基于 ARM 处理器的 UART 设计

杨　雪　郭家虎
(安徽理工大学　安徽　淮南　232001)

【摘　要】UART 是用于控制计算机与串行设备的装置,在嵌入式系统中它操作简单、工作可靠、抗干扰强。本文介绍了一种近距离的通信方法,讲述了 UART 的原理和软件设计的改进,在 ARM 中采用串口通信技术,具有使用方便、程序简单、可读性好、工作效率高等优点,可以广泛应用于基于串行通信的各种场合。结果表明,此设计有助于提高数据传输的实时性。

【关键词】通用异步收发器;异步通信;ARM;中断

Design of UART Based on ARM Processor

Yang Xue　Guo Jiahu
(**Anhui University of Science and Technology, Huainan 232001,China**)

【Abstract】UART is an equipment which is used to control computer and serial device. It is easy to operate, work reliably and has strong anti-jamming. This article introduces a method of short distance communication. Then UART's theory and software flow are expounded. Use technology of serial communication in ARM to make the system convenient, easy, readable and efficient. It can be used in the circumstances based on serial communication. The result indicates it can help to enhance the transmitted data's real-time performance.

【Key words】UART; asynchronous communication; ARM; interrupt

1.引言

UART(Universal Asynchronous Receiver Transmitter) 通用异步收发器在通信、控制等领域得到了广泛的应用。UART 作为微机系统 I/O 接口中重要的组成部分,主要进行串行和并行数据流间的转换,它与微处理器的总线接口是并行连接的,与外界是串行连接的[1]。

RS-232 被定义为一种在低速率串行通信中增加通信距离的单端标准(由于采取不平衡传输方式,即所谓单端通讯)。ARM 系统需要通过串口进行程序调试,目前它是 PC 机与通信工业中应用最广泛的一种串行接口。为此,本文提出了串口及中断处理,可以显著提高系统处理数据的效率,增强系统稳定性。

2.UART 原理

串行通信是指将构成字符的每个二进制数据位依据一定的顺序逐位进行传送的通信方法。在串行通信中,有两种基本的通信方式:异步通信和同步通信,这里我们采用的是异步通信。

2.1 异步串行通信标准

所谓串行通信是指外设与计算机间使用一根数据信号线,数据在传输过程中是通过一位一位地在一个数据信号线传输实现通信的,每一位数据都占有一个固定的时间长度。在 UART 中,数据位是以字符为传送单位的,数据的前、后要有起始位、停止位来实现字符的同步,另外可以在停止位的前面加上一个比特的校验位来对所传输的字符加以确认,所以收发双方采取异步通信措施[2]。

当发送一个字符代码时,字符前面要加一个起始信号,其长度为一个码元,极性为"0",即空号极性,字符后面要加一个终止符号,其长度为 1~2 个码元,极性为"1",即传号极性。加上起始终止信号后,可以区分出其要传输的字符,传送时可以连续发送,也可以单独发送。其数据格式如图 1 所示。

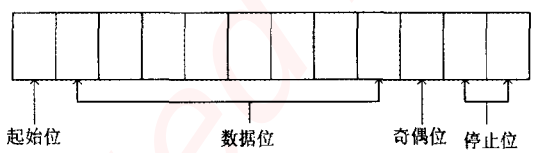

图 1　异步串行通信的数据格式

当接收设备收到起始位后,紧接着就会收到数据位。数据位的个数可以是 5、6、7 或者 8 位的数据。在字符数据传送过程中,数据位从最低位开始传输。

数据发送完之后,可以发送奇偶校验位,奇偶校验位用于有限差错检测。如果选择奇偶校验,则数据位和奇偶位的逻辑"1"的个数为偶数;若为奇校验,逻辑"1"的个数为奇数。在奇偶位或者数据位(此时没有奇偶校验时)之后发送停止位,可以是 1 位、1.5 位或者 2 位的低电平,接收设备收到停止位后,通信线路便恢复逻辑"1"状态,直到下一个字符数据的起始位到来。停止位是一个数据的结束标志。

