目 录

1. 实验4 点阵字符显示和用户操作界面设计 1

1.1 实验目的 1

1.2 实验主要器材和设备 1

1.3 实验任务技术解决方案的简要说明 1

1.3.1 实验任务4\_1 1

1.3.2 实验任务4\_2 1

1.3.3 实验任务4\_3 3

1.4 实验核心代码清单 4

1.4.1 实验任务4\_1 4

1.4.2 实验任务4\_2 8

1.4.3 实验任务4\_3 16

# 实验4 点阵字符显示和用户操作界面设计

## 实验目的

学习如何查阅厂商技术资料，掌握点阵液晶显示屏电路模块的应用开发方法；

学会基于有限状态机(FSM，Finite State Machine)逻辑设计用户操作界面(UI，User Interface)功能的一般方法；

学习借助专业测量工具仪器评估实验作品技术性能的实验技巧；

培养在技术团队中以专业态度扮演个人角色和履行分工职责的能力。

## 实验主要器材和设备

电脑；TM4C1294NCPDT实验板卡；A2000TM4扩展板；

调频发射实验单元板或调频接收实验单元板（两种单元板均带有JLX12864G-086-PC型液晶显示屏电路模块）；

台式稳压电源或USB接口供电设备（输出电压DC5V，输出电流>0.5A）；数字示波器；多用电表。

## 实验任务技术解决方案的简要说明

### 实验任务4\_1

图1是关于本任务的程序流程示意图。

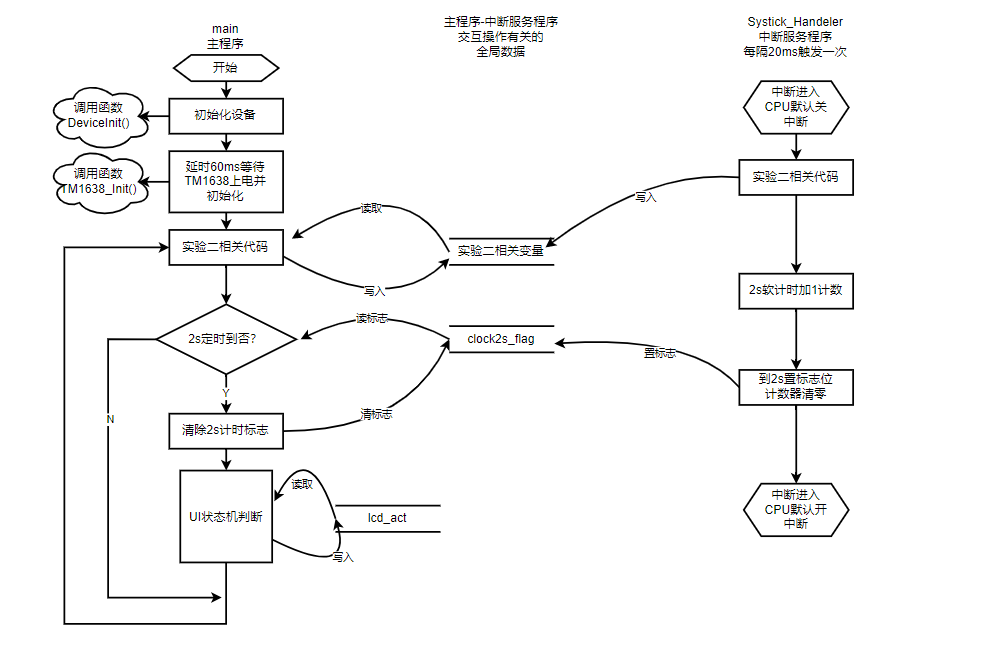


图1 关于实验任务4\_1的程序流程示意图

### 实验任务4\_2

关于状态ACT005处理的源代码如下。

|  |  |
| --- | --- |
| 1 | /\*\* |
| 2 | \* @brief UI状态机ACT005状态处理 |
| 3 | \* 光标在工作参数十分位的位置 |
| 4 | \* |
| 5 | \*/ |
| 6 | void ui\_proc005(void) |
| 7 | { |
| 8 | // 当"右"键按下：光标移到"模式#"，下一状态ACT001 |
| 9 | if (key\_RIGHT\_flag) |
| 10 | { |
| 11 | key\_RIGHT\_flag = 0; |
| 12 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 13 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 1); |
| 14 | ui\_state = 0x001; |
| 15 | } |
| 16 | // 当"左"键按下：光标移到个位，下一状态ACT003 |
| 17 | else if (key\_LEFT\_flag) |
| 18 | { |
| 19 | key\_LEFT\_flag = 0; |
| 20 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 21 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 1); |
| 22 | ui\_state = 0x003; |
| 23 | } |
| 24 | // 当"+"键按下：十分位数按1、2、...、9、0、1、...正序循环切换，留在本状态 |
| 25 | else if (key\_INCREASE\_flag) |
| 26 | { |
| 27 | key\_INCREASE\_flag = 0; |
| 28 | if (++((act[0]->str[5])[0]) > '9') |
| 29 | { |
| 30 | (act[0]->str[5])[0] = '0'; |
| 31 | } |
| 32 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 33 | } |
| 34 | // 当"-"键按下：十分位数按9、8、...、0、9、8、...逆序循环切换，留在本状态 |
| 35 | else if (key\_DECREASE\_flag) |
| 36 | { |
| 37 | key\_DECREASE\_flag = 0; |
| 38 | if (--((act[0]->str[5])[0]) < '0') |
| 39 | { |
| 40 | (act[0]->str[5])[0] = '9'; |
| 41 | } |
| 42 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 43 | } |
| 44 | else if (key\_ENTER\_flag) |
| 45 | { |
| 46 | key\_ENTER\_flag = 0; |
| 47 | } |
| 48 |  |
| 49 | // 当10秒无操作：工作参数十分位反白效果解除，下一状态ACT0 |
| 50 | if (NOKEY\_clock10s\_flag) |
| 51 | { |
| 52 | NOKEY\_clock10s\_flag = 0; |
| 53 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 54 | ui\_state = 0x0; |
| 55 | } |
| 56 | } |
| 57 |  |

### 实验任务4\_3

关于本任务的功能逻辑细化分析示于表1。

表1 实验任务4\_3各状态的功能逻辑细化分析

| **序号** | **状态代号** | **画面描述** | **状态转移 触发条件** | **状态转移前 完成相应操作** | **下一状态** |
| --- | --- | --- | --- | --- | --- |
| 1 | ACT0 | 开机初始画面，不显示光标 | 任意按键 | “设置”做反白效果（光标） | ACT005 |
| 2 | ACT005 | 光标在“设置”的位置 | “确定”键操作 | 光标移到ACT1的“工作参数” | ACT100 |
| 5秒无键操作 | “设置”反白效果解除 | ACT0 |
| 3 | ACT100 | 光标在“工作模式”的位置 | “←”键操作 | 光标移到“返回” | ACT102 |
| “→”键操作 | 光标移到“工作参数” | ACT101 |
| “确定”键操作 | 光标移到ACT2的“模式#” | ACT201 |
| 4 | ACT101 | 光标在“工作参数”的位置 | “←”键操作 | 光标移到“工作模式” | ACT100 |
| “→”键操作 | 光标移到“返回” | ACT102 |
| “确定”键操作 | 光标移到ACT3的工作参数的个位 | ACT301 |
| 5 | ACT102 | 光标在“返回”的位置 | “←”键操作 | 光标移到“工作参数” | ACT101 |
| “→”键操作 | 光标移到“工作模式” | ACT100 |
| “确定”键操作 | 显示开机初始画面 | ACT0 |
| 6 | ACT201 | 光标在“模式#”的位置 | “←”键操作 | 光标移到“取消” | ACT203 |
| “→”键操作 | 光标移到“确定” | ACT202 |
| “+”键操作 | “模式#”中的字母按A→B→C→A正序循环切换 | ACT201（留在本状态） |
| “-”键操作 | “模式#”中的字母按C→B→A→C逆序循环切换 | ACT201（留在本状态） |
| 7 | ACT202 | 光标在“确定”的位置 | “←”键操作 | 光标移到“模式#” | ACT201 |
| “→”键操作 | 光标移到“取消” | ACT203 |
| “确定”键操作 | 保存更改；光标移到ACT1的“工作模式” | ACT100 |
| 8 | ACT203 | 光标在“取消”的位置 | “←”键操作 | 光标移到“确定” | ACT202 |
| “→”键操作 | 光标移到“模式#” | ACT201 |
| “确定”键操作 | 撤销更改；光标移到ACT1的“工作模式” | ACT100 |
| 9 | ACT301 | 光标在工作参数个位的位置 | “←”键操作 | 光标移到“取消” | ACT306 |
| “→”键操作 | 光标移到工作参数的十分位 | ACT303 |
| “+”键操作 | 个位数按1→2→…→9→0→1正序循环切换 | ACT301 （留在本状态） |
| “-”键操作 | 个位数按9→8→…→0→9逆序循环切换 | ACT301 （留在本状态） |

续表1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **状态代号** | **画面描述** | **状态转移 触发条件** | **状态转移前 完成相应操作** | **下一状态** |
| 10 | ACT303 | 光标在工作参数十分位的位置 | “←”键操作 | 光标移到工作参数的个位 | ACT301 |
| “→”键操作 | 光标移到“确定” | ACT305 |
| “+”键操作 | 十分位数按1→2→…→9→0→1正序循环切换 | ACT303 （留在本状态） |
| “-”键操作 | 十分位数按9→8→…→0→9逆序循环切换 | ACT303 （留在本状态） |
| 11 | ACT305 | 光标在“确定”的位置 | “←”键操作 | 光标移到工作参数的十分位 | ACT303 |
| “→”键操作 | 光标移到“取消” | ACT306 |
| “确定”键操作，且参数值合法 | 保存更改；光标移到ACT1的“工作参数” | ACT101 |
| “确定”键操作，且参数值非法 | 显示警示画面ACT4 | ACT4 |
| 12 | ACT306 | 光标在“取消”的位置 | “←”键操作 | 光标移到“确定” | ACT305 |
| “→”键操作 | 光标移到工作参数的个位 | ACT301 |
| “确定”键操作 | 光标移到ACT1的“工作参数” | ACT101 |
| 13 | ACT4 | 显示警示画面 | 5秒计时到 | 光标移到ACT3的工作参数的个位 | ACT301 |

