上海电力大学

嵌入式系统设计应用

实验名称：实验五矩阵键盘输入控制实验

专业班级：集成电路设计与集成系统

姓　　名：某同学

学　　号：2021xxxx

2024 年 6 月 9 日

一、实验要求

1. 运行例程实验实验 11　矩阵键盘输入 + 串口输出 +LCD 液晶显示，观察实验现象
2. 看懂源程序
3. 修改源程序，按下各个键时分别在串口和 LCD 液晶屏上显示如图所示符号
4. 撰写实验报告、把修改的程序截图、实验现象的裁图或者图片整理到报告中

0	1	2	3
4	5	6	7
8	9	+	-
*	/	=	(

图 1: 实验要求

二、实验内容及结果

1. 编写代码

修改 `main.c` 文件，实现按下各个键时分别在串口和 LCD 液晶屏上显示如图所示符号。主要功能包括初始化 LED、GPIO、LCD 和串口通信。它还包括一个按键扫描功能，用于检测按键输入，并将按键值通过串口输出显示在 LCD 上。以下是代码的重要部分的简要说明：

1. `LED_GPIO_Config()`：配置 LED 相关的 GPIO。
2. `GPIO_Configuration()`：配置 GPIO。
3. `NT35510_Init()`：初始化 LCD 显示。
4. `Debu_USART_Config()`：配置调试用的串口。
5. `NT35510_GramScan(6)`：设置 LCD 显示模式。
6. `Key_scan()`：扫描按键输入并返回按键值。
7. `LCD_Test()`：用于测试 LCD 显示的函数。

在 main 函数中，通过一个无限循环，不断调用 LCD_Test() 和 Key_scan() 函数，以及一个延时函数 Delay() 来实现连续的按键扫描和显示更新。

