工程实践与科技创新 II-A
（信工电科教学班）
实验报告

实验 4　点阵字符显示和用户操作界面设计

实验 5　发送-接收联合系统实验

组号：03B

成员姓名：危国锐(组长)、于严谦

完成时间：2021 年 6 月 19 日

上海交通大学 电子信息与电气工程学院

目　　录

1. 实验 4 点阵字符显示和用户操作界面设计

1.1 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。

1.2 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。

1.3 实验任务技术解决方案的简要说明

1.3.1 实验任务 4_1

图 1 是关于本任务的程序流程示意图。

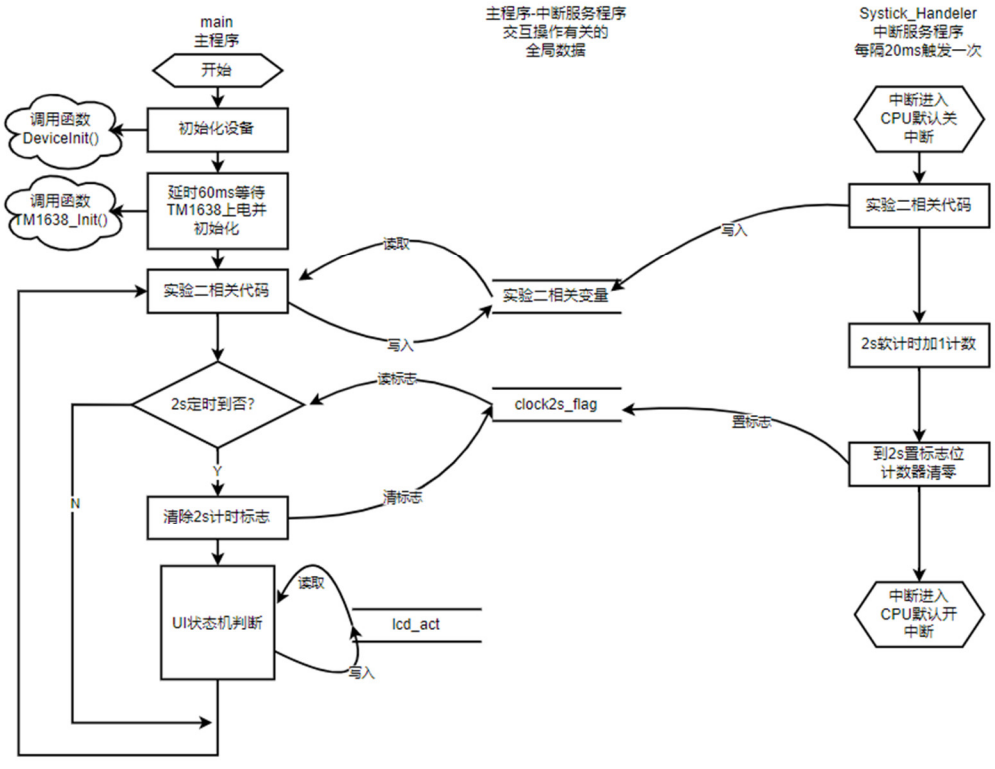

图 1 关于实验任务 4_1 的程序流程示意图

1.3.2 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
```

```
 * 光标在工作参数十分位的位置
 *
 */
void ui_proc005(void)
{
    // 当"右"键按下：光标移到"模式#"，下一状态 ACT001
    if (key_RIGHT_flag)
    {
        key_RIGHT_flag = 0;
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 0);
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 1);
        ui_state = 0x001;
    }
    // 当"左"键按下：光标移到个位，下一状态 ACT003
    else if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 0);
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 1);
        ui_state = 0x003;
    }
    // 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
    else if (key_INCREASE_flag)
    {
        key_INCREASE_flag = 0;
        if (++((act[0]->str[5])[0]) > '9')
        {
            (act[0]->str[5])[0] = '0';
        }
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 1);
    }
    // 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
    else if (key_DECREASE_flag)
    {
        key_DECREASE_flag = 0;
        if (--((act[0]->str[5])[0]) < '0')
        {
            (act[0]->str[5])[0] = '9';
        }
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 1);
    }
    else if (key_ENTER_flag)
    {
        key_ENTER_flag = 0;
    }

    // 当 10 秒无操作：工作参数十分位反白效果解除，下一状态 ACT0
    if (NOKEY_clock10s_flag)
    {
        NOKEY_clock10s_flag = 0;
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 0);
        ui_state = 0x0;
    }
}

```

1.3.3 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

1.4 实验核心代码清单

1.4.1 实验任务 4_1

