**ncu2**

嵌入式系统I

**综合设计报告**

基于RT-Thread 的智慧图书管理系统设计

**学生姓名： 邓德聪**

**学 号： 6003122146**

**学院名称： 信息工程学院**

**专业班级： 电子信息工程221班**

**2025年 6月**

1. 设计目标分析、主要功能和主要指标

1.1 拟解决问题

传统图书管理依赖人工登记，效率低且易出错。

图书借阅、归还流程缺乏自动化，用户体验差，且容易丢失图书。

库存管理不实时，无法快速查询书籍状态。

1.2 主要功能

图书查询：通过 LCD 界面的触摸输入法输入书名，查询书籍库存状态（可借 / 已借出）。

借阅管理：通过 RFID 刷卡实现图书借出，系统自动更新库存。

归还管理：RFID 刷卡归还图书，同步更新书籍状态。

库存管理：支持新增、删除书籍信息，实时显示库存列表。

1.3 主要指标

响应速度：按键操作响应时间＜100ms，RFID 识别时间＜300ms。

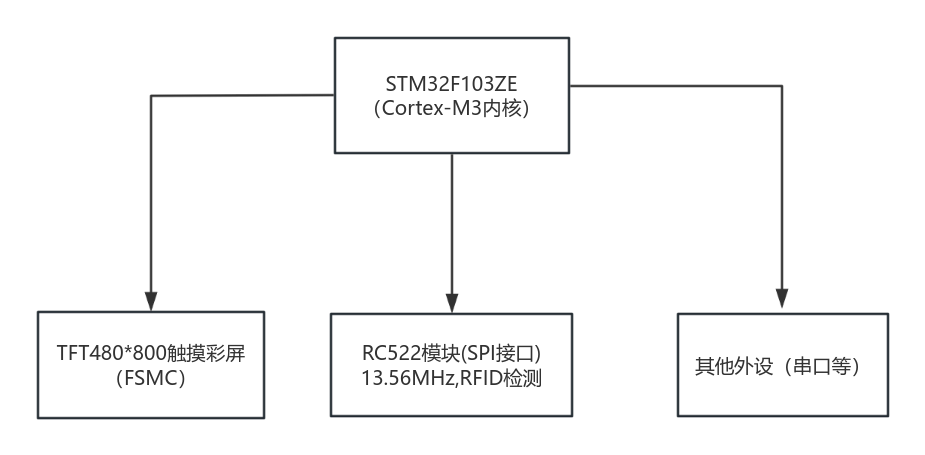
存储容量：支持至少 50 本图书信息存储（可通过外存扩展）。

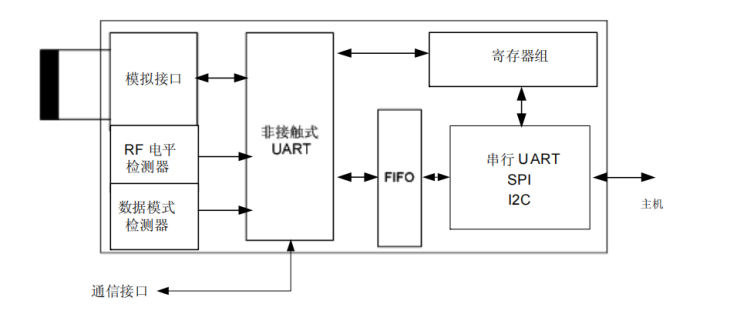
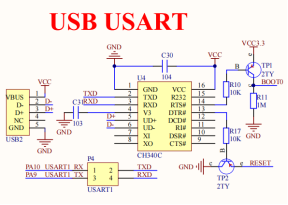
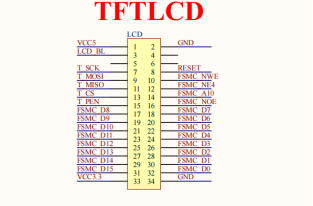
显示效果：LCD 界面分辨率 480×800，支持图片、GUI界面显示。

功耗：系统待机功耗＜100mW，工作功耗＜500mW。

2. 系统硬件模块组成

本系统采用基于STM32F103ZE的玄武开发板来完成，主要调用了TFTLCD、USART、RFID模块。





3. 基于RTT的软件模块组成

**3.1 线程规划**

|  |  |  |  |
| --- | --- | --- | --- |
| 线程名称 | 优先级 | 栈大小 | 功能描述 |
| lcd\_thread | 1 | 512B | LCD主界面显示与触摸处理 |
| find\_thread | 3 | 512B | 图书查询逻辑处理 |
| borrow\_thread | 3 | 512B | 借阅流程控制（RFID 刷卡） |
| giveback\_thread | 3 | 512B | 归还流程控制（RFID 刷卡） |
| change\_thread | 3 | 512B | 库存管理（增删书籍总量） |