## 实验核心代码清单

### 实验任务4\_1

|  |  |
| --- | --- |
| 1 | /\*\* |
| 2 | \* @file exp4\_1.c |
| 3 | \* @author 上海交通大学电子工程系实验教学中心; Guorui Wei (313017602@qq.com) |
| 4 | \* @brief 实验4\_1 |
| 5 | \* @version 0.1 |
| 6 | \* @date 2021-06-05 |
| 7 | \* |
| 8 | \* @copyright 2020-2021, 上海交通大学电子工程系实验教学中心 |
| 9 | \* |
| 10 | \*/ |
| 11 |  |
| 12 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 13 | // |
| 14 | // 头文件 |
| 15 | // |
| 16 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 17 | #include <stdint.h> |
| 18 | #include <stdbool.h> |
| 19 | #include "inc/hw\_memmap.h"       // 基址宏定义 |
| 20 | #include "inc/hw\_types.h"        // 数据类型宏定义，寄存器访问函数 |
| 21 | #include "driverlib/debug.h"     // 调试用 |
| 22 | #include "driverlib/gpio.h"      // 通用IO口宏定义 |
| 23 | #include "driverlib/pin\_map.h"   // TM4C系列MCU外围设备管脚宏定义 |
| 24 | #include "driverlib/sysctl.h"    // 系统控制定义 |
| 25 | #include "driverlib/systick.h"   // SysTick Driver 原型 |
| 26 | #include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型 |
| 27 | #include "JLX12864.h"            // 与控制JLX12864G有关的函数 |
| 28 | #include "tm1638.h"              // 与控制TM1638芯片有关的函数 |
| 29 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 30 | // |
| 31 | // 宏定义 |
| 32 | // |
| 33 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 34 | #define SYSTICK\_FREQUENCY 50 // SysTick频率为50Hz，即循环定时周期20ms |
| 35 |  |
| 36 | #define V\_T100ms 5  // 0.1s软件定时器溢出值，5个20ms |
| 37 | #define V\_T500ms 25 // 0.5s软件定时器溢出值，25个20ms |
| 38 | #define V\_T2s 100   // 2.0s软定时器溢出值，100个20ms |
| 39 |  |
| 40 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 41 | // |
| 42 | // 函数原型声明 |
| 43 | // |
| 44 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 45 | void GPIOInit(void);    // GPIO初始化 |
| 46 | void SysTickInit(void); // 设置SysTick中断 |
| 47 | void DevicesInit(void); // MCU器件初始化，注：会调用上述函数 |
| 48 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 49 | // |
| 50 | // 变量定义 |
| 51 | // |
| 52 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 53 |  |
| 54 | // 软件定时器计数 |
| 55 | uint8\_t clock100ms = 0; |
| 56 | uint8\_t clock500ms = 0; |
| 57 | uint8\_t clock2s = 0; |
| 58 |  |
| 59 | // 软件定时器溢出标志 |
| 60 | uint8\_t clock100ms\_flag = 0; |
| 61 | uint8\_t clock500ms\_flag = 0; |
| 62 | uint8\_t clock2s\_flag = 0; |
| 63 |  |
| 64 | // 测试用计数器 |
| 65 | uint32\_t test\_counter = 0; |
| 66 |  |
| 67 | // 8位数码管显示的数字或字母符号 |
| 68 | // 注：板上数码位从左到右序号排列为4、5、6、7、0、1、2、3 |
| 69 | uint8\_t digit[8] = {' ', ' ', ' ', ' ', '\_', ' ', '\_', ' '}; |
| 70 |  |
| 71 | // 8位小数点 1亮  0灭 |
| 72 | // 注：板上数码位小数点从左到右序号排列为4、5、6、7、0、1、2、3 |
| 73 | uint8\_t pnt = 0x04; |
| 74 |  |
| 75 | // 8个LED指示灯状态，0灭，1亮 |
| 76 | // 注：板上指示灯从左到右序号排列为7、6、5、4、3、2、1、0 |
| 77 | //     对应元件LED8、LED7、LED6、LED5、LED4、LED3、LED2、LED1 |
| 78 | uint8\_t led[] = {1, 1, 1, 1, 1, 1, 1, 0}; |
| 79 |  |
| 80 | // 当前按键值 |
| 81 | uint8\_t key\_code = 0; |
| 82 |  |
| 83 | // 系统时钟频率 |
| 84 | uint32\_t ui32SysClock; |
| 85 |  |
| 86 | uint8\_t lcd\_act = 0; // LCD屏幕状态机当前状态 |
| 87 |  |
| 88 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 89 | // |
| 90 | // 主程序 |
| 91 | // |
| 92 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 93 | int main(void) |
| 94 | { |
| 95 | uint8\_t temp, i; |
| 96 | DevicesInit(); //  MCU器件初始化 |
| 97 |  |
| 98 | while (clock100ms < 3) |
| 99 | ;           // 延时>60ms,等待TM1638上电完成 |
| 100 | TM1638\_Init();  // 初始化TM1638 |
| 101 | initial\_lcd();  // 初始化JLX12864 |
| 102 | clear\_screen(); //clear all dots |
| 103 |  |
| 104 | while (1) |
| 105 | { |
| 106 | if (clock100ms\_flag == 1) // 检查0.1秒定时是否到 |
| 107 | { |
| 108 | clock100ms\_flag = 0; |
| 109 | // 每0.1秒累加计时值在数码管上以十进制显示，有键按下时暂停计时 |
| 110 | if (key\_code == 0) |
| 111 | { |
| 112 | if (++test\_counter >= 10000) |
| 113 | test\_counter = 0; |
| 114 | digit[0] = test\_counter / 1000;     // 计算百位数 |
| 115 | digit[1] = test\_counter / 100 % 10; // 计算十位数 |
| 116 | digit[2] = test\_counter / 10 % 10;  // 计算个位数 |
| 117 | digit[3] = test\_counter % 10;       // 计算百分位数 |
| 118 | } |
| 119 | } |
| 120 |  |
| 121 | if (clock500ms\_flag == 1) // 检查0.5秒定时是否到 |
| 122 | { |
| 123 | clock500ms\_flag = 0; |
| 124 | // 8个指示灯以走马灯方式，每0.5秒向右（循环）移动一格 |
| 125 | temp = led[0]; |
| 126 | for (i = 0; i < 7; i++) |
| 127 | led[i] = led[i + 1]; |
| 128 | led[7] = temp; |
| 129 | } |
| 130 |  |
| 131 | if (clock2s\_flag) |
| 132 | { |
| 133 | clock2s\_flag = 0; |
| 134 | TEST\_H; |
| 135 | switch (lcd\_act) |
| 136 | { |
| 137 | case 0: |
| 138 | ++lcd\_act; |
| 139 | clear\_screen(); |
| 140 | display\_128x64(xiaohui); |
| 141 | break; |
| 142 | case 1: |
| 143 | ++lcd\_act; |
| 144 | clear\_screen(); |
| 145 | display\_GB2312\_string(1, 1, "12864,带中文字库", false);  //在第 1 页，第 1 列，显示一串 16x16 点阵汉字或 8x16 的 ASCII 字 |
| 146 | display\_GB2312\_string(3, 1, "16X16 简体汉字库,", false); //显示一串 16x16 点阵汉字或 8x16 的 ASCII 字.以下雷同 |
| 147 | display\_GB2312\_string(5, 1, "或 8X16 点阵 ASCII,", false); |
| 148 | display\_GB2312\_string(7, 1, "或 5x8 点阵 ASCII 码", false); |
| 149 | break; |
| 150 | case 2: |
| 151 | ++lcd\_act; |
| 152 | clear\_screen(); |
| 153 | display\_GB2312\_string(1, 1, "晶联讯成立于二零", true); |
| 154 | display\_GB2312\_string(3, 1, "零四年十一月七日", true); |
| 155 | display\_GB2312\_string(5, 1, "主要生产液晶模块", true); |
| 156 | display\_GB2312\_string(7, 1, "品质至上真诚服务", true); |
| 157 | break; |
| 158 | case 3: |
| 159 | ++lcd\_act; |
| 160 | display\_GB2312\_string(1, 1, "GB2312 简体字库及", true); |
| 161 | display\_GB2312\_string(3, 1, "有图型功能，可自", false); |
| 162 | display\_GB2312\_string(5, 1, "编大字或图像或生", true); |
| 163 | display\_GB2312\_string(7, 1, "僻字，例如： ", false); |
| 164 | display\_graphic\_16x16(7, 97, jiong1); //在第 7 页，第 81 列显示单个自编生僻汉字“囧” |
| 165 | display\_graphic\_16x16(7, 113, lei1);  //显示单个自编生僻汉字"畾“ |
| 166 | break; |
| 167 | case 4: |
| 168 | ++lcd\_act; |
| 169 | display\_GB2312\_string(1, 1, "<!@#$%^&\*()\_-+]/", false);   //在第 1 页，第 1 列，显示一串 16x16 点阵汉字或 8\*16 的 ASCII 字 |
| 170 | display\_string\_5x8(3, 1, "<!@#$%^&\*()\_-+]/;.,?[", true); //在第 3 页，第 1 列，显示一串 5x8 点阵的 ASCII 字 |
| 171 | display\_string\_5x8(4, 1, "JLX electronics Co., ", false); //显示一串 5x8 点阵的 ASCII 字 |
| 172 | display\_string\_5x8(5, 1, "Ltd. established at ", true);  //显示一串 5x8 点阵的 ASCII 字 |
| 173 | display\_string\_5x8(6, 1, "year 2004.Focus LCM. ", false); //显示一串 5x8 点阵的 ASCII 字 |
| 174 | display\_string\_5x8(7, 1, "TEL:0755-29784961 ", true);    //显示一串 5x8 点阵的 ASCII 字 |
| 175 | display\_string\_5x8(8, 1, "FAX:0755-29784964 ", false);    //显示一串 5x8 点阵的 ASCII 字 |
| 176 | break; |
| 177 | case 5: |
| 178 | ++lcd\_act; |
| 179 | display\_GB2312\_string(1, 1, "啊阿埃挨哎唉哀皑", true); //在第 1 页，第 1 列，显示一串 16x16 点阵汉字或 8x16 的 ASCII 字 |
| 180 | display\_GB2312\_string(3, 1, "癌蔼矮艾碍爱隘鞍", false); //显示一串 16x16 点阵汉字或 8x16 的ASCII 字.以下雷同 |
| 181 | display\_GB2312\_string(5, 1, "氨安俺按暗岸胺案", true); |
| 182 | display\_GB2312\_string(7, 1, "肮昂盎凹敖熬翱袄", false); |
| 183 | break; |
| 184 | case 6: |
| 185 | lcd\_act = 0; |
| 186 | display\_GB2312\_string(1, 1, "鬟鬣麽麾縻麂麇麈", false); |
| 187 | display\_GB2312\_string(3, 1, "麋麒鏖麝麟黛黜黝", true); |
| 188 | display\_GB2312\_string(5, 1, "黠黟黢黩黧黥黪黯", false); |
| 189 | display\_GB2312\_string(7, 1, "鼢鼬鼯鼹鼷鼽鼾齄", true); |
| 190 | break; |
| 191 | default: |
| 192 | break; |
| 193 | } |
| 194 | TEST\_L; |
| 195 | } |
| 196 | } |
| 197 | } |
| 198 |  |
| 199 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 200 | // |
| 201 | // 函数原型：void GPIOInit(void) |
| 202 | // 函数功能：GPIO初始化。使能PortK，设置PK4,PK5为输出；使能PortM，设置PM0为输出。 |
| 203 | //          （PK4连接TM1638的STB，PK5连接TM1638的DIO，PM0连接TM1638的CLK） |
| 204 | // 函数参数：无 |
| 205 | // 函数返回值：无 |
| 206 | // |
| 207 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 208 | void GPIOInit(void) |
| 209 | { |
| 210 | //配置TM1638芯片管脚 |
| 211 | SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOK); // 使能端口 K |
| 212 | while (!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOK)) |
| 213 | { |
| 214 | }; // 等待端口 K准备完毕 |
| 215 |  |
| 216 | SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOM); // 使能端口 M |
| 217 | while (!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOM)) |
| 218 | { |
| 219 | }; // 等待端口 M准备完毕 |
| 220 |  |
| 221 | // 设置端口 K的第4,5位（PK4,PK5）为输出引脚       PK4-STB  PK5-DIO |
| 222 | GPIOPinTypeGPIOOutput(GPIO\_PORTK\_BASE, GPIO\_PIN\_4 | GPIO\_PIN\_5); |
| 223 | // 设置端口 M的第0位（PM0）为输出引脚   PM0-CLK |
| 224 | GPIOPinTypeGPIOOutput(GPIO\_PORTM\_BASE, GPIO\_PIN\_0); |
| 225 | } |
| 226 |  |
| 227 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 228 | // |
| 229 | // 函数原型：SysTickInit(void) |
| 230 | // 函数功能：设置SysTick中断 |
| 231 | // 函数参数：无 |
| 232 | // 函数返回值：无 |
| 233 | // |
| 234 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 235 | void SysTickInit(void) |
| 236 | { |
| 237 | SysTickPeriodSet(ui32SysClock / SYSTICK\_FREQUENCY); // 设置心跳节拍,定时周期20ms |
| 238 | SysTickEnable();                                    // SysTick使能 |
| 239 | SysTickIntEnable();                                 // SysTick中断允许 |
| 240 | } |
| 241 |  |
| 242 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 243 | // |
| 244 | // 函数原型：void DevicesInit(void) |
| 245 | // 函数功能：CU器件初始化，包括系统时钟设置、GPIO初始化和SysTick中断设置 |
| 246 | // 函数参数：无 |
| 247 | // 函数返回值：无 |
| 248 | // |
| 249 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 250 | void DevicesInit(void) |
| 251 | { |
| 252 | // 使用外部25MHz主时钟源，经过PLL，然后分频为20MHz |
| 253 | ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN | |
| 254 | SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480), |
| 255 | 20000000); |
| 256 |  |
| 257 | GPIOInit();        // GPIO初始化 |
| 258 | SysTickInit();     // 设置SysTick中断 |
| 259 | IntMasterEnable(); // 总中断允许 |
| 260 | } |
| 261 |  |
| 262 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 263 | // |
| 264 | // 函数原型：void SysTick\_Handler(void) |
| 265 | // 函数功能：SysTick中断服务程序 |
| 266 | // 函数参数：无 |
| 267 | // 函数返回值：无 |
| 268 | // |
| 269 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 270 | void SysTick\_Handler(void) // 定时周期为20ms |
| 271 | { |
| 272 | // 0.1秒钟软定时器计数 |
| 273 | if (++clock100ms >= V\_T100ms) |
| 274 | { |
| 275 | clock100ms\_flag = 1; // 当0.1秒到时，溢出标志置1 |
| 276 | clock100ms = 0; |
| 277 | } |
| 278 |  |
| 279 | // 0.5秒钟软定时器计数 |
| 280 | if (++clock500ms >= V\_T500ms) |
| 281 | { |
| 282 | clock500ms\_flag = 1; // 当0.5秒到时，溢出标志置1 |
| 283 | clock500ms = 0; |
| 284 | } |
| 285 |  |
| 286 | // 2.0秒钟软定时器计数 |
| 287 | if (++clock2s >= V\_T2s) |
| 288 | { |
| 289 | clock2s\_flag = 1; // 当2.0秒到时，溢出标志置1 |
| 290 | clock2s = 0; |
| 291 | } |
| 292 |  |
| 293 | // 刷新全部数码管和LED指示灯 |
| 294 | TM1638\_RefreshDIGIandLED(digit, pnt, led); |
| 295 |  |
| 296 | // 检查当前键盘输入，0代表无键操作，1-9表示有对应按键 |
| 297 | // 键号显示在一位数码管上 |
| 298 | key\_code = TM1638\_Readkeyboard(); |
| 299 |  |
| 300 | digit[5] = key\_code; |
| 301 | } |
| 302 |  |