```
/**
 * @file exp4_1.c
 * @author 上海交通大学电子工程系实验教学中心; Guorui Wei (313017602@qq.com)
 * @brief 实验 4_1
 * @version 0.1
 * @date 2021-06-05
 *
 * @copyright 2020-2021, 上海交通大学电子工程系实验教学中心
 *
 */

//*****************************************************************************
//
// 头文件
//
//*****************************************************************************
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"        // 基址宏定义
#include "inc/hw_types.h"         // 数据类型宏定义，寄存器访问函数
#include "driverlib/debug.h"      // 调试用
#include "driverlib/gpio.h"       // 通用 IO 口宏定义
#include "driverlib/pin_map.h"    // TM4C 系列 MCU 外围设备管脚宏定义
#include "driverlib/sysctl.h"     // 系统控制定义
#include "driverlib/systick.h"    // SysTick Driver 原型
#include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型
#include "JLX12864.h"             // 与控制 JLX12864G 有关的函数
#include "tm1638.h"               // 与控制 TM1638 芯片有关的函数
//*****************************************************************************
//
// 宏定义
//
//*****************************************************************************
```

```
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, "麈麒麈麝麟黛黜黝", true);
188                 display_GB2312_string(5, 1, "點黟黡黩黧黥黪黯", false);
189                 display_GB2312_string(7, 1, "黝鼬鼯鼹鼷鼽鼾齄", 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
```

```
// 函数参数：无
// 函数返回值：无
//
//*****************************************************************************
void DevicesInit(void)
{
    // 使用外部 25MHz 主时钟源，经过 PLL，然后分频为 20MHz
    ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
                                       SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480),
                                      20000000);

    GPIOInit();         // GPIO 初始化
    SysTickInit();      // 设置 SysTick 中断
    IntMasterEnable(); // 总中断允许
}

//*****************************************************************************
//
// 函数原型：void SysTick_Handler(void)
// 函数功能：SysTick 中断服务程序
// 函数参数：无
// 函数返回值：无
//
//*****************************************************************************
void SysTick_Handler(void) // 定时周期为 20ms
{
    // 0.1 秒钟软定时器计数
    if (++clock100ms >= V_T100ms)
    {
        clock100ms_flag = 1; // 当 0.1 秒到时，溢出标志置 1
        clock100ms = 0;
    }

    // 0.5 秒钟软定时器计数
    if (++clock500ms >= V_T500ms)
    {
        clock500ms_flag = 1; // 当 0.5 秒到时，溢出标志置 1
        clock500ms = 0;
    }

    // 2.0 秒钟软定时器计数
    if (++clock2s >= V_T2s)
    {
        clock2s_flag = 1; // 当 2.0 秒到时，溢出标志置 1
        clock2s = 0;
    }

    // 刷新全部数码管和 LED 指示灯
    TM1638_RefreshDIGIandLED(digit, pnt, led);

    // 检查当前键盘输入，0 代表无键操作，1-9 表示有对应按键
    // 键号显示在一位数码管上
    key_code = TM1638_Readkeyboard();

    digit[5] = key_code;
}
```

1.4.2 实验任务 4_2

```
/**
 * @file exp4_2.c
 * @author 上海交通大学电子工程系实验教学中心; Guorui Wei (313017602@qq.com)
 * @brief 实验 4_2
 * @version 0.1
 * @date 2021-06-05
 *
 * @copyright 2020-2021, 上海交通大学电子工程系实验教学中心
 *
 */

//*****************************************************************************
//
// 头文件
//
//*****************************************************************************
```

```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"        // 基址宏定义
#include "inc/hw_types.h"         // 数据类型宏定义，寄存器访问函数
#include "driverlib/debug.h"      // 调试用
#include "driverlib/gpio.h"       // 通用 IO 口宏定义
#include "driverlib/pin_map.h"    // TM4C 系列 MCU 外围设备管脚宏定义
#include "driverlib/sysctl.h"     // 系统控制定义
#include "driverlib/systick.h"    // SysTick Driver 原型
#include "driverlib/interrupt.h" // NVIC Interrupt Controller Driver 原型
#include "JLX12864.h"             // 与控制 JLX12864G 有关的函数
#include "tm1638.h"               // 与控制 TM1638 芯片有关的函数
#include "string.h"               //