**3.2 线程同步与通信**

本实验主要采用了二值信号量用在线程间的同步功能，还用了互斥量保护线程间的共同资源。

信号量（Semaphore）：

find\_sem：触发查询线程，由 LCD 线程释放，查询线程获取。

borrow\_sem：触发借阅线程，由 LCD 线程释放，借阅线程获取。

giveback\_sem:触发还书线程，由 LCD 线程释放，还书线程获取。

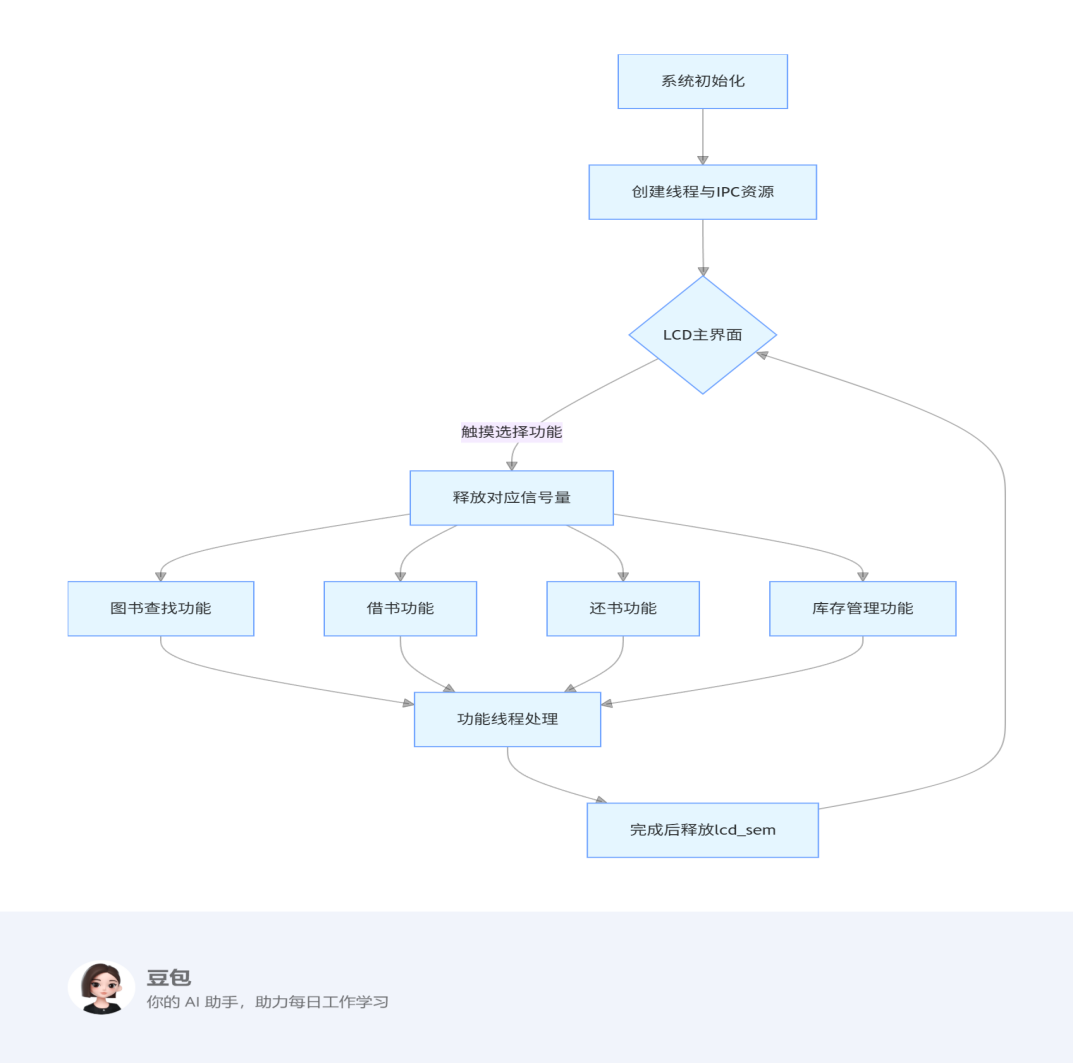
change\_sem:触发库存管理线程，由LCD线程释放，change\_thread获取。

