湖南科技大学计算机科学与工程学院

嵌入式 实验报告

**专业班级：** 22计科6班

**姓 名：** 周俊哲

**学 号：** 2205010711

**指导教师：** 黄卫红

**时 间**：

**地 点**：

|  |  |  |  |
| --- | --- | --- | --- |
| 指导教师评语：    **成绩： 等级：**  **签名：**  **年 月 日** | | | |
| 实验名称 | | 嵌入式操作系统应用设计 | | | |
| 实验性质  （必修、选修） | |  | 实验类型（验证、设计、创新、综合） |  | |
| 实验课时 | | 16 | 实验日期 | 1、  2、  3、  4、 | |
| 实验仪器设备以及实验软硬件要求 | | 硬件：EBD-A23学习开发板（或其带数码管和LED的开发板）。  软件：相关可用开发环境。 | | | |
| 实验目的 | | 了解嵌入式实时操作系统的基本概念和特点，并能够运用实时操作系统编写嵌入式程序。同时，通过本实验让学生对实时性和可靠性的要求有更深入的理解。实验的内容包括以下几个方面：  1. 实时操作系统的概念和基本特点，嵌入式操作系统移植；  2. 实时操作系统的任务调度机制；  3. 实时操作系统的信号量和消息队列；  4. 在实时操作系统uC/os上编写一个简单的示例程序包含三个以上任务，在信号量、邮箱、消息队列任等OS功能中选一至两项应用。 | | | |
| 1、实验内容（实验原理、运用的理论知识、算法、程序、步骤和方法）  1 专题索引  1  1.1 专题 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  1  2 结构体索引  1  2.1 结构体 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  1  3 专题文档  2  3.1 系统配置 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  2  3.1.1 详细描述 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  2  3.2 全局变量 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  2  3.2.1 详细描述 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  2  3.3 硬件抽象层 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  3  3.3.1 详细描述 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  3  3.3.2 函数说明 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  3  4 结构体说明  4  4.1 MSG结构体 参考 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  4  4.1.1 详细描述 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  5  4.1.2 结构体成员变量说明 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  5  Index 7  1 专题索引  1.1 专题  这里是所有专题及其简介:  系统配置 2  全局变量 2  硬件抽象层 3  2 结构体索引  2.1 结构体  这里列出了所有结构体， 并附带简要说明:  MSG  任务间通信消息结构体 4    3 专题文档  3.1 系统配置  系统全局配置参数  宏定义  • #define TASK STK SIZE 512  任务栈大小  • #define MSG QUEUE SIZE 10  消息队列容量  • #define MEM POOL SIZE 20  内存分区大小  3.1.1 详细描述  系统全局配置参数  3.2 全局变量  系统全局资源定义  变量  • OS STK LED TaskStk [TASK STK SIZE]  LED任务堆栈  • OS STK Display TaskStk [TASK STK SIZE]  显示任务堆栈  • OS STK Input TaskStk [TASK STK SIZE]  输入任务堆栈  • OS STK Monitor TaskStk [TASK STK SIZE]  监控任务堆栈  • OS EVENT ∗ MsgQueue  消息队列指针  • OS MEM ∗ MsgPool  内存分区指针  • OS EVENT ∗ LedSem  LED控制信号量  3.2.1 详细描述  系统全局资源定义    3.3 硬件抽象层  硬件平台相关函数实现  函数  • void BSP Init (void)  初始化硬件平台  • void BSP LED Set (INT8U pattern)  设置LED显示模式  • void BSP 7Seg Display (INT8U num)  数码管显示数字  • INT8U BSP Key Scan (void)  扫描按键状态  • void BSP WDT Feed (void)  喂看门狗  3.3.1 详细描述  硬件平台相关函数实现  3.3.2 函数说明  BSP 