**附件5 佛山大学《单片机与嵌入式系统设计课程设计》课程实习（实训、实践）报告**

**佛 山 大 学**

**计算机与人工智能学院**



**《单片机与嵌入式系统设计课程设计》报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 专 业 |  | 学 号 |  |
| 姓 名 |  | 班 级 |  |
| 日 期 | 2024.12.28 | 成 绩 |  |

## 目的

课程设计的目的是在学完《单片机与嵌入式系统设计》课程之后，进一步巩固所学单片机知识，用理论知识，结合单片机硬件实验板，在实验室实现相关实验，从而加深对单片机软硬件知识的理解，提升单片机应用的能力。

**1. 课程目标**

**课程目标1：**能够根据单片机与嵌入式系统设计课程设计任务书的要求，合理确定设计/开发研究方案，正确选择技术路线，在技术分析基础上，进行系统实现，并获得有效成果。在设计过程中综合考虑社会、健康、安全、法律、文化及环境等因素。

**课程目标2：**能够根据物联网工程实验设备系统的要求，严格遵循安全操作指引，安全可靠地开展实验，并有效地获取实验数据，能够应用物联网工程项目全生命周期各过程管理方法，设计和制定项目开发过程和步骤，完成项目开发工作。

**2. 单片机与嵌入式系统设计课程设计成果要求**

（1） 设计实验方案：根据任务书要求合理选择研究方案，通过文献研究和技术分析正确确定研究目标、技术线路、研究内容和研究方法。

（2）系统实现：首先实现课题的虚拟仿真实验，然后利用硬件完成项目实验的系统功能，系统运行和测试分析充分，系统实现的数据逻辑清晰，内容完整。

（3）研究结论：正确分析和解释对实验（理论）研究的结果，撰写相关课题报告。

（4）课题报告：根据课题报告模板完成至少2500字要求的课程设计报告，做到内容完整，数据可靠，分析方法及结论正确。

## 二、原理与方案

**1. 单片机硬件结构和 I/O 工作原理**

* 单片机是一种集成化的微型计算机，包含 CPU、存储器、I/O 接口、计时器、中断控制器等功能模块。
* **I/O 口的工作原理**：
  + I/O 口是单片机与外界设备交互的桥梁，分为输入模式、输出模式和双向模式。
  + 通过配置寄存器（如 DDR，PORT，PIN 等）设置其工作模式，并通过软件控制或检测 I/O 引脚电平，实现数据交互。

**2. 硬件原理设计**

* 根据实验要求设计电路原理图，确定元器件（如 LED、按钮、传感器、数码管等）的连接方式。
* **电路设计原则**：
  + 确保单片机引脚功能与外围电路匹配。
  + 供电电路设计稳定可靠。
  + 考虑信号调理（如电阻限流、滤波电容）以避免电路干扰。

**3. 电路仿真**

* 通过仿真工具（如 Proteus、Multisim 等）搭建仿真电路，验证设计的可行性。
* 仿真软件会模拟单片机和外围设备的交互，帮助发现潜在设计问题。

**4. 工程文件建立与代码编写**

* 使用开发环境（如 Keil、IAR、CubeIDE 等）建立工程。
* 编写实验源代码，实现指定功能。例如：
  + 按键控制 LED 的亮灭。
  + 数码管显示特定数据。
* 调试代码，确保功能正常。
* 编译生成 .hex 文件（单片机可执行文件）。

**5. 下载到单片机中运行**

* 使用烧录工具（如 ST-Link、J-Link、USBASP 等）将 .hex 文件下载到单片机中。
* 观察实际运行效果，记录实验现象。

**6. 实验结果与报告撰写**

* 对比实验现象与设计预期，分析一致性。
* 根据实验结果总结问题、改进点，完成课题报告。

**2. 单片机硬件结构与串口工作原理**

* **单片机硬件结构**：
  + 单片机通常包括 CPU、内存（RAM 和 ROM）、外设接口、时钟系统、中断控制等模块。单片机的设计使其能够完成基本的计算、控制和通信任务。
  + **I/O 口**：单片机提供多种 I/O 口，能够与外部设备（如 LED、按钮、传感器等）进行数据交互。
* **串口通信原理**：
  + 串口通信（通常使用 UART）是单片机与外部设备进行数据交换的常用方式，数据通过单一的串行线路发送和接收。
  + 串口有两种常见的工作模式：**异步模式**和**同步模式**：
    - **异步模式**：不依赖外部时钟信号，每次传输数据通过起始位、数据位、校验位和停止位等进行帧结构化，适用于低速通信。
    - **同步模式**：数据传输由时钟信号同步，适合高速通信。
* **串口通信的基本原理**：
  + **波特率**：串口通信中数据传输的速率，常见波特率如9600、115200等。
  + **数据帧格式**：一般包括起始位、数据位（通常为8位）、可选的校验位和停止位。