//*****************************************************************************
//
// 宏定义
//
//*****************************************************************************
#define SYSTICK_FREQUENCY 50  // SysTick 频率为 50Hz，即循环定时周期 20ms
#define V_T100ms 5            // 0.1s 软件定时器溢出值，5 个 20ms
#define V_T500ms 25           // 0.5s 软件定时器溢出值，25 个 20ms
#define V_T2s 100             // 2.0s 软定时器溢出值，100 个 20ms
#define V_T10s 500            // 10.0s 软定时器溢出值，500 个 20ms
#define LCD_MAX_BLOCK 15      // 显示屏上的最大 8x8 分区数
#define LCD_MAX_BLOCK_CHAR 15 // 显示屏上每个分区的最大字符数

//*****************************************************************************
//
// 函数原型声明
//
//*****************************************************************************
void GPIOInit(void);    // GPIO 初始化
void SysTickInit(void); // 设置 SysTick 中断
void DevicesInit(void); // MCU 器件初始化，注：会调用上述函数

/**
 * UI 状态机相关函数
 */

void ui_state_proc(uint16_t ui_state);
void ui_proc0(void);
void ui_proc001(void);
void ui_proc003(void);
void ui_proc005(void);
void ENTER_detect(void);
void LEFT_detect(void);
void RIGHT_detect(void);
void INCREASE_detect(void);
void DECREASE_detect(void);

//*****************************************************************************
//
// 变量定义
//
//*****************************************************************************

// 软件定时器计数
uint8_t clock100ms = 0;
uint8_t clock500ms = 0;
uint8_t clock2s = 0;
uint16_t NOKEY_clock10s = 0;

// 软件定时器溢出标志
uint8_t clock100ms_flag = 0;
uint8_t clock500ms_flag = 0;
uint8_t clock2s_flag = 0;
uint8_t NOKEY_clock10s_flag = 0;

/**
 * 按键事件标志
 */
uint8_t key_LEFT_flag = 0;
uint8_t key_RIGHT_flag = 0;
```

```
uint8_t key_INCREASE_flag = 0;
uint8_t key_DECREASE_flag = 0;
uint8_t key_ENTER_flag = 0;

// 测试用计数器
uint32_t test_counter = 0;

// 8 位数码管显示的数字或字母符号
// 注：板上数码位从左到右序号排列为 4、5、6、7、0、1、2、3
uint8_t digit[8] = {' ', ' ', ' ', ' ', '_', ' ', '_', ' '};

// 8 位小数点 1 亮  0 灭
// 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
uint8_t pnt = 0x04;

// 8 个 LED 指示灯状态，0 灭，1 亮
// 注：板上指示灯从左到右序号排列为 7、6、5、4、3、2、1、0
//     对应元件 LED8、LED7、LED6、LED5、LED4、LED3、LED2、LED1
uint8_t led[] = {1, 1, 1, 1, 1, 1, 1, 0};

// 当前按键值
uint8_t key_code = 0;
uint8_t pre_key_code; // 上一按键值

// 系统时钟频率
uint32_t ui32SysClock;

/**
 * 用户界面（UI）状态机相关变量定义
 */

uint16_t ui_state = 0x0; // 用户界面（UI）状态机当前状态

/**
 * @brief 状态参数结构体
 *
 */
struct ACT_T
{
    uint8_t row_page[LCD_MAX_BLOCK];                      // 显示屏上每个分区的起始行页号
    uint8_t col_page[LCD_MAX_BLOCK];                      // 显示屏上每个分区的起始列页号
    unsigned char str[LCD_MAX_BLOCK][LCD_MAX_BLOCK_CHAR]; // 显示屏上每个分区的显示内容
    const uint8_t SIZE;                                   // 显示屏上有效分区的数量
};

struct ACT_T act0 = {
    {3, 3, 5, 5, 5, 5, 5},
    {1, 11, 1, 11, 12, 13, 14},
    {"工作模式：", "模式 A", "工作参数：", "1", ".", "1", "Hz"},
    7};

struct ACT_T *act[] = {&act0};

//*****************************************************************************
//
// 主程序
//
//*****************************************************************************
int main(void)
{
    uint8_t temp, i;
    DevicesInit(); //  MCU 器件初始化

    while (clock100ms < 3)
        ;           // 延时>60ms,等待 TM1638 上电完成
    TM1638_Init();  // 初始化 TM1638
    initial_lcd();  // 初始化 JLX12864
    clear_screen(); //clear all dots

    while (1)
    {
        if (clock100ms_flag == 1) // 检查 0.1 秒定时是否到
        {
            clock100ms_flag = 0;
```