lcd\_sem：功能线程完成后通知 LCD 线程刷新界面。

互斥量（Mutex）：

lcd\_mutex：保护 LCD 资源，避免多线程同时访问导致显示错乱。

**3.3流程图**



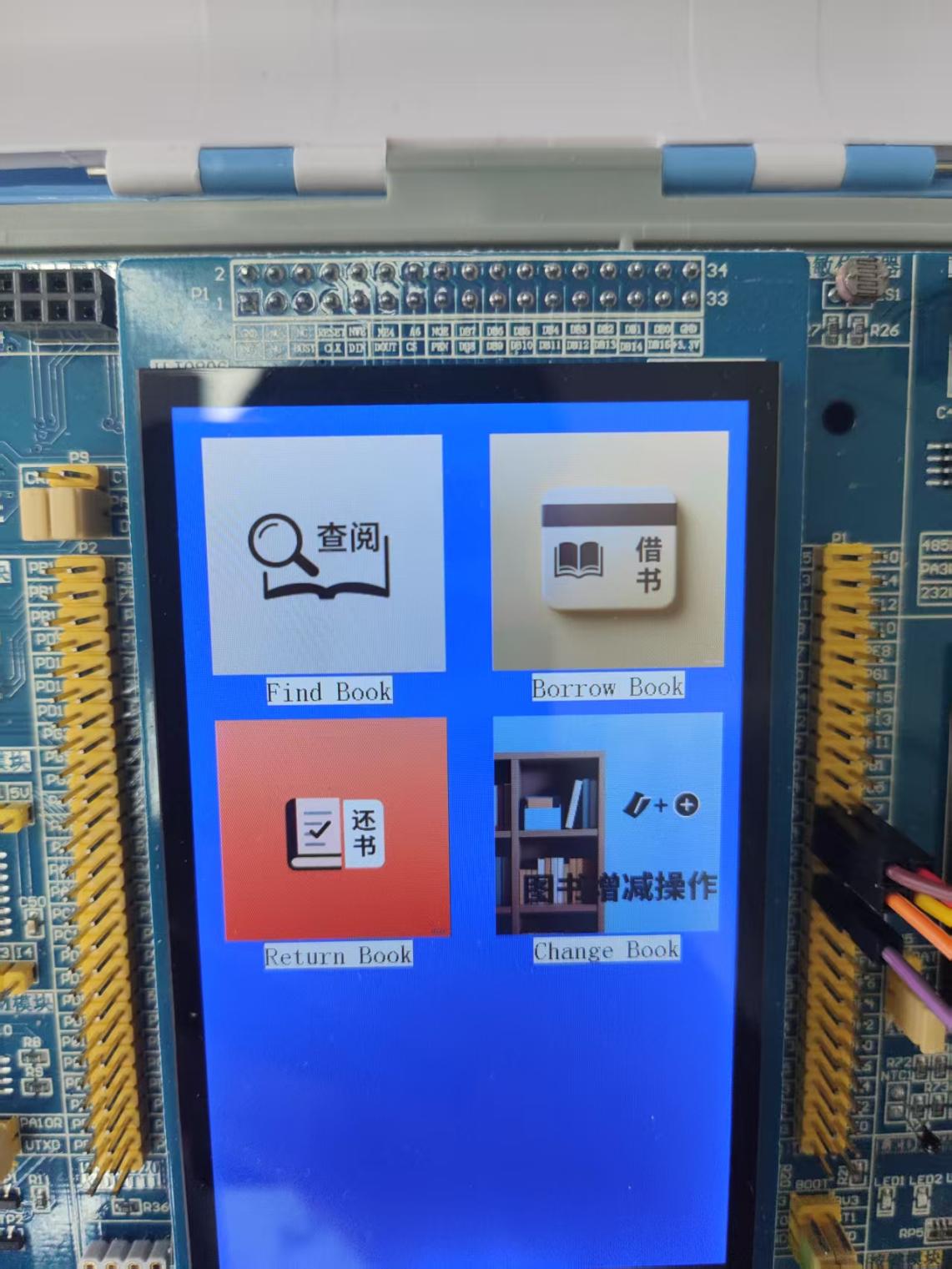
**4. 测试和分析**

**4.1 功能模块调试​**

4.1.1 LCD 界面测试​

测试用例：点击主界面 “查询”“借阅”“归还”“管理” 图标，验证界面跳转是否正常。​

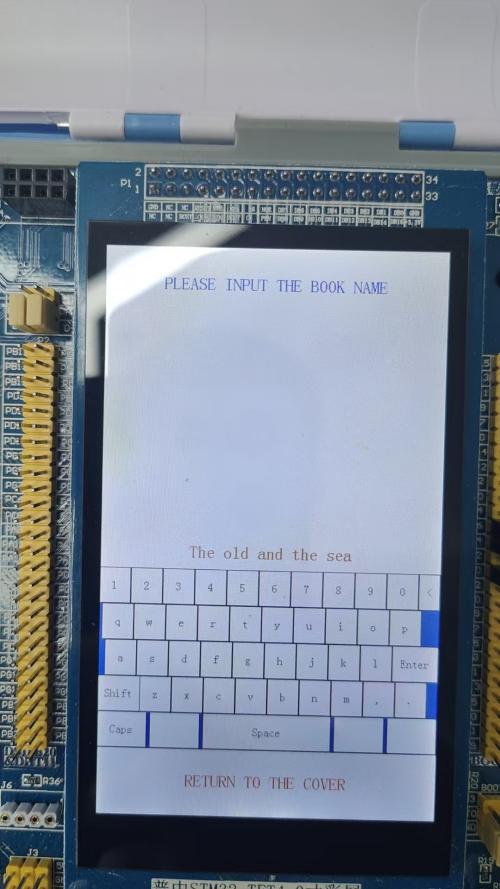
结果：图标点击响应准确，界面切换无卡顿，文字显示清晰。



4.1.2 查询功能测试

测试用例：进入查询功能，验证输入法的输入是否流畅、准确，输入法的“Shift”切换是否正常，查询结果是否准确，响应是否迅速。​

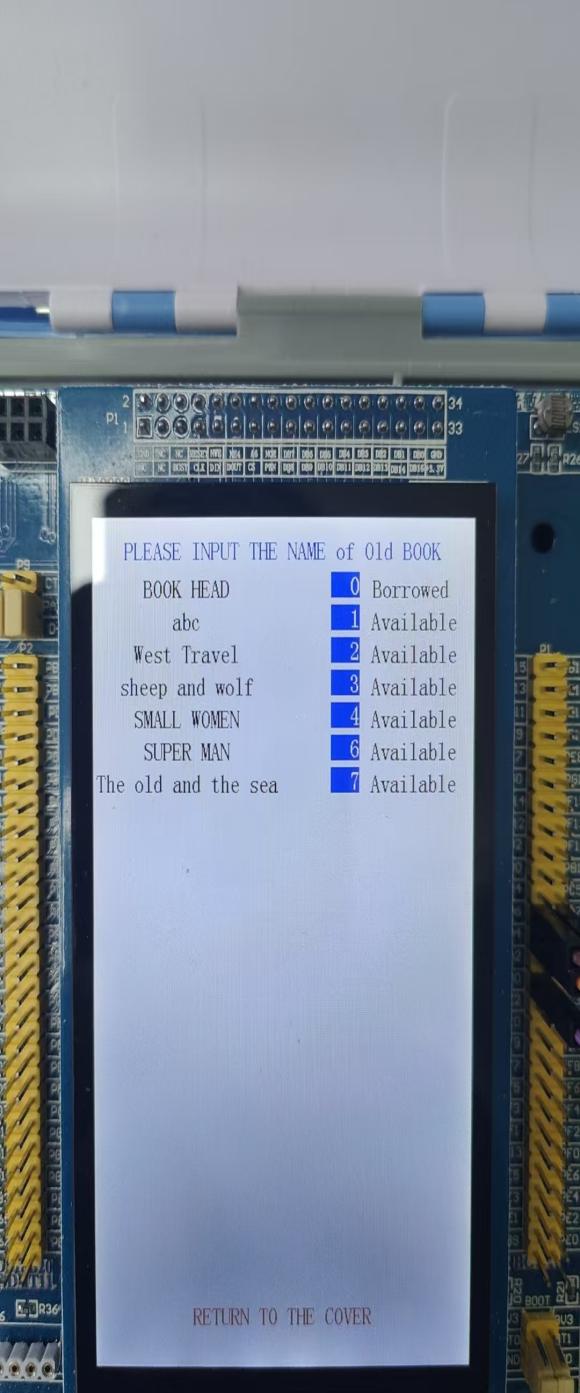
结果：界面刷新无卡顿，文字显示清晰，输入法准确流畅、切换自然，输入字符能够实时刷新，查询结果准确。

4.1.3库存增添功能测试

测试用例：通过串口输入新增书籍名称（如 “Chinese Book”），验证是否成功添加至库存，库存中新增的图书信息是否正确。同样，通过串口输入要删除的图书，验证删除后库存内容是否正确。​

结果：新增 / 删除操作响应时间＜1秒，库存列表实时刷新。下图展示的是初始的图书库、删减了English book后的图书库和增加了Chinese book的图书库。



4.1.4 RFID 模块测试​

测试用例：刷学生卡执行借阅 / 归还操作，检查系统能否正确识别卡号及卡片内容，卡片内信息读取是否正确。​