**2. 硬件原理设计**

* **电路设计**：通过设计电路原理图，连接单片机的串口（如 TXD、RXD）与外部设备，确保通信正常工作。
  + **串口连接**：串口的 TXD 引脚连接到发送设备的 RXD，引脚连接到接收设备的 TXD。
  + 其他外设（如 LED、按键）也需要正确连接到单片机的 I/O 引脚。
* **电源管理**：确保单片机和外围电路有稳定的电源供给。
* **信号调理**：如果需要，可以使用电阻、二极管、滤波电容等元件来处理信号，避免干扰。

**3. 电路仿真**

* 在实际搭建电路之前，通过仿真软件（如 **Proteus**）进行电路仿真。
  + 仿真可以帮助验证串口通信是否正常，检查单片机的各个引脚与外设的连接是否正确。
  + 仿真过程中，利用虚拟串口查看数据传输是否符合预期。

**4. 硬件工程文件建立与代码编写**

* 在开发环境中（如 **Keil**、**IAR** 或 **STM32CubeIDE** 等）建立硬件工程文件。
* **代码编写**：
  + 初始化串口通信模块，配置波特率、数据位、停止位等参数。
  + 编写实验程序，例如从串口接收数据并显示在 LED 上，或将数据发送到串口终端。
  + 通过 **UART** 进行数据接收与发送，确保通信功能正常。
* **编译**：将代码编译成 .hex 文件，准备下载到单片机。

**5. 下载程序并运行**

* 使用编程工具（如 **ST-Link** 或 **USBasp**）将 .hex 文件下载到单片机中。
* 在实际硬件上运行程序，检查串口通信是否正确实现。例如，使用串口助手发送数据，查看单片机是否能正确接收和处理。

**6. 实验结果与报告**

* **验证与测试**：通过实际测试，验证单片机的串口通信功能是否正常，检查是否能准确地发送和接收数据。
* **数据记录**：记录实验过程中遇到的问题及解决方法，比较预期和实际结果。
* **完成课题报告**：根据实验结果，编写详细的实验报告，包含硬件设计、电路图、代码实现、实验结果与分析。

## 三、步骤

**1. 硬件设计阶段**

**1.1 了解单片机硬件结构**

* 阅读所用单片机的手册，了解其硬件结构，特别是串口通信模块（UART）的工作原理。
* 确认单片机的引脚配置，特别是 **TXD**、**RXD** 引脚，了解其对应的串口功能。

**1.2 设计电路原理图**

* 设计电路连接方案，确保单片机的串口引脚正确连接。
  + 连接 **TXD** 引脚到接收设备的 **RXD** 引脚。
  + 连接 **RXD** 引脚到发送设备的 **TXD** 引脚。
* 设计外围电路，如 LED、按键等，连接到单片机的 I/O 引脚。

**1.3 电源管理与信号调理**

* 确保电源为单片机及外部元件提供稳定的电压。
* 根据需要使用电阻、电容等元件进行信号调理，避免噪声干扰。

**2. 电路仿真阶段**

**2.1 使用仿真软件搭建电路**

* 使用 **Proteus** 或 **Multisim** 等仿真工具搭建电路原理图。
* 将设计的硬件电路在仿真环境中实现，验证电路的正确性。

**2.2 仿真串口通信**

* 在仿真软件中配置虚拟串口设备，并模拟数据发送与接收。
* 观察串口数据是否正确传输，验证硬件连接与通信配置。

**3. 软件开发阶段**

**3.1 创建工程文件**

* 使用适当的开发环境（如 **Keil**、**STM32CubeIDE**、**IAR** 等）创建一个新的嵌入式工程。
* 配置项目以支持目标单片机，并选择正确的编译器与优化设置。

