实验 1：点阵 LED 显示实验

1. 实验目的

1)掌握 LPC2000 系列 ARM7 处理器中 SPI 模块的基本原理及应用

2) 掌握点阵 LED 的驱动方法。

2. 实验环境

硬件：PC 机

SmartSOPC 教学实验开发平台

QuickARM 核心板

软件：ADS1.2 集成开发环境

μC/OS-Ⅱ 操作系统（2.52）

3. 实验内容

在点阵 LED 上显示汉字“广州致远电子有限公司”

4. 实验步骤

1) 使用 ARM Executabl Image for USOCII（for QuickARM）工程模板在 Demo Program 目录下建立工程 Lattice_LED.

2) 在 ADS 的项目窗口添加 GPIO 和 Lattice 文件组；然后把 Driver\GPIO \目录下的 GPIO.h 和 GPIO.c 文件添加到 GPIO 文件组，将 Software Package\Lattice 目录下的 Lattice.h 保存到 Lattice 文件组。

3) 调整外部总线速率，修改 Startup.s 文件中的 ResetInit 函数，BCFG0 和 BCFG1 寄存器的数据更改为 0x10001460

a) LDR R0，=BCFG0

b) LDR R1，=0x10001460

c) STR R1，[R0]

d) LDR R0，=BCFG1

e) LDR R1，=0x10001460

f) STR R1，[R0]

4) 打开工程窗口 user 组中的 main.c 文件，编写实验程序并保存。

5) 编译连接工程，并启动 AXD 进行 JTAG 仿真调试。

6) 跳线 JP7 短接，跳线 JP6 的 P7.0 短接 BEEP，JP6 中的 P0.4、P0.6 和 P0.18 分别连接到 COM13 端的 SCK、SI 和 STR。全速运行程序，程序将会在 main.c 的主函数中停止（因为 main 函数起始处默认设置有断点）。

7) 全速运行程序，观察点阵 LED 上显示的数据。

5. 实验方式

每位同学独立上机编程实验，实验指导教师现场指导。

6. 参考内容

实验原理：

点阵 LED 的工作原理如图 5.10 所示，其每个点都需要一个 LED，因此对于 16X16 的点阵 LED 来说，需要 16X16=256 个 LED。如果需要点亮某一点，只需将该点所在的阳极置“1”，阴极置“0”即可。点阵 LED 一般采用扫描的方式显示，通常采用的扫描方法有：点扫描、行扫描和列扫描。