2.2 RS-232C 接口

RS 是英文"推荐标准"的缩写,232 为标识号,C 表示修改次数。RS-232C 总线标准设有 25 条信号线,有一个主通道和一个辅助通道,在大多数情况下主要是用主通道。对于一般双工通信,仅需几条信号线就可实现,如一条发送线、一条接收线和一条地线[3]。

RS-232C 串行接口总线适用于:设备之间的通信距离不大于 15m,传输率最大 20kbps,规定的数据传输速率为每秒 50、75、100、150、300、600、1200、2400、4800、9600、19200 波特。RS-232C 采用负逻辑,即逻辑"1":-5V~-15V;逻辑"0":+5V~+15V。

一个完整的 RS-232C 接口有 22 根线,采用标准的 25 芯插头座。它规定连接电缆和机械、电气特性、信号功能及传送过程等,这些都是物理层的。图中的左边是微机串行接口电路中的主芯片 UART,它是 TTL 器件,下面是 RS-232C 连接器,RS-232C 所有的输入、输出信号都要分别经过 MC1488 和 MC1498 转换器,进行电平转换后才能送到连接器上去或从连接器上送进来,如图 2 所示。

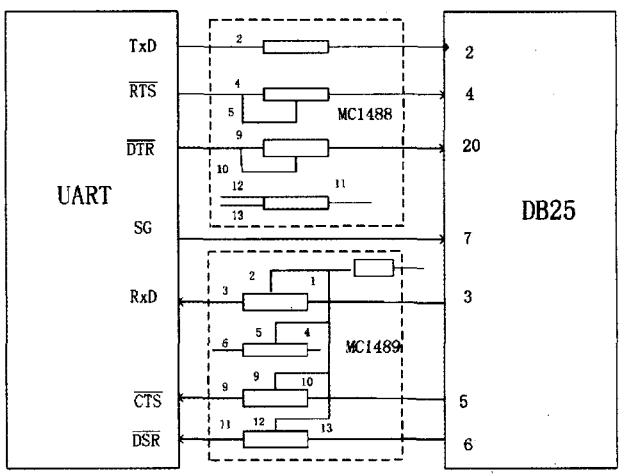

图 2　UART 与连接器示意图

2.3 串行通信接口的基本任务

(1) 实现数据格式化:来自 CPU 的是普通的并行数据,接口电路应具有实现不同串行通信方式下数据格式化的任务。在异步通信方式下,接口自动生成起止式的帧数据格式。在面向字符的同步方式下,接口要在待传送的数据块前加上同步字符。

(2) 进行串/并转换:串行传送,数据是一位一位串行传送的,而计算机处理数据是并行数据。当数据由计算机送至数据发送器时,首先把串行数据转换为并行数据才能送入计算机处理。串/并转换是串行接口电路的重要任务。

(3) 控制数据传输速率：串行通信接口电路应具有对数据传输速率——波特率进行选择和控制的能力。

(4) 进行错误检测：在发送时接口电路对传送的字符数据自动生成奇偶校验位或其他校验码。在接收时，接口电路检查字符的奇偶校验或其他校验码，确定是否发生传送错误。

3.软件设计

3.1 设置串口控制信号

串口设备初始化设备包括：波特率、校验位和停止位设置。

设置波特率代码如下(波特率设置为 19200)：

```
Struct termios Opt;
Tcgetattr(fd,& Opt);
Cfsetispeed(& Opt,B19200);
Cfsetospeed(& Opt,B19200);
Tcsetattr(fd,TCANOW,& Opt);
```

设置校验位和停止位的代码(设置串口为无校验 8 位)：

```
Option.c_cflag &=~PARENB;
Option.c_cflag &=~CSTOPB;
Option.c_cflag &=~CSIZE;
Option.c_cflag |=~CS8;
```

设置串口 1 停止位代码：

```
Option.c_cflag &=~CSTOPB;
```

设置串口 2 停止位代码：

```
Option.c_cflag |=CSTOPB;
```

3.2 读入串口控制信号

发送数据操作代码：

```
char buffer[1024];
int Length=1024;
int nByte;
nByte = write(fd, buffer, Length);
```