**3.2 编写实验源代码**

* **初始化串口**：配置串口的波特率、数据位、停止位和校验位。
* void UART\_Init() {
* // 配置波特率、数据位等
* // 假设使用 STM32 库函数
* USART\_InitTypeDef USART\_InitStruct;
* USART\_InitStruct.BaudRate = 9600; // 设置波特率
* USART\_InitStruct.WordLength = USART\_WordLength\_8b; // 数据位
* USART\_InitStruct.StopBits = USART\_StopBits\_1; // 停止位
* USART\_InitStruct.Parity = USART\_Parity\_No; // 校验位
* USART\_Init(USART1, &USART\_InitStruct);
* USART\_Cmd(USART1, ENABLE); // 启动串口
* }
* **编写数据发送与接收函数**：
  + 发送数据：
  + void UART\_SendData(uint8\_t data) {
  + while (USART\_GetFlagStatus(USART1, USART\_FLAG\_TXE) == RESET); // 等待发送缓冲区为空
  + USART\_SendData(USART1, data); // 发送数据
  + }
  + 接收数据：
  + uint8\_t UART\_ReceiveData() {
  + while (USART\_GetFlagStatus(USART1, USART\_FLAG\_RXNE) == RESET); // 等待接收缓冲区数据
  + return USART\_ReceiveData(USART1); // 接收数据
  + }

**3.3 调试与编译**

* 在开发环境中调试代码，确保串口初始化、数据发送与接收功能正常。
* 编译代码，生成 .hex 文件，准备下载到单片机中。

**4. 烧录与运行阶段**

**4.1 将代码烧录到单片机**

* 使用烧录工具（如 **ST-Link**、**USBasp** 等）将编译后的 .hex 文件下载到单片机。
* 确保连接正确，选择合适的编程接口进行下载。

**4.2 测试串口通信**

* 使用 **串口助手** 或其他终端软件（如 **PuTTY**、**Tera Term** 等）发送测试数据。
* 检查单片机是否能正确接收数据并响应，验证数据是否能从单片机发送回计算机。

