|  |  |  |  |
| --- | --- | --- | --- |
| **学生学号** | 0121902960525 | **实验课成绩** |  |

**武汉理工大学**

**学 生 实 验 报 告 书**

**实验课程名称 单片机原理及应用**

**开 课 学 院 计算机科学与技术学院**

**指导老师姓名 陆丽萍**

**学 生 姓 名 周航**

**学生专业班级 物联网1902班**

2020 — 2021 学年 第二学期

实验课程名称： 单片机原理及应用

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | **GPIO控制实验** | | | 实验成绩 |  |
| 实验者 | 周航 | 专业班级 | 物联网1902班 | 组别 |  |
| 同组者 |  | | | 实验日期 | 月 日 |
| 第一部分：实验分析与设计（可加页）  一、实验内容描述（问题域描述）   1. 实现开发板上的 LED 等按照一定规律“亮、灭”，掌握 STM32 基本 GPIO 口的使用。 2. 按下 STM32 开发板上的 Key1，翻转 LED1 与 LED2 的状态。本实验不仅需要例 程的知识，还需要学习将 GPIO 模式配置为上拉输入，以及使用 GPIO\_ReadInputDataBit( ) 读取按键输入   二、实验基本原理与设计（包括实验方案设计，实验手段的确定，试验步骤等，用硬件逻辑或者算法描述）  本实验是 STM32 开发最基础的实验，如图 3.4 所示，3 个 LED 灯分别接 GPIOD 的引 脚 PD5、PD6、PD7，实现 LED 灯的闪烁功能。    图3.4 LED灯硬件连接图  完成上述功能的步骤为：  （1）使能 I/O 口时钟，调用函数为 RCC\_APB2PeriphClockCmd( )；  （2）初始化 I/O 参数，调用函数 GPIO\_Init( )；  （3）操作 I/O，调用函数 GPIO\_SetBits( )和 GPIO\_ResetBits。  要实现按下KEY1实现灯的翻转，需要进行以下步骤的修改：   1. 初始化KEY1的管脚，调用 Key\_GPIO\_Config()，其中模式配置为上拉输入. 2. 在main函数中调用GPIO\_ReadInputDataBit( )读取按键输入。因为上拉输入，按键按下时为低电平，当读取到低电平时翻转灯的状态。   在例程上修改的部分主要如下：  #define KEY\_ON 0  #define LED\_3\_PORT GPIOA  void Key\_GPIO\_Config()  {  GPIO\_InitTypeDef GPIO\_InitStructure;  //打开GPIOA的外设时钟  RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOA ,ENABLE);  GPIO\_InitStructure.GPIO\_Pin = KEY1; // 选择操作的IO口  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU; // 上拉输入  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; // IO口最大输出速率  GPIO\_Init(LED\_3\_PORT, &GPIO\_InitStructure); // 完成初始化  }  int main(void)  {  LED\_Init();  Key\_GPIO\_Config();  while(1)  {  if(GPIO\_ReadInputDataBit(LED\_3\_PORT,KEY1)==KEY\_ON)  {  delayms(300); //延时消抖  if(GPIO\_ReadInputDataBit(LED\_3\_PORT,KEY1)==KEY\_ON) //检测按键按下  {  GPIO\_WriteBit(LED\_PORT,LED1,(BitAction)((1-GPIO\_ReadOutputDataBit(LED\_PORT,LED1))));  //翻转灯1的状态  GPIO\_WriteBit(LED\_PORT,LED2,(BitAction)((1-GPIO\_ReadOutputDataBit(LED\_PORT,LED2))));  //翻转灯2的状态  }  }  }  }  三、主要仪器设备及耗材  安装了Windows10或其它版本的Windows操作系统的PC机1台 | | | | | |
| 第二部分：实验调试与结果分析（可加页）  一、调试过程（包括调试方法描述、实验数据记录，实验现象记录，实验过程发现的问题等）  1．修改例程程序，在软件上编译，如果编译报错就改正相应错误  2．修改完异常和错误后，连接STM32开发板，将程序下载到开发板上  3．断电重启STM32开发板，进行实验操作，刚开始 LED1和LED2均不亮，按下KEY1后，都被点亮，再次按下，都被熄灭，依次循环。  实验过程发现的问题：   1. 按键按下没有反应，原因是没有配置相应的端口 2. 按键按下没有反应，原因是没有修改模式为上拉输入     二、实验结果及分析（包括结果描述、实验现象分析、影响因素讨论、综合分析和结论等）  图1：打开开发板，LED1,LED2均不亮  图2：按下KEY1，LED1，LED2点亮 图三：按下KEY1，LED1,LED2熄灭    图1：初始状态 图2：翻转状态  图三：再次翻转状态  三、实验小结、建议及体会  本次实验我了解了跑马灯的工作原理，也学习了一些底层硬件的结构。通过这次实验，我学会了如何配置GPIO寄存器的引脚，明白了相应的配置有什么作用，并对例程的程序进行了修改，达到用按键控制LED灯目的，增强了我的实践动手能力，扩宽了我的知识视野。 | | | | | |