### 实验任务4\_2

|  |  |
| --- | --- |
| 1 | /\*\* |
| 2 | \* @file exp4\_2.c |
| 3 | \* @author 上海交通大学电子工程系实验教学中心; Guorui Wei (313017602@qq.com) |
| 4 | \* @brief 实验4\_2 |
| 5 | \* @version 0.1 |
| 6 | \* @date 2021-06-05 |
| 7 | \* |
| 8 | \* @copyright 2020-2021, 上海交通大学电子工程系实验教学中心 |
| 9 | \* |
| 10 | \*/ |
| 11 |  |
| 12 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 13 | // |
| 14 | // 头文件 |
| 15 | // |
| 16 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 17 | #include <stdint.h> |
| 18 | #include <stdbool.h> |
| 19 | #include "inc/hw\_memmap.h"       // 基址宏定义 |
| 20 | #include "inc/hw\_types.h"        // 数据类型宏定义，寄存器访问函数 |
| 21 | #include "driverlib/debug.h"     // 调试用 |
| 22 | #include "driverlib/gpio.h"      // 通用IO口宏定义 |
| 23 | #include "driverlib/pin\_map.h"   // TM4C系列MCU外围设备管脚宏定义 |
| 24 | #include "driverlib/sysctl.h"    // 系统控制定义 |
| 25 | #include "driverlib/systick.h"   // SysTick Driver 原型 |
| 26 | #include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型 |
| 27 | #include "JLX12864.h"            // 与控制JLX12864G有关的函数 |
| 28 | #include "tm1638.h"              // 与控制TM1638芯片有关的函数 |
| 29 | #include "string.h"              // |
| 30 |  |
| 31 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 32 | // |
| 33 | // 宏定义 |
| 34 | // |
| 35 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 36 | #define SYSTICK\_FREQUENCY 50  // SysTick频率为50Hz，即循环定时周期20ms |
| 37 | #define V\_T100ms 5            // 0.1s软件定时器溢出值，5个20ms |
| 38 | #define V\_T500ms 25           // 0.5s软件定时器溢出值，25个20ms |
| 39 | #define V\_T2s 100             // 2.0s软定时器溢出值，100个20ms |
| 40 | #define V\_T10s 500            // 10.0s软定时器溢出值，500个20ms |
| 41 | #define LCD\_MAX\_BLOCK 15      // 显示屏上的最大8x8分区数 |
| 42 | #define LCD\_MAX\_BLOCK\_CHAR 15 // 显示屏上每个分区的最大字符数 |
| 43 |  |
| 44 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 45 | // |
| 46 | // 函数原型声明 |
| 47 | // |
| 48 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 49 | void GPIOInit(void);    // GPIO初始化 |
| 50 | void SysTickInit(void); // 设置SysTick中断 |
| 51 | void DevicesInit(void); // MCU器件初始化，注：会调用上述函数 |
| 52 |  |
| 53 | /\*\* |
| 54 | \* UI状态机相关函数 |
| 55 | \*/ |
| 56 |  |
| 57 | void ui\_state\_proc(uint16\_t ui\_state); |
| 58 | void ui\_proc0(void); |
| 59 | void ui\_proc001(void); |
| 60 | void ui\_proc003(void); |
| 61 | void ui\_proc005(void); |
| 62 | void ENTER\_detect(void); |
| 63 | void LEFT\_detect(void); |
| 64 | void RIGHT\_detect(void); |
| 65 | void INCREASE\_detect(void); |
| 66 | void DECREASE\_detect(void); |
| 67 |  |
| 68 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 69 | // |
| 70 | // 变量定义 |
| 71 | // |
| 72 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 73 |  |
| 74 | // 软件定时器计数 |
| 75 | uint8\_t clock100ms = 0; |
| 76 | uint8\_t clock500ms = 0; |
| 77 | uint8\_t clock2s = 0; |
| 78 | uint16\_t NOKEY\_clock10s = 0; |
| 79 |  |
| 80 | // 软件定时器溢出标志 |
| 81 | uint8\_t clock100ms\_flag = 0; |
| 82 | uint8\_t clock500ms\_flag = 0; |
| 83 | uint8\_t clock2s\_flag = 0; |
| 84 | uint8\_t NOKEY\_clock10s\_flag = 0; |
| 85 |  |
| 86 | /\*\* |
| 87 | \* 按键事件标志 |
| 88 | \*/ |
| 89 | uint8\_t key\_LEFT\_flag = 0; |
| 90 | uint8\_t key\_RIGHT\_flag = 0; |
| 91 | uint8\_t key\_INCREASE\_flag = 0; |
| 92 | uint8\_t key\_DECREASE\_flag = 0; |
| 93 | uint8\_t key\_ENTER\_flag = 0; |
| 94 |  |
| 95 | // 测试用计数器 |
| 96 | uint32\_t test\_counter = 0; |
| 97 |  |
| 98 | // 8位数码管显示的数字或字母符号 |
| 99 | // 注：板上数码位从左到右序号排列为4、5、6、7、0、1、2、3 |
| 100 | uint8\_t digit[8] = {' ', ' ', ' ', ' ', '\_', ' ', '\_', ' '}; |
| 101 |  |
| 102 | // 8位小数点 1亮  0灭 |
| 103 | // 注：板上数码位小数点从左到右序号排列为4、5、6、7、0、1、2、3 |
| 104 | uint8\_t pnt = 0x04; |
| 105 |  |
| 106 | // 8个LED指示灯状态，0灭，1亮 |
| 107 | // 注：板上指示灯从左到右序号排列为7、6、5、4、3、2、1、0 |
| 108 | //     对应元件LED8、LED7、LED6、LED5、LED4、LED3、LED2、LED1 |
| 109 | uint8\_t led[] = {1, 1, 1, 1, 1, 1, 1, 0}; |
| 110 |  |
| 111 | // 当前按键值 |
| 112 | uint8\_t key\_code = 0; |
| 113 | uint8\_t pre\_key\_code; // 上一按键值 |
| 114 |  |
| 115 | // 系统时钟频率 |
| 116 | uint32\_t ui32SysClock; |
| 117 |  |
| 118 | /\*\* |
| 119 | \* 用户界面（UI）状态机相关变量定义 |
| 120 | \*/ |
| 121 |  |
| 122 | uint16\_t ui\_state = 0x0; // 用户界面（UI）状态机当前状态 |
| 123 |  |
| 124 | /\*\* |
| 125 | \* @brief 状态参数结构体 |
| 126 | \* |
| 127 | \*/ |
| 128 | struct ACT\_T |
| 129 | { |
| 130 | uint8\_t row\_page[LCD\_MAX\_BLOCK];                      // 显示屏上每个分区的起始行页号 |
| 131 | uint8\_t col\_page[LCD\_MAX\_BLOCK];                      // 显示屏上每个分区的起始列页号 |
| 132 | unsigned char str[LCD\_MAX\_BLOCK][LCD\_MAX\_BLOCK\_CHAR]; // 显示屏上每个分区的显示内容 |
| 133 | const uint8\_t SIZE;                                   // 显示屏上有效分区的数量 |
| 134 | }; |
| 135 |  |
| 136 | struct ACT\_T act0 = { |
| 137 | {3, 3, 5, 5, 5, 5, 5}, |
| 138 | {1, 11, 1, 11, 12, 13, 14}, |
| 139 | {"工作模式：", "模式A", "工作参数：", "1", ".", "1", "Hz"}, |
| 140 | 7}; |
| 141 |  |
| 142 | struct ACT\_T \*act[] = {&act0}; |
| 143 |  |
| 144 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 145 | // |
| 146 | // 主程序 |
| 147 | // |
| 148 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 149 | int main(void) |
| 150 | { |
| 151 | uint8\_t temp, i; |
| 152 | DevicesInit(); //  MCU器件初始化 |
| 153 |  |
| 154 | while (clock100ms < 3) |
| 155 | ;           // 延时>60ms,等待TM1638上电完成 |
| 156 | TM1638\_Init();  // 初始化TM1638 |
| 157 | initial\_lcd();  // 初始化JLX12864 |
| 158 | clear\_screen(); //clear all dots |
| 159 |  |
| 160 | while (1) |
| 161 | { |
| 162 | if (clock100ms\_flag == 1) // 检查0.1秒定时是否到 |
| 163 | { |
| 164 | clock100ms\_flag = 0; |
| 165 | // 每0.1秒累加计时值在数码管上以十进制显示，有键按下时暂停计时 |
| 166 | if (key\_code == 0) |
| 167 | { |
| 168 | if (++test\_counter >= 10000) |
| 169 | test\_counter = 0; |
| 170 | digit[0] = test\_counter / 1000;     // 计算百位数 |
| 171 | digit[1] = test\_counter / 100 % 10; // 计算十位数 |
| 172 | digit[2] = test\_counter / 10 % 10;  // 计算个位数 |
| 173 | digit[3] = test\_counter % 10;       // 计算百分位数 |
| 174 | } |
| 175 | } |
| 176 |  |
| 177 | if (clock500ms\_flag == 1) // 检查0.5秒定时是否到 |
| 178 | { |
| 179 | clock500ms\_flag = 0; |
| 180 | // 8个指示灯以走马灯方式，每0.5秒向右（循环）移动一格 |
| 181 | temp = led[0]; |
| 182 | for (i = 0; i < 7; i++) |
| 183 | led[i] = led[i + 1]; |
| 184 | led[7] = temp; |
| 185 | } |
| 186 |  |
| 187 | ui\_state\_proc(ui\_state); |
| 188 | } |
| 189 | } |
| 190 |  |
| 191 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 192 | // |
| 193 | // 函数原型：void GPIOInit(void) |
| 194 | // 函数功能：GPIO初始化。使能PortK，设置PK4,PK5为输出；使能PortM，设置PM0为输出。 |
| 195 | //          （PK4连接TM1638的STB，PK5连接TM1638的DIO，PM0连接TM1638的CLK） |
| 196 | // 函数参数：无 |
| 197 | // 函数返回值：无 |
| 198 | // |
| 199 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 200 | void GPIOInit(void) |
| 201 | { |
| 202 | //配置TM1638芯片管脚 |
| 203 | SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOK); // 使能端口 K |
| 204 | while (!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOK)) |
| 205 | { |
| 206 | }; // 等待端口 K准备完毕 |
| 207 |  |
| 208 | SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOM); // 使能端口 M |
| 209 | while (!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOM)) |
| 210 | { |
| 211 | }; // 等待端口 M准备完毕 |
| 212 |  |
| 213 | // 设置端口 K的第4,5位（PK4,PK5）为输出引脚       PK4-STB  PK5-DIO |
| 214 | GPIOPinTypeGPIOOutput(GPIO\_PORTK\_BASE, GPIO\_PIN\_4 | GPIO\_PIN\_5); |
| 215 | // 设置端口 M的第0位（PM0）为输出引脚   PM0-CLK |
| 216 | GPIOPinTypeGPIOOutput(GPIO\_PORTM\_BASE, GPIO\_PIN\_0); |
| 217 | } |
| 218 |  |
| 219 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 220 | // |
| 221 | // 函数原型：SysTickInit(void) |
| 222 | // 函数功能：设置SysTick中断 |
| 223 | // 函数参数：无 |
| 224 | // 函数返回值：无 |
| 225 | // |
| 226 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 227 | void SysTickInit(void) |
| 228 | { |
| 229 | SysTickPeriodSet(ui32SysClock / SYSTICK\_FREQUENCY); // 设置心跳节拍,定时周期20ms |
| 230 | SysTickEnable();                                    // SysTick使能 |
| 231 | SysTickIntEnable();                                 // SysTick中断允许 |
| 232 | } |
| 233 |  |
| 234 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 235 | // |
| 236 | // 函数原型：void DevicesInit(void) |
| 237 | // 函数功能：CU器件初始化，包括系统时钟设置、GPIO初始化和SysTick中断设置 |
| 238 | // 函数参数：无 |
| 239 | // 函数返回值：无 |
| 240 | // |
| 241 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 242 | void DevicesInit(void) |
| 243 | { |
| 244 | // 使用外部25MHz主时钟源，经过PLL，然后分频为20MHz |
| 245 | ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN | |
| 246 | SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480), |
| 247 | 20000000); |
| 248 |  |
| 249 | GPIOInit();        // GPIO初始化 |
| 250 | SysTickInit();     // 设置SysTick中断 |
| 251 | IntMasterEnable(); // 总中断允许 |
| 252 | } |
| 253 |  |
| 254 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 255 | // |
| 256 | // 函数原型：void SysTick\_Handler(void) |
| 257 | // 函数功能：SysTick中断服务程序 |
| 258 | // 函数参数：无 |
| 259 | // 函数返回值：无 |
| 260 | // |
| 261 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 262 | void SysTick\_Handler(void) // 定时周期为20ms |
| 263 | { |
| 264 | // 0.1秒钟软定时器计数 |
| 265 | if (++clock100ms >= V\_T100ms) |
| 266 | { |