```
            // 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
            if (key_code == 0)
            {
                if (++test_counter >= 10000)
                    test_counter = 0;
                digit[0] = test_counter / 1000;      // 计算百位数
                digit[1] = test_counter / 100 % 10; // 计算十位数
                digit[2] = test_counter / 10 % 10;  // 计算个位数
                digit[3] = test_counter % 10;        // 计算百分位数
            }
        }

        if (clock500ms_flag == 1) // 检查 0.5 秒定时是否到
        {
            clock500ms_flag = 0;
            // 8 个指示灯以走马灯方式，每 0.5 秒向右（循环）移动一格
            temp = led[0];
            for (i = 0; i < 7; i++)
                led[i] = led[i + 1];
            led[7] = temp;
        }

        ui_state_proc(ui_state);
    }
}

//*****************************************************************************
//
// 函数原型：void GPIOInit(void)
// 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。
//           （PK4 连接 TM1638 的 STB，PK5 连接 TM1638 的 DIO，PM0 连接 TM1638 的 CLK）
// 函数参数：无
// 函数返回值：无
//
//*****************************************************************************
void GPIOInit(void)
{
    //配置 TM1638 芯片管脚
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOK); // 使能端口 K
    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOK))
    {
    }; // 等待端口 K 准备完毕

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOM); // 使能端口 M
    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOM))
    {
    }; // 等待端口 M 准备完毕

    // 设置端口 K 的第 4,5 位（PK4,PK5）为输出引脚        PK4-STB   PK5-DIO
    GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_4 | GPIO_PIN_5);
    // 设置端口 M 的第 0 位（PM0）为输出引脚    PM0-CLK
    GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_0);
}

//*****************************************************************************
//
// 函数原型：SysTickInit(void)
// 函数功能：设置 SysTick 中断
// 函数参数：无
// 函数返回值：无
//
//*****************************************************************************
void SysTickInit(void)
{
    SysTickPeriodSet(ui32SysClock / SYSTICK_FREQUENCY); // 设置心跳节拍,定时周期 20ms
    SysTickEnable();                                    // SysTick 使能
    SysTickIntEnable();                                 // SysTick 中断允许
}

//*****************************************************************************
//
// 函数原型：void DevicesInit(void)
// 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
// 函数参数：无
```

```
// 函数返回值：无
//
//*****************************************************************************
void DevicesInit(void)
{
    // 使用外部 25MHz 主时钟源，经过 PLL，然后分频为 20MHz
    ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
                                       SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480),
                                      20000000);

    GPIOInit();         // GPIO 初始化
    SysTickInit();      // 设置 SysTick 中断
    IntMasterEnable(); // 总中断允许
}

//*****************************************************************************
//
// 函数原型：void SysTick_Handler(void)
// 函数功能：SysTick 中断服务程序
// 函数参数：无
// 函数返回值：无
//
//*****************************************************************************
void SysTick_Handler(void) // 定时周期为 20ms
{
    // 0.1 秒钟软定时器计数
    if (++clock100ms >= V_T100ms)
    {
        clock100ms_flag = 1; // 当 0.1 秒到时，溢出标志置 1
        clock100ms = 0;
    }

    // 0.5 秒钟软定时器计数
    if (++clock500ms >= V_T500ms)
    {
        clock500ms_flag = 1; // 当 0.5 秒到时，溢出标志置 1
        clock500ms = 0;
    }

    // 刷新全部数码管和 LED 指示灯
    TM1638_RefreshDIGIandLED(digit, pnt, led);

    // 检查当前键盘输入，0 代表无键操作，1-9 表示有对应按键
    // 键号显示在一位数码管上
    pre_key_code = key_code;          // 保存上一按键值
    key_code = TM1638_Readkeyboard(); // 更新当前按键值

    digit[5] = key_code;

    ENTER_detect();
    LEFT_detect();
    RIGHT_detect();
    INCREASE_detect();
    DECREASE_detect();

    // 10.0 秒钟软定时器计数
    if (!key_code && ++NOKEY_clock10s >= V_T10s) // 当无键按下时
    {
        NOKEY_clock10s_flag = 1; // 当 10.0 秒到时，溢出标志置 1
        NOKEY_clock10s = 0;
    }
    // 若有键按下，则 10.0 秒计数清零
    if (key_code)
    {
        NOKEY_clock10s = 0;
    }
}