结果：RFID 识别成功率＞99%，借阅 / 归还状态更新及时。​​

**4.2 核心代码说明**​

在本次综合设计实验中，我主要负责搭建项目工程的整体框架、查询功能、库存管理功能，最后完善了项目的总体功能，以下是我负责部分的核心代码说明。

4.2.1 线程间同步

通过二值信号量的同步功能来实现，示例（LCD 线程触发查询功能）：​

// LCD线程中根据触摸索引释放信号量

switch(lcd\_index) {

case 0: rt\_sem\_release(find\_sem); break; // 查询功能

// 其他功能信号量释放...

}

// 查询线程等待信号量触发

rt\_sem\_take(find\_sem, RT\_WAITING\_FOREVER);

//查询功能.......

rt\_sem\_take(lcd\_sem, RT\_WAITING\_FOREVER);//功能完成，释放信号量

4.2.2 互斥量保护 LCD 操作​

// 进入LCD操作前获取互斥量

rt\_mutex\_take(lcd\_mutex, RT\_WAITING\_FOREVER);

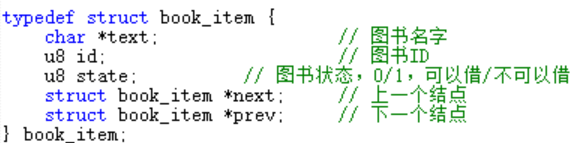
LCD\_Clear(WHITE); // 清屏

gui\_show\_strmid(...); // 显示文字

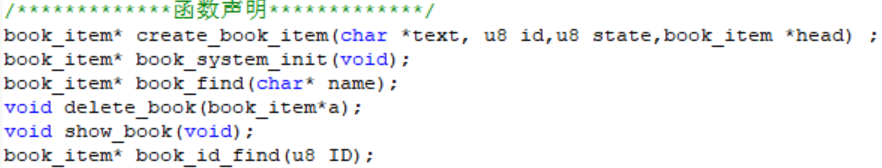
rt\_mutex\_release(lcd\_mutex); // 释放互斥量

