\_font 设置字体，lv\_label\_set\_text 指定显示的中文姓名。初始位置设定在屏幕顶部较靠上的位置（如 (0, 10)）。

**学号标签**：类似方式创建，用于显示学号字符串，位置相对于姓名标签下移一点（例如 (0, 40)）。

**图片对象**：位置设定在屏幕中部偏下。

**（3）线程创建与启动** 在同一函数中，创建了两个线程：

**名称为“rolling”的线程，**其入口函数是 rolling\_thread\_entry，负责实现姓名跑马灯效果。该线程接收姓名标签对象作为参数。

**名称为“changing”的线程，**其入口函数是 changing\_thread\_entry，负责处理学号颜色的周期性变换。线程接收学号标签对象作为参数。 每个线程都分配了 1024 字节的堆栈、设定了优先级和时间片参数，确保系统调度时各线程能按预定时间进行处理。

**3. 姓名跑马灯效果的实现**

**（1）运动轨迹设计** 在 rolling\_thread\_entry 中，先定义了两个位置变量：

**start\_pos：**起始位置，设定为负值（例如 -60），使得标签初始时完全位于屏幕左侧之外；

**end\_pos：**结束位置，设定为一个大于屏幕宽度的值（例如 240），确保标签完全移出右侧。

**（2）动画实现方式** 采用一个无限循环来不断更新姓名标签的位置：

在每个循环内，通过一个 for 循环，每次将标签的 x 坐标增加 1 个像素，从起始位置逐步移向结束位置。

每一步之间加入延时，利用 HAL\_Delay(interval) 控制刷新间隔。延时时间由整个移动周期 5 秒分成固定步数（代码中大致分为 300 步，每步间隔大约 16～20 毫秒），保证运动平滑且总时间为 5 秒。

当标签到达结束位置后，通过 lv\_obj\_set\_x 将位置重置回 start\_pos，从而形成循环滚动的无缝动画效果。

**4. 学号颜色定时变换的实现**

**（1）颜色数组的定义**

在 changing\_thread\_entry 中，先定义了一个包含六种颜色的数组，颜色通过 lv\_color\_make 函数生成，包含黑色、红色、绿色、蓝色、黄色和紫色。

**（2）定时切换逻辑**

线程进入无限循环，在每个周期中，使用 lv\_obj\_set\_style\_text\_color 将学号标签的文本颜色设置为数组中当前索引所对应的颜色。

随后通过简单的算术运算 (color\_index + 1) % color\_count 更新颜色索引，实现循环使用数组内的颜色。

最后调用 rt\_thread\_mdelay(60000) 延时 60 秒，使得颜色更换的周期为 1 分钟。

**5. 线程调度与实时性保障**

**（1）线程优先级与时间片**

代码中创建线程时均设置了相同的优先级，这保证了两个线程能获得 CPU 时间而不会互相抢占过多资源，同时时间片参数的设置使得 RT-Thread 能合理分配执行时间。

**（2）延时函数的使用**

在滚动线程中使用 HAL\_Delay 控制动画帧率，在颜色切换线程中使用 rt\_thread\_mdelay 控制周期性任务。两种延时函数分别适用于硬件级别的延时和 RTOS 任务延时，保证了系统的实时响应和调度稳定性。

**6. 代码模块化与可维护性**

**（1）功能分离**

将姓名滚动和学号颜色变换分别放在独立线程中，有利于未来扩展和维护。每个线程只关注自己的任务，逻辑清晰，减少了代码间的耦合。

**（2）注释和日志**

在代码中添加了必要的注释，解释每个变量和逻辑的含义，便于开发人员和后续维护者快速理解程序运行流程。线程创建时也有错误日志输出，方便调试线程初始化失败的情况。

**（3）资源管理**

使用 LVGL 提供的 API 进行对象创建和样式设置，确保在图形显示方面调用的是稳定的底层接口。依赖 RT-Thread 的线程管理机制来保证多任务环境下的协作调度。