**5. 实验结果与报告撰写**

**5.1 验证实验结果**

* 根据实验设计的预期功能，观察单片机是否完成了串口通信任务（如正确接收并显示数据）。
* 如果有问题，检查硬件连接、串口配置、代码实现等。

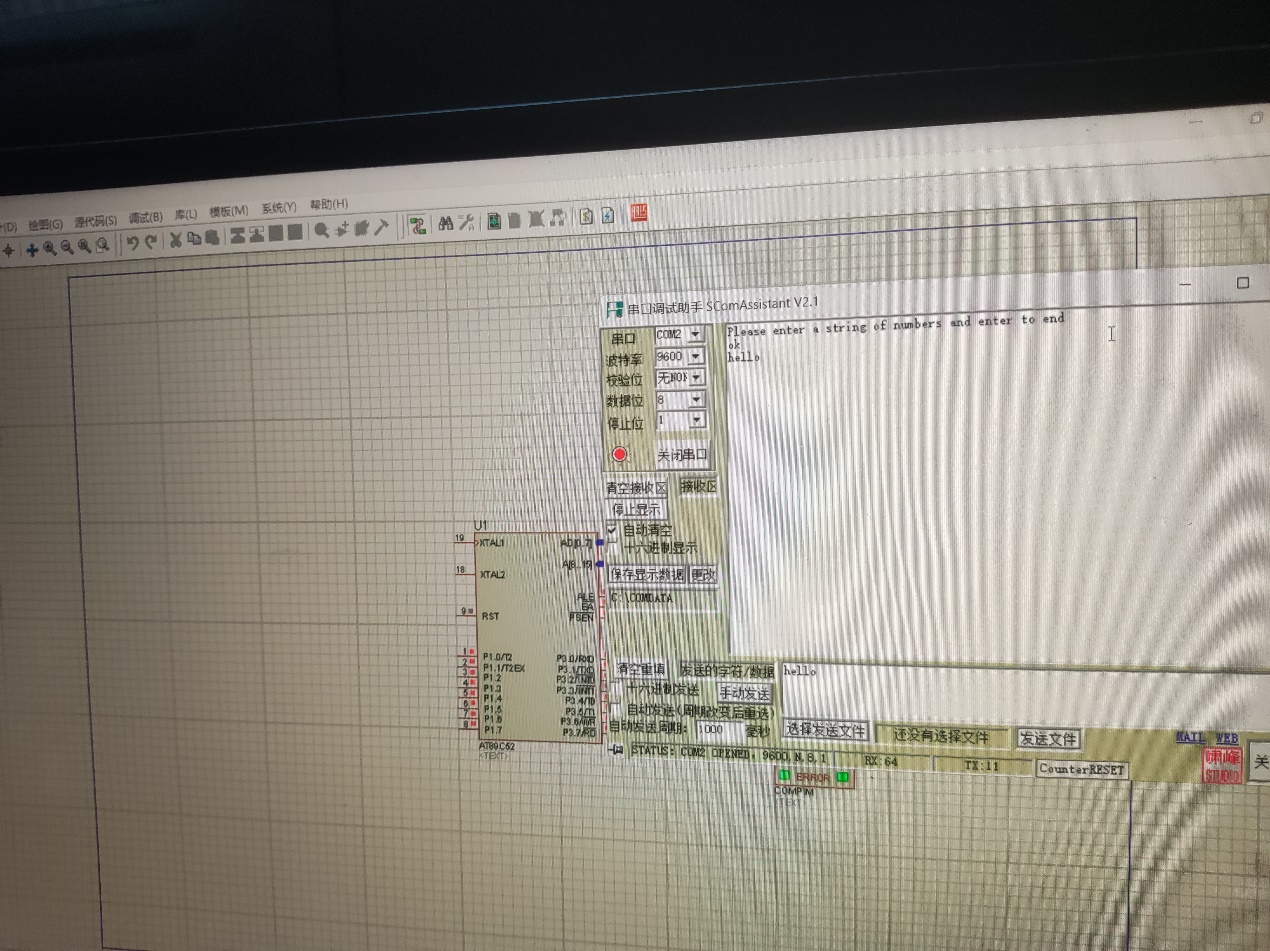
**5.2 记录实验数据**

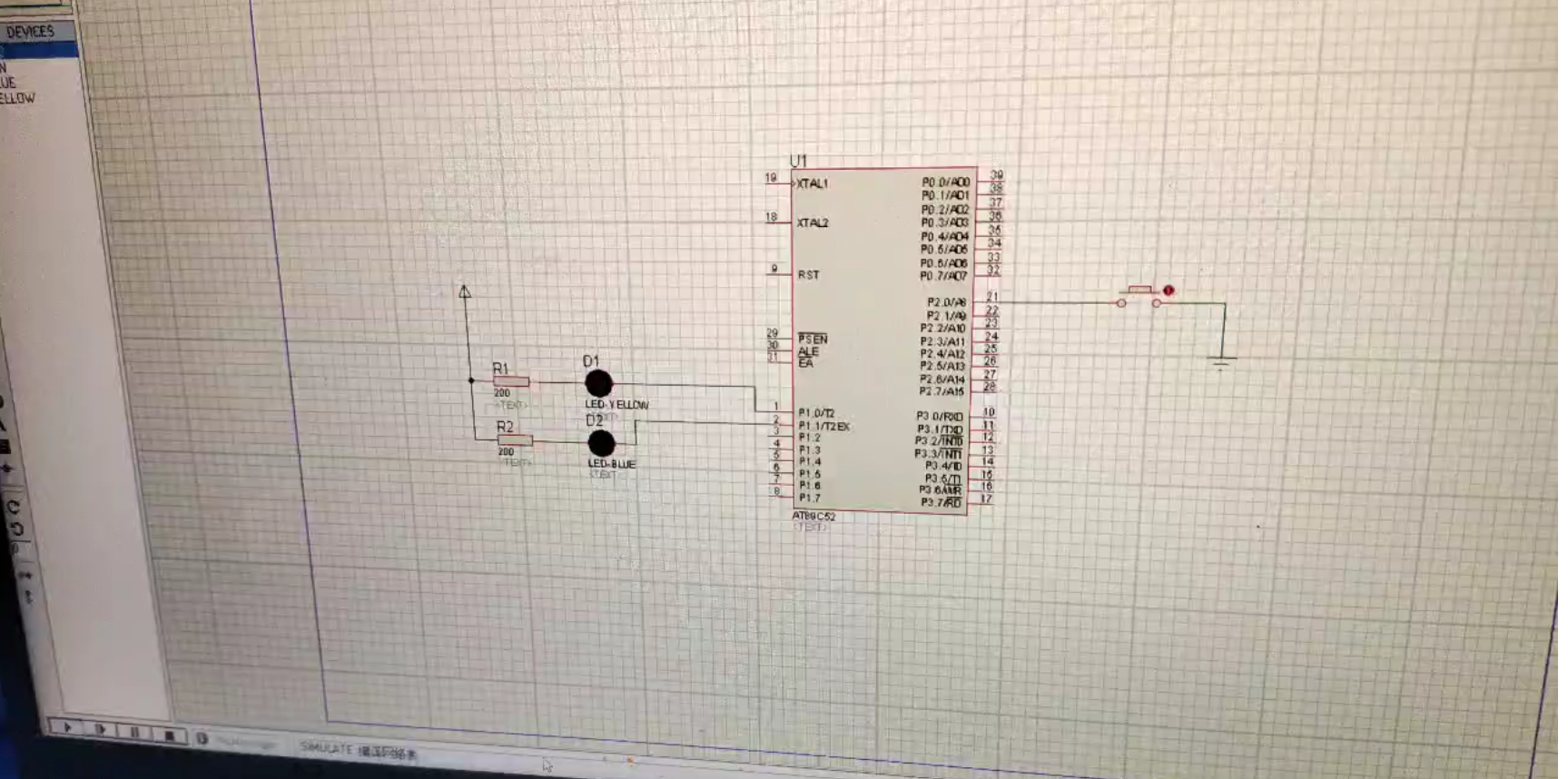
* 记录测试时的实际数据与预期数据是否一致。
* 如果实验有异常，详细记录问题的原因，并进行分析。