| 267 | clock100ms\_flag = 1; // 当0.1秒到时，溢出标志置1 |
| 268 | clock100ms = 0; |
| 269 | } |
| 270 |  |
| 271 | // 0.5秒钟软定时器计数 |
| 272 | if (++clock500ms >= V\_T500ms) |
| 273 | { |
| 274 | clock500ms\_flag = 1; // 当0.5秒到时，溢出标志置1 |
| 275 | clock500ms = 0; |
| 276 | } |
| 277 |  |
| 278 | // 刷新全部数码管和LED指示灯 |
| 279 | TM1638\_RefreshDIGIandLED(digit, pnt, led); |
| 280 |  |
| 281 | // 检查当前键盘输入，0代表无键操作，1-9表示有对应按键 |
| 282 | // 键号显示在一位数码管上 |
| 283 | pre\_key\_code = key\_code;          // 保存上一按键值 |
| 284 | key\_code = TM1638\_Readkeyboard(); // 更新当前按键值 |
| 285 |  |
| 286 | digit[5] = key\_code; |
| 287 |  |
| 288 | ENTER\_detect(); |
| 289 | LEFT\_detect(); |
| 290 | RIGHT\_detect(); |
| 291 | INCREASE\_detect(); |
| 292 | DECREASE\_detect(); |
| 293 |  |
| 294 | // 10.0秒钟软定时器计数 |
| 295 | if (!key\_code && ++NOKEY\_clock10s >= V\_T10s) // 当无键按下时 |
| 296 | { |
| 297 | NOKEY\_clock10s\_flag = 1; // 当10.0秒到时，溢出标志置1 |
| 298 | NOKEY\_clock10s = 0; |
| 299 | } |
| 300 | // 若有键按下，则10.0秒计数清零 |
| 301 | if (key\_code) |
| 302 | { |
| 303 | NOKEY\_clock10s = 0; |
| 304 | } |
| 305 | } |
| 306 |  |
| 307 | /\*\* |
| 308 | \* @brief UI状态机处理函数 |
| 309 | \* |
| 310 | \* @param ui\_state UI状态机当前状态 |
| 311 | \*/ |
| 312 | void ui\_state\_proc(uint16\_t ui\_state) |
| 313 | { |
| 314 | switch (ui\_state) |
| 315 | { |
| 316 | case 0x0: // ACT0 |
| 317 | ui\_proc0(); |
| 318 | break; |
| 319 | case 0x001: //ACT001 |
| 320 | ui\_proc001(); |
| 321 | break; |
| 322 | case 0x003: // ACT003 |
| 323 | ui\_proc003(); |
| 324 | break; |
| 325 | case 0x005: // ACT005 |
| 326 | ui\_proc005(); |
| 327 | break; |
| 328 | default: |
| 329 | ui\_state = 0; |
| 330 | break; |
| 331 | } |
| 332 | } |
| 333 |  |
| 334 | /\*\* |
| 335 | \* @brief UI状态机ACT0状态处理 |
| 336 | \* 开机初始画面，不显示光标 |
| 337 | \* |
| 338 | \*/ |
| 339 | void ui\_proc0(void) |
| 340 | { |
| 341 | uint8\_t i = 0; |
| 342 | // 显示开机初始画面，无光标 |
| 343 | for (i = 0; i < act[0]->SIZE; ++i) |
| 344 | { |
| 345 | display\_GB2312\_string(act[0]->row\_page[i], act[0]->col\_page[i] \* 8 - 7, act[0]->str[i], 0); |
| 346 | } |
| 347 |  |
| 348 | // 当有任意按键被按下："模式#"做反白效果（光标），转移至状态ACT001 |
| 349 | if (!pre\_key\_code && key\_code) |
| 350 | { |
| 351 | key\_LEFT\_flag = key\_RIGHT\_flag = key\_INCREASE\_flag = key\_DECREASE\_flag = key\_ENTER\_flag = 0; |
| 352 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 1); |
| 353 | ui\_state = 0x001; |
| 354 | } |
| 355 | } |
| 356 |  |
| 357 | /\*\* |
| 358 | \* @brief UI状态机ACT001状态处理 |
| 359 | \* 光标在工作模式选择位置 |
| 360 | \* |
| 361 | \*/ |
| 362 | void ui\_proc001(void) |
| 363 | { |
| 364 | // 当"右"键按下：光标移到工作参数的个位，下一状态ACT003 |
| 365 | if (key\_RIGHT\_flag) |
| 366 | { |
| 367 | key\_RIGHT\_flag = 0; |
| 368 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 0); |
| 369 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 1); |
| 370 | ui\_state = 0x003; |
| 371 | } |
| 372 | // 当"左"键按下：光标移到工作参数的十分位，下一状态ACT005 |
| 373 | else if (key\_LEFT\_flag) |
| 374 | { |
| 375 | key\_LEFT\_flag = 0; |
| 376 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 0); |
| 377 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 378 | ui\_state = 0x005; |
| 379 | } |
| 380 | // 当"+"键按下："模式#"按A、B、C、A正序循环切换，留在本状态 |
| 381 | else if (key\_INCREASE\_flag) |
| 382 | { |
| 383 | key\_INCREASE\_flag = 0; |
| 384 | if (++((act[0]->str[1])[strlen((const char \*)(act[0]->str[1])) - 1]) > 'C') |
| 385 | { |
| 386 | (act[0]->str[1])[strlen((const char \*)(act[0]->str[1])) - 1] = 'A'; |
| 387 | } |
| 388 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 1); |
| 389 | } |
| 390 | // 当"-"键按下："模式#"按C、B、A、C逆序循环切换，留在本状态 |
| 391 | else if (key\_DECREASE\_flag) |
| 392 | { |
| 393 | key\_DECREASE\_flag = 0; |
| 394 | if (--((act[0]->str[1])[strlen((const char \*)(act[0]->str[1])) - 1]) < 'A') |
| 395 | { |
| 396 | (act[0]->str[1])[strlen((const char \*)(act[0]->str[1])) - 1] = 'C'; |
| 397 | } |
| 398 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 1); |
| 399 | } |
| 400 |  |
| 401 | // 当10秒无操作："模式#"反白效果解除，下一状态ACT0 |
| 402 | if (NOKEY\_clock10s\_flag) |
| 403 | { |
| 404 | NOKEY\_clock10s\_flag = 0; |
| 405 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 0); |
| 406 | ui\_state = 0x0; |
| 407 | } |
| 408 | } |
| 409 |  |
| 410 | /\*\* |
| 411 | \* @brief UI状态机ACT003状态处理 |
| 412 | \* 光标在工作参数个位的位置 |
| 413 | \* |
| 414 | \*/ |
| 415 | void ui\_proc003(void) |
| 416 | { |
| 417 | // 当"右"键按下：光标移到十分位，下一状态ACT005 |
| 418 | if (key\_RIGHT\_flag) |
| 419 | { |
| 420 | key\_RIGHT\_flag = 0; |
| 421 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 0); |
| 422 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 423 | ui\_state = 0x005; |
| 424 | } |
| 425 | // 当"左"键按下：光标移到"模式#"，下一状态ACT001 |
| 426 | else if (key\_LEFT\_flag) |
| 427 | { |
| 428 | key\_LEFT\_flag = 0; |
| 429 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 0); |
| 430 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 1); |
| 431 | ui\_state = 0x001; |
| 432 | } |
| 433 | // 当"+"键按下：个位数按1、2、...、9、0、1、...正序循环切换，留在本状态 |
| 434 | else if (key\_INCREASE\_flag) |
| 435 | { |
| 436 | key\_INCREASE\_flag = 0; |
| 437 | if (++((act[0]->str[3])[0]) > '9') |
| 438 | { |
| 439 | (act[0]->str[3])[0] = '0'; |
| 440 | } |
| 441 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 1); |
| 442 | } |
| 443 | // 当"-"键按下：个位数按9、8、...、0、9、8、...逆序循环切换，留在本状态 |
| 444 | else if (key\_DECREASE\_flag) |
| 445 | { |
| 446 | key\_DECREASE\_flag = 0; |
| 447 | if (--((act[0]->str[3])[0]) < '0') |
| 448 | { |
| 449 | (act[0]->str[3])[0] = '9'; |
| 450 | } |
| 451 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 1); |
| 452 | } |
| 453 | else if (key\_ENTER\_flag) |
| 454 | { |
| 455 | key\_ENTER\_flag = 0; |
| 456 | } |
| 457 |  |
| 458 | // 当10秒无操作：个位反白效果解除，下一状态ACT0 |
| 459 | if (NOKEY\_clock10s\_flag) |
| 460 | { |
| 461 | NOKEY\_clock10s\_flag = 0; |
| 462 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 0); |
| 463 | ui\_state = 0x0; |
| 464 | } |
| 465 | } |
| 466 |  |
| 467 | /\*\* |
| 468 | \* @brief UI状态机ACT005状态处理 |
| 469 | \* 光标在工作参数十分位的位置 |
| 470 | \* |
| 471 | \*/ |
| 472 | void ui\_proc005(void) |
| 473 | { |
| 474 | // 当"右"键按下：光标移到"模式#"，下一状态ACT001 |
| 475 | if (key\_RIGHT\_flag) |
| 476 | { |
| 477 | key\_RIGHT\_flag = 0; |
| 478 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 479 | display\_GB2312\_string(act[0]->row\_page[1], act[0]->col\_page[1] \* 8 - 7, act[0]->str[1], 1); |
| 480 | ui\_state = 0x001; |
| 481 | } |
| 482 | // 当"左"键按下：光标移到个位，下一状态ACT003 |
| 483 | else if (key\_LEFT\_flag) |
| 484 | { |
| 485 | key\_LEFT\_flag = 0; |
| 486 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 487 | display\_GB2312\_string(act[0]->row\_page[3], act[0]->col\_page[3] \* 8 - 7, act[0]->str[3], 1); |
| 488 | ui\_state = 0x003; |
| 489 | } |
| 490 | // 当"+"键按下：十分位数按1、2、...、9、0、1、...正序循环切换，留在本状态 |
| 491 | else if (key\_INCREASE\_flag) |
| 492 | { |
| 493 | key\_INCREASE\_flag = 0; |
| 494 | if (++((act[0]->str[5])[0]) > '9') |
| 495 | { |
| 496 | (act[0]->str[5])[0] = '0'; |
| 497 | } |
| 498 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 499 | } |
| 500 | // 当"-"键按下：十分位数按9、8、...、0、9、8、...逆序循环切换，留在本状态 |
| 501 | else if (key\_DECREASE\_flag) |
| 502 | { |
| 503 | key\_DECREASE\_flag = 0; |
| 504 | if (--((act[0]->str[5])[0]) < '0') |
| 505 | { |
| 506 | (act[0]->str[5])[0] = '9'; |
| 507 | } |
| 508 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 509 | } |
| 510 | else if (key\_ENTER\_flag) |
| 511 | { |
| 512 | key\_ENTER\_flag = 0; |
| 513 | } |
| 514 |  |
| 515 | // 当10秒无操作：工作参数十分位反白效果解除，下一状态ACT0 |
| 516 | if (NOKEY\_clock10s\_flag) |
| 517 | { |
| 518 | NOKEY\_clock10s\_flag = 0; |
| 519 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 520 | ui\_state = 0x0; |
| 521 | } |
| 522 | } |
| 523 |  |
| 524 | /\*\* |
| 525 | \* @brief "确定"键按键检测 |
| 526 | \* |
| 527 | \*/ |
| 528 | void ENTER\_detect(void) |
| 529 | { |
| 530 | if (pre\_key\_code != 5 && key\_code == 5) |
| 531 | { |
| 532 | key\_ENTER\_flag = 1; |
| 533 | } |
| 534 | } |
| 535 |  |
| 536 | /\*\* |
| 537 | \* @brief "+"键按键检测 |
| 538 | \* |
| 539 | \*/ |
| 540 | void INCREASE\_detect(void) |
| 541 | { |
| 542 | if (pre\_key\_code != 2 && key\_code == 2) |
| 543 | { |
| 544 | key\_INCREASE\_flag = 1; |
| 545 | } |
| 546 | } |
| 547 |  |
| 548 | /\*\* |
| 549 | \* @brief "-"键按键检测 |
| 550 | \* |
| 551 | \*/ |
| 552 | void DECREASE\_detect(void) |
| 553 | { |
| 554 | if (pre\_key\_code != 8 && key\_code == 8) |
| 555 | { |
| 556 | key\_DECREASE\_flag = 1; |
| 557 | } |
| 558 | } |
| 559 |  |
| 560 | /\*\* |
| 561 | \* @brief "左"键按键检测 |
| 562 | \* |
| 563 | \*/ |
| 564 | void LEFT\_detect(void) |
| 565 | { |
| 566 | if (pre\_key\_code != 4 && key\_code == 4) |
| 567 | { |
| 568 | key\_LEFT\_flag = 1; |
| 569 | } |
| 570 | } |
| 571 |  |
| 572 | /\*\* |
| 573 | \* @brief "右"键按键检测 |
| 574 | \* |
| 575 | \*/ |
| 576 | void RIGHT\_detect(void) |
| 577 | { |
| 578 | if (pre\_key\_code != 6 && key\_code == 6) |
| 579 | { |
| 580 | key\_RIGHT\_flag = 1; |
| 581 | } |
| 582 | } |
| 583 |  |