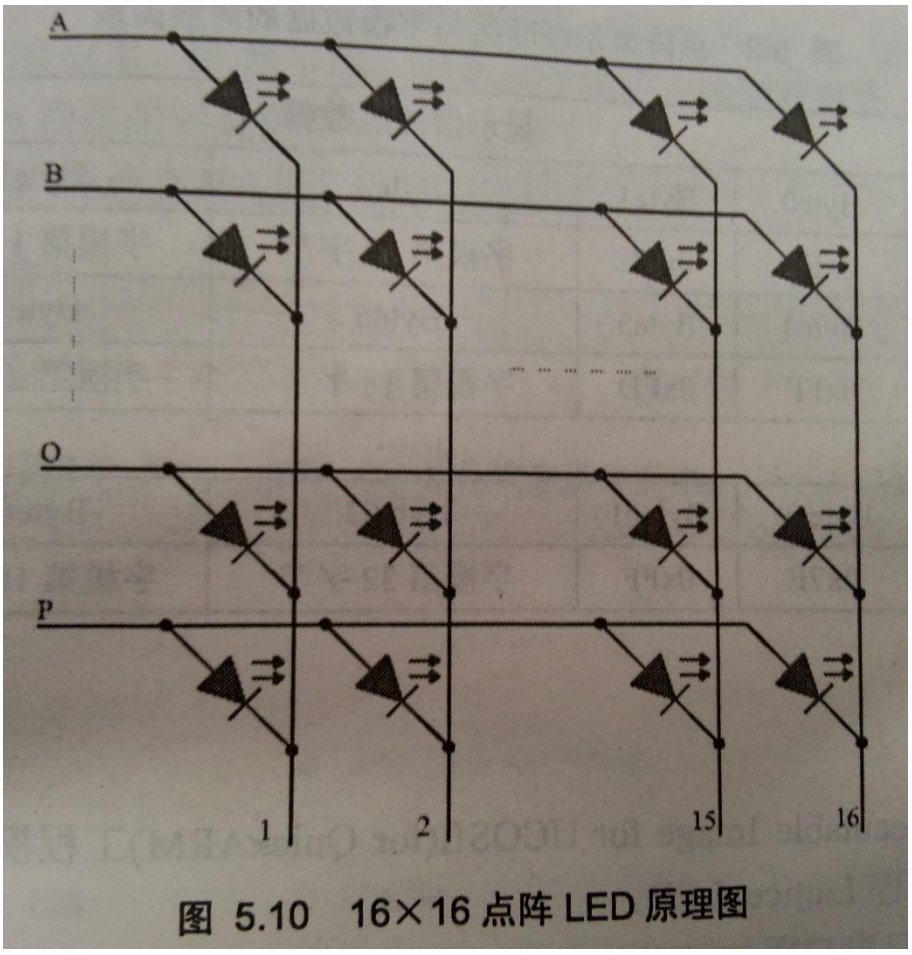


图 5.10　16×16 点阵 LED 原理图

本实验中，我们采用“列扫描”的方式。由于主板上的 16X16 点阵 LED 由 74HC595 驱动，所以显示一列数据需要发送 4 字节数据，一屏数据所需要的缓冲区大小为 64 字节。

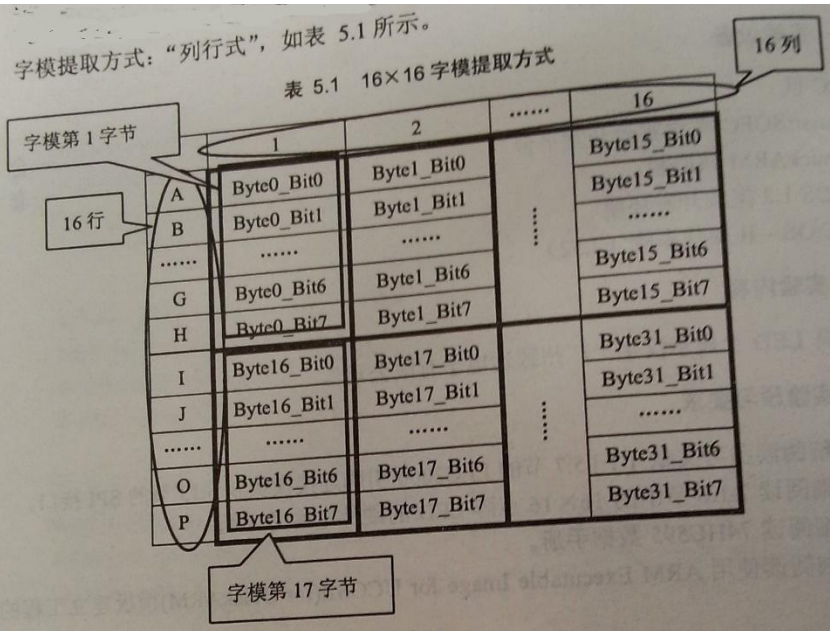
程序中开辟了两块 64 字节的缓冲区：LED_Point_Buf 和 LED_Point_Buf_Bak.

LED_Point_Buf：一屏数据的显示缓冲区

LED_Point_Buf_Bak：显示数据备份缓冲区，用来更新显示缓冲区，实现画面移动。

字模提取方式：“列行式”，如表 5.1 所示。

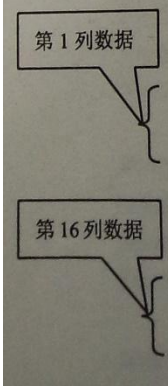
表 5.1　16×16 字模提取方式



	1	2	……	16
A	Byte0_Bit0	Byte1_Bit0		Byte15_Bit0
B	Byte0_Bit1	Byte1_Bit1		Byte15_Bit1
……	……	……		……
G	Byte0_Bit6	Byte1_Bit6		Byte15_Bit6
H	Byte0_Bit7	Byte1_Bit7		Byte15_Bit7
I	Byte16_Bit0	Byte17_Bit0		Byte31_Bit0
J	Byte16_Bit1	Byte17_Bit1		Byte31_Bit1
……	……	……		……
O	Byte16_Bit6	Byte17_Bit6		Byte31_Bit6
P	Byte16_Bit7	Byte17_Bit7		Byte31_Bit7

用 16×16 字模软件提取一个字，会得到 32 个字节的数据，而一个字在 16×16 点阵 LED 上显示需要 64 个字节，所以还要将字模数据转换为显示数据。在使用 74HC595 传送数据时，一列信息需要 4 字节数据，发送顺序为：列（16～9）＋列（8～1）＋行（P～I）＋行（H～A）。16×16 点阵 LED 显示缓冲区与字模数据的对应关系如表 5.2 所示。

表 5.2　点阵显示缓冲区与字模数据的对应关系



	显示缓冲区数据			
字节	Byte0	Byte1	Byte2	Byte3
数据	0xFF	0xFE	字模第 17 字节	字模第 1 字节
字节	Byte4	Byte5	Byte6	Byte7
数据	0xFF	0xFD	字模第 18 字节	字模第 2 字节
	……			
字节	Byte60	Byte61	Byte62	Byte63
数据	0x7F	0xFF	字模第 32 字节	字模第 16 字节