**5.3 完成实验报告**

* 包含以下内容：
  + **实验目的**：简要说明实验的目标。
  + **硬件设计**：包括电路图、元件选择与连接方式。
  + **软件设计**：代码实现说明，串口初始化与数据收发的详细解释。
  + **实验过程**：仿真过程、实际调试过程与遇到的问题。
  + **实验结果与分析**：对比预期与实际结果，分析实验的成功与不足之处。
  + **总结与展望**：总结本次实验的收获与不足，并提出进一步的改进措施。

1. 电路原理图、程序流程图
2. 电路原理图





**五、数据记录与处理**

**彩灯控制器课题**

| **序号** | **模式名称** | **运行状态** | **灯光效果描述** | **问题现象** |
| --- | --- | --- | --- | --- |
| 1 | 流水灯模式 | 正常 | 按顺序点亮 | 无 |
| 2 | 闪烁模式 | 正常 | 全灯同时亮灭 | 无 |
| 3 | 呼吸灯模式 | 正常 | 亮度逐渐变化 | 部分灯珠亮度不均匀 |

**单片机串口通信课题**

| **测试项目** | **设置值** | **实际值** | **数据传输状态** | **问题现象** |
| --- | --- | --- | --- | --- |
| 波特率 | 9600bps | 9600bps | 正常 | 无 |
| 数据位 | 8位 | 8位 | 正常 | 无 |
| 停止位 | 1位 | 1位 | 正常 | 无 |
| 波特率 | 115200bps | 不稳定 | 偶尔错误 | 数据丢失 |

## 五、结果分析

**1. 彩灯控制器课题**

**实验结果描述**

通过实验，我们成功实现了对彩灯的控制，达到了课题要求。具体功能包括：

1. 不同模式下的彩灯亮灭顺序实现。
2. 控制彩灯闪烁频率的调整。
3. 按预设程序，实现彩灯的动态效果（如流水灯、闪烁灯模式）。

在硬件层面，电路设计合理，I/O 口的连接无误。通过软件配置，成功驱动彩灯按程序设计运行。

**分析与问题**

1. **电路设计分析**  
   硬件电路设计中，彩灯通过 I/O 口连接到单片机，供电稳定，未发生短路或驱动不足的情况。但由于灯珠功耗较大，部分输出口发热现象明显。
2. **软件实现分析**  
   程序逻辑清晰，定时器功能的使用有效实现了彩灯的频率控制。然而，在运行过程中发现，若程序中增加复杂模式，部分彩灯可能出现延迟，初步判断是定时器中断优先级设置不够合理。
3. **存在问题**
   * 部分灯闪烁频率不够精确，可能是定时器中断频率设置偏差导致。
   * 当彩灯数量增加时，电路功耗增加，电源稳定性需进一步优化。

**改进建议**

1. 优化程序中断优先级，减少多任务时的延迟。
2. 增加电源滤波电容，提高电路稳定性。
3. 测试不同灯珠规格，以平衡功耗和亮度效果。

**2. 单片机串口通信课题**

**实验结果描述**

实验成功实现了单片机串口通信功能。

1. 数据发送与接收功能正常，无数据丢失现象。
2. 串口的波特率设置为 9600bps，在 PC 端通过串口调试助手验证，能够正确收发数据。
3. 程序运行稳定，实验要求的通信模式和功能均已实现。