实验课程名称： 单片机原理及应用

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | EXIT中断实验 | | | 实验成绩 |  |
| 实验者 | 周航 | 专业班级 | 物联网1902班 | 组别 |  |
| 同组者 |  | | | 实验日期 | 月 日 |
| 第一部分：实验分析与设计（可加页）  一、实验内容描述（问题域描述）  本实验将通过按键产生中断来实现控制 LED 灯的亮灭，这节重点是如何设置一个外部中断：当按下 KEY1 的时候，LED1亮，松开后灭；当按下 KEY3 的时候，LED 3亮，松开后灭  将 KEY1 换为 KEY2，利用中断方式，实现 LED 灯的闪烁  二、实验基本原理与设计（包括实验方案设计，实验手段的确定，试验步骤等，用硬件逻辑或者算法描述）    实现以上工程需要以下几个步骤：  （1）开启GPIO口复用时钟，  （2）初始化GPIO口输入 ；  （3）配置中断分组（NVIC），并使能中断；  （4）设置GPIO口与中断线的映射关系，设置触发条件 ；  （5）编写中断服务程序。  要实现将 KEY1 换为 KEY2，利用中断方式，实现 LED 灯的闪烁，需要完成以下步骤：   1. 修改程序将KEY2配置成中断，具体的做法是首先配置KEY2引脚，然后连接EXTI Line8到PA8。KEY3引脚的配置不用修改。调用KEY\_Init()实现。 2. 由于KEY1和KEY2连接的外部中断线都是EXITLine8,所以其中断服务程序可以不用修改   在例程上修改的部分主要如下：  //配置KEY2，KEY3引脚以及中断  void KEY\_Init(void)  {  GPIO\_InitTypeDef GPIO\_InitStructure;  EXTI\_InitTypeDef EXTI\_InitStructure;  RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOA |RCC\_APB2Periph\_GPIOD| RCC\_APB2Periph\_AFIO ,ENABLE);  //打开GPIOA和GPIOD的外设时钟  //配置KEY3  GPIO\_InitStructure.GPIO\_Pin = KEY3; //选择操作的IO口  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING; //浮空输入  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //IO口最大输出速率  GPIO\_Init(KEY1\_3\_PORT, &GPIO\_InitStructure); //完成初始化    //配置KEY2  GPIO\_InitStructure.GPIO\_Pin = KEY2; //选择操作的IO口  GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING; //浮空输入  GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //IO口最大输出速率  GPIO\_Init(LED\_PORT, &GPIO\_InitStructure); //完成初始化      GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOD, GPIO\_PinSource8); //连接EXTI Line8到PA8  GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource4); //连接EXTI Line4到PA4  EXTI\_ClearITPendingBit(EXTI\_Line4);  EXTI\_ClearITPendingBit(EXTI\_Line8);  EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt; //中断模式  EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling; //下降沿触发  EXTI\_InitStructure.EXTI\_Line = EXTI\_Line4 | EXTI\_Line8 ;  EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;  EXTI\_Init(&EXTI\_InitStructure); //初始化EXTI  }  三、主要仪器设备及耗材  安装了Windows10或其它版本的Windows操作系统的PC机1台 | | | | | |
| 第二部分：实验调试与结果分析（可加页）  一、调试过程（包括调试方法描述、实验数据记录，实验现象记录，实验过程发现的问题等）  1．修改例程程序，在软件上编译，如果编译报错就改正相应错误  2．修改完异常和错误后，连接STM32开发板，将程序下载到开发板上  3．断电重启STM32开发板，进行实验操作，刚开始 LED1和LED2均不亮，按下KEY2触发中断，LED1被点亮，再次按下,LED1熄灭。按下KEY3触发中断，LED3被点亮，再次按下,LED3熄灭  实验过程发现的问题：  （1）在用KEY3控制LED3时，没有按下switch按键，导致没有反应。  二、实验结果及分析（包括结果描述、实验现象分析、影响因素讨论、综合分析和结论等）  图1：初始状态，LED均熄灭，等待中断触发  图2：按下KEY2，LED1点亮 图三：按下KEY3，LED3点亮    图1：初始状态 图2：KEY2中断触发  图三：KEY3中断触发  三、实验小结、建议及体会  本次实验我学习中断的原理，更进一步了解了GPIO寄存器的引脚配置及相应的功能。  通过分析例程函数，我明白了如何将GPIO的引脚配置成中断，相应的引脚连接哪一根外部中断线，以及抢占优先级和响应优先级的功能和配置方法，增强了我的实践动手能力，扩宽了我的知识视野。 | | | | | |