参考程序：

点阵 LED 显示实验的参考程序如程序清单 5.12、程序清单 5.13 所示。

程序清单 5.12　点阵 LED 显示实验参考程序——main 函数

```
#include "config.h"
#include "stdlib.h"
#include "GPIO.h"
#include "Lattice.h"                    // 点阵 LED 显示字库

#define    TaskStkLengh    128          // 定义用户任务的堆栈长度
#define    Lattice_CS      (1 << 18)    // 点阵 LED 控制芯片，74HC595 片选信号线，P0.18
/*定义 SPI 的管脚*/
```

```
#define    SCK0      (1<<8)                  // SCK0   为 P0.4 的第二功能
#define    MISO0     (1<<10)                 // MISO0  为 P0.5 的第二功能
#define    MOSI0     (1<<12)                 // MOSI0  为 P0.6 的第二功能
#define    SSEL0     (1<<14)                 // SSEL0  为 P0.7 的第二功能
OS_STK   TaskStk [TaskStkLengh];             // Define the Task0 stack 定义用户任务 0 的堆栈
uint8    LED_Point_Buf[64];                  // 点阵 LED 缓冲区
uint8    LED_Point_Buf_Bak[64];              // 点阵 LED 备份缓冲区，用来作为移动显示
uint8    Remove_Times;                       // 移动显示的列数
uint8    Remove_Bytes;                       // 需要显示的"字"的个数
void  Task0(void *pdata);                    // Task0 任务 0
/*********************************************************************************************
** 函数名称：main()
** 函数功能：点阵 LED 显示实验。
** 说明：  P0.4 短接 SCK，P0.6 短接 SI，P0.18 短接 STR，P0.7 短接 BEEP，JP7 短接。
*********************************************************************************************/
int main (void)
{
    OSInit ();
    OSTaskCreate (Task0,(void *)0, &TaskStk[TaskStkLengh - 1], 2);
    OSStart ();
    return 0;
}
```

程序清单 5.13　点阵 LED 显示实验参考程序——点阵 LED 显示任务

```
/*********************************************************************************************
** 任务名称：Task0
** 任务功能：点阵 LED 显示。
*********************************************************************************************/
void Task0(void *pdata)
{
    uint32   dly;
    TargetInit ();
    pdata = pdata;
    P0_GPIOInit(Lattice_CS,1);          // 将 74HC595（点阵 LED）的片选线设置为 GPIO 输出模式
    SPI_Init(1,400000,0);               // SPI 主机接口，通信速率 400K，中断禁止
    Change_Buf((uint8 *)(&(Lattice_TBL1[0])));       // 第一个字是"广"
    for(dly = 0;dly < 50;dly++)  Disp_Buf_Hard();
    while(1)
    {
        Update_Buf();                   // 更新显示信息，实现画面的移动
        OSTimeDly(1);
        for(dly = 0;dly < 50;dly++)     // 一屏数据显示 50 次
            Disp_Buf_Hard();
```

7. 相关软件下载

无

8. 实验报告要求

1) **实验内容**：（实验题目，实验要求）
    a) 实验题目：
    b) 实验要求：
2) **实验过程**：
    a) （实验原理，设计思路，主要相关参数设置和函数功能说明，实验解决方案。。。）
3) **实验结果：**（结果分析，心得体会。。。）

注：共三大项，具体每一项可根据自己的报告内容分条叙述。（括号里内容仅供参考）

实验 2：LCD 显示

1. 实验目的

1) 掌握 LPC2000 系列 ARM7 处理器中 EMC 的基本原理及应用。
2) 掌握 ST7920 的操作方法。

2. 实验环境

硬件：PC 机
SmartSOPC 教学实验开发平台
QuickARM 核心板
软件：ADS1.2 集成开发环境
μC/OS-ⅠⅠ 操作系统（2.52）

3. 实验内容

在 128X64 的点阵液晶上显示 5 屏数据，按下 KEY1，显示上一屏数据；按下 KEY5，显示下一屏数据。

4. 实验步骤

（1）启动 ADS1.2，使用 ARM Executabl Image for USOCII（for QuickARM）工程模板在 Demo Program 目录下建立工程 LCD_Disp。
（2）在 ADS 的项目窗口添加 GPIO 和 LCD 文件组；然后把 Driver\GPIO \目录下的 GPIO.h 和 GPIO.c 文件添加到 GPIO 文件组，将 Software Package\LCD 目录下的 ST7920.c、ST7920.h 和 LCD.h 保存到 LCD 文件组。
（3）打开工程窗口 user 组中的 main.c 文件，编写实验程序并保存。
（4）编译连接工程，并启动 AXD 进行 JTAG 仿真调试。
（5）跳线 JP6 的 L_EN 短接 LCD_EN、P1.21 短接 LCD LIGHT、P4.0 短接 KEY5、P0.18 短接 KEY1.全速运行程序，程序将会在 main.c 的主函数中停止（因为 main 函数起始处默认设置有断点）。
（6）全速运行程序，按 KEY1 和 KEY5，并观察 LCD 的显示。