**7．软件流程图：流程阶段分解**

**A. 初始阶段**

**开始 → 系统初始化**

程序启动后首先执行系统及硬件的初始化。包括：

【1】RT-Thread内核初始化、任务调度与系统资源配置。

【2】硬件外设初始化，如LCD显示模块、GPIO、串口等（如果有需要）。

**B. GUI初始化阶段**

**调用GUI初始化函数 → lv\_user\_gui\_init()**  
进入GUI初始化阶段后，程序完成以下操作：

**创建显示对象**

***姓名标签*：**调用lv\_label\_create创建姓名显示对象，并用lv\_obj\_set\_style\_text\_font设置自定义字体；初始位置设置为屏幕上方（如 (0, 10)）。

***学号标签*：**同样方式创建学号显示对象，设置字体和初始位置（如 (0, 40)）。

***图片对象*：**调用lv\_img\_create创建图片显示对象，加载资源并设置显示位置（如 (51, 60)）。

**线程创建与启动**

创建**“滚动线程”**，其入口函数为rolling\_thread\_entry，负责控制姓名标签的跑马灯效果。

创建**“颜色切换线程”**，其入口函数为changing\_thread\_entry，负责每隔1分钟切换学号文本颜色。

**C. 姓名跑马灯线程处理（滚动线程）**

**进入滚动线程 → rolling\_thread\_entry()**  
具体流程如下：

**初始位置设置**

设置姓名标签的起始位置为屏幕左侧之外（例如 start\_pos = -60）。

定义结束位置为屏幕右侧之外（例如 end\_pos = 240）。

**运动动画循环**

采用for循环，每次将标签的x坐标增加1像素。

每次更新后调用延时函数（如HAL\_Delay(interval)，其中interval大约为20毫秒），确保整个运动周期为5秒（大致300步）。

**重置与循环**

当姓名标签移动到结束位置后，立即将位置重置回起始点，实现无缝循环滚动。

**D. 学号颜色切换线程处理（颜色切换线程）**

**进入颜色切换线程 → changing\_thread\_entry()**  
具体流程如下：

**颜色数组定义**

定义包含六种颜色（黑、红、绿、蓝、黄、紫）的数组，用于轮换显示学号文本颜色。

**颜色切换循环**

在无限循环中，首先调用lv\_obj\_set\_style\_text\_color将当前颜色应用到学号标签。然后调用延时函数rt\_thread\_mdelay(60 \* 1000)实现60秒延时。