/**
 * @brief UI 状态机处理函数
 *
 * @param ui_state UI 状态机当前状态
 */
void ui_state_proc(uint16_t ui_state)
{
```

```
    switch (ui_state)
    {
    case 0x0: // ACT0
        ui_proc0();
        break;
    case 0x001: //ACT001
        ui_proc001();
        break;
    case 0x003: // ACT003
        ui_proc003();
        break;
    case 0x005: // ACT005
        ui_proc005();
        break;
    default:
        ui_state = 0;
        break;
    }
}

/**
 * @brief UI 状态机 ACT0 状态处理
 * 开机初始画面，不显示光标
 *
 */
void ui_proc0(void)
{
    uint8_t i = 0;
    // 显示开机初始画面，无光标
    for (i = 0; i < act[0]->SIZE; ++i)
    {
        display_GB2312_string(act[0]->row_page[i], act[0]->col_page[i] * 8 - 7, act[0]->str[i], 0);
    }

    // 当有任意按键被按下："模式#"做反白效果（光标），转移至状态 ACT001
    if (!pre_key_code && key_code)
    {
        key_LEFT_flag = key_RIGHT_flag = key_INCREASE_flag = key_DECREASE_flag = key_ENTER_flag = 0;
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 1);
        ui_state = 0x001;
    }
}

/**
 * @brief UI 状态机 ACT001 状态处理
 * 光标在工作模式选择位置
 *
 */
void ui_proc001(void)
{
    // 当"右"键按下：光标移到工作参数的个位，下一状态 ACT003
    if (key_RIGHT_flag)
    {
        key_RIGHT_flag = 0;
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 0);
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 1);
        ui_state = 0x003;
    }
    // 当"左"键按下：光标移到工作参数的十分位，下一状态 ACT005
    else if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 0);
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 1);
        ui_state = 0x005;
    }
    // 当"+"键按下："模式#"按 A、B、C、A 正序循环切换，留在本状态
    else if (key_INCREASE_flag)
    {
        key_INCREASE_flag = 0;
        if (++((act[0]->str[1])[strlen((const char *)(act[0]->str[1])) - 1]) > 'C')
        {
            (act[0]->str[1])[strlen((const char *)(act[0]->str[1])) - 1] = 'A';
        }
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 1);
    }
```

```
    // 当"-"键按下："模式#"按 C、B、A、C 逆序循环切换，留在本状态
    else if (key_DECREASE_flag)
    {
        key_DECREASE_flag = 0;
        if (--((act[0]->str[1])[strlen((const char *)(act[0]->str[1])) - 1]) < 'A')
        {
            (act[0]->str[1])[strlen((const char *)(act[0]->str[1])) - 1] = 'C';
        }
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 1);
    }

    // 当 10 秒无操作："模式#"反白效果解除，下一状态 ACT0
    if (NOKEY_clock10s_flag)
    {
        NOKEY_clock10s_flag = 0;
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 0);
        ui_state = 0x0;
    }
}

/**
 * @brief UI 状态机 ACT003 状态处理
 * 光标在工作参数个位的位置
 *
 */
void ui_proc003(void)
{
    // 当"右"键按下：光标移到十分位，下一状态 ACT005
    if (key_RIGHT_flag)
    {
        key_RIGHT_flag = 0;
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 0);
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 1);
        ui_state = 0x005;
    }
    // 当"左"键按下：光标移到"模式#"，下一状态 ACT001
    else if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 0);
        display_GB2312_string(act[0]->row_page[1], act[0]->col_page[1] * 8 - 7, act[0]->str[1], 1);
        ui_state = 0x001;
    }
    // 当"+"键按下：个位数按 1、2、...、9、0、1、...正序循环切换，留在本状态
    else if (key_INCREASE_flag)
    {
        key_INCREASE_flag = 0;
        if (++((act[0]->str[3])[0]) > '9')
        {
            (act[0]->str[3])[0] = '0';
        }
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 1);
    }
    // 当"-"键按下：个位数按 9、8、...、0、9、8、...逆序循环切换，留在本状态
    else if (key_DECREASE_flag)
    {
        key_DECREASE_flag = 0;
        if (--((act[0]->str[3])[0]) < '0')
        {
            (act[0]->str[3])[0] = '9';
        }
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 1);
    }
    else if (key_ENTER_flag)
    {
        key_ENTER_flag = 0;
    }