4.2.3 对图书的各类处理逻辑实现代码

本质就是创建了一个双向链表，链表的结点是一个包含图书信息的结构体，如下：

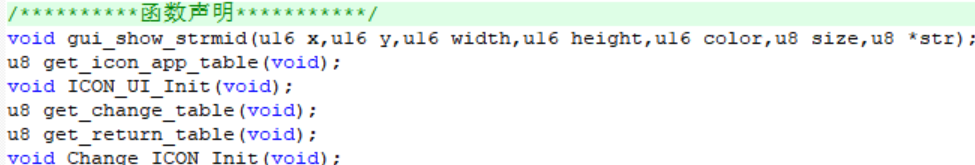


对图书进行操作处理的所有代码放在book.c文件里，具体包含如下函数，主是对图书的各类操作，函数功能正如函数名所描述的。至于函数实现的各个操作就是对链表的结点动态增减、遍历等等了，这里不多说。



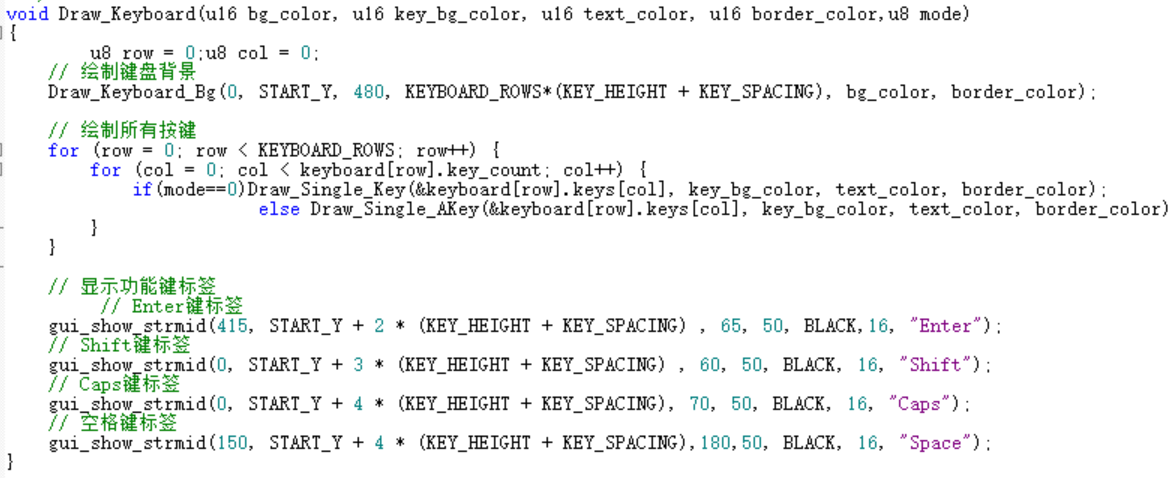
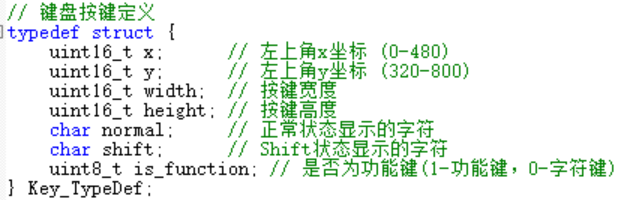
4.2.4 简易GUI界面制作

我简单的移植几个gui的代码，并在这个基础上自己写了一些要用到的函数。diy\_gui.c文件具体包含如下函数，函数功能正如函数名所描述的，主要负责GUI界面的显示和触摸功能。



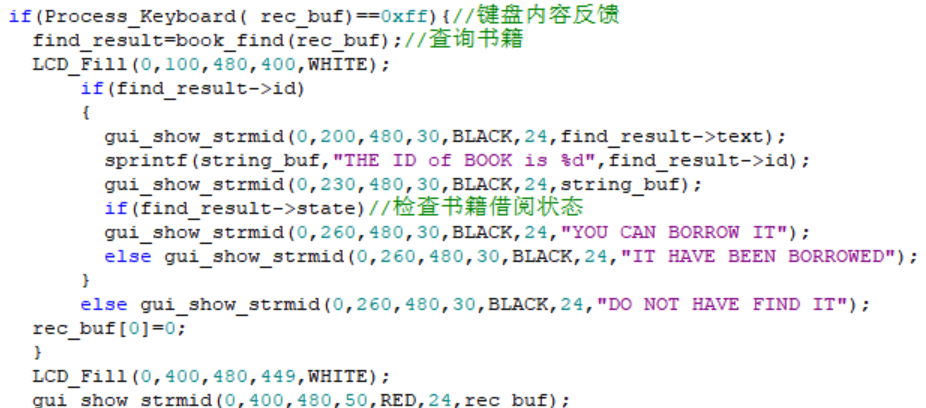
4.2.5英文键盘实现

英文键盘主要包括GUI界面的显示以及输入字符串的功能。为了灵活的显示键盘的每个按键和准确的识别触摸按键表时的字符，必须考虑每个按键的具体作用范围，这就必须避免硬编码的出现，需要学习先进的编程思想。在这部分代码中的编写过程中，真正体会到了模块化编程的重要性，减少硬编码更是需要在编程过程着重思考的。只有这样，代码才不是只适用于当前项目，而是可以移植到任何适用的项目。这里展示一个按键的结构体以及绘制键盘的函数代码，其他代码见附录。