延时结束后，更新颜色索引（通过循环索引实现数组内颜色轮换），继续下一次颜色切换。

**E. 循环机制与实时响应**

**多线程独立运行**

两个线程（滚动线程和颜色切换线程）并发运行，各自完成动画与颜色变换任务，不会互相阻塞。

系统保持实时响应，确保LCD上始终呈现平滑的跑马灯效果和定时颜色切换。

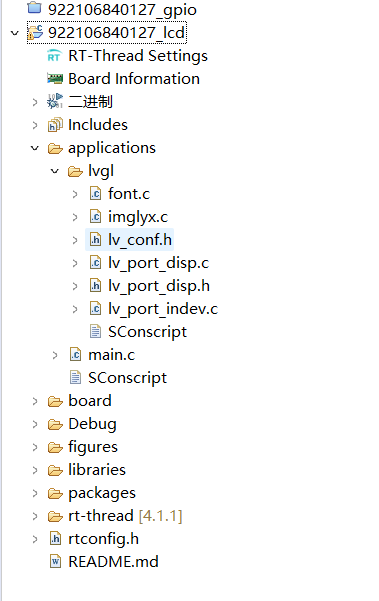
**F. 结束阶段**

**系统退出或复位**

当系统接收到关闭或复位信号时，各线程结束任务，系统资源释放，LCD显示停止更新。

**8. 具体功能设计与代码实现**

**（1）项目文件结构**

****

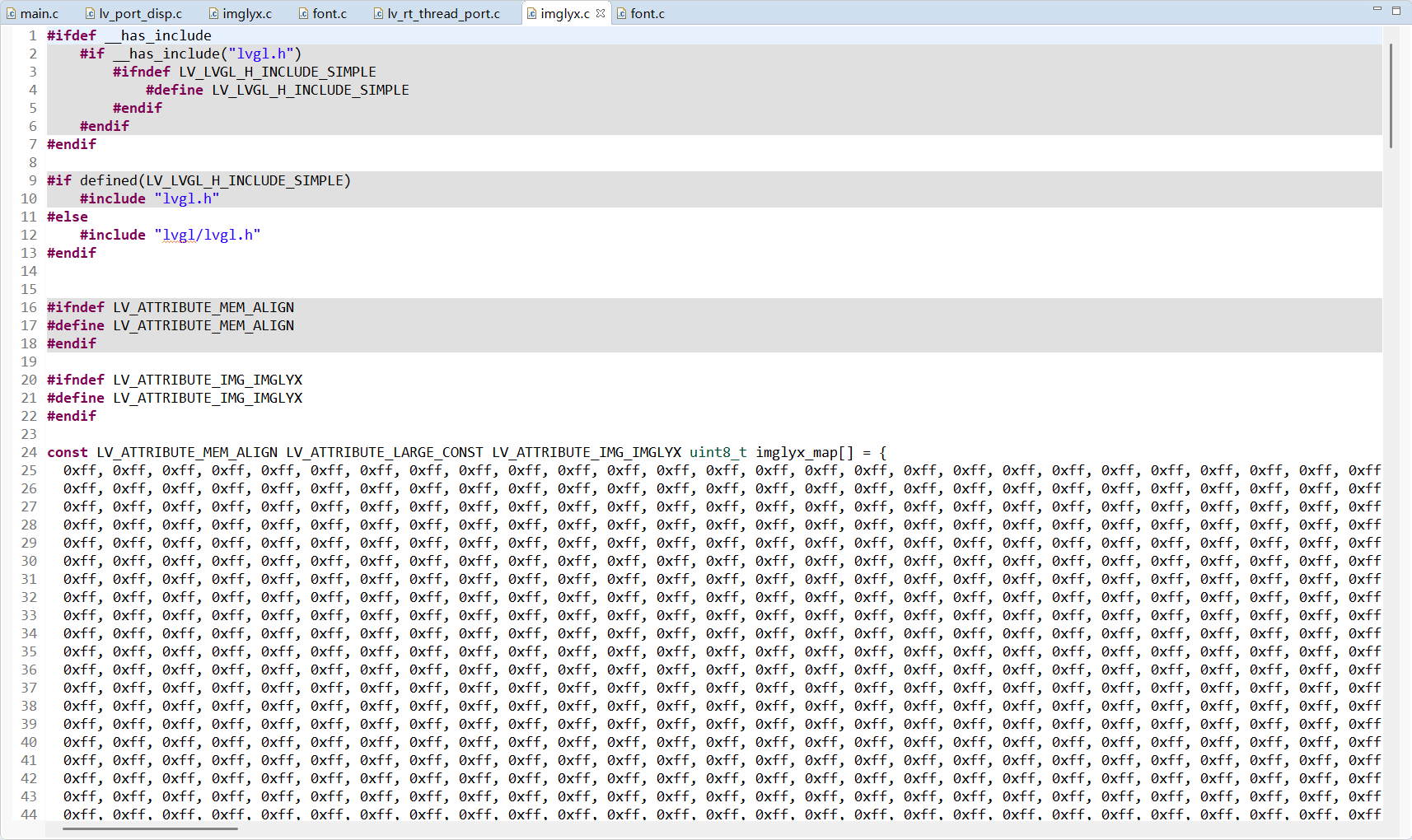
**（2）font.c以及imglyx.c**

功能：用于存放字体以及证件照导入生成的字模以及图模，由于代码内容过长故而截图一部分如下，具体就是给相关的内容进行编码用于LCD展示。

1. font.c



1. imglyx.c



**9. main.c代码实现**

/\*

\* Copyright (c) 2006-2021, RT-Thread Development Team

\*

\* SPDX-License-Identifier: Apache-2.0

\*

\* Change Logs:

\* Date Author Notes

\* 2023-5-10 ShiHao first version

\*/

#include <rtthread.h>

#include <rtdevice.h>

#include <board.h>

#define DBG\_TAG "main"

#define DBG\_LVL DBG\_LOG

#include <rtdbg.h>

#include <lvgl.h>

/\* 导入字体和图片资源 \*/

LV\_FONT\_DECLARE(font)

LV\_IMG\_DECLARE(imglyx)