按键扫描函数 Key_scan() 通过设置和重置 GPIO 位来检测哪个按键被按下，并将按键值存储在 key_value 变量中，然后通过串口输出该值。每个按键对应一个字符，例如'0'到'9'、'+'、'-'、'*'、'/'、'=' 和'('。这些字符随后通过串口输出，并显示在 LCD 上。

```c
/**
  ******************************************************************************
  * @file    main.c

  * @brief   按键扫描 + 串口输出 +LCD 液晶显示
        * @ 键盘 C1-C4 接 PB4——PB7,R1-R4 接 PB0——PB3
  ******************************************************************************

  ******************************************************************************
  */

#include "stm32f4xx.h"
#include "./usart/bsp_debug_usart.h"
#include "./led/bsp_led.h"
#include "./lcd/bsp_nt35510_lcd.h"

static void LCD_Test(void);
static void Delay(__IO uint32_t nCount);
void Printf_Charater(void);
uint16_t Key_scan(void);

/**
 * @brief  主函数
 * @param  无
 * @retval 无
 */
int key_value = 0;
int main(void)
{
        LED_GPIO_Config();
        GPIO_Configuration();
        NT35510_Init(); // LCD 初始化

        Debug_USART_Config();

        printf("\r\n ********** 按键扫描 + 串口输出 +LCD 液晶显示 ********** \r\n");
        printf("\r\n 本程序不支持中文 \r\n");

        // 其中 0、3、5、6 模式适合从左至右显示文字，
        // 不推荐使用其它模式显示文字          其它模式显示文字会有镜像效果
        // 其中 6 模式为大部分液晶例程的默认显示方向
        NT35510_GramScan(6);

        while (1)
        {
                LCD_Test(); // LCD
```

```
47                  Key_scan(); // 按键扫描
48                  // printf("%d\r\n",key_value);
49                  Delay(0x22FFFF);
50          }
51  }
52
53  uint16_t Key_scan(void)
54  {
55          // 第一行置 1，其它行置为 0；这时哪一列为 1，则代表哪一列被按下
56          GPIO_SetBits(L1_PORT, L1_PIN);
57          GPIO_ResetBits(L2_PORT, L2_PIN);
58          GPIO_ResetBits(L3_PORT, L3_PIN);
59          GPIO_ResetBits(L4_PORT, L4_PIN);
60
61          if (R1_PORT->IDR & R1_PIN) // 判断第一列是否被按下
62          {
63                  while (R1_PORT->IDR & R1_PIN)
64                          ; // 判断第一列是否已经弹起
65                  key_value = '0';
66                  printf("%c\r\n", key_value);
67          }
68          if (R2_PORT->IDR & R2_PIN) // 判断第二列是否被按下
69          {
70                  while (R2_PORT->IDR & R2_PIN)
71                          ; // 判断第二列是否已经弹起
72                  key_value = '1';
73                  printf("%c\r\n", key_value);
74          }
75          if (R3_PORT->IDR & R3_PIN) // 判断第三列是否已经弹起
76          {
77                  while (R3_PORT->IDR & R3_PIN)
78                          ; // 判断第三列是否已经弹起
79                  key_value = '2';
80                  printf("%c\r\n", key_value);
81          }
82          if (R4_PORT->IDR & R4_PIN) // 判断第四列是否已经弹起
83          {
84                  while (R4_PORT->IDR & R4_PIN)
85                          ; // 判断第四列是否已经弹起
86                  key_value = '3';
87                  printf("%c\r\n", key_value);
88          }
89          // 第二行置 1，其它行置为 0；这时哪一列为 1，则代表哪一列被按下
90          GPIO_SetBits(L2_PORT, L2_PIN);
91          GPIO_ResetBits(L1_PORT, L1_PIN);
92          GPIO_ResetBits(L3_PORT, L3_PIN);
93          GPIO_ResetBits(L4_PORT, L4_PIN);
94
95          if (R1_PORT->IDR & R1_PIN) // 判断第一列是否被按下
96          {
97                  while (R1_PORT->IDR & R1_PIN)
98                          ; // 判断第一列是否已经弹起
99                  key_value = '4';
100                 printf("%c\r\n", key_value);
```

```
101         }
102         if (R2_PORT->IDR & R2_PIN) // 判断第二列是否被按下
103         {
104                 while (R2_PORT->IDR & R2_PIN)
105                         ; // 判断第二列是否已经弹起
106                 key_value = '5';
107                 printf("%c\r\n", key_value);
108         }
109         if (R3_PORT->IDR & R3_PIN) // 判断第三列是否已经弹起
110         {
111                 while (R3_PORT->IDR & R3_PIN)
112                         ; // 判断第三列是否已经弹起
113                 key_value = '6';
114                 printf("%c\r\n", key_value);
115         }
116         if (R4_PORT->IDR & R4_PIN) // 判断第四列是否已经弹起
117         {
118                 while (R4_PORT->IDR & R4_PIN)
119                         ; // 判断第四列是否已经弹起
120                 key_value = '7';
121                 printf("%c\r\n", key_value);
122         }
123
124         // 第三行置 1，其它行置为 0；这时哪一列为 1，则代表哪一列被按下
125         GPIO_SetBits(L3_PORT, L3_PIN);
126         GPIO_ResetBits(L1_PORT, L1_PIN);
127         GPIO_ResetBits(L2_PORT, L2_PIN);
128         GPIO_ResetBits(L4_PORT, L4_PIN);
129
130         if (R1_PORT->IDR & R1_PIN) // 判断第一列是否被按下
131         {
132                 while (R1_PORT->IDR & R1_PIN)
133                         ; // 判断第一列是否已经弹起
134                 key_value = '8';
135                 printf("%c\r\n", key_value);
136         }
137         if (R2_PORT->IDR & R2_PIN) // 判断第二列是否被按下
138         {
139                 while (R2_PORT->IDR & R2_PIN)
140                         ; // 判断第二列是否已经弹起
141                 key_value = '9';
142                 printf("%c\r\n", key_value);
143         }
144         if (R3_PORT->IDR & R3_PIN) // 判断第三列是否已经弹起
145         {
146                 while (R3_PORT->IDR & R3_PIN)
147                         ; // 判断第三列是否已经弹起
148                 key_value = '+';
149                 printf("%c\r\n", key_value);
150         }
151         if (R4_PORT->IDR & R4_PIN) // 判断第四列是否已经弹起
152         {
153                 while (R4_PORT->IDR & R4_PIN)
154                         ; // 判断第四列是否已经弹起
```

```
155                 key_value = '-';
156                 printf("%c\r\n", key_value);
157         }
158 
159         // 第四行置 1，其它行置为 0；这时哪一列为 1，则代表哪一列被按下
160         GPIO_SetBits(L4_PORT, L4_PIN);
161         GPIO_ResetBits(L1_PORT, L1_PIN);
162         GPIO_ResetBits(L2_PORT, L2_PIN);
163         GPIO_ResetBits(L3_PORT, L3_PIN);
164 
165         if (R1_PORT->IDR & R1_PIN) // 判断第一列是否被按下
166         {
167                 while (R1_PORT->IDR & R1_PIN)
168                         ; // 判断第一列是否已经弹起
169                 key_value = '*';
170                 printf("%c\r\n", key_value);
171         }
172         if (R2_PORT->IDR & R2_PIN) // 判断第二列是否被按下
173         {
174                 while (R2_PORT->IDR & R2_PIN)
175                         ; // 判断第二列是否已经弹起
176                 key_value = '/';
177                 printf("%c\r\n", key_value);
178         }
179         if (R3_PORT->IDR & R3_PIN) // 判断第三列是否已经弹起
180         {
181                 while (R3_PORT->IDR & R3_PIN)
182                         ; // 判断第三列是否已经弹起
183                 key_value = '=';
184                 printf("%c\r\n", key_value);
185         }
186         if (R4_PORT->IDR & R4_PIN) // 判断第四列是否已经弹起
187         {
188                 while (R4_PORT->IDR & R4_PIN)
189                         ; // 判断第四列是否已经弹起
190                 key_value = '(';
191                 printf("%c\r\n", key_value);
192         }
193 
194         return key_value;
195 }
196 
197 /* 用于测试各种液晶的函数 */
198 void LCD_Test(void)
199 {
200         /* 演示显示变量 */
201         char dispBuff[100];
202 
203         NT35510_ClearLine(LINE(7)); /* 清除单行文字 */
204         sprintf(dispBuff, "Key sequence number:%c", key_value);
205         NT35510_DispStringLine_EN(LINE(7), dispBuff); /* 显示单行文字 */
206 }
207 
208 /**
```

```
209   * @brief  简单延时函数
210   * @param  nCount : 延时计数值
211   * @retval 无
212   */
213  static void Delay(__IO uint32_t nCount)
214  {
215          for (; nCount != 0; nCount--)
216                  ;
217  }
218
219  /*********************************************END OF FILE**********************/
```