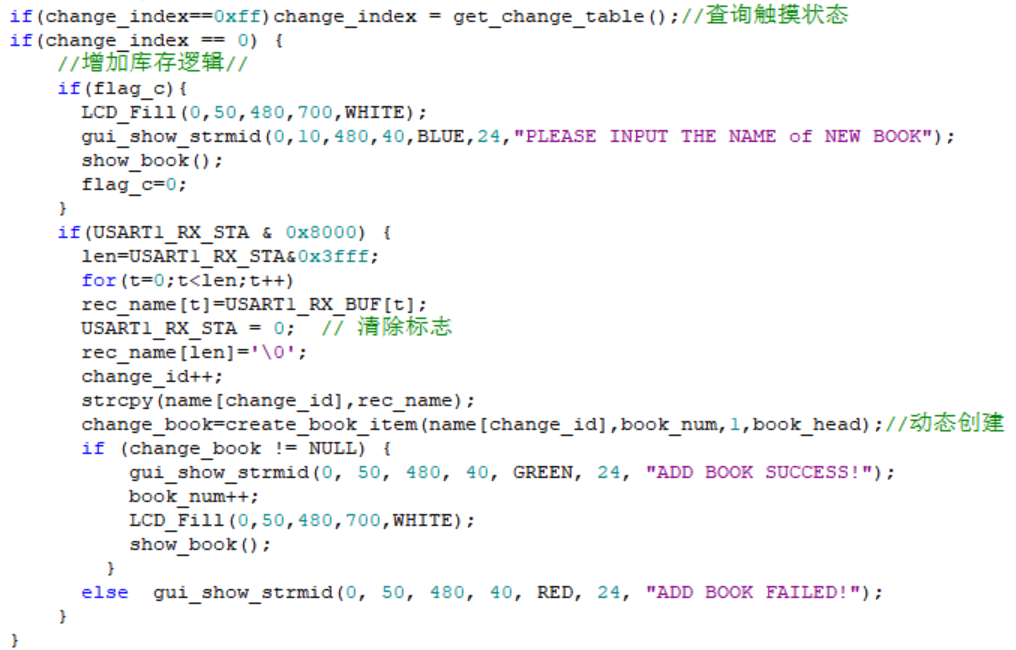
4.2.6 查询功能

通过编写的英文输入法来输入书籍的书名来查询，关于输入法的和查询的具体相关代码我放在附录里，这里展示查询线程函数中的核心代码。实质是遍历链表操作。



4.2.7 库存管理功能

直接对图书库的库存进行操作，增减图书的总数量，其实就是对双向链表进行增加或删减结点的动态操作。增加和删减操作的主要区别是一个是动态申请结点内存，另一个释放结点内，其他操作差不多，所以这里展示增加库存图书的核心代码。具体代码见附录



5. 附录

1. 核心代码

**main.c部分：**

查询功能线程**：**

/\*\*\*\*\*\* 查询功能线程 \*\*\*\*\*\*\*\*\*/

static void find\_thread\_entry(void\* parameter)

{

static char name[30];

static book\_item\* find\_result=NULL;

static u8 len=0,t;

static char string\_buf[30];

static char rec\_buf[30];

while(1) {

rt\_sem\_take(find\_sem, RT\_WAITING\_FOREVER);

rt\_mutex\_take(lcd\_mutex, RT\_WAITING\_FOREVER);

LCD\_Clear(WHITE);

gui\_show\_strmid(0,30,480,40,BLUE,24,"PLEASE INPUT THE BOOK NAME");

gui\_show\_strmid(0,700,480,100,RED,24,"RETURN TO THE COVER");

Show\_Default\_Keyboard(0);//绘画键盘

rt\_mutex\_release(lcd\_mutex);

while(1) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ if(Process\_Keyboard( rec\_buf)==0xff){//键盘内容反馈

find\_result=book\_find(rec\_buf);//查询书籍

LCD\_Fill(0,100,480,400,WHITE);

if(find\_result->id)

{

gui\_show\_strmid(0,200,480,30,BLACK,24,find\_result->text);

sprintf(string\_buf,"THE ID of BOOK is %d",find\_result->id);

gui\_show\_strmid(0,230,480,30,BLACK,24,string\_buf);

if(find\_result->state)//检查书籍借阅状态

gui\_show\_strmid(0,260,480,30,BLACK,24,"YOU CAN BORROW IT");

else gui\_show\_strmid(0,260,480,30,BLACK,24,"IT HAVE BEEN BORROWED");

}

else gui\_show\_strmid(0,260,480,30,BLACK,24,"DO NOT HAVE FIND IT");

rec\_buf[0]=0;

}

LCD\_Fill(0,400,480,449,WHITE);

gui\_show\_strmid(0,400,480,50,RED,24,rec\_buf);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(get\_return\_table()) {

rt\_sem\_release(lcd\_sem);

break;

}

rt\_thread\_delay(20);