实验课程名称： 单片机原理及应用

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 串行数据通信试验 | | | 实验成绩 |  |
| 实验者 | 周航 | 专业班级 | 物联网1902班 | 组别 |  |
| 同组者 |  | | | 实验日期 | 月 日 |
| 第一部分：实验分析与设计（可加页）  一、实验内容描述（问题域描述）  学习 STM32 的串口发送和接收数据，掌握 STM32 的串口通信  实现 USART 查询方式实现通信，即循环接收到上位机发过来的字符，然后再回传给上位机  单片机每次接收上位机发送 5 个字符，然后回传给上位机  二、实验基本原理与设计（包括实验方案设计，实验手段的确定，试验步骤等，用硬件逻辑或者算法描述）：  USART，即通用同步/异步串行接收/发送器，常被称作串口。在 STM32 的参考手册中，串口被描述成通用同步异步收发器(USART)，它提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。USART 利用分数波特率发生器提供宽范围的波特率选择。它支持同步单向通信和半双工单线通信，也支持 LIN（局部互联网），智能卡协 议和 IrDA（红外数据组织）SIR ENDEC 规范，以及调制解调器(CTS/RTS)操作。它还允许 多处理器通信。还可以使用 DMA 方式，实现高速数据通信。  USART 通过 3 个引脚与其他设备连接在一起，任何 USART 双向通信至少需要2个引脚：接受数据输入(RX)和发送数据输出(TX)。  RX: 接受数据串行输入。通过过采样技术来区别数据和噪音，从而恢复数据。  TX: 发送数据输出。当发送器被禁止时，输出引脚恢复到它的 I/O 端口配置。当发送器 被激活，并且不发送数据时，TX 引脚处于高电平。在单线和智能卡模式里，此 I/O 口被同 时用于数据的发送和接收。  完成串口通信功能的步骤为：  （1）使能 I/O 口时钟，调用函数为 RCC\_APB2PeriphClockCmd( )；  （2）初始化 GPIO，调用函数 GPIO\_Init( )；  （3）初始化 USART，调用函数 USART\_Init()与 USART\_Cmd；  （4）实现串口通信，调用函数为 USART\_SendData()与 USART\_ReceiveData()。  要实现单片机每次接收上位机发送的5 个字符，然后回传给上位机的功能，需要完成以下步骤：  （1）在函数中定义一个数组，利用数组存储接收到的字符，这里数组就像一个缓冲区，缓冲区的大小为5，当单片机接收到5个字符时，才会回传给上位机，否则不会回传。当一次输入的数据超过5个，只会接收前5个数据。这里也可以设置扩大数组来获得更大的缓冲区。  在例程上修改的部分主要如下：  int main(void)  {  uint16\_t i=0;  uint16\_t temp[5]; //存储从上位机传来的字符  //中间配置GPIO和USART1和例程一致    //实现串口通信  while(1)  {  GPIO\_SetBits(GPIOD,GPIO\_Pin\_5);  RX\_status=USART\_GetFlagStatus(USART1,USART\_FLAG\_RXNE);  if(RX\_status==SET)  {  if(i<5)  temp[i++] = USART\_ReceiveData(USART1);  //如果接收到的数据小于5个，将其存入数组  if(i==5)  {  for(i=0;i<5;i++)  {  USART\_SendData(USART1,temp[j++]);  delayms(5); //延时确保输出完成  }  i=0;  }  while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)==RESET);  //判断是否输出完毕  GPIO\_ResetBits(GPIOD,GPIO\_Pin\_5);  }  }  }  三、主要仪器设备及耗材  安装了Windows10或其它版本的Windows操作系统的PC机1台  第二部分：实验调试与结果分析（可加页）  一、调试过程（包括调试方法描述、实验数据记录，实验现象记录，实验过程发现的问题等）  1．修改例程程序，在软件上编译，如果编译报错就改正相应错误  2．修改完异常和错误后，连接STM32开发板，将程序下载到开发板上  3．断电重启STM32开发板，进行实验操作，在输入框输入一个字符，然后点击发送，正确的情况应该是按四次发送窗口都不会回显，当按第五次时，窗口一次回显5个发送的字符。  实验过程发现的问题有：  1．发送的数据异常，具体表现是后一个字符串的输出内容包含了前一个字符串的输出内容，解决方法让数组重新开始计数来覆盖输入的数据。  二、实验结果及分析（包括结果描述、实验现象分析、影响因素讨论、综合分析和结论等）  打开串口调试助手 ：  发送键按3次，没有输出结果。  发送键按5次，输出存入缓冲区的5个字符。    三、实验小结、建议及体会  本次实验我学习了串口的相关知识，明白了如何去配置一个串口，学习了USART外设的模式操纵，数据位设置，通信标志位等功能，加深了我对USART设备操作的理解，增强了我的实践动手能力，扩宽了我的知识视野，更加体会到单片机功能的强大。 | | | | | |