5. 实验方式

每位同学独立上机编程实验，实验指导教师现场指导。

6. 参考内容

参考程序：

程序清单 5.14　LCD 显示实验——main 函数

```
#include "config.h"
#include "stdlib.h"
#include "GPIO.h"                       //  GPIO 底层驱动
#include "LCD.h"                        //  MG12864 图形字模
#include "ST7920.h"                     //  ST7920 操作软件包

#define    TaskStkLengh    128          //  定义用户任务的堆栈长度
#define    KEY1            (1 << 18)    //  KEY1 控制引脚 P0.18
#define    KEY5            (1 <<  4)    //  KEY5 控制引脚 P0.4
#define    Next            1            //  按下 KEY5，显示下一屏
#define    Pre             5            //  按下 KEY1，显示上一屏

OS_STK   TaskStk [TaskStkLengh];        //  Define the Task0 stack  定义用户任务 0 的堆栈
OS_STK   Task1Stk [TaskStkLengh];       //  Define the Task1 stack  定义用户任务 1 的堆栈
OS_EVENT     *OS_Mbox;                  //  设置一个邮箱，用来在 Task0 与 Task1 之间进行通信

void  Task0(void *pdata);               //  Task0  任务 0
void  Task1(void *pdata);               //  Task1  任务 1

int main (void)
{
    OSInit ();
    OS_Mbox = OSMboxCreate ((void *)0);             // 建立一个邮箱，用来传递按键信息
    OSTaskCreate (Task0,(void *)0, &TaskStk[TaskStkLengh - 1], 2);
    OSStart ();
    return 0;
}
```

程序清单 5.15　LCD 显示实验——LCD 显示任务

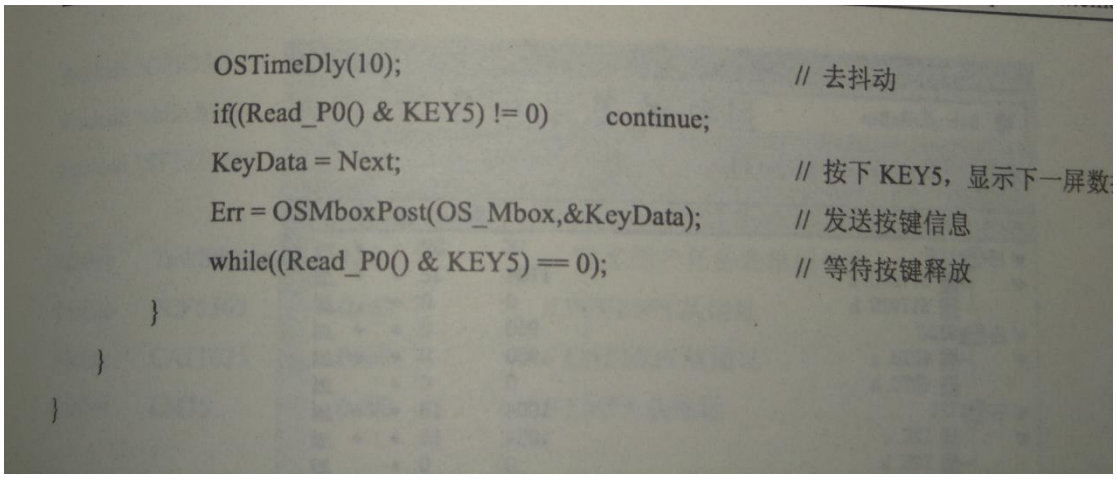
```
/*********************************************************************************************
** 任务名称：Task0
** 任务功能：LCD 显示
*********************************************************************************************/
```

```
void Task0 (void *pdata)
{
    uint8 ret_code;
    uint8 *Rcv_Data;
    uint8 Err;
    uint8 picture = 0;                          // 显示画面的编号 0 ～ 4

    pdata = pdata;
    TargetInit ();
    P1_GPIOInit(LCD_LIGHT,1);                   // LCD 背光控制引脚设置为 GPIO 输出模式
    P0_GPIOInit(KEY1 | KEY5,0);                 // KEY1 和 KEY5 控制引脚设置为 GPIO 输入模式
    OSTaskCreate (Task1,(void *)0, &Task1Stk[TaskStkLengh - 1], 3);

    LCD_init( );
    LCD_BacklightCon(LCD_BACKLIGHT_ON);         // 打开背光
    ST7920_ClearScreen();
    LCD_PlotPic(PowerOnPicture);                // 显示第一屏画面
    while(1)
    {
        Rcv_Data = OSMboxPend(OS_Mbox,0,&Err);
        if(*Rcv_Data == Next)                   // 按下 KEY5，显示下一屏画面
            picture = (picture + 1) % 5;
        if(*Rcv_Data == Pre)                    // 按下 KEY1，显示上一屏画面
        {
            if(picture == 0)   picture = 4;
            else               picture--;
        }
        switch(picture)
        {
            case 0:                             // 显示第 1 幅图片
                ST7920_ClearScreen();
                LCD_PlotPic(PowerOnPicture);
                break;
            case 1:                             // 显示第 2 幅图片
                ST7920_ClearScreen();
                LCD_PlotPic(ZhiYuanLogo);
                break;
            case 2:                             // 显示第 3 幅图片
                ST7920_ClearScreen();
                LCD_PlotPic(Logo);
                break;
            case 3:                             // 显示第 4 幅图片
                ST7920_ClearScreen();
                ST7920_ClearGDRAM();
```

```
                ret_code = LCD_printf(0,0,(uint8 *)"CPU:LPC2220",strlen("CPU:LPC2200"));
                ret_code = LCD_printf(1,0,(uint8 *)"片内 RAM:64K 字节",
                                      strlen("片内 RAM:64K 字节"));
                ret_code = LCD_printf(2,0,(uint8 *)"片外 RAM:128K 字节",
                                      strlen("片外 RAM:128K 字节"));
                ret_code = LCD_printf(3,0,(uint8 *)"片外 FLASH:2M 字节",
                                      strlen("片外 FLASH:2M 字节"));
                break;
            case 4:                                   // 显示第 5 幅图片
                ST7920_ClearScreen();
                LCD_printf(0,0,(uint8 *)"欢迎使用 EDA/SOPC/DSP/MCU/ARM/SOC
                           教学实验开发平台广州致远电子公司",
                           strlen("欢迎使用 EDA/SOPC/DSP/MCU/ARM/SOC
                           教学实验开发平台广州致远电子公司"));
                break;
            default:
                break;
        }
    }
}
```