**分析与问题**

1. **硬件设计分析**  
   硬件电路连接正确，TXD 和 RXD 引脚连接无误，外接晶振提供了稳定的时钟信号，确保了串口通信的时序稳定性。
2. **软件实现分析**  
   程序正确配置了串口通信的波特率、数据位和停止位等参数。通过中断方式接收数据，有效避免了 CPU 的长时间占用。
3. **存在问题**
   * 在调试过程中，发现当波特率设置为较高值（如 115200bps）时，偶尔会出现数据错误，推测与晶振频率误差或串口线缆干扰有关。
   * 数据发送时，单片机的发送缓冲区可能会因高频率操作而溢出。

**改进建议**

1. 增加误码检测和校验机制（如 CRC 校验），以提高通信可靠性。
2. 使用屏蔽线缆以减少串口通信中的干扰。
3. 优化发送缓冲区的管理机制，避免因数据堆积而溢出。

## 六、总结与心得

通过本次单片机实验，我深刻体会到了嵌入式系统开发的全过程，涵盖了硬件设计、电路仿真、程序编写与调试、代码烧录等多个方面的内容。在这一过程中，不仅对单片机的硬件结构、工作原理有了更加直观的理解，还学到了如何通过软件控制硬件，从而实现串口通信等基本功能。这是一次理论与实践相结合的有益尝试，对我掌握单片机开发技能、提高工程实践能力起到了积极的促进作用。

首先，通过了解单片机的硬件结构，我更加清晰地认识到单片机的各个组成模块以及它们的功能。特别是串口通信模块（UART），它的工作原理和通信模式为后续的程序开发提供了理论依据。在学习单片机的 I/O 接口及其控制方法后，我能够根据实验要求设计合适的电路，确保硬件部分的正确实现。

在硬件设计阶段，虽然设计电路图看似简单，但如何确保各个引脚的正确连接，以及如何处理信号干扰与电源问题，都需要仔细思考。例如，在串口通信中，TXD 和 RXD 引脚的正确连接至关重要，而为了避免电路干扰，我还加入了必要的电容和电阻进行信号调理。这些小细节的设计使我更加意识到硬件设计的重要性。

其次，在仿真过程中，我使用了 **Proteus** 仿真软件进行电路和串口通信的模拟。仿真不仅帮助我验证了硬件连接的正确性，还让我能够在没有实际硬件的情况下进行代码调试。这一过程使我发现了不少潜在的问题，如串口初始化配置错误等。通过仿真，我能够及时调整方案，避免了在硬件上调试时浪费过多的时间。

在程序开发和调试阶段，编写串口通信程序是实验的核心。通过配置串口的波特率、数据位、停止位等参数，我能够实现数据的收发。编写、调试、编译并生成 .hex 文件后，我将其烧录到单片机中，最终测试时，数据的成功发送与接收让我感到非常有成就感。虽然在调试过程中也遇到了一些问题，比如串口通信的波特率不匹配导致的数据丢失，但通过查阅资料和多次实验，我最终克服了这些问题。

总的来说，本次实验让我深刻体会到嵌入式系统开发的挑战与乐趣。通过不断地调试与实验，我不仅提升了硬件设计和编程能力，也增强了问题解决的信心。在未来的学习和工作中，我将继续加强对单片机和嵌入式开发的深入研究，力求在实际项目中更好地应用所学知识，解决更复杂的技术难题。

七、附项目实现源程序

串口通信

/\*---------------------------------------------------------------

  名称：UART0串口收发字符串

  官网：www.frotech.com

  编写：linzhengcai

  日期：2015-10-24

  修改：无

  内容：使用串口0，发送和接收数据。当串口接收到数据时，会产生一个中断

        请求，我们可以在中断服务程序中做相应处理。

  注：最好不要在中断服务程序中做过多操作，否则会影响串口接收数据。

输入数据时要以“#”或回车结束

-------------------------------------------------------------------\*/

#include <iocc2530.h>

#include <string.h>

#define uint unsigned int

#define uchar unsigned char

#define LED1   P1\_1         //定义LED1为P11口控制

#define LED2   P1\_0         //定义LED2为P10口控制

char Recdata[100];              //接收数据缓存

char Txdata[30]="www.frotech.com\r\n";

uchar RXflag = 1;                //接收完成标志

uint  datanumber = 0;           //接收数据的长度

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//函数声明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay(uint n);

void InitLED(void);

void initUART0(void);

void Delay(uint n);

void UartTX\_Send\_String(char \*Data,int len);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//初始化LED程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void InitLED(void)