7Seg Display()  void BSP 7Seg Display (  INT8U num)  数码管显示数字  参数  警告  输入超过9时将不更新显示  BSP Init()  void BSP Init (  void )  初始化硬件平台  初始化系统时钟、 GPIO、 外设等  注解  必须在所有任务创建前调用    BSP Key Scan()  INT8U BSP Key Scan (  void )  扫描按键状态  返回  INT8U 按键值（ 0-9） 或0xFF表示无按键  返回当前按下的按键编号  BSP LED Set()  void BSP LED Set (  INT8U pattern)  设置LED显示模式  参数  根据输入位图控制LED状态：  • bit0: LED1  • bit1: LED2  • bit2: LED3  • bit3: LED4  BSP WDT Feed()  void BSP WDT Feed (  void )  喂看门狗  重置看门狗定时器防止系统复位  4 结构体说明  4.1 MSG结构体 参考  任务间通信消息结构体    成员变量  • INT8U display value  显示数值 取值范围0-9  • INT8U led pattern  4.1.1 详细描述  任务间通信消息结构体  用于在LED任务和显示任务之间传递控制参数  4.1.2 结构体成员变量说明  led pattern  INT8U MSG::led pattern  LED显示模式  • 0x01: 单闪模式  • 0x02: 双闪模式  • 0x03: 流水灯模式  该结构体的文档由以下文件生成:  • shiyan7.c    Index  BSP 7Seg Display  硬件抽象层, 3 BSP Init  硬件抽象层, 3 BSP Key Scan  硬件抽象层, 3 BSP LED Set  硬件抽象层, 4 BSP WDT Feed  硬件抽象层, 4  led pattern  MSG, 5  MSG, 4  led pattern, 5  全局变量, 2  硬件抽象层, 3  BSP 7Seg Display, 3 BSP Init, 3  BSP Key Scan, 3 BSP LED Set, 4 BSP WDT Feed, 4  系统配置, 2  程序代码如下: /\*\*  \* @file main.c  \* @brief 嵌入式实验7：基于uC/OS-II的多任务协作系统  \* @author 周俊哲  \* @date 2025-05-18  \* @version 2.0  \*  \* @details  \* 本程序实现基于uC/OS-II实时操作系统的多任务协作系统，包含：  \* - LED控制任务  \* - 数码管显示任务  \* - 按键处理任务  \* - 系统监控任务  \* 使用消息队列和信号量实现任务间通信  \*/  #include "bsp.h" // 硬件抽象层  #include "includes.h"  /\*\*  \* @defgroup 系统配置  \* @brief 系统全局配置参数  \* @{  \*/  #define TASK\_STK\_SIZE 512 ///< 任务栈大小  #define MSG\_QUEUE\_SIZE 10 ///< 消息队列容量  #define MEM\_POOL\_SIZE 20 ///< 内存分区大小  /\*\* @} \*/ // end of 系统配置  /\*\*  \* @defgroup 全局变量  \* @brief 系统全局资源定义  \* @{  \*/  OS\_STK LED\_TaskStk[TASK\_STK\_SIZE]; ///< LED任务堆栈  OS\_STK Display\_TaskStk[TASK\_STK\_SIZE]; ///< 显示任务堆栈  OS\_STK Input\_TaskStk[TASK\_STK\_SIZE]; ///< 输入任务堆栈  OS\_STK Monitor\_TaskStk[TASK\_STK\_SIZE]; ///< 监控任务堆栈  OS\_EVENT \*MsgQueue; ///< 消息队列指针  OS\_MEM \*MsgPool; ///< 内存分区指针  OS\_EVENT \*LedSem; ///< LED控制信号量  /\*\* @} \*/ // end of 全局变量  /\*\*  \* @struct MSG  \* @brief 任务间通信消息结构体  \* @ingroup 数据结构  \*  \* @details 用于在LED任务和显示任务之间传递控制参数  \*/  typedef struct {  INT8U display\_value; ///< 显示数值 取值范围0-9  INT8U led\_pattern; /\*\*< LED显示模式  - 0x01: 单闪模式  - 0x02: 双闪模式  - 0x03: 流水灯模式 \*/  } MSG;  /\* 函数前置声明 \*/  static void App\_CreateResources(void);  void LED\_Task(void \*pdata);  void Display\_Task(void \*pdata);  void Input\_Task(void \*pdata);  void Monitor\_Task(void \*pdata);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief 系统主函数  \* @details 程序入口，完成系统初始化和任务创建  \*  \* @return int 程序返回值（实际不会返回）  \*  \* @note 本函数不会正常返回，将一直运行在操作系统的任务调度中  \*/  int main(void) {  /\* 硬件初始化 \*/  BSP\_Init();  /\* 初始化uC/OS-II \*/  OSInit();  /\* 创建系统资源 \*/  App\_CreateResources();  /\* 创建应用任务 \*/  OSTaskCreate(LED\_Task, NULL, &LED\_TaskStk[TASK\_STK\_SIZE - 1], 3);  OSTaskCreate(Display\_Task, NULL, &Display\_TaskStk[TASK\_STK\_SIZE - 1], 4);  OSTaskCreate(Input\_Task, NULL, &Input\_TaskStk[TASK\_STK\_SIZE - 1], 5);  OSTaskCreate(Monitor\_Task, NULL, &Monitor\_TaskStk[TASK\_STK\_SIZE - 1], 2);  /\* 启动多任务调度 \*/  OSStart();  return 0;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 系统资源初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief 创建系统资源  \* @ingroup 系统初始化  \*  \* @details 初始化系统运行所需的通信资源和内存管理：  \* - 创建消息内存池  \* - 创建消息队列  \* - 创建信号量  \*  \* @warning 必须在任务创建前调用  \*/  static void App\_CreateResources(void) {  INT8U err;  void \*MemPartition[MEM\_POOL\_SIZE];  /\* 创建内存分区 \*/  MsgPool = OSMemCreate(MemPartition, MEM\_POOL\_SIZE, sizeof(MSG), &err);  /\* 创建消息队列 \*/  MsgQueue = OSQCreate(&MemPartition[0], MSG\_QUEUE\_SIZE);  /\* 创建信号量 \*/  LedSem = OSSemCreate(0);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LED控制任务 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief LED控制任务函数  \* @param pdata 任务参数（未使用）  \* @ingroup 应用任务  \*  \* @details 任务工作流程：  \* 1. 等待信号量触发  \* 2. 从消息队列获取控制参数  \* 3. 根据模式更新LED显示  \* 4. 延时500ms  \*  \* @note LED模式由消息中的led\_pattern字段决定  \*/  void LED\_Task(void \*pdata) {  INT8U err;  MSG \*pmsg;  (void)pdata; // 消除未使用参数警告  while (1) {  /\* 等待信号量 \*/  OSSemPend(LedSem, 0, &err);  /\* 从消息队列获取消息 \*/  pmsg = (MSG \*)OSQPend(MsgQueue, 0, &err);  /\* 执行LED控制 \*/  switch (pmsg->led\_pattern) {  case 0x01: // 单闪模式  BSP\_LED\_Set(0x01);  OSTimeDlyHMSM(0, 0, 0, 500);  BSP\_LED\_Set(0x00);  break;  case 0x02: // 双闪模式  BSP\_LED\_Set(0x03);  OSTimeDlyHMSM(0, 0, 0, 250);  BSP\_LED\_Set(0x00);  OSTimeDlyHMSM(0, 0, 0, 250);  break;  case 0x03: // 流水灯模式  for (INT8U i = 0; i < 4; i++) {  BSP\_LED\_Set(0x01 << i);  OSTimeDlyHMSM(0, 0, 0, 200);  }  break;  default:  break;  }  /\* 释放消息内存 \*/  OSMemPut(MsgPool, pmsg);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 数码管显示任务 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief 数码管显示任务函数  \* @param pdata 任务参数（未使用）  \* @ingroup 应用任务  \*  \* @details 任务工作流程：  \* 1. 从消息队列获取显示数值  \* 2. 更新数码管显示  \* 3. 