****

**嵌入式系统 实验四**

**实 验 报 告**

指导老师： 贺建彪 戴训华

学 院： 计算机学院

专 业： 物联网工程

班 级： 物联网1802

学 号： 8208181125 8213180228

姓 名： 王灏洋 王云鹏

## 实验目的

1. 掌握UART串口的工作原理；
2. 掌握Cortex-M7的UART串口配置方法；
3. 通过实验掌握printf()函数重定向的方法；
4. 通过实验掌握Cortex-M7串口通信及调试方法。

## 实验设备

* 硬件：ARM Cortex-M7实验平台，ULINK2 USB-JTAG仿真器套件，PC机。
* 软件：µVision IDE for ARM集成开发环境，Windows 98/2000/NT/XP。

## 实验要求

编写程序，对指定UART端口进行初始化，完成串口通信相关寄存器的配置，完成串口数据的发送与接收。

实验中将printf()函数的输出重定向至串口，使得通过调用printf()函数即可实现向串口发送数据的功能。在实验过程中学习Cortex-M7中UART相关寄存器的设置、初始化（重点掌握波特率、起始位、校验位等串口通信参数的设置）以及printf()函数重定向的方法，学习使用串口对程序进行调试的方法。

## 实验原理

* UART通信协议

UART（Universal Asynchronous Receiver and Transmitter）通用异步收发器是异步串行通信接口的总称，支持RS-232、RS-422、RS-485等接口标准规范和总线标准规范。

UART作为异步串口通信协议的一种，工作原理是将传输数据的每个字符一位接一位的传输。其中每一位(Bit)的意义如下：

起始位：先发出一个逻辑“0”的信号，表示传输字符的开始。

数据位：紧接着起始位之后。数据位的个数可以是4、5、6、7、8等，构成一个字符。从最低位开始传送，靠时钟定位。

奇偶校验位：数据位加上这一位后，使得“1”的位数应为偶数(偶校验)或奇数(奇校验)，以此来校验数据传送的正确性。

停止位：它是一个字符数据的结束标志。可以是1位、1.5位、2位的高电平。由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，并且提供计算机校正时钟同步的机会。适用于停止位的位数越多，不同时钟同步的容忍程度越大，但是数据传输率同时也越慢。

空闲位：处于逻辑“1”状态，表示当前线路上没有数据传送。

UART协议传输时序如图3-25所示。

|  |
| --- |
|  |

发送数据过程：空闲状态，线路处于高电位；当收到发送数据指令后，拉低线路一个数据位的时间T，接着数据按低位到高位依次发送，数据发送完毕后，接着发送奇偶校验位和停止位（停止位为高电位），一帧数据发送结束。

波特率是衡量数据传输速率的指标，表示每秒传送数据的字符数，单位为Baud。UART的接收和发送是按照相同的波特率进行收发的。波特率发生器产生的时钟频率不是波特率时钟频率，而是波特率时钟频率的16倍，目的是为在接收时进行精确地采样，以提取出异步的串行数据。根据给定的晶振时钟和要求的波特率，可以算出波特率分频计数值。

* printf()函数重定向

标准库函数的默认输出设备是显示器，因此必须对重新定义printf()函数中与串口输出相关的函数，才能通过调用printf()函数向串口发送数据。

使用以下代码可以完成printf()函数的重定义：

#ifdef \_\_GNUC\_\_

/\* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf

set to 'Yes') calls \_\_io\_putchar() \*/

#define PUTCHAR\_PROTOTYPE int \_\_io\_putchar(int ch)

#else

#define PUTCHAR\_PROTOTYPE int fputc(int ch, FILE \*f)

#endif /\* \_\_GNUC\_\_ \*/

/\* Retargets the C library printf function to the UART \*/

PUTCHAR\_PROTOTYPE

{

/\* Place your implementation of fputc here \*/

/\* e.g. write a character to the COM1 and Loop until the end of transmission \*/

HAL\_UART\_Transmit(&Uart\_Handle, (uint8\_t \*)&ch, 1, 0xFFFF);

return ch;

}

由于GNU编译器中printf()调用的是putchar()函数执行底层输出任务，所以前半段代码使用宏定义可以兼容不同编译器。后半段即重定义过程，使用HAL\_UART\_Transmit()函数完成fputc()或putchar()函数的底层输出任务。HAL\_UART\_Transmit()函数的具体定义可以在实验例程“03\_UART”中查看。

## 实现内容和步骤

* 准备实验环境

使用ULINK2 USB-JTAG仿真器连接ARM Cortex-M7实验板与PC，实验板一侧接右下方的P1接口。使用串口线，连接实验板右侧的串口J3和PC机的串口。

* 串口接收设置

在PC机上运行windows自带的超级终端串口通信程序（波特率115200 、1 位停止位、无校验位、无硬件流控制）；或者使用其它串口通信程序

* 打开实验例程

拷贝实验平台附带程序“04\_UART”，使用µVision IDE for ARM通过 ULINK2 USB-JTAG仿真器连接实验板，打开工程文件，编译链接工程，根据本实验指导书中2.3.2小节中“编译配置”部分对工程进行配置（工程默认已经配置正确），点击MDK的Project菜单，选择Rebuild all target files进行编译，编译成功后，点击Debug菜单，选择Start/Stop Debug Session项或点击工具栏中的图标，下载工程生成的.axf 文件到目标板的 RAM中调试运行。

* 观察实验结果

结合实验内容和相关资料，使用一些调试命令，观察程序运行。注意观察PC中超级终端显示信息，根据提示使用键盘输入数据，MCU接收到数据后通过串口将收到的数据重新发送至PC，并点亮LED。

## 结果演示

* 原始代码结果演示

|  |
| --- |
|  |

* 修改代码结果演示：成功完成实验内容

|  |
| --- |
|  |

## 总结收获

首先证明王灏洋和王云鹏同学来上课了。

|  |
| --- |
|  |

通过这次实验，我们感受到了嵌入式系统的魅力。刚开始的时候，我们还不知道到底输入是如何进行的，后来发现通过HAL\_UART\_Receive就可以进行读取，而通过控制buffer的大小，就可以控制输入多少字符进行一次显示，通过这样一个洞察，我成功完成了实时显示，总的来说，从刚开始时候的迷茫，到后来时候的有些懂了，再到后来的，可以修改代码，这个过程真的还蛮有趣。

这次实验中的成功与失败都给了我们丰富的体验，让我们体会到开发的不易，工艺打磨的艰辛。实际应用中的bug往往出乎意料，所以我们只有将知识掌握的更加牢固，将能力提升到更高的水平，才能够在实际应用中披荆斩棘，化腐朽为神奇。而成功的体验更是让我们体会到从知识到实践的喜悦，这种将自己所学化为现实的感觉无可替代，给了我们更加充分的自信心。我相信，这次实验对于别人来说是一小步，但对我个人来说是一大步。利用这次实验带给我的体验，我将有更加浓厚的兴趣学习开发嵌入式相关的应用系统展望未来，我们会更加努力学习课程的知识，为未来的职业生涯打下坚实的基础。

## 附录：源代码

* 我们通过截图的方式进行代码分析

|  |
| --- |
|  |