}

}

}

**库存管理线程：**

/\*\*\*\*\*\*\* 库存管理线程 \*\*\*\*\*\*\*\*\*\*\*\*/

static void change\_thread\_entry(void\* parameter)

{

u8 change\_index = 0xff;

static u8 change\_id,len,t,i,flag\_c;

static char rec\_name[30];

book\_item\* change\_book= book\_head;

static char string\_buf[30];

change\_id=(u8)book\_num;

while(1) {

rt\_sem\_take(change\_sem, RT\_WAITING\_FOREVER);

rt\_mutex\_take(lcd\_mutex, RT\_WAITING\_FOREVER);

Change\_ICON\_Init();//界面初始化

rt\_mutex\_release(lcd\_mutex);

change\_index=0xff;flag\_c=1;

while(1) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*库存管理逻辑\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(change\_index==0xff)change\_index = get\_change\_table();//查询触摸状态

if(change\_index == 0) {

//增加库存逻辑//

if(flag\_c){

LCD\_Fill(0,50,480,700,WHITE);

gui\_show\_strmid(0,10,480,40,BLUE,24,"PLEASE INPUT THE NAME of NEW BOOK");

show\_book();

flag\_c=0;

}

if(USART1\_RX\_STA & 0x8000) {

len=USART1\_RX\_STA&0x3fff;

for(t=0;t<len;t++)

rec\_name[t]=USART1\_RX\_BUF[t];

USART1\_RX\_STA = 0; // 清除标志

rec\_name[len]='\0';

change\_id++;

strcpy(name[change\_id],rec\_name);

change\_book=create\_book\_item(name[change\_id],book\_num,1,book\_head);//动态创建

if (change\_book != NULL) {

gui\_show\_strmid(0, 50, 480, 40, GREEN, 24, "ADD BOOK SUCCESS!");

book\_num++;

LCD\_Fill(0,50,480,700,WHITE);

show\_book();

}

else gui\_show\_strmid(0, 50, 480, 40, RED, 24, "ADD BOOK FAILED!"); }

}

else if(change\_index == 1) {

// 减少库存逻辑//

if(flag\_c){

LCD\_Fill(0,50,480,700,WHITE);

gui\_show\_strmid(0,10,480,40,BLUE,24,"PLEASE INPUT THE NAME of Old BOOK");

show\_book();//展示图书库存

flag\_c=0;

}

if(USART1\_RX\_STA & 0x8000) {

len=USART1\_RX\_STA&0x3fff;

for(t=0;t<len;t++)

rec\_name[t]=USART1\_RX\_BUF[t];

USART1\_RX\_STA = 0; // 清除标志

rec\_name[len]='\0';

change\_book=book\_find(rec\_name);

if(change\_book->id!=0){

delete\_book(change\_book) ;

gui\_show\_strmid(0,50,480,40,GREEN,24,"DELETE BOOK SUCCEE!");

book\_num--;

LCD\_Fill(0,50,480,700,WHITE);

show\_book();

}

else gui\_show\_strmid(0,50,480,40,RED,24,"DELETE BOOK FAIL");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(get\_return\_table()) {

rt\_sem\_release(lcd\_sem);

break;

}

rt\_thread\_delay(20);

}

}

}

**book.c:（部分）**

//加书

book\_item\* create\_book\_item(char \*text, u8 id,u8 state,book\_item \*head)

{

book\_item \*last = head->prev;

book\_item \*item = (book\_item\*)malloc(sizeof(book\_item));

if(item==NULL)return item;

item->text = text;

item->id = id;

item->state=state;

item->next = item->prev = item;//链表初始化

last->next = item;

item->prev = last;

item->next =head;

head->prev = item;//这里是向前插入,方便遍历

return item;

}

//减书

void delete\_book(book\_item\*a)

{

if(a->id==0)return;

a->prev->next=a->next;

a->next->prev=a->prev;

a->next=a;

a->prev=a;

free(a);

}

//初始化

book\_item\* book\_system\_init(void)

{

u8 i=0;

//建个头节点

book\_head = (book\_item\*)malloc(sizeof(book\_item));

book\_head->text = "BOOK HEAD";

book\_head->id = 0;

book\_head->state=0;

book\_head->next = book\_head->prev = book\_head;//链表初始化

for(i=1;i<=book\_num;i++){

book\_stock[i-1]=create\_book\_item(name[i],i,1,book\_head) ;

}

return book\_head;

}

//找书（书名）

book\_item\* book\_find(char\* name)

{

book\_item\* a=book\_head;

while(1)

{

a=a->next;

if((!strcmp(a->text,name))||a->id==0)

break;

}

return a;

}

//找书（书号）

book\_item\* book\_id\_find(u8 ID)

{

book\_item\* a=book\_head;

while(1)

{

a=a->next;

if(ID==a->id||a->id==0)

break;

}

return a;

}

//展示图书库存

void show\_book(void)

{

u32 i;

book\_item\* a=book\_head;

for(i=0;;i++){

gui\_show\_strmid(0,50+i\*30,240,30,BLACK,24,a->text);

LCD\_ShowNum(300,50+i\*30,a->id,3,24);

if(a->state)gui\_show\_strmid(350,50+i\*30,100,30,BLACK,24,"Available");

else gui\_show\_strmid(350,50+i\*30,100,30,BLACK,24,"Borrowed");

a=a->next;

if(a->id==0)break;