实验课程名称： 单片机原理及应用

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | AD转换实验 | | | 实验成绩 |  |
| 实验者 | 周航 | 专业班级 | 物联网1902班 | 组别 |  |
| 同组者 |  | | | 实验日期 | 月 日 |
| 第一部分：实验分析与设计（可加页）  一、实验内容描述（问题域描述）  学习利用 STM32 的 ADC1 通道 0 来采集外部的电压值，掌握 STM32 中 ADC 的配置与使用  实现模数转换功能，并通过 USART1 输出转换后的数字量信息  二、实验基本原理与设计（包括实验方案设计，实验手段的确定，试验步骤等，用硬件逻辑或者算法描述）：  STM32 中实现 AD 功能，主要是对结构体 ADC\_InitTypeDef 的初始化，理解其每个成员的功能，本实验使用 ADC1 的通道 0 来进行 AD 转换，其详细设置步骤如下：  （1）开启 PA 口时钟，设置 PA0 为模拟输入；  （2）使能 ADC1 时钟，设置 ADC1 的工作模式；  （3）设置 ADC1 规则序列的相关信息；  （4）启动校准并等待完成；  （5）开启 AD 转换器，开始采集。    ADC电路原理图    将程序下载到开发板，打开串口助手，设置波特率为：115200， 8个数据位， 1个停止位，无硬件流，无校验位，此时程序通过串口会循环打印通道 0的 AD值，调节电位器， AD值会伴随着变化，同时， LED也会有相应的变化，当采集的 AD值小于2048的时候， LED1灭，LED2亮，LED3灭；当 AD值大于 20 4 8的时候，LED1亮，LED2灭，LED3灭；当 AD值等于 20 4 8的时候，LED1灭，LED2灭，LED3亮。  完成上述功能的步骤是：   1. 初始化LED灯，调用LED\_Init() 2. 初始化串口，调用USART1\_Init() 3. 初始化ADC，调用ADC\_Configuration()，本次实验采集的时ADC1的0通道，且是软件触发模数转换。 4. 在main函数中根据采集到的AD值进行判断和操作   本次实验直接采用例程的代码，观察实验现象。  三、主要仪器设备及耗材  安装了Windows10或其它版本的Windows操作系统的PC机1台 | | | | | |
| 第二部分：实验调试与结果分析（可加页）  一、调试过程（包括调试方法描述、实验数据记录，实验现象记录，实验过程发现的问题等）  1．编译例程程序。  2．连接STM32开发板，将程序下载到开发板上  3．断电重启STM32开发板，进行实验操作，在例程程序中，需要在串口调试助手上监测从STM32开发板发来的信息并转动点位器观察LED灯的亮灭情况是否符合设定的初值  二、实验结果及分析（包括结果描述、实验现象分析、影响因素讨论、综合分析和结论等）  打开串口调试助手，接收STM32发来的AD值信息：    根据实验要求：  当 AD 值小于 2048 的时候， LED1 灭，LED2 亮，LED3 灭；  当 AD 值大于 2048 的时候，LED1 亮，LED2 灭，LED3 灭；  当 AD 值等于 2048 的时候，LED1 灭，LED2 灭，LED3 亮：    分别为LED1亮和LED2亮，满足要求  三、实验小结、建议及体会  本次实验我学习了如何利用STM32的ADC1通道0来采集外部的电压值，掌握STM32中ADC的配置与使用。我明白了ADC是一种逐次逼近型模拟数字转换器。它能将通道采集到的电压值转换成相应的数字量信息并通过串口发送显示。 | | | | | |