程序清单 5.16 LCD 显示实验——按键检测任务

```
/*********************************************************************************************
** 任务名称：Task1
** 任务功能：按键检测任务。
*********************************************************************************************/
void Task1 (void *pdata)
{
    uint8 KeyData;
    uint8 Err;
    pdata = pdata;
    while(1)
    {
        if((Read_P0() & KEY1) == 0)                   // 检测按键
        {
            OSTimeDly(10);                            // 去抖动
            if((Read_P0() & KEY1) != 0)     continue;
            KeyData = Pre;                            // 按下 KEY1，显示上一屏数据
            Err = OSMboxPost(OS_Mbox,&KeyData);       // 发送按键信息
            while((Read_P0() & KEY1) == 0);           // 等待按键释放
        }
        if((Read_P0() & KEY5) == 0)
        {
```



```
            OSTimeDly(10);                                  // 去抖动
            if((Read_P0() & KEY5) != 0)      continue;
            KeyData = Next;                                 // 按下 KEY5，显示下一屏数
            Err = OSMboxPost(OS_Mbox,&KeyData);             // 发送按键信息
            while((Read_P0() & KEY5) == 0);                 // 等待按键释放
        }
    }
}
```

7. 相关软件下载

无

8. 实验报告要求

1) **实验内容**：（实验题目，实验要求）
   a) 实验题目：
   b) 实验要求：
2) **实验过程**：
   a) （实验原理，设计思路，主要相关参数设置和函数功能说明，实验解决方案。。。）
3) **实验结果：**（结果分析，心得体会。。。）

注：共三大项，具体每一项可根据自己的报告内容分条叙述。（括号里内容仅供参考）

实验 3：七段数码管显示实验

1. 实验目的

掌握七段数码管的驱动方法

2. 实验环境

硬件：PC 机
SmartSOPC 教学实验开发平台
QuickARM 核心板
软件：ADS1.2 集成开发环境
μC/OS-II 操作系统（2.52）

3. 实验内容

在数码管上显示“LPC2220F”