2. 下载运行

使用 FlyMCU.exe 下载程序到 STM32 开发版上，观察实验现象。

3. 实验现象

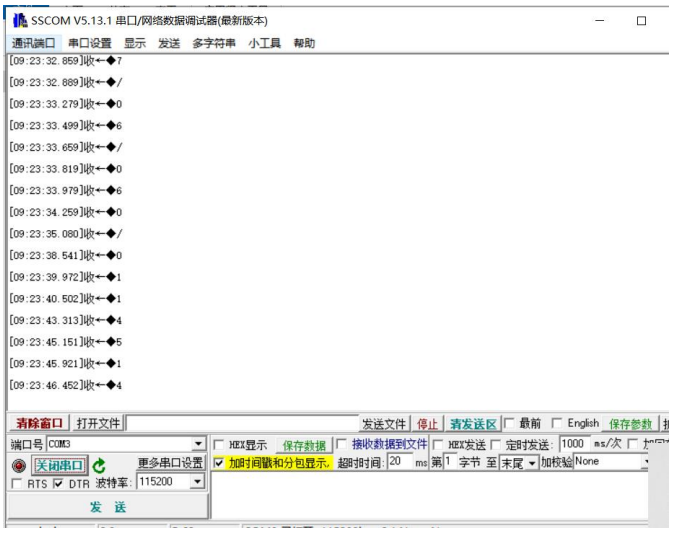

图 2: 串口输出结果

三、实验小结

本次实验通过集成 LED 指示灯、按键输入、串口通信以及 LCD 显示，实验构建了一个交互式系统。系统能够响应用户的按键输入，并将结果实时显示在 LCD 屏幕上，同时通过串口输出相关信息。