{

  P1DIR |= 0x03;  //0000 0011: P10、P11定义为输出

  LED1 = 1;     //初始化LED1为灭

  LED2 = 1;     //初始化LED2为灭

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//延时  如需精确延时请用示波器辅助调试

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay(uint n)

{

    uint i,j;

    for(j = 0; j < 10 ; j++)

    {

      for(i = 0; i < n ; i++);

    }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

初始化串口0函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initUART0(void)

{

    CLKCONCMD &= ~0x40;              //设置系统时钟源为32MHZ晶振

    while(CLKCONSTA & 0x40);         //等待晶振稳定

    CLKCONCMD &= ~0x47;              //设置系统主时钟频率为32MHZ

    PERCFG = 0x00;         //位置1 P0口

    P0SEL = 0x3c;              //P0用作串口

    P2DIR &= ~0XC0;                  //P0优先作为UART0

    U0CSR |= 0x80;         //串口设置为UART方式

    U0GCR  |= 9;         //波特率设为19200

    U0BAUD |= 59;

    UTX0IF = 1;                      //UART0 TX中断标志初始置位1

    U0CSR |= 0X40;         //允许接收

    IEN0 |= 0x84;        //开总中断，接收中断

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

串口发送字符串函数

参数：\*Data要发送的字符串；len字符串的长度

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UartTX\_Send\_String(char \*Data,int len)

{

  int j;

  for(j=0;j<len;j++)

  {

    U0DBUF = \*Data++;    //将数据放入串口缓冲区

    while(UTX0IF == 0);  //等待数据发送完成

    UTX0IF = 0;          //清除中断标志

  }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

主函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

  InitLED();

  initUART0();

  while(1)

  {

    if(RXflag == 1)   //接收到数据

    {

      LED2 = 0;           //LED2接收状态指示

      RXflag = 0;              //标志清零

      Delay(5000);

      UartTX\_Send\_String(Recdata,datanumber);

      datanumber=0;

      LED2 = 1;

    }

    else

    {

      UartTX\_Send\_String(Txdata,strlen(Txdata));

      Delay(50000);

      Delay(50000);

      Delay(50000);

    }

    LED1 = ! LED1;

  }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 串口接收中断处理程序:

一旦有数据从串口传至CC2530,则进入中断，将接收到的数据赋值给变量temp.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#pragma vector = URX0\_VECTOR

 \_\_interrupt void UART0\_ISR(void)

{

  uchar temp;

  temp = U0DBUF;

  if(temp == '#'||temp == '\n')           //‘#’结束标识符

  {

    Recdata[datanumber++] = '\n';

    RXflag = 1;          //接收完成

  }

  else

    Recdata[datanumber++] = temp;

  URX0IF = 0;   //清中断标志

}

彩灯

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*  晶振频率    ：选取用RC 11.0592MHz

\*  文件名      ：IO.c

\*  功能说明    ：STC12C5A16S2单片机的I/O口演示实验

\*  制作        ：www.frotech.com

\*  变更记录    ：2013.04.27

\*  变更内容    ：新建造

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <STC12C5A60S2.h>

#define Main\_Fosc 11059200L      //定义外部晶振的频率

sbit  P1\_0 = P1^0;               //定义特殊功能寄存器位变量P1\_0为P1口的第0位

void delay\_ms(unsigned char ms); //声明毫秒级延时函数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*  函数名：main( void )

\*  输入：无

\*  输出：无

\*  功能描述：实验LED的D1灯循环闪烁

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

      P1\_0 = 1;                 //P1\_0输出使得D1初始状态为灭

      while(1)

      {

          delay\_ms(5000);       //延时

          P1\_0 =~P1\_0 ;         //对P1\_0口的状态循环取反，使其亮，灭循环变化

      }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*  函数名：delay\_ms(unsigned char ms)

\*  输入：延时的时间

\*  输出：无

\*  功能描述：实现毫秒级延时

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay\_ms(unsigned char ms)

{

    unsigned int i;

    do{

        i = Main\_Fosc /1400;

        while(--i);

    }while(--ms);

}