4. 实验步骤

1) 使用 ARM Executabl Image for USOCII（for QuickARM）工程模板在 Demo Program 目录下建立工程 Digital_LED.
2) 在 ADS 的项目窗口添加 GPIO 和 LED 的文件组，然后把 Driver\GPIO \ 目录下的 GPIO.h 和 GPIO.c 文件添加到 GPIO 文件组，将 Software Package\LED 目录下的 LED.h 保存到 LED 文件组。
3) 调整外部总线速率，修改 Startup.s 文件中的 ResetInit 函数，BCFG0 和 BCFG1 寄存器的数据更改位 0x10001460
   a) LDR R0，=BCFG0
   b) LDR R1，=0x10001460
   c) STR R1，[R0]
   d) LDR R0，=BCFG1
   e) LDR R1，=0x10001460
   f) STR R1，[R0]
4) 打开工程窗口 user 组中的 main.c 文件，编写实验程序并保存。
5) 编译连接工程，并启动 AXD 进行 JTAG 仿真调试。
6) 全速运行程序，观察数码管上显示的数据。

5. 实验方式

每位同学独立上机编程实验，实验指导教师现场指导。

6. 参考内容

实验预习要求：

仔细阅读 ARM Executabl Image for USOCII（for QuickARM）模板建立工程的步骤。

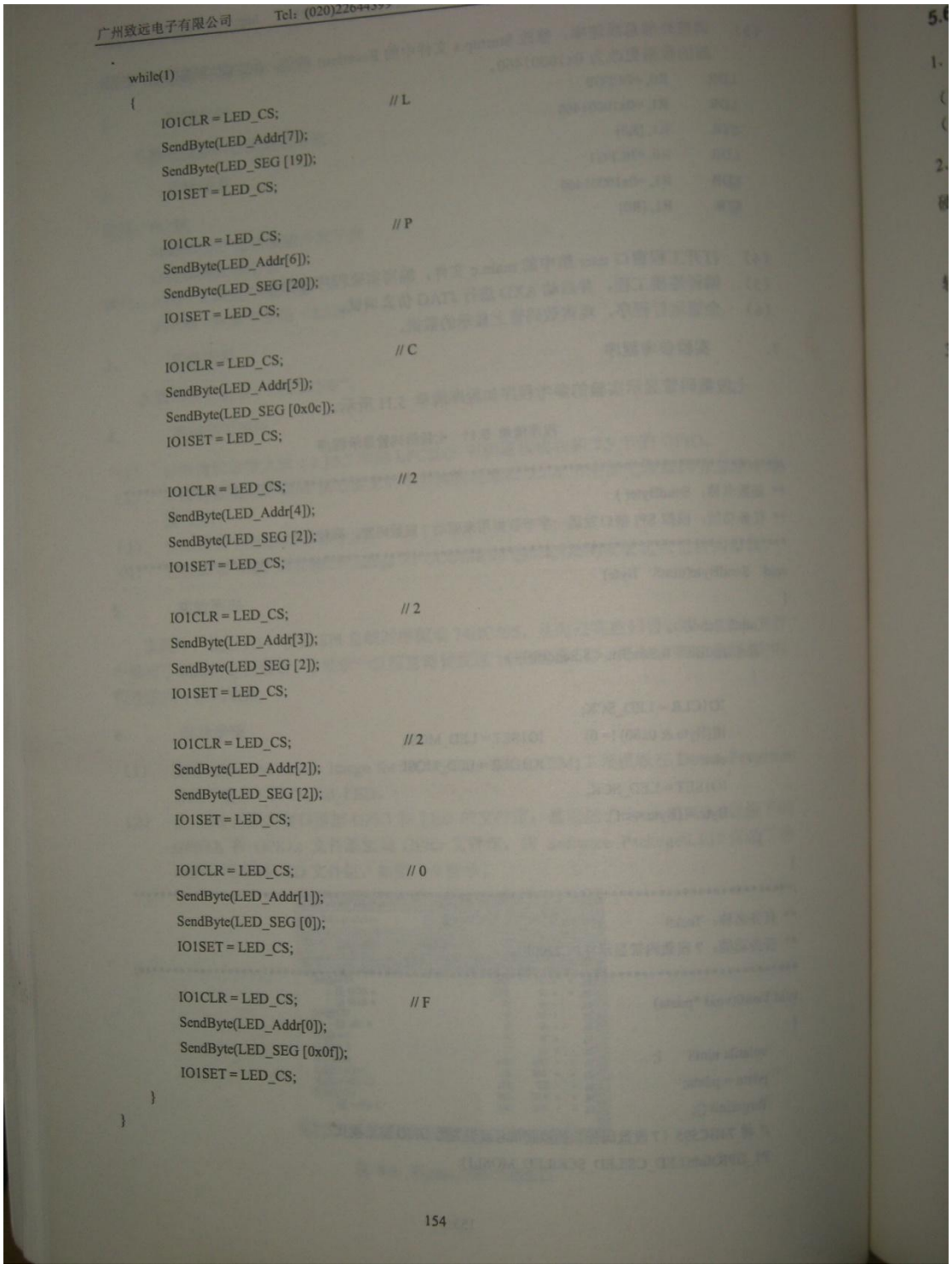
实验原理：

实验中，使用 IO 口模拟 SPI 总线时序驱动 74HC595，从而点亮数码管。在 LED.h 文件中给出了相应的字模数据。每显示一位信息需要发送 16 位数据