实验课程名称： 单片机原理及应用

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | 定时器实验 | | | 实验成绩 |  |
| 实验者 | 周航 | 专业班级 | 物联网1902班 | 组别 |  |
| 同组者 |  | | | 实验日期 | 月 日 |
| 第一部分：实验分析与设计（可加页）  一、实验内容描述（问题域描述）  掌握 STM32 中通用定时器使用，实现开发板上的 LED 等按照一定规律“亮、灭”  实现通过 TIM2 的定时功能，使得 LED 灯按照 1s 的时间间隔来闪烁  改变定时器的周期，实现 PWM 功能的实现，使 PWM 信号的周期为4秒，占空比为75%，即 LED 灯亮3秒，灭1秒  二、实验基本原理与设计（包括实验方案设计，实验手段的确定，试验步骤等，用硬件逻辑或者算法描述）：  实现上述工程需要以下几个步骤：  （1）开启TIM2时钟；  （2）初始化定时器参数,设置自动重装值，分频系数，计数方式等；  （3）设置 TIM2\_DIER 允许更新中断；  （4）TIM2中断优先级设置；  （5）使能 TIM2  （6）编写中断服务程序在中断产生后，处理相关中断。通过状态寄存器的值来判断此次产生的中断属于什么类型。然后执行相关的操作，这里使用的是更新（溢出）中断，所以在状态寄存器 SR 的最低位。在处理完中断之后应该向 TIM2\_SR 的最低位写 0，来清除该中断标志。  实现定时器控制灯亮3s，熄1s，需要实现以下步骤:   1. 已知定时器配置的周期为1s，可以让LED灯亮三个定时器周期，熄一个定时器周期，可以定义一个全局变量count来记录灯亮的定时器周期数   在例程上修改的部分主要如下：  //TIM2中断处理函数  u8 count=1;  void TIM2\_IRQHandler(void)  {  u8 ReadValue;  //检测是否发生溢出更新事件  if(TIM\_GetITStatus(TIM2,TIM\_IT\_Update)!=RESET)  {  //清除TIM2的中断待处理位  TIM\_ClearITPendingBit(TIM2,TIM\_FLAG\_Update);    //将LED1管脚输出数值写入ReadValue  ReadValue=GPIO\_ReadOutputDataBit(GPIOD,GPIO\_Pin\_5);  if(ReadValue==0)  GPIO\_SetBits(GPIOD,GPIO\_Pin\_5);  else  {  if(count<3)  {  count++;  }  else  {  count=1;  GPIO\_ResetBits(GPIOD,GPIO\_Pin\_5);  }  }  }  }  三、主要仪器设备及耗材  安装了Windows10或其它版本的Windows操作系统的PC机1台  第二部分：实验调试与结果分析（可加页）  一、调试过程（包括调试方法描述、实验数据记录，实验现象记录，实验过程发现的问题等）  1．修改例程程序，在软件上编译，如果编译报错就改正相应错误  2．修改完异常和错误后，连接STM32开发板，将程序下载到开发板上  3．断电重启STM32开发板，进行实验操作，观察 LED1，大致估计其亮灭的时间。  实验过程发现的问题：  主要是这种处理方式比较难想到。  二、实验结果及分析（包括结果描述、实验现象分析、影响因素讨论、综合分析和结论等）  经过观察和测量，LED1每亮3s，熄1s.  三、实验小结、建议及体会  本实验我学习了TIM定时器的相关知识和相关操作，包括定时器的定时功能的实现原理和方法、定时周期的设定和中断执行周期固定的操作，加深了我对TIM这一模块以及固件函数库的理解和运用，扩宽了我的知识视野，更加体会到单片机功能的强大。 | | | | | |