    // 当 10 秒无操作：个位反白效果解除，下一状态 ACT0
    if (NOKEY_clock10s_flag)
    {
        NOKEY_clock10s_flag = 0;
        display_GB2312_string(act[0]->row_page[3], act[0]->col_page[3] * 8 - 7, act[0]->str[3], 0);
        ui_state = 0x0;
    }
}
```

```
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 
```

1.4.3 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  //
```

```
// 主程序
//
//*****************************************************************************
int main(void)
{
    uint8_t temp, i;
    DevicesInit(); //  MCU 器件初始化

    while (clock100ms < 3)
        ;           // 延时>60ms,等待 TM1638 上电完成
    TM1638_Init();  // 初始化 TM1638
    initial_lcd();  // 初始化 JLX12864
    clear_screen(); //clear all dots

    while (1)
    {
        if (clock100ms_flag == 1) // 检查 0.1 秒定时是否到
        {
            clock100ms_flag = 0;
            // 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
            if (key_code == 0)
            {
                if (++test_counter >= 10000)
                    test_counter = 0;
                digit[0] = test_counter / 1000;     // 计算百位数
                digit[1] = test_counter / 100 % 10; // 计算十位数
                digit[2] = test_counter / 10 % 10;  // 计算个位数
                digit[3] = test_counter % 10;       // 计算百分位数
            }
        }

        if (clock500ms_flag == 1) // 检查 0.5 秒定时是否到
        {
            clock500ms_flag = 0;
            // 8 个指示灯以走马灯方式，每 0.5 秒向右（循环）移动一格
            temp = led[0];
            for (i = 0; i < 7; i++)
                led[i] = led[i + 1];
            led[7] = temp;
        }

        ui_state_proc(ui_state);
    }
}

//*****************************************************************************
//
// 函数原型：void GPIOInit(void)
// 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。
//           （PK4 连接 TM1638 的 STB，PK5 连接 TM1638 的 DIO，PM0 连接 TM1638 的 CLK）
// 函数参数：无
// 函数返回值：无
//
//*****************************************************************************
void GPIOInit(void)
{
    //配置 TM1638 芯片管脚
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOK); // 使能端口 K
    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOK))
    {
    }; // 等待端口 K 准备完毕

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOM); // 使能端口 M
    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOM))
    {
    }; // 等待端口 M 准备完毕

    // 设置端口 K 的第 4,5 位（PK4,PK5）为输出引脚       PK4-STB  PK5-DIO
    GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_4 | GPIO_PIN_5);
    // 设置端口 M 的第 0 位（PM0）为输出引脚   PM0-CLK
    GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_0);
}

//*****************************************************************************
//
```

```
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
```

```
    if (!key_code && ++NOKEY_clock5s >= V_T5s) // 当无键按下时
    {
        NOKEY_clock5s_flag = 1; // 当 5.0 秒到时，溢出标志置 1
        NOKEY_clock5s = 0;
    }
    // 若有键按下，则 5.0 秒计数清零
    if (key_code)
    {
        NOKEY_clock5s = 0;
    }

    // ACT4（警示画面）的 5.0 秒钟软定时器计数
    if (ui_state == 0x4 && ++ACT4_clock5s >= V_T5s) // 当无键按下时
    {
        ACT4_clock5s_flag = 1; // 当 5.0 秒到时，溢出标志置 1
        ACT4_clock5s = 0;
    }
}

/**
 * @brief UI 状态机处理函数
 *
 * @param ui_state UI 状态机当前状态
 */
void ui_state_proc(uint16_t ui_state)
{
    switch (ui_state)
    {
    case 0x0: // ACT0
        ui_proc0();
        break;
    case 0x005: //ACT005
        ui_proc005();
        break;
    case 0x100: // ACT100
        ui_proc100();
        break;
    case 0x101: // ACT101
        ui_proc101();
        break;
    case 0x102: // ACT102
        ui_proc102();
        break;
    case 0x201:
        ui_proc201();