（即两个字节），发送顺序：位置数据+段码数据。

参考程序：

```
/*********************************************************************************************
** 函数名称：SendByte( )
** 任务功能：模拟 SPI 接口发送一字节数据用来驱动 7 段数码管。高位先发
*********************************************************************************************/
void  SendByte(uint8   Byte)
{
    uint8 SendBit;
    for(SendBit = 0;SendBit < 8;SendBit++)
    {
        IO1CLR = LED_SCK;
        if((Byte & 0x80) != 0)        IO1SET = LED_MOSI;
        else                          IO1CLR = LED_MOSI;
        IO1SET = LED_SCK;
        Byte = (Byte << 1);
    }
}
/*********************************************************************************************
** 任务名称：Task0
** 任务功能：7 段数码管显示"LPC2200F"。
*********************************************************************************************/
void Task0(void *pdata)
{
    volatile uint8      i;
    pdata = pdata;
    TargetInit ();
    // 将 74HC595（7 段数码管）的控制信号线设置为 GPIO 输出模式
    P1_GPIOInit(LED_CS|LED_SCK|LED_MOSI,1);
```



```
while(1)
{
    IO1CLR = LED_CS;                    // L
    SendByte(LED_Addr[7]);
    SendByte(LED_SEG [19]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // P
    SendByte(LED_Addr[6]);
    SendByte(LED_SEG [20]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // C
    SendByte(LED_Addr[5]);
    SendByte(LED_SEG [0x0c]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // 2
    SendByte(LED_Addr[4]);
    SendByte(LED_SEG [2]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // 2
    SendByte(LED_Addr[3]);
    SendByte(LED_SEG [2]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // 2
    SendByte(LED_Addr[2]);
    SendByte(LED_SEG [2]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // 0
    SendByte(LED_Addr[1]);
    SendByte(LED_SEG [0]);
    IO1SET = LED_CS;

    IO1CLR = LED_CS;                    // F
    SendByte(LED_Addr[0]);
    SendByte(LED_SEG [0x0f]);
    IO1SET = LED_CS;
  }
}
```


7. 相关软件下载

无

8. 实验报告要求

1) **实验内容**：（实验题目，实验要求）
    a) 实验题目：
    b) 实验要求：
2) **实验过程**：
    a) （实验原理，设计思路，主要相关参数设置和函数功能说明，实验解决方案。。。）
3) **实验结果：**（结果分析，心得体会。。。）

注：共三大项，具体每一项可根据自己的报告内容分条叙述。（括号里内容仅供参考）

实验 4：A/D 转换实验

1. 实验目的

通过实验，掌握 LPC2000 系列 ARM7 微控制器的 WDT 功能及其使用方法。

2. 实验环境

硬件：PC 机
SmartSOPC 教学实验开发平台
QuickARM 核心板
软件：ADS1.2 集成开发环境

3. 实验内容

初始化并运行 WDT，然后控制核心板上的 LED1 显示并进行喂狗处理，若核心板上的 KEY1 被按下，CPU 进入死循环，等待 WDT 溢出复位

4. 实验步骤

1) 启动 ADS1.2，使用 ARM Executabl Image for for QuickARM 工程模板建立一个工程 WDT。
2) 在 main.c 文件中，编写实验程序
3) 选用 RelOutChip 生成目标，然后编译连接工程
4) 将核心板插到 SmartSOPC 上。将 SmartSPOC 板上的 A2 区的 JP6 的 KEY1 和 BEEP 短接，将 A6 区的 JP7 短接。将核心板上的 JP1(ISP)跳线断开。
5) 选择 Project-->Debug，启动 AXD 进行 JTAG 仿真调试。
6) 全速运行程序，蜂鸣器首先鸣叫一声，然后 LED1 闪烁，当按下 KEY1 后，WDT 溢出复位。当 WDT 复位后，停止程序(AXD 软件按下 Stop 按钮)将会停在不正确的地址上。