### 实验任务4\_3

|  |  |
| --- | --- |
| 1 | /\*\* |
| 2 | \* @file exp4\_3.c |
| 3 | \* @author 上海交通大学电子工程系实验教学中心; Guorui Wei (313017602@qq.com) |
| 4 | \* @brief 实验4\_3 |
| 5 | \* @version 0.1 |
| 6 | \* @date 2021-06-05 |
| 7 | \* |
| 8 | \* @copyright 2020-2021, 上海交通大学电子工程系实验教学中心 |
| 9 | \* |
| 10 | \*/ |
| 11 |  |
| 12 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 13 | // |
| 14 | // 头文件 |
| 15 | // |
| 16 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 17 | #include <stdint.h> |
| 18 | #include <stdbool.h> |
| 19 | #include "inc/hw\_memmap.h"       // 基址宏定义 |
| 20 | #include "inc/hw\_types.h"        // 数据类型宏定义，寄存器访问函数 |
| 21 | #include "driverlib/debug.h"     // 调试用 |
| 22 | #include "driverlib/gpio.h"      // 通用IO口宏定义 |
| 23 | #include "driverlib/pin\_map.h"   // TM4C系列MCU外围设备管脚宏定义 |
| 24 | #include "driverlib/sysctl.h"    // 系统控制定义 |
| 25 | #include "driverlib/systick.h"   // SysTick Driver 原型 |
| 26 | #include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型 |
| 27 | #include "JLX12864.h"            // 与控制JLX12864G有关的函数 |
| 28 | #include "tm1638.h"              // 与控制TM1638芯片有关的函数 |
| 29 | #include "string.h"              // |
| 30 |  |
| 31 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 32 | // |
| 33 | // 宏定义 |
| 34 | // |
| 35 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 36 | #define SYSTICK\_FREQUENCY 50  // SysTick频率为50Hz，即循环定时周期20ms |
| 37 | #define V\_T100ms 5            // 0.1s软件定时器溢出值，5个20ms |
| 38 | #define V\_T500ms 25           // 0.5s软件定时器溢出值，25个20ms |
| 39 | #define V\_T2s 100             // 2.0s软定时器溢出值，100个20ms |
| 40 | #define V\_T5s 250             // 5.0s软定时器溢出值，250个20ms |
| 41 | #define V\_T10s 500            // 10.0s软定时器溢出值，500个20ms |
| 42 | #define LCD\_MAX\_BLOCK 15      // 显示屏上的最大8x8分区数 |
| 43 | #define LCD\_MAX\_BLOCK\_CHAR 15 // 显示屏上每个分区的最大字符数 |
| 44 |  |
| 45 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 46 | // |
| 47 | // 函数原型声明 |
| 48 | // |
| 49 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 50 | void GPIOInit(void);    // GPIO初始化 |
| 51 | void SysTickInit(void); // 设置SysTick中断 |
| 52 | void DevicesInit(void); // MCU器件初始化，注：会调用上述函数 |
| 53 |  |
| 54 | /\*\* |
| 55 | \* UI状态机相关函数 |
| 56 | \*/ |
| 57 |  |
| 58 | void ui\_state\_proc(uint16\_t ui\_state); |
| 59 | void ui\_proc0(void); |
| 60 | void ui\_proc005(void); |
| 61 | void ui\_proc100(void); |
| 62 | void ui\_proc101(void); |
| 63 | void ui\_proc102(void); |
| 64 | void ui\_proc201(void); |
| 65 | void ui\_proc202(void); |
| 66 | void ui\_proc203(void); |
| 67 | void ui\_proc301(void); |
| 68 | void ui\_proc303(void); |
| 69 | void ui\_proc305(void); |
| 70 | void ui\_proc306(void); |
| 71 | void ui\_proc4(void); |
| 72 | void ENTER\_detect(void); |
| 73 | void LEFT\_detect(void); |
| 74 | void RIGHT\_detect(void); |
| 75 | void INCREASE\_detect(void); |
| 76 | void DECREASE\_detect(void); |
| 77 |  |
| 78 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 79 | // |
| 80 | // 变量定义 |
| 81 | // |
| 82 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 83 |  |
| 84 | // 软件定时器计数 |
| 85 | uint8\_t clock100ms = 0; |
| 86 | uint8\_t clock500ms = 0; |
| 87 | uint8\_t clock2s = 0; |
| 88 | uint8\_t NOKEY\_clock5s = 0; |
| 89 | uint8\_t ACT4\_clock5s = 0; |
| 90 |  |
| 91 | // 软件定时器溢出标志 |
| 92 | uint8\_t clock100ms\_flag = 0; |
| 93 | uint8\_t clock500ms\_flag = 0; |
| 94 | uint8\_t clock2s\_flag = 0; |
| 95 | uint8\_t NOKEY\_clock5s\_flag = 0; |
| 96 | uint8\_t ACT4\_clock5s\_flag = 0; |
| 97 |  |
| 98 | /\*\* |
| 99 | \* 按键事件标志 |
| 100 | \*/ |
| 101 | uint8\_t key\_LEFT\_flag = 0; |
| 102 | uint8\_t key\_RIGHT\_flag = 0; |
| 103 | uint8\_t key\_INCREASE\_flag = 0; |
| 104 | uint8\_t key\_DECREASE\_flag = 0; |
| 105 | uint8\_t key\_ENTER\_flag = 0; |
| 106 |  |
| 107 | // 测试用计数器 |
| 108 | uint32\_t test\_counter = 0; |
| 109 |  |
| 110 | // 8位数码管显示的数字或字母符号 |
| 111 | // 注：板上数码位从左到右序号排列为4、5、6、7、0、1、2、3 |
| 112 | uint8\_t digit[8] = {' ', ' ', ' ', ' ', '\_', ' ', '\_', ' '}; |
| 113 |  |
| 114 | // 8位小数点 1亮  0灭 |
| 115 | // 注：板上数码位小数点从左到右序号排列为4、5、6、7、0、1、2、3 |
| 116 | uint8\_t pnt = 0x04; |
| 117 |  |
| 118 | // 8个LED指示灯状态，0灭，1亮 |
| 119 | // 注：板上指示灯从左到右序号排列为7、6、5、4、3、2、1、0 |
| 120 | //     对应元件LED8、LED7、LED6、LED5、LED4、LED3、LED2、LED1 |
| 121 | uint8\_t led[] = {1, 1, 1, 1, 1, 1, 1, 0}; |
| 122 |  |
| 123 | // 当前按键值 |
| 124 | uint8\_t key\_code = 0; |
| 125 | uint8\_t pre\_key\_code; // 上一按键值 |
| 126 |  |
| 127 | // 系统时钟频率 |
| 128 | uint32\_t ui32SysClock; |
| 129 |  |
| 130 | /\*\* |
| 131 | \* 用户界面（UI）状态机相关变量定义 |
| 132 | \*/ |
| 133 |  |
| 134 | uint16\_t ui\_state = 0x0; // 用户界面（UI）状态机当前状态 |
| 135 |  |
| 136 | /\*\* |
| 137 | \* @brief 状态参数结构体 |
| 138 | \* |
| 139 | \*/ |
| 140 | struct ACT\_T |
| 141 | { |
| 142 | uint8\_t row\_page[LCD\_MAX\_BLOCK];                      // 显示屏上每个分区的起始行页号 |
| 143 | uint8\_t col\_page[LCD\_MAX\_BLOCK];                      // 显示屏上每个分区的起始列页号 |
| 144 | unsigned char str[LCD\_MAX\_BLOCK][LCD\_MAX\_BLOCK\_CHAR]; // 显示屏上每个分区的显示内容 |
| 145 | const uint8\_t SIZE;                                   // 显示屏上有效分区的数量 |
| 146 | }; |
| 147 |  |
| 148 | struct ACT\_T act0 = { |
| 149 | {3, 3, 3, 3, 3, 7}, |
| 150 | {3, 11, 12, 13, 14, 1}, |
| 151 | {"模式A", "1", ".", "1", "Hz", "设置"}, |
| 152 | 6}; |
| 153 |  |
| 154 | struct ACT\_T act1 = { |
| 155 | {3, 5, 7}, |
| 156 | {3, 3, 13}, |
| 157 | {"工作模式", "工作参数", "返回"}, |
| 158 | 3}; |
| 159 |  |
| 160 | struct ACT\_T act2 = { |
| 161 | {3, 3, 7, 7}, |
| 162 | {1, 11, 1, 13}, |
| 163 | {"工作模式：", "模式A", "确定", "取消"}, |
| 164 | 4}; |
| 165 |  |
| 166 | struct ACT\_T act3 = { |
| 167 | {3, 3, 3, 3, 3, 7, 7}, |
| 168 | {1, 11, 12, 13, 14, 1, 13}, |
| 169 | {"工作参数：", "1", ".", "1", "Hz", "确定", "取消"}, |
| 170 | 7}; |
| 171 |  |
| 172 | struct ACT\_T act4 = { |
| 173 | {3}, |
| 174 | {1}, |
| 175 | {"工作参数不合法"}, |
| 176 | 1}; |
| 177 |  |
| 178 | struct ACT\_T \*act[] = {&act0, &act1, &act2, &act3, &act4}; |
| 179 |  |
| 180 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 181 | // |
| 182 | // 主程序 |
| 183 | // |
| 184 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 185 | int main(void) |
| 186 | { |
| 187 | uint8\_t temp, i; |
| 188 | DevicesInit(); //  MCU器件初始化 |
| 189 |  |
| 190 | while (clock100ms < 3) |
| 191 | ;           // 延时>60ms,等待TM1638上电完成 |
| 192 | TM1638\_Init();  // 初始化TM1638 |
| 193 | initial\_lcd();  // 初始化JLX12864 |
| 194 | clear\_screen(); //clear all dots |
| 195 |  |
| 196 | while (1) |
| 197 | { |
| 198 | if (clock100ms\_flag == 1) // 检查0.1秒定时是否到 |
| 199 | { |
| 200 | clock100ms\_flag = 0; |
| 201 | // 每0.1秒累加计时值在数码管上以十进制显示，有键按下时暂停计时 |
| 202 | if (key\_code == 0) |
| 203 | { |
| 204 | if (++test\_counter >= 10000) |
| 205 | test\_counter = 0; |
| 206 | digit[0] = test\_counter / 1000;     // 计算百位数 |
| 207 | digit[1] = test\_counter / 100 % 10; // 计算十位数 |
| 208 | digit[2] = test\_counter / 10 % 10;  // 计算个位数 |
| 209 | digit[3] = test\_counter % 10;       // 计算百分位数 |
| 210 | } |
| 211 | } |
| 212 |  |
| 213 | if (clock500ms\_flag == 1) // 检查0.5秒定时是否到 |
| 214 | { |
| 215 | clock500ms\_flag = 0; |
| 216 | // 8个指示灯以走马灯方式，每0.5秒向右（循环）移动一格 |
| 217 | temp = led[0]; |
| 218 | for (i = 0; i < 7; i++) |
| 219 | led[i] = led[i + 1]; |
| 220 | led[7] = temp; |
| 221 | } |
| 222 |  |
| 223 | ui\_state\_proc(ui\_state); |
| 224 | } |
| 225 | } |
| 226 |  |
| 227 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 228 | // |
| 229 | // 函数原型：void GPIOInit(void) |
| 230 | // 函数功能：GPIO初始化。使能PortK，设置PK4,PK5为输出；使能PortM，设置PM0为输出。 |
| 231 | //          （PK4连接TM1638的STB，PK5连接TM1638的DIO，PM0连接TM1638的CLK） |
| 232 | // 函数参数：无 |
| 233 | // 函数返回值：无 |
| 234 | // |
| 235 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 236 | void GPIOInit(void) |
| 237 | { |
| 238 | //配置TM1638芯片管脚 |
| 239 | SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOK); // 使能端口 K |
| 240 | while (!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOK)) |
| 241 | { |
| 242 | }; // 等待端口 K准备完毕 |
| 243 |  |
| 244 | SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOM); // 使能端口 M |
| 245 | while (!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOM)) |
| 246 | { |
| 247 | }; // 等待端口 M准备完毕 |
| 248 |  |
| 249 | // 设置端口 K的第4,5位（PK4,PK5）为输出引脚       PK4-STB  PK5-DIO |
| 250 | GPIOPinTypeGPIOOutput(GPIO\_PORTK\_BASE, GPIO\_PIN\_4 | GPIO\_PIN\_5); |
| 251 | // 设置端口 M的第0位（PM0）为输出引脚   PM0-CLK |
| 252 | GPIOPinTypeGPIOOutput(GPIO\_PORTM\_BASE, GPIO\_PIN\_0); |
| 253 | } |
| 254 |  |
| 255 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 256 | // |