        break;
    case 0x202:
        ui_proc202();
        break;
    case 0x203:
        ui_proc203();
        break;
    case 0x301:
        ui_proc301();
        break;
    case 0x303:
        ui_proc303();
        break;
    case 0x305:
        ui_proc305();
        break;
    case 0x306:
        ui_proc306();
        break;
    case 0x4: // ACT4
        ui_proc4();
        break;
    default:
        ui_state = 0x0;
        break;
    }
}

/**
 * @brief UI 状态机 ACT0 状态处理
 * 开机初始画面，不显示光标
```

```
 *
 */
void ui_proc0(void)
{
    uint8_t i = 0;
    // 显示开机初始画面，无光标
    for (i = 0; i < act[0]->SIZE; ++i)
    {
        display_GB2312_string(act[0]->row_page[i], act[0]->col_page[i] * 8 - 7, act[0]->str[i], 0);
    }

    // 当有任意按键被按下："设置"做反白效果（光标），转移至状态 ACT005
    if (!pre_key_code && key_code)
    {
        key_LEFT_flag = key_RIGHT_flag = key_INCREASE_flag = key_DECREASE_flag = key_ENTER_flag = 0;
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 1);
        ui_state = 0x005;
    }
}

/**
 * @brief UI 状态机 ACT005 状态处理
 * 光标在"设置"的位置
 *
 */
void ui_proc005(void)
{
    uint8_t i = 0;
    // 当"确定"键按下：光标移到 ACT1 的"工作参数"，下一状态 ACT100
    if (key_ENTER_flag)
    {
        key_ENTER_flag = 0;
        // 清屏，显示 ACT1 的画面
        clear_screen();
        for (i = 0; i < act[1]->SIZE; ++i)
        {
            display_GB2312_string(act[1]->row_page[i], act[1]->col_page[i] * 8 - 7, act[1]->str[i], 0);
        }
        // 显示光标
        display_GB2312_string(act[1]->row_page[0], act[1]->col_page[0] * 8 - 7, act[1]->str[0], 1);
        ui_state = 0x100;
    }
    else if (key_LEFT_flag || key_RIGHT_flag || key_INCREASE_flag || key_DECREASE_flag)
    {
        key_LEFT_flag = key_RIGHT_flag = key_DECREASE_flag = key_INCREASE_flag = 0;
    }

    // 当 5 秒无操作："设置"反白效果解除，下一状态 ACT0
    if (NOKEY_clock5s_flag)
    {
        NOKEY_clock5s_flag = 0;
        display_GB2312_string(act[0]->row_page[5], act[0]->col_page[5] * 8 - 7, act[0]->str[5], 0);
        ui_state = 0x0;
    }
}

/**
 * @brief UI 状态机 ACT100 状态处理
 * 光标在"工作模式"的位置
 *
 */
void ui_proc100(void)
{
    uint8_t i = 0;
    // 当"左"键按下：光标移到"返回"，下一状态 ACT102
    if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[1]->row_page[0], act[1]->col_page[0] * 8 - 7, act[1]->str[0], 0);
        display_GB2312_string(act[1]->row_page[2], act[1]->col_page[2] * 8 - 7, act[1]->str[2], 1);
        ui_state = 0x102;
    }
    // 当"右"键按下：光标移到"工作参数"，下一状态 ACT101
    else if (key_RIGHT_flag)
    {
        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         }
```

```
    // 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
    else if (key_ENTER_flag)
    {
        key_ENTER_flag = 0;
        // 撤销对"模式#"的更改
        strcpy((char *)(act[2]->str[1]), (const char *)(act[0]->str[0]));
        // 清屏，显示 ACT1 的画面，光标移到 ACT1 的"工作模式"
        clear_screen();
        for (i = 0; i < act[1]->SIZE; ++i)
        {
            display_GB2312_string(act[1]->row_page[i], act[1]->col_page[i] * 8 - 7, act[1]->str[i], 0);
        }
        display_GB2312_string(act[1]->row_page[0], act[1]->col_page[0] * 8 - 7, act[1]->str[0], 1);
        ui_state = 0x100;
    }
    else if (key_INCREASE_flag || key_DECREASE_flag)
    {
        key_DECREASE_flag = key_INCREASE_flag = 0;
    }
}

/**
 * @brief UI 状态机 ACT301 状态处理
 * 光标在工作参数个位的位置
 *
 */
void ui_proc301(void)
{
    // 当"左"键按下：光标移到"取消"，下一状态 ACT306
    if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[3]->row_page[1], act[3]->col_page[1] * 8 - 7, act[3]->str[1], 0);
        display_GB2312_string(act[3]->row_page[6], act[3]->col_page[6] * 8 - 7, act[3]->str[6], 1);
        ui_state = 0x306;