读取串口数据使用文件操作 read()函数读取，如果设置为原始模式传输数据，read 函数返回的字符数就是实际串口收到的字符数。可以使用操作文件的函数来实现异步读取，如 fcntl，或者 select 等来操作。

读取串口数据操作代码：

```
char buff[1024];
int Len=1024;
int readByte = read(fd, buff, Len);
```

执行完成这些操作后需要执行关闭串口操作代码：

```
Close(fd);
```

3.3 中断程序

中断及中断处理程序直接影响到嵌入式系统的性能、安全性及稳定性。与 ARM 通信时，程序中要对串口进行初始化和配置，这样才能使其正常工作，发送命令后数据包是自动发送的，正常工作时会自动发送数据包[4]。

程序代码如下：

```
Void SerialInter () interrupt 4 //串口中断
 {
  EA = 0;  //关中断
   if (RI0)
 {
   RI0 = 0;
   if ((SeRecStart= = 0)&(SBUF0 = = 0xfb))
 {
   SeRecNum = 0;
   SeRecStart = 1;
   SeRec[SeRecNum] = SBUF0;
 }
   if ((SeRecStart)&(SeRecNum ! = 0)) //接收数据包数据
   SeRec[SeRecNum] = SBUF0;
   SeRecNum++;
   if ((SBUF = = 0xf7)&(SeRecNum = = SeRecNum = = SeRec[1]))
 {
   SeRecStart = 0;
   SeRecEnd = 1;
   RecRight = 1;
  }
 if ((SeRecNum>2)&(SeRecNum>SeRec[1]))
   {
     SeRecStart = 0;
     SeRecEnd = 1;
     RecRight = 0;
   }
 }
  EA = 1; //中断结束，开中断
}
```

3.4 调试程序

在调试 ARM 之前，必须对串口工作情况进行调试，经过串口初始化和程序运行，将 PC 与 ARM 串口连接，设置好 PC 的串口号和波特率，在串口发送查询状态命令，并得到返回的数据包，表明与 ARM 通信正常[5]。

由于篇幅限制，只列出总线读写的调试命令。

代码如下：

```
If (scomm! =0xff)
Switch(scomm)
{ .
  .
  .
  case 0xd0: //总线读取(16 位)
    put_string("Read Word:");
    put_num32(SYSW_(address));
    put_string("h at ");
    put_num32(address);
    put_string("h\n");
    break;
  case 0xe0: //写总线(16 位)
    SYSW_(address) =data;
    put_string("Write Word:");
    put_num32(data);
    put_string("h at ");
    put_num32(address);
    put_string("h\n");
    break;
  default:
    put_string("Not Support! \n");
}
```

4.结束语

针对嵌入式系统应用，目前的 UART 器件普遍存在操作复杂、引脚多、价格昂贵等弱点，不能满足和适应嵌入式系统的需要，本文对它进行了改进，通过改写串口程序和中断程序，有助于数据实时性能提高以及降低程序开发的难度。它的软件设计简单，利用设计中的芯片进行电平转换，具有收发数据快的特点，使系统有较强的实时性。该设计在所实现的平台上得到了成功的应用。

【参考文献】

[1]时晨，张伟功.基于 AMBA 总线 UART IP 核的设计与实现[J].计算机应用，2003，23:36-38.

[2]杨晓斌，赵花荣，赵明生.通用异步串行接口的 VHDL 实用化设计[J].微计算机信息.2005,32:124-126.

[3]李雪莉，张兆莉，史晓龙.面向嵌入式 Linux 系统的软中断设计与实现[J].微计算机信息.2007，23: 60-61.

[4]Michael Beck, Harald Bohme, Mirko Dziadzka. Linux Kernel Programming 3rd Edition.[M]. 2004，11:27-28.

作者简介：杨雪(1984—)，女，湖南湘阴人，硕士研究生，主要研究领域为嵌入式系统；

郭家虎(1974—)，男，硕士生导师，主要研究领域为嵌入式系统和电力电子等。

[责任编辑：汤静]

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道

7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码

14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现

12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计

Hardware:

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究

Programming:

1. 计算机软件基础数据结构 - 算法