}

}

**diy\_gui.c:（部分）**

//判断哪个应用APP图标被按下，并返回对应索引

u8 get\_icon\_app\_table(void)

{

u8 index=0xff;

static u16 curxpos=0;//当前tp按下的x坐标

static u16 curypos=0;//当前tp按下的y坐标

static u8 curtpsta=0;//触摸按下标志

u8 i=0,j=0;

tp\_dev.scan(0); //扫描

if(tp\_dev.sta&TP\_PRES\_DOWN)//有按键被按下

{

curxpos=tp\_dev.x[0]; //记录当前坐标

curypos=tp\_dev.y[0]; //记录当前坐标

curtpsta=1;

}

else //按键松开了

{

if(curtpsta)//之前有按下

{

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

if(curxpos>(PIC\_ICON\_APP\_XSTART+(PIC\_ICON\_APP\_XSPACE+PIC\_ICON\_APP\_WIDTH)\*j)

&& curxpos<(PIC\_ICON\_APP\_XSTART+(PIC\_ICON\_APP\_XSPACE+PIC\_ICON\_APP\_WIDTH)\*j+PIC\_ICON\_APP\_WIDTH)

&& curypos>(PIC\_ICON\_APP\_YSTART+(PIC\_ICON\_APP\_YSPACE+PIC\_ICON\_APP\_HEIGHT)\*i)

&& curypos<(PIC\_ICON\_APP\_YSTART+(PIC\_ICON\_APP\_YSPACE+PIC\_ICON\_APP\_HEIGHT)\*i+PIC\_ICON\_APP\_HEIGHT))

{

index=j+2\*i;//得到选中的编号

break;

}

}

}

}

curtpsta=0;//清空标志

}

return index;

}

//

void Change\_ICON\_Init(void)

{

LCD\_Clear(WHITE);

gui\_show\_strmid(change\_xstart,change\_ystart,change\_xwidth,change\_yheight,GREEN,24,"ADD SOME BOOK TO THE LIBRARY");

gui\_show\_strmid(change\_xstart,change\_ystart+change\_yheight,change\_xwidth,change\_yheight,GREEN,24,"DELETE SOME BOOK FROM LIBARY");

gui\_show\_strmid(0,700,480,100,RED,24,"RETURN TO THE COVER");

}

u8 get\_change\_table(void)

{

u8 index=0xff;

static u16 curxpos=0;//当前tp按下的x坐标

static u16 curypos=0;//当前tp按下的y坐标

static u8 curtpsta=0;//触摸按下标志

u8 i=0,j=0;

tp\_dev.scan(0); //扫描

if(tp\_dev.sta&TP\_PRES\_DOWN)//有按键被按下

{

curxpos=tp\_dev.x[0]; //记录当前坐标

curypos=tp\_dev.y[0]; //记录当前坐标

curtpsta=1;

}

else //按键松开了

{

if(curtpsta)//之前有按下

{

for(i=0;i<2;i++){

if(curypos>change\_ystart+i\*change\_yheight&&curypos<change\_ystart+(i+1)\*change\_yheight)index=i;

}

}

curtpsta=0;//清空标志

}

return index;

}

u8 get\_return\_table(void)

{

u8 index=0;

static u16 curxpos=0;//当前tp按下的x坐标

static u16 curypos=0;//当前tp按下的y坐标

static u8 curtpsta=0;//触摸按下标志

u8 i=0,j=0;

tp\_dev.scan(0); //扫描

if(tp\_dev.sta&TP\_PRES\_DOWN)//有按键被按下

{

curxpos=tp\_dev.x[0]; //记录当前坐标

curypos=tp\_dev.y[0]; //记录当前坐标

curtpsta=1;

}

else //按键松开了

{

if(curtpsta)//之前有按下

{

if(curypos>700)index=1;

}

curtpsta=0;//清空标志

}

return index;

}

**Keyboard.c:(部分)**

//按键结果处理函数

u8 Process\_Keyboard(char\* buf)

{

static u8 shift\_state = 0;

static u8 caps\_state = 0;

char key=0;

static char i;

key = Detect\_Keyboard\_Touch(shift\_state ^ caps\_state);

switch(key) {

case 0:return 0;break;

case FUNC\_SHIFT:

shift\_state ^= 1; // 切换Shift状态

Show\_Default\_Keyboard(shift\_state ^ caps\_state); // 重绘键盘

break;

case FUNC\_CAPS:

caps\_state ^= 1; // 切换Caps状态

Show\_Default\_Keyboard(shift\_state ^ caps\_state); // 重绘键盘

break;

case FUNC\_ENTER:

buf[i]='\0';//字符串输入完了，这里不用作换行

i=0;

return 0xff;

break;

case FUNC\_BACKSPACE:

if(i)buf[--i]='\0';

break;

case FUNC\_SPACE:

buf[i++]=' ';

buf[i]=0;

break;

default:

if(key > 0) {

// 处理普通字符输入

buf[i++]=key;

buf[i]=0;

return (u8)buf[i-1];

}

}

}

2. 作品图片