| 257 | // 函数原型：SysTickInit(void) |
| 258 | // 函数功能：设置SysTick中断 |
| 259 | // 函数参数：无 |
| 260 | // 函数返回值：无 |
| 261 | // |
| 262 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 263 | void SysTickInit(void) |
| 264 | { |
| 265 | SysTickPeriodSet(ui32SysClock / SYSTICK\_FREQUENCY); // 设置心跳节拍,定时周期20ms |
| 266 | SysTickEnable();                                    // SysTick使能 |
| 267 | SysTickIntEnable();                                 // SysTick中断允许 |
| 268 | } |
| 269 |  |
| 270 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 271 | // |
| 272 | // 函数原型：void DevicesInit(void) |
| 273 | // 函数功能：CU器件初始化，包括系统时钟设置、GPIO初始化和SysTick中断设置 |
| 274 | // 函数参数：无 |
| 275 | // 函数返回值：无 |
| 276 | // |
| 277 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 278 | void DevicesInit(void) |
| 279 | { |
| 280 | // 使用外部25MHz主时钟源，经过PLL，然后分频为20MHz |
| 281 | ui32SysClock = SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN | |
| 282 | SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480), |
| 283 | 20000000); |
| 284 |  |
| 285 | GPIOInit();        // GPIO初始化 |
| 286 | SysTickInit();     // 设置SysTick中断 |
| 287 | IntMasterEnable(); // 总中断允许 |
| 288 | } |
| 289 |  |
| 290 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 291 | // |
| 292 | // 函数原型：void SysTick\_Handler(void) |
| 293 | // 函数功能：SysTick中断服务程序 |
| 294 | // 函数参数：无 |
| 295 | // 函数返回值：无 |
| 296 | // |
| 297 | //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| 298 | void SysTick\_Handler(void) // 定时周期为20ms |
| 299 | { |
| 300 | // 0.1秒钟软定时器计数 |
| 301 | if (++clock100ms >= V\_T100ms) |
| 302 | { |
| 303 | clock100ms\_flag = 1; // 当0.1秒到时，溢出标志置1 |
| 304 | clock100ms = 0; |
| 305 | } |
| 306 |  |
| 307 | // 0.5秒钟软定时器计数 |
| 308 | if (++clock500ms >= V\_T500ms) |
| 309 | { |
| 310 | clock500ms\_flag = 1; // 当0.5秒到时，溢出标志置1 |
| 311 | clock500ms = 0; |
| 312 | } |
| 313 |  |
| 314 | // 刷新全部数码管和LED指示灯 |
| 315 | TM1638\_RefreshDIGIandLED(digit, pnt, led); |
| 316 |  |
| 317 | // 检查当前键盘输入，0代表无键操作，1-9表示有对应按键 |
| 318 | // 键号显示在一位数码管上 |
| 319 | pre\_key\_code = key\_code;          // 保存上一按键值 |
| 320 | key\_code = TM1638\_Readkeyboard(); // 更新当前按键值 |
| 321 |  |
| 322 | digit[5] = key\_code; |
| 323 |  |
| 324 | ENTER\_detect(); |
| 325 | LEFT\_detect(); |
| 326 | RIGHT\_detect(); |
| 327 | INCREASE\_detect(); |
| 328 | DECREASE\_detect(); |
| 329 |  |
| 330 | // 5.0秒钟软定时器计数 |
| 331 | if (!key\_code && ++NOKEY\_clock5s >= V\_T5s) // 当无键按下时 |
| 332 | { |
| 333 | NOKEY\_clock5s\_flag = 1; // 当5.0秒到时，溢出标志置1 |
| 334 | NOKEY\_clock5s = 0; |
| 335 | } |
| 336 | // 若有键按下，则5.0秒计数清零 |
| 337 | if (key\_code) |
| 338 | { |
| 339 | NOKEY\_clock5s = 0; |
| 340 | } |
| 341 |  |
| 342 | // ACT4（警示画面）的5.0秒钟软定时器计数 |
| 343 | if (ui\_state == 0x4 && ++ACT4\_clock5s >= V\_T5s) // 当无键按下时 |
| 344 | { |
| 345 | ACT4\_clock5s\_flag = 1; // 当5.0秒到时，溢出标志置1 |
| 346 | ACT4\_clock5s = 0; |
| 347 | } |
| 348 | } |
| 349 |  |
| 350 | /\*\* |
| 351 | \* @brief UI状态机处理函数 |
| 352 | \* |
| 353 | \* @param ui\_state UI状态机当前状态 |
| 354 | \*/ |
| 355 | void ui\_state\_proc(uint16\_t ui\_state) |
| 356 | { |
| 357 | switch (ui\_state) |
| 358 | { |
| 359 | case 0x0: // ACT0 |
| 360 | ui\_proc0(); |
| 361 | break; |
| 362 | case 0x005: //ACT005 |
| 363 | ui\_proc005(); |
| 364 | break; |
| 365 | case 0x100: // ACT100 |
| 366 | ui\_proc100(); |
| 367 | break; |
| 368 | case 0x101: // ACT101 |
| 369 | ui\_proc101(); |
| 370 | break; |
| 371 | case 0x102: // ACT102 |
| 372 | ui\_proc102(); |
| 373 | break; |
| 374 | case 0x201: |
| 375 | ui\_proc201(); |
| 376 | break; |
| 377 | case 0x202: |
| 378 | ui\_proc202(); |
| 379 | break; |
| 380 | case 0x203: |
| 381 | ui\_proc203(); |
| 382 | break; |
| 383 | case 0x301: |
| 384 | ui\_proc301(); |
| 385 | break; |
| 386 | case 0x303: |
| 387 | ui\_proc303(); |
| 388 | break; |
| 389 | case 0x305: |
| 390 | ui\_proc305(); |
| 391 | break; |
| 392 | case 0x306: |
| 393 | ui\_proc306(); |
| 394 | break; |
| 395 | case 0x4: // ACT4 |
| 396 | ui\_proc4(); |
| 397 | break; |
| 398 | default: |
| 399 | ui\_state = 0x0; |
| 400 | break; |
| 401 | } |
| 402 | } |
| 403 |  |
| 404 | /\*\* |
| 405 | \* @brief UI状态机ACT0状态处理 |
| 406 | \* 开机初始画面，不显示光标 |
| 407 | \* |
| 408 | \*/ |
| 409 | void ui\_proc0(void) |
| 410 | { |
| 411 | uint8\_t i = 0; |
| 412 | // 显示开机初始画面，无光标 |
| 413 | for (i = 0; i < act[0]->SIZE; ++i) |
| 414 | { |
| 415 | display\_GB2312\_string(act[0]->row\_page[i], act[0]->col\_page[i] \* 8 - 7, act[0]->str[i], 0); |
| 416 | } |
| 417 |  |
| 418 | // 当有任意按键被按下："设置"做反白效果（光标），转移至状态ACT005 |
| 419 | if (!pre\_key\_code && key\_code) |
| 420 | { |
| 421 | key\_LEFT\_flag = key\_RIGHT\_flag = key\_INCREASE\_flag = key\_DECREASE\_flag = key\_ENTER\_flag = 0; |
| 422 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 1); |
| 423 | ui\_state = 0x005; |
| 424 | } |
| 425 | } |
| 426 |  |
| 427 | /\*\* |
| 428 | \* @brief UI状态机ACT005状态处理 |
| 429 | \* 光标在"设置"的位置 |
| 430 | \* |
| 431 | \*/ |
| 432 | void ui\_proc005(void) |
| 433 | { |
| 434 | uint8\_t i = 0; |
| 435 | // 当"确定"键按下：光标移到ACT1的"工作参数"，下一状态ACT100 |
| 436 | if (key\_ENTER\_flag) |
| 437 | { |
| 438 | key\_ENTER\_flag = 0; |
| 439 | // 清屏，显示ACT1的画面 |
| 440 | clear\_screen(); |
| 441 | for (i = 0; i < act[1]->SIZE; ++i) |
| 442 | { |
| 443 | display\_GB2312\_string(act[1]->row\_page[i], act[1]->col\_page[i] \* 8 - 7, act[1]->str[i], 0); |
| 444 | } |
| 445 | // 显示光标 |
| 446 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 1); |
| 447 | ui\_state = 0x100; |
| 448 | } |
| 449 | else if (key\_LEFT\_flag || key\_RIGHT\_flag || key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 450 | { |
| 451 | key\_LEFT\_flag = key\_RIGHT\_flag = key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 452 | } |
| 453 |  |
| 454 | // 当5秒无操作："设置"反白效果解除，下一状态ACT0 |
| 455 | if (NOKEY\_clock5s\_flag) |
| 456 | { |
| 457 | NOKEY\_clock5s\_flag = 0; |
| 458 | display\_GB2312\_string(act[0]->row\_page[5], act[0]->col\_page[5] \* 8 - 7, act[0]->str[5], 0); |
| 459 | ui\_state = 0x0; |
| 460 | } |
| 461 | } |
| 462 |  |
| 463 | /\*\* |
| 464 | \* @brief UI状态机ACT100状态处理 |
| 465 | \* 光标在"工作模式"的位置 |
| 466 | \* |
| 467 | \*/ |
| 468 | void ui\_proc100(void) |
| 469 | { |
| 470 | uint8\_t i = 0; |
| 471 | // 当"左"键按下：光标移到"返回"，下一状态ACT102 |
| 472 | if (key\_LEFT\_flag) |
| 473 | { |
| 474 | key\_LEFT\_flag = 0; |
| 475 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 0); |
| 476 | display\_GB2312\_string(act[1]->row\_page[2], act[1]->col\_page[2] \* 8 - 7, act[1]->str[2], 1); |
| 477 | ui\_state = 0x102; |
| 478 | } |
| 479 | // 当"右"键按下：光标移到"工作参数"，下一状态ACT101 |
| 480 | else if (key\_RIGHT\_flag) |
| 481 | { |
| 482 | key\_RIGHT\_flag = 0; |
| 483 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 0); |
| 484 | display\_GB2312\_string(act[1]->row\_page[1], act[1]->col\_page[1] \* 8 - 7, act[1]->str[1], 1); |
| 485 | ui\_state = 0x101; |
| 486 | } |
| 487 | // 当"确定"键按下：光标移到ACT2的"模式#"，下一状态ACT201 |
| 488 | else if (key\_ENTER\_flag) |
| 489 | { |
| 490 | key\_ENTER\_flag = 0; |
| 491 | // 清屏，显示ACT2的画面 |
| 492 | clear\_screen(); |
| 493 | for (i = 0; i < act[2]->SIZE; ++i) |
| 494 | { |
| 495 | display\_GB2312\_string(act[2]->row\_page[i], act[2]->col\_page[i] \* 8 - 7, act[2]->str[i], 0); |
| 496 | } |
| 497 | // 显示光标 |
| 498 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 1); |
| 499 | ui\_state = 0x201; |
| 500 | } |
| 501 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 502 | { |
| 503 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 504 | } |
| 505 | } |
| 506 |  |
| 507 | /\*\* |
| 508 | \* @brief UI状态机ACT101状态处理 |
| 509 | \* 光标在"工作参数"的位置 |
| 510 | \* |
| 511 | \*/ |
| 512 | void ui\_proc101(void) |
| 513 | { |
| 514 | uint8\_t i = 0; |
| 515 | // 当"左"键按下：光标移到"工作模式"，下一状态ACT100 |
| 516 | if (key\_LEFT\_flag) |
| 517 | { |
| 518 | key\_LEFT\_flag = 0; |
| 519 | display\_GB2312\_string(act[1]->row\_page[1], act[1]->col\_page[1] \* 8 - 7, act[1]->str[1], 0); |
| 520 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 1); |
| 521 | ui\_state = 0x100; |
| 522 | } |
| 523 | // 当"右"键按下：光标移到"返回"，下一状态ACT102 |
| 524 | else if (key\_RIGHT\_flag) |
| 525 | { |
| 526 | key\_RIGHT\_flag = 0; |
| 527 | display\_GB2312\_string(act[1]->row\_page[1], act[1]->col\_page[1] \* 8 - 7, act[1]->str[1], 0); |
| 528 | display\_GB2312\_string(act[1]->row\_page[2], act[1]->col\_page[2] \* 8 - 7, act[1]->str[2], 1); |
| 529 | ui\_state = 0x102; |
| 530 | } |
| 531 | // 当"确定"键按下：光标移到ACT3的工作参数个位的位置，下一状态ACT301 |