/\* 线程函数声明 \*/

void rolling\_thread\_entry(void \*ptr);

void changing\_thread\_entry(void \*ptr);

/\* GUI 初始化函数 \*/

void lv\_user\_gui\_init(void)

{

/\* 创建姓名标签 \*/

lv\_obj\_t \*obj = lv\_label\_create(lv\_scr\_act());

lv\_obj\_set\_style\_text\_font(obj, &font, 0);

lv\_label\_set\_text(obj, "刘宇翔");

lv\_obj\_set\_pos(obj, 0, 10);

/\* 创建学号标签 \*/

lv\_obj\_t \*obj2 = lv\_label\_create(lv\_scr\_act());

lv\_obj\_set\_style\_text\_font(obj2, &font, 0);

lv\_label\_set\_text(obj2, "922106840127");

lv\_obj\_set\_pos(obj2, 0, 40);

/\* 创建图片 \*/

lv\_obj\_t \*obj3 = lv\_img\_create(lv\_scr\_act());

lv\_img\_set\_src(obj3, &imglyx);

lv\_obj\_set\_pos(obj3, 51, 60);

/\* 创建并启动滚动线程 \*/

rt\_thread\_t thread = rt\_thread\_create("rolling", rolling\_thread\_entry, obj, 1024, 1, 5);

if (thread != RT\_NULL)

{

rt\_thread\_startup(thread);

}

else

{

LOG\_E("Failed to create rolling thread!");

}

/\* 创建并启动颜色变换线程 \*/

rt\_thread\_t thread2 = rt\_thread\_create("changing", changing\_thread\_entry, obj2, 1024, 1, 5);

if (thread2 != RT\_NULL)

{

rt\_thread\_startup(thread2);

}

else

{

LOG\_E("Failed to create changing thread!");

}

}

/\* 姓名滚动线程 \*/

void rolling\_thread\_entry(void \*ptr)

{

lv\_obj\_t\* obj = (lv\_obj\_t\*) ptr;

int32\_t start\_pos = -60; // 起始位置（完全左出）

int32\_t end\_pos = 240; // 结束位置（完全右出）

const int interval = 5000 / 300; // 5秒内移动300步，每步间隔20ms

while (1)

{

for (int16\_t x = start\_pos; x <= end\_pos; x += 1)

{

lv\_obj\_set\_pos(obj, x, 20);

HAL\_Delay(interval);

}

lv\_obj\_set\_x(obj, start\_pos); // 重置回左侧

}

}

/\* 颜色定义 \*/

#define COLOR\_BLACK lv\_color\_make(0x00, 0x00, 0x00)

#define COLOR\_RED lv\_color\_make(0xFF, 0x00, 0x00)

#define COLOR\_GREEN lv\_color\_make(0x00, 0xFF, 0x00)

#define COLOR\_BLUE lv\_color\_make(0x00, 0x00, 0xFF)

#define COLOR\_YELLOW lv\_color\_make(0xFF, 0xFF, 0x00)

#define COLOR\_PURPLE lv\_color\_make(0x80, 0x00, 0x80)

/\* 学号颜色变换线程 \*/

void changing\_thread\_entry(void \*ptr)

{

lv\_obj\_t\* obj = (lv\_obj\_t\*) ptr;

const lv\_color\_t colors[] = { COLOR\_BLACK, COLOR\_RED, COLOR\_GREEN, COLOR\_BLUE, COLOR\_YELLOW, COLOR\_PURPLE };

uint8\_t color\_index = 0;

const uint16\_t color\_count = sizeof(colors) / sizeof(colors[0]);

while (1)

{

lv\_obj\_set\_style\_text\_color(obj, colors[color\_index], LV\_PART\_MAIN); // 设置颜色

color\_index = (color\_index + 1) % color\_count; // 更新索引

rt\_thread\_mdelay(60 \* 1000); // 60秒延时

}

}

/\* 主函数 \*/

int main(void)

{

return 0;

}

# 3 实验结果分析与总结

以下是我实验结果的过程截图，可以看到学号的颜色变色了，由于60秒的视频占据存储空间过大，从而只截取了切换字体颜色的几秒钟作为实验视频验证。