延时1秒  \*  \* @warning 显示值超过9时将不更新显示  \*/  void Display\_Task(void \*pdata) {  INT8U err;  MSG \*pmsg;  (void)pdata;  while (1) {  pmsg = (MSG \*)OSQPend(MsgQueue, 0, &err);  if (pmsg->display\_value <= 9) {  BSP\_7Seg\_Display(pmsg->display\_value);  }  OSMemPut(MsgPool, pmsg);  OSTimeDlyHMSM(0, 0, 1, 0);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 按键处理任务 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief 按键处理任务函数  \* @param pdata 任务参数（未使用）  \* @ingroup 应用任务  \*  \* @details 任务工作流程：  \* 1. 扫描按键状态  \* 2. 检测到有效按键时创建消息  \* 3. 发送消息到队列  \* 4. 触发信号量  \* 5. 延时100ms防抖  \*  \* @note 按键值直接作为显示数值，LED模式循环切换  \*/  void Input\_Task(void \*pdata) {  INT8U key\_val;  INT8U err;  MSG \*pmsg;  (void)pdata;  while (1) {  key\_val = BSP\_Key\_Scan();  if (key\_val != 0xFF) {  pmsg = (MSG \*)OSMemGet(MsgPool, &err);  if (err == OS\_NO\_ERR) {  pmsg->display\_value = key\_val;  pmsg->led\_pattern = 0x01 + (key\_val % 3);  OSQPost(MsgQueue, pmsg);  OSSemPost(LedSem);  }  }  OSTimeDlyHMSM(0, 0, 0, 100);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 系统监控任务 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @brief 系统监控任务函数  \* @param pdata 任务参数（未使用）  \* @ingroup 系统任务  \*  \* @details 任务工作流程：  \* 1. 执行系统状态检查  \* 2. 喂看门狗  \* 3. 延时5秒  \*  \* @note 需要启用OS\_TASK\_STAT\_EN宏定义  \*/  void Monitor\_Task(void \*pdata) {  (void)pdata;  while (1) {  #if OS\_TASK\_STAT\_EN > 0  OS\_TaskStat();  #endif  BSP\_WDT\_Feed();  OSTimeDlyHMSM(0, 0, 5, 0);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 硬件抽象层 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* @defgroup BSP 硬件抽象层  \* @brief 硬件平台相关函数实现  \* @{  \*/  /\*\*  \* @brief 初始化硬件平台  \* @details 初始化系统时钟、GPIO、外设等  \* @note 必须在所有任务创建前调用  \*/  void BSP\_Init(void) {  // 具体实现依赖于目标硬件  }  /\*\*  \* @brief 设置LED显示模式  \* @param pattern LED模式位图  \*  \* @details 根据输入位图控制LED状态：  \* - bit0: LED1  \* - bit1: LED2  \* - bit2: LED3  \* - bit3: LED4  \*/  void BSP\_LED\_Set(INT8U pattern) {  // 具体实现依赖于目标硬件  }  /\*\*  \* @brief 数码管显示数字  \* @param num 要显示的数字（0-9）  \*  \* @warning 输入超过9时将不更新显示  \*/  void BSP\_7Seg\_Display(INT8U num) {  // 具体实现依赖于目标硬件  }  /\*\*  \* @brief 扫描按键状态  \* @return INT8U 按键值（0-9）或0xFF表示无按键  \*  \* @details 返回当前按下的按键编号  \*/  INT8U BSP\_Key\_Scan(void) {  // 具体实现依赖于目标硬件  return 0xFF;  }  /\*\*  \* @brief 喂看门狗  \* @details 重置看门狗定时器防止系统复位  \*/  void BSP\_WDT\_Feed(void) {  // 具体实现依赖于目标硬件  }  /\*\* @} \*/ // end of BSP | | | | | |
| 实验结果与分析  了解嵌入式实时操作系统的基本概念和特点，并能够运用实时操作系统编写嵌入式程序 | | | | | |