| 532 | else if (key\_ENTER\_flag) |
| 533 | { |
| 534 | key\_ENTER\_flag = 0; |
| 535 | // 清屏，显示ACT3的画面 |
| 536 | clear\_screen(); |
| 537 | for (i = 0; i < act[3]->SIZE; ++i) |
| 538 | { |
| 539 | display\_GB2312\_string(act[3]->row\_page[i], act[3]->col\_page[i] \* 8 - 7, act[3]->str[i], 0); |
| 540 | } |
| 541 | // 显示光标 |
| 542 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 1); |
| 543 | ui\_state = 0x301; |
| 544 | } |
| 545 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 546 | { |
| 547 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 548 | } |
| 549 | } |
| 550 |  |
| 551 | /\*\* |
| 552 | \* @brief UI状态机ACT102状态处理 |
| 553 | \* 光标在"返回"的位置 |
| 554 | \* |
| 555 | \*/ |
| 556 | void ui\_proc102(void) |
| 557 | { |
| 558 | uint8\_t i = 0; |
| 559 | // 当"左"键按下：光标移到"工作参数"，下一状态ACT101 |
| 560 | if (key\_LEFT\_flag) |
| 561 | { |
| 562 | key\_LEFT\_flag = 0; |
| 563 | display\_GB2312\_string(act[1]->row\_page[2], act[1]->col\_page[2] \* 8 - 7, act[1]->str[2], 0); |
| 564 | display\_GB2312\_string(act[1]->row\_page[1], act[1]->col\_page[1] \* 8 - 7, act[1]->str[1], 1); |
| 565 | ui\_state = 0x101; |
| 566 | } |
| 567 | // 当"右"键按下：光标移到"工作模式"，下一状态ACT100 |
| 568 | else if (key\_RIGHT\_flag) |
| 569 | { |
| 570 | key\_RIGHT\_flag = 0; |
| 571 | display\_GB2312\_string(act[1]->row\_page[2], act[1]->col\_page[2] \* 8 - 7, act[1]->str[2], 0); |
| 572 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 1); |
| 573 | ui\_state = 0x100; |
| 574 | } |
| 575 | // 当"确定"键按下：显示开机初始画面，下一状态ACT0 |
| 576 | else if (key\_ENTER\_flag) |
| 577 | { |
| 578 | key\_ENTER\_flag = 0; |
| 579 | // 清屏，显示ACT0的画面 |
| 580 | clear\_screen(); |
| 581 | for (i = 0; i < act[0]->SIZE; ++i) |
| 582 | { |
| 583 | display\_GB2312\_string(act[0]->row\_page[i], act[0]->col\_page[i] \* 8 - 7, act[0]->str[i], 0); |
| 584 | } |
| 585 | ui\_state = 0x0; |
| 586 | } |
| 587 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 588 | { |
| 589 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 590 | } |
| 591 | } |
| 592 |  |
| 593 | /\*\* |
| 594 | \* @brief UI状态机ACT201状态处理 |
| 595 | \* 光标在"模式#"的位置 |
| 596 | \* |
| 597 | \*/ |
| 598 | void ui\_proc201(void) |
| 599 | { |
| 600 | // 当"左"键按下：光标移到"取消"，下一状态ACT203 |
| 601 | if (key\_LEFT\_flag) |
| 602 | { |
| 603 | key\_LEFT\_flag = 0; |
| 604 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 0); |
| 605 | display\_GB2312\_string(act[2]->row\_page[3], act[2]->col\_page[3] \* 8 - 7, act[2]->str[3], 1); |
| 606 | ui\_state = 0x203; |
| 607 | } |
| 608 | // 当"右"键按下：光标移到"确定"，下一状态ACT202 |
| 609 | else if (key\_RIGHT\_flag) |
| 610 | { |
| 611 | key\_RIGHT\_flag = 0; |
| 612 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 0); |
| 613 | display\_GB2312\_string(act[2]->row\_page[2], act[2]->col\_page[2] \* 8 - 7, act[2]->str[2], 1); |
| 614 | ui\_state = 0x202; |
| 615 | } |
| 616 | // 当"+"键按下："模式#"按A、B、C、A正序循环切换，留在本状态 |
| 617 | else if (key\_INCREASE\_flag) |
| 618 | { |
| 619 | key\_INCREASE\_flag = 0; |
| 620 | if (++((act[2]->str[1])[strlen((const char \*)(act[2]->str[1])) - 1]) > 'C') |
| 621 | { |
| 622 | (act[2]->str[1])[strlen((const char \*)(act[2]->str[1])) - 1] = 'A'; |
| 623 | } |
| 624 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 1); |
| 625 | } |
| 626 | // 当"-"键按下："模式#"按C、B、A、C逆序循环切换，留在本状态 |
| 627 | else if (key\_DECREASE\_flag) |
| 628 | { |
| 629 | key\_DECREASE\_flag = 0; |
| 630 | if (--((act[2]->str[1])[strlen((const char \*)(act[2]->str[1])) - 1]) < 'A') |
| 631 | { |
| 632 | (act[2]->str[1])[strlen((const char \*)(act[2]->str[1])) - 1] = 'C'; |
| 633 | } |
| 634 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 1); |
| 635 | } |
| 636 | else if (key\_ENTER\_flag) |
| 637 | { |
| 638 | key\_ENTER\_flag = 0; |
| 639 | } |
| 640 | } |
| 641 |  |
| 642 | /\*\* |
| 643 | \* @brief UI状态机ACT202状态处理 |
| 644 | \* 光标在"确定"的位置 |
| 645 | \* |
| 646 | \*/ |
| 647 | void ui\_proc202(void) |
| 648 | { |
| 649 | uint8\_t i = 0; |
| 650 | // 当"左"键按下：光标移到"模式#"，下一状态ACT201 |
| 651 | if (key\_LEFT\_flag) |
| 652 | { |
| 653 | key\_LEFT\_flag = 0; |
| 654 | display\_GB2312\_string(act[2]->row\_page[2], act[2]->col\_page[2] \* 8 - 7, act[2]->str[2], 0); |
| 655 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 1); |
| 656 | ui\_state = 0x201; |
| 657 | } |
| 658 | // 当"右"键按下：光标移到"取消"，下一状态ACT203 |
| 659 | else if (key\_RIGHT\_flag) |
| 660 | { |
| 661 | key\_RIGHT\_flag = 0; |
| 662 | display\_GB2312\_string(act[2]->row\_page[2], act[2]->col\_page[2] \* 8 - 7, act[2]->str[2], 0); |
| 663 | display\_GB2312\_string(act[2]->row\_page[3], act[2]->col\_page[3] \* 8 - 7, act[2]->str[3], 1); |
| 664 | ui\_state = 0x203; |
| 665 | } |
| 666 | // 当"确定"键按下：将当前更改同步到ACT0画面的"模式#"，光标移到ACT1的"工作模式"，下一状态ACT100 |
| 667 | else if (key\_ENTER\_flag) |
| 668 | { |
| 669 | key\_ENTER\_flag = 0; |
| 670 | // 将当前更改同步到ACT0画面的"模式#" |
| 671 | strcpy((char \*)(act[0]->str[0]), (const char \*)(act[2]->str[1])); |
| 672 | // 清屏，显示ACT1的画面 |
| 673 | clear\_screen(); |
| 674 | for (i = 0; i < act[1]->SIZE; ++i) |
| 675 | { |
| 676 | display\_GB2312\_string(act[1]->row\_page[i], act[1]->col\_page[i] \* 8 - 7, act[1]->str[i], 0); |
| 677 | } |
| 678 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 1); |
| 679 | ui\_state = 0x100; |
| 680 | } |
| 681 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 682 | { |
| 683 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 684 | } |
| 685 | } |
| 686 |  |
| 687 | /\*\* |
| 688 | \* @brief UI状态机ACT203状态处理 |
| 689 | \* 光标在"取消"的位置 |
| 690 | \* |
| 691 | \*/ |
| 692 | void ui\_proc203(void) |
| 693 | { |
| 694 | uint8\_t i = 0; |
| 695 | // 当"左"键按下：光标移到"确定"，下一状态ACT202 |
| 696 | if (key\_LEFT\_flag) |
| 697 | { |
| 698 | key\_LEFT\_flag = 0; |
| 699 | display\_GB2312\_string(act[2]->row\_page[3], act[2]->col\_page[3] \* 8 - 7, act[2]->str[3], 0); |
| 700 | display\_GB2312\_string(act[2]->row\_page[2], act[2]->col\_page[2] \* 8 - 7, act[2]->str[2], 1); |
| 701 | ui\_state = 0x202; |
| 702 | } |
| 703 | // 当"右"键按下：光标移到"模式#"，下一状态ACT201 |
| 704 | else if (key\_RIGHT\_flag) |
| 705 | { |
| 706 | key\_RIGHT\_flag = 0; |
| 707 | display\_GB2312\_string(act[2]->row\_page[3], act[2]->col\_page[3] \* 8 - 7, act[2]->str[3], 0); |
| 708 | display\_GB2312\_string(act[2]->row\_page[1], act[2]->col\_page[1] \* 8 - 7, act[2]->str[1], 1); |
| 709 | ui\_state = 0x201; |
| 710 | } |
| 711 | // 当"确定"键按下：撤销对"模式#"的更改，光标移到ACT1的"工作模式"，下一状态ACT100 |
| 712 | else if (key\_ENTER\_flag) |
| 713 | { |
| 714 | key\_ENTER\_flag = 0; |
| 715 | // 撤销对"模式#"的更改 |
| 716 | strcpy((char \*)(act[2]->str[1]), (const char \*)(act[0]->str[0])); |
| 717 | // 清屏，显示ACT1的画面，光标移到ACT1的"工作模式" |
| 718 | clear\_screen(); |
| 719 | for (i = 0; i < act[1]->SIZE; ++i) |
| 720 | { |
| 721 | display\_GB2312\_string(act[1]->row\_page[i], act[1]->col\_page[i] \* 8 - 7, act[1]->str[i], 0); |
| 722 | } |
| 723 | display\_GB2312\_string(act[1]->row\_page[0], act[1]->col\_page[0] \* 8 - 7, act[1]->str[0], 1); |
| 724 | ui\_state = 0x100; |
| 725 | } |
| 726 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 727 | { |
| 728 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 729 | } |
| 730 | } |
| 731 |  |
| 732 | /\*\* |
| 733 | \* @brief UI状态机ACT301状态处理 |
| 734 | \* 光标在工作参数个位的位置 |
| 735 | \* |
| 736 | \*/ |
| 737 | void ui\_proc301(void) |
| 738 | { |
| 739 | // 当"左"键按下：光标移到"取消"，下一状态ACT306 |
| 740 | if (key\_LEFT\_flag) |
| 741 | { |
| 742 | key\_LEFT\_flag = 0; |
| 743 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 0); |
| 744 | display\_GB2312\_string(act[3]->row\_page[6], act[3]->col\_page[6] \* 8 - 7, act[3]->str[6], 1); |
| 745 | ui\_state = 0x306; |
| 746 | } |
| 747 | // 当"右"键按下：光标移到工作参数十分位的位置，下一状态ACT303 |
| 748 | else if (key\_RIGHT\_flag) |
| 749 | { |
| 750 | key\_RIGHT\_flag = 0; |
| 751 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 0); |
| 752 | display\_GB2312\_string(act[3]->row\_page[3], act[3]->col\_page[3] \* 8 - 7, act[3]->str[3], 1); |
| 753 | ui\_state = 0x303; |
| 754 | } |
| 755 | // 当"+"键按下：个位数按1、2、...、9、0、1、...正序循环切换，留在本状态 |
| 756 | else if (key\_INCREASE\_flag) |
| 757 | { |
| 758 | key\_INCREASE\_flag = 0; |
| 759 | if (++((act[3]->str[1])[0]) > '9') |
| 760 | { |
| 761 | (act[3]->str[1])[0] = '0'; |
| 762 | } |
| 763 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 1); |
| 764 | } |
| 765 | // 当"-"键按下：个位数按9、8、...、0、9、8、...逆序循环切换，留在本状态 |
| 766 | else if (key\_DECREASE\_flag) |
| 767 | { |
| 768 | key\_DECREASE\_flag = 0; |
| 769 | if (--((act[3]->str[1])[0]) < '0') |