    }
    // 当"右"键按下：光标移到工作参数十分位的位置，下一状态 ACT303
    else if (key_RIGHT_flag)
    {
        key_RIGHT_flag = 0;
        display_GB2312_string(act[3]->row_page[1], act[3]->col_page[1] * 8 - 7, act[3]->str[1], 0);
        display_GB2312_string(act[3]->row_page[3], act[3]->col_page[3] * 8 - 7, act[3]->str[3], 1);
        ui_state = 0x303;
    }
    // 当"+"键按下：个位数按 1、2、...、9、0、1、...正序循环切换，留在本状态
    else if (key_INCREASE_flag)
    {
        key_INCREASE_flag = 0;
        if (++((act[3]->str[1])[0]) > '9')
        {
            (act[3]->str[1])[0] = '0';
        }
        display_GB2312_string(act[3]->row_page[1], act[3]->col_page[1] * 8 - 7, act[3]->str[1], 1);
    }
    // 当"-"键按下：个位数按 9、8、...、0、9、8、...逆序循环切换，留在本状态
    else if (key_DECREASE_flag)
    {
        key_DECREASE_flag = 0;
        if (--((act[3]->str[1])[0]) < '0')
        {
            (act[3]->str[1])[0] = '9';
        }
        display_GB2312_string(act[3]->row_page[1], act[3]->col_page[1] * 8 - 7, act[3]->str[1], 1);
    }
    else if (key_ENTER_flag)
    {
        key_ENTER_flag = 0;
    }
}

/**
 * @brief UI 状态机 ACT303 状态处理
 * 光标在工作参数十分位的位置
 *
 */
void ui_proc303(void)
```

```
{
    // 当"左"键按下：光标移到工作参数个位的位置，下一状态 ACT301
    if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[3]->row_page[3], act[3]->col_page[3] * 8 - 7, act[3]->str[3], 0);
        display_GB2312_string(act[3]->row_page[1], act[3]->col_page[1] * 8 - 7, act[3]->str[1], 1);
        ui_state = 0x301;
    }
    // 当"右"键按下：光标移到"确定"，下一状态 ACT305
    else if (key_RIGHT_flag)
    {
        key_RIGHT_flag = 0;
        display_GB2312_string(act[3]->row_page[3], act[3]->col_page[3] * 8 - 7, act[3]->str[3], 0);
        display_GB2312_string(act[3]->row_page[5], act[3]->col_page[5] * 8 - 7, act[3]->str[5], 1);
        ui_state = 0x305;
    }
    // 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
    else if (key_INCREASE_flag)
    {
        key_INCREASE_flag = 0;
        if (++((act[3]->str[3])[0]) > '9')
        {
            (act[3]->str[3])[0] = '0';
        }
        display_GB2312_string(act[3]->row_page[3], act[3]->col_page[3] * 8 - 7, act[3]->str[3], 1);
    }
    // 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
    else if (key_DECREASE_flag)
    {
        key_DECREASE_flag = 0;
        if (--((act[3]->str[3])[0]) < '0')
        {
            (act[3]->str[3])[0] = '9';
        }
        display_GB2312_string(act[3]->row_page[3], act[3]->col_page[3] * 8 - 7, act[3]->str[3], 1);
    }
    else if (key_ENTER_flag)
    {
        key_ENTER_flag = 0;
    }
}

/**
 * @brief UI 状态机 ACT305 状态处理
 * 光标在"确定"的位置
 *
 */
void ui_proc305(void)
{
    // 当"左"键按下：光标移到工作参数十分位的位置，下一状态 ACT303
    if (key_LEFT_flag)
    {
        key_LEFT_flag = 0;
        display_GB2312_string(act[3]->row_page[5], act[3]->col_page[5] * 8 - 7, act[3]->str[5], 0);
        display_GB2312_string(act[3]->row_page[3], act[3]->col_page[3] * 8 - 7, act[3]->str[3], 1);
        ui_state = 0x303;
    }
    // 当"右"键按下：光标移到"取消"，下一状态 ACT306
    else if (key_RIGHT_flag)
    {
        key_RIGHT_flag = 0;
        display_GB2312_string(act[3]->row_page[5], act[3]->col_page[5] * 8 - 7, act[3]->str[5], 0);
        display_GB2312_string(act[3]->row_page[6], act[3]->col_page[6] * 8 - 7, act[3]->str[6], 1);
        ui_state = 0x306;
    }
    // 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
    else if (key_ENTER_flag)
    {
        uint8_t i;
        uint8_t num = ((act[3]->str[1])[0] - '0') * 10 + (act[3]->str[3])[0] - '0';

        key_ENTER_flag = 0;
        // 非法参数：显示警示画面 ACT4，下一状态 ACT4
        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
```