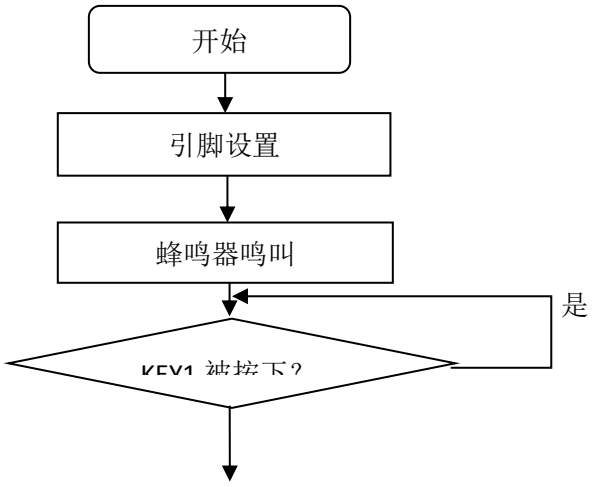
5. 实验方式

每位同学独立上机编程实验，实验指导教师现场指导。

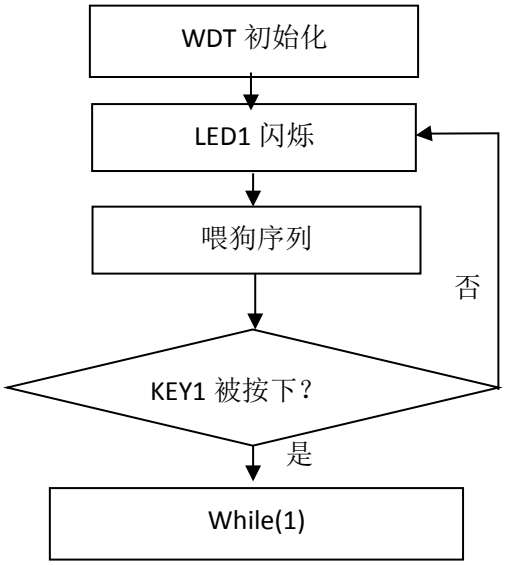
6. 参考内容

实验原理：

WDT 复位试验程序的实现流程如下图所示



否



参考程序：

```
#include "config.h"

#define BEEP 1<<7     //P0.7 脚控制蜂鸣器
#define KEY1 1<<18    //P0.18 脚接 Key1
#define LED1 1<<22    //P1.22 脚控制核心板上的 LED1
/**
*名称: void Delay(uint32 dly)
*功能:延时函数
*参数:uint32 dly      延时时间
**/
void Delay(uint32 dly){
      uint32 i;
      for(;dly>0;dly--){
            for(i=0;i<50;i++);
      }
}
/**
*名称: void Beep(uint32 t)
*功能:交流蜂鸣器鸣叫函数
*参数:uint32 t      鸣叫时间
**/
void Beep(uint32 t){
      for(;t>0;t--){
            IO0CLR=BEEP;
            Delay(2);
            IO0SET=BEEP;
            Delay(2);
      }
```

```
}
/**
*名称: void WDT_Feed(void)
*功能:喂狗函数
*说明:喂狗序列不能被打断，因此必须保证在执行喂狗序列时禁止中断响应
**/
void WDT_Feed(void) {
    WDFEED=0xAA;
    WDFEED=0x55;
}
/**
*名称: main()
*功能:看门狗溢出实验
*说明:将 SmartSPOC 板上的 A2 区的 JP6 的 KEY1 和 BEEP 短接，将 A6 区的 JP7 短
接
**/
int main(void)
{
    PINSEL2&=(~(1<<3));        //使能 P1.16~P1.25 为 GPIO
    IO1DIR=LED1;                 //LED1 设置为输出
    IO1SET=LED1;                 //熄灭 LED1

    PINSEL0=0;
    IO0DIR=BEEP;
    IO0SET=BEEP;                 //关闭蜂鸣器

    Beep(100);                   //蜂鸣器鸣叫
    while((IO0PIN&KEY1)==0);//等待 KEY1 按键弹起

    WDTC=(Fpclk)/4;              //设定溢出时间为 1 秒钟、time=((Fpclk)/4)*tpclk*4
    WDMOD=0x03;
    WDT_Feed();

    while(1){
        while((IO0PIN&KEY1)==0);//检测 KEY1 是否按下。
                                //若被按下了则进入死循环，等待 WDT 溢
出
        WDT_Feed();

        if(IO1SET&LED1){          //LED 闪烁
            IO1CLR=LED1;
        }else{
            IO1SET=LED1;
        }
```

```
            Delay(500);
        }
        return 0;
}
```

思考：

(1) WDTINT 终端标志可以使用软件清 0 吗？

(2)若系统进入掉电模式，WDT 还能继续运行下去吗？

7. 相关软件下载

无

8. 实验报告要求

1) **实验内容**：（实验题目，实验要求）
   a) 实验题目：
   b) 实验要求：
2) **实验过程**：
   a) （实验原理，设计思路，主要相关参数设置和函数功能说明，实验解决方案。。。）
3) **实验结果：**（结果分析，心得体会。。。）

注：共三大项，具体每一项可根据自己的报告内容分条叙述。（括号里内容仅供参考）