| 770 | { |
| 771 | (act[3]->str[1])[0] = '9'; |
| 772 | } |
| 773 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 1); |
| 774 | } |
| 775 | else if (key\_ENTER\_flag) |
| 776 | { |
| 777 | key\_ENTER\_flag = 0; |
| 778 | } |
| 779 | } |
| 780 |  |
| 781 | /\*\* |
| 782 | \* @brief UI状态机ACT303状态处理 |
| 783 | \* 光标在工作参数十分位的位置 |
| 784 | \* |
| 785 | \*/ |
| 786 | void ui\_proc303(void) |
| 787 | { |
| 788 | // 当"左"键按下：光标移到工作参数个位的位置，下一状态ACT301 |
| 789 | if (key\_LEFT\_flag) |
| 790 | { |
| 791 | key\_LEFT\_flag = 0; |
| 792 | display\_GB2312\_string(act[3]->row\_page[3], act[3]->col\_page[3] \* 8 - 7, act[3]->str[3], 0); |
| 793 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 1); |
| 794 | ui\_state = 0x301; |
| 795 | } |
| 796 | // 当"右"键按下：光标移到"确定"，下一状态ACT305 |
| 797 | else if (key\_RIGHT\_flag) |
| 798 | { |
| 799 | key\_RIGHT\_flag = 0; |
| 800 | display\_GB2312\_string(act[3]->row\_page[3], act[3]->col\_page[3] \* 8 - 7, act[3]->str[3], 0); |
| 801 | display\_GB2312\_string(act[3]->row\_page[5], act[3]->col\_page[5] \* 8 - 7, act[3]->str[5], 1); |
| 802 | ui\_state = 0x305; |
| 803 | } |
| 804 | // 当"+"键按下：十分位数按1、2、...、9、0、1、...正序循环切换，留在本状态 |
| 805 | else if (key\_INCREASE\_flag) |
| 806 | { |
| 807 | key\_INCREASE\_flag = 0; |
| 808 | if (++((act[3]->str[3])[0]) > '9') |
| 809 | { |
| 810 | (act[3]->str[3])[0] = '0'; |
| 811 | } |
| 812 | display\_GB2312\_string(act[3]->row\_page[3], act[3]->col\_page[3] \* 8 - 7, act[3]->str[3], 1); |
| 813 | } |
| 814 | // 当"-"键按下：十分位数按9、8、...、0、9、8、...逆序循环切换，留在本状态 |
| 815 | else if (key\_DECREASE\_flag) |
| 816 | { |
| 817 | key\_DECREASE\_flag = 0; |
| 818 | if (--((act[3]->str[3])[0]) < '0') |
| 819 | { |
| 820 | (act[3]->str[3])[0] = '9'; |
| 821 | } |
| 822 | display\_GB2312\_string(act[3]->row\_page[3], act[3]->col\_page[3] \* 8 - 7, act[3]->str[3], 1); |
| 823 | } |
| 824 | else if (key\_ENTER\_flag) |
| 825 | { |
| 826 | key\_ENTER\_flag = 0; |
| 827 | } |
| 828 | } |
| 829 |  |
| 830 | /\*\* |
| 831 | \* @brief UI状态机ACT305状态处理 |
| 832 | \* 光标在"确定"的位置 |
| 833 | \* |
| 834 | \*/ |
| 835 | void ui\_proc305(void) |
| 836 | { |
| 837 | // 当"左"键按下：光标移到工作参数十分位的位置，下一状态ACT303 |
| 838 | if (key\_LEFT\_flag) |
| 839 | { |
| 840 | key\_LEFT\_flag = 0; |
| 841 | display\_GB2312\_string(act[3]->row\_page[5], act[3]->col\_page[5] \* 8 - 7, act[3]->str[5], 0); |
| 842 | display\_GB2312\_string(act[3]->row\_page[3], act[3]->col\_page[3] \* 8 - 7, act[3]->str[3], 1); |
| 843 | ui\_state = 0x303; |
| 844 | } |
| 845 | // 当"右"键按下：光标移到"取消"，下一状态ACT306 |
| 846 | else if (key\_RIGHT\_flag) |
| 847 | { |
| 848 | key\_RIGHT\_flag = 0; |
| 849 | display\_GB2312\_string(act[3]->row\_page[5], act[3]->col\_page[5] \* 8 - 7, act[3]->str[5], 0); |
| 850 | display\_GB2312\_string(act[3]->row\_page[6], act[3]->col\_page[6] \* 8 - 7, act[3]->str[6], 1); |
| 851 | ui\_state = 0x306; |
| 852 | } |
| 853 | // 当"确定"键按下：检查参数值合法性，若非法，则显示警示画面ACT4，下一状态ACT4；若合法，则将当前更改同步到ACT0画面，光标移到ACT1的"工作参数"，下一状态ACT101 |
| 854 | else if (key\_ENTER\_flag) |
| 855 | { |
| 856 | uint8\_t i; |
| 857 | uint8\_t num = ((act[3]->str[1])[0] - '0') \* 10 + (act[3]->str[3])[0] - '0'; |
| 858 |  |
| 859 | key\_ENTER\_flag = 0; |
| 860 | // 非法参数：显示警示画面ACT4，下一状态ACT4 |
| 861 | if (num < 10 || num > 90) |
| 862 | { |
| 863 | // 显示ACT4的画面，下一状态ACT4 |
| 864 | clear\_screen(); |
| 865 | for (i = 0; i < act[4]->SIZE; ++i) |
| 866 | { |
| 867 | display\_GB2312\_string(act[4]->row\_page[i], act[4]->col\_page[i] \* 8 - 7, act[4]->str[i], 0); |
| 868 | } |
| 869 | ACT4\_clock5s\_flag = ACT4\_clock5s = 0; // ACT4计数器初始化 |
| 870 | ui\_state = 0x4; |
| 871 | } |
| 872 | // 合法参数：将当前更改同步到ACT0画面，显示ACT1的画面，光标移到ACT1的"工作参数"，下一状态ACT101 |
| 873 | else |
| 874 | { |
| 875 | strcpy((char \*)(act[0]->str[1]), (const char \*)(act[3]->str[1])); |
| 876 | strcpy((char \*)(act[0]->str[3]), (const char \*)(act[3]->str[3])); |
| 877 | clear\_screen(); |
| 878 | for (i = 0; i < act[1]->SIZE; ++i) |
| 879 | { |
| 880 | display\_GB2312\_string(act[1]->row\_page[i], act[1]->col\_page[i] \* 8 - 7, act[1]->str[i], 0); |
| 881 | } |
| 882 | display\_GB2312\_string(act[1]->row\_page[1], act[1]->col\_page[1] \* 8 - 7, act[1]->str[1], 1); |
| 883 | ui\_state = 0x101; |
| 884 | } |
| 885 | } |
| 886 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 887 | { |
| 888 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 889 | } |
| 890 | } |
| 891 |  |
| 892 | /\*\* |
| 893 | \* @brief UI状态机ACT306状态处理 |
| 894 | \* 光标在"取消"的位置 |
| 895 | \* |
| 896 | \*/ |
| 897 | void ui\_proc306(void) |
| 898 | { |
| 899 | // 当"左"键按下：光标移到"确定"，下一状态ACT305 |
| 900 | if (key\_LEFT\_flag) |
| 901 | { |
| 902 | key\_LEFT\_flag = 0; |
| 903 | display\_GB2312\_string(act[3]->row\_page[6], act[3]->col\_page[6] \* 8 - 7, act[3]->str[6], 0); |
| 904 | display\_GB2312\_string(act[3]->row\_page[5], act[3]->col\_page[5] \* 8 - 7, act[3]->str[5], 1); |
| 905 | ui\_state = 0x305; |
| 906 | } |
| 907 | // 当"右"键按下：光标移到工作参数个位的位置，下一状态ACT301 |
| 908 | else if (key\_RIGHT\_flag) |
| 909 | { |
| 910 | key\_RIGHT\_flag = 0; |
| 911 | display\_GB2312\_string(act[3]->row\_page[6], act[3]->col\_page[6] \* 8 - 7, act[3]->str[6], 0); |
| 912 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 1); |
| 913 | ui\_state = 0x301; |
| 914 | } |
| 915 | // 当"确定"键按下：撤销对工作参数的更改，光标移到ACT1的"工作参数"，下一状态ACT101 |
| 916 | else if (key\_ENTER\_flag) |
| 917 | { |
| 918 | uint8\_t i = 0; |
| 919 |  |
| 920 | key\_ENTER\_flag = 0; |
| 921 | // 撤销对工作参数的更改 |
| 922 | strcpy((char \*)(act[3]->str[1]), (const char \*)(act[0]->str[1])); |
| 923 | strcpy((char \*)(act[3]->str[3]), (const char \*)(act[0]->str[3])); |
| 924 | // 显示ACT1的画面，光标移到ACT1的"工作参数"，下一状态ACT101 |
| 925 | clear\_screen(); |
| 926 | for (i = 0; i < act[1]->SIZE; ++i) |
| 927 | { |
| 928 | display\_GB2312\_string(act[1]->row\_page[i], act[1]->col\_page[i] \* 8 - 7, act[1]->str[i], 0); |
| 929 | } |
| 930 | display\_GB2312\_string(act[1]->row\_page[1], act[1]->col\_page[1] \* 8 - 7, act[1]->str[1], 1); |
| 931 | ui\_state = 0x101; |
| 932 | } |
| 933 | else if (key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 934 | { |
| 935 | key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 936 | } |
| 937 | } |
| 938 |  |
| 939 | /\*\* |
| 940 | \* @brief UI状态机ACT4状态处理 |
| 941 | \* 显示警示画面，维持5秒 |
| 942 | \* |
| 943 | \*/ |
| 944 | void ui\_proc4(void) |
| 945 | { |
| 946 | uint8\_t i = 0; |
| 947 | // 五秒时间到：显示ACT3的界面，光标移到ACT3的工作参数个位的位置，下一状态ACT301 |
| 948 | if (ACT4\_clock5s\_flag) |
| 949 | { |
| 950 | ACT4\_clock5s\_flag = 0; |
| 951 | // 光标移到ACT3的工作参数个位的位置，下一状态ACT301 |
| 952 | clear\_screen(); |
| 953 | for (i = 0; i < act[3]->SIZE; ++i) |
| 954 | { |
| 955 | display\_GB2312\_string(act[3]->row\_page[i], act[3]->col\_page[i] \* 8 - 7, act[3]->str[i], 0); |
| 956 | } |
| 957 | display\_GB2312\_string(act[3]->row\_page[1], act[3]->col\_page[1] \* 8 - 7, act[3]->str[1], 1); |
| 958 | ui\_state = 0x301; |
| 959 | } |
| 960 |  |
| 961 | if (key\_ENTER\_flag || key\_LEFT\_flag || key\_RIGHT\_flag || key\_INCREASE\_flag || key\_DECREASE\_flag) |
| 962 | { |
| 963 | key\_ENTER\_flag = key\_LEFT\_flag = key\_RIGHT\_flag = key\_DECREASE\_flag = key\_INCREASE\_flag = 0; |
| 964 | } |
| 965 | } |
| 966 |  |
| 967 | /\*\* |
| 968 | \* @brief "确定"键按键检测 |
| 969 | \* |
| 970 | \*/ |
| 971 | void ENTER\_detect(void) |
| 972 | { |
| 973 | if (pre\_key\_code != 5 && key\_code == 5) |
| 974 | { |
| 975 | key\_ENTER\_flag = 1; |
| 976 | } |
| 977 | } |
| 978 |  |
| 979 | /\*\* |
| 980 | \* @brief "+"键按键检测 |
| 981 | \* |
| 982 | \*/ |
| 983 | void INCREASE\_detect(void) |
| 984 | { |
| 985 | if (pre\_key\_code != 2 && key\_code == 2) |
| 986 | { |
| 987 | key\_INCREASE\_flag = 1; |
| 988 | } |
| 989 | } |
| 990 |  |
| 991 | /\*\* |
| 992 | \* @brief "-"键按键检测 |
| 993 | \* |
| 994 | \*/ |
| 995 | void DECREASE\_detect(void) |
| 996 | { |
| 997 | if (pre\_key\_code != 8 && key\_code == 8) |
| 998 | { |
| 999 | key\_DECREASE\_flag = 1; |
| 1000 | } |
| 1001 | } |
| 1002 |  |
| 1003 | /\*\* |
| 1004 | \* @brief "左"键按键检测 |
| 1005 | \* |
| 1006 | \*/ |
| 1007 | void LEFT\_detect(void) |
| 1008 | { |
| 1009 | if (pre\_key\_code != 4 && key\_code == 4) |
| 1010 | { |
| 1011 | key\_LEFT\_flag = 1; |
| 1012 | } |
| 1013 | } |
| 1014 |  |
| 1015 | /\*\* |
| 1016 | \* @brief "右"键按键检测 |
| 1017 | \* |
| 1018 | \*/ |
| 1019 | void RIGHT\_detect(void) |
| 1020 | { |
| 1021 | if (pre\_key\_code != 6 && key\_code == 6) |
| 1022 | { |
| 1023 | key\_RIGHT\_flag = 1; |
| 1024 | } |
| 1025 | } |
| 1026 |  |