 DOI: https://doi.org/10.12349/iser.v4i2.1225

Design of a RS485 Slave Station Based on μC/OS-Ⅱ and AT91RM9200

Xuesong Zhang

China National Electric Apparatus Research Institute Co., Ltd., Guangzhou, Guangdong, 510860, China

Abstract

The design of RS485 slave station based on μC/OS- Ⅱ and AT91RM9200 is used to transmit data between modules in industrial equipment. In this paper, the data transmission from USART to Application layer is implemented by message queues of μC/OS- Ⅱ and peripheral data controller in AT91RM9200 which can reduce overhead of the processor. The data frame will be parsed through the Modbus-RTU protocol. It is proved that the design is stable and reliable in practical work.

Keywords

μC/OS-; RS485 slave station; ModBus-RTU

基于 μC/OS–II 和 AT91RM9200 的 RS485 从站设计

张雪松

中国电器科学研究院股份有限公司，中国 · 广东 广州 510860

摘　要

论文针对工业设备内部各模块之间经常需要通过RS485进行数据交互的问题，在AT91RM9200微处理器平台上实现了一种基于μC/OS-Ⅱ系统下的RS485从站设计。论文利用系统消息队列完成了数据从AT91RM9200微处理器串口到应用层的数据传输，同时使用外设数据控制器降低了数据收发过程中的微处理器资源开销，数据则采用Modbus-RTU协议进行解析处理。实践证明该设计是稳定可靠的。

关键词

μC/OS-Ⅱ；RS485从站；ModBus-RTU

1 引言

随着工业技术的不断发展，设备的自动化、集成化、智能化、模块化的程度越来越高，控制方式也逐渐从集中控制向分布式控制方式发展。设备内部常常配备了不同功能的嵌入式模块，用于完成不同的任务。模块之间需要通过大量的数据交互协作。在工业应用中，模块本身的功能都比较繁杂，常常通过采用实时操作系统（RTOS）创建多个并发任务来实现，通信任务作为其中的一种，具有数据量大、数据交互频繁、可靠性、抗干扰性和实时性要求高的特点，但同时又不能因此而过多地影响系统本身的性能。在 RS485 从站的设计中，一般都是采用串口接收中断、发送中断和定时器中断来实现，中断次数较多，而且操作相对烦琐。论文主要针对上述问题，介绍了一种在 μC/OS-II 下的 RS485 从站设计方法。

【作者简介】张雪松（1987–），男，中国四川乐山人，硕士，工程师，从事电池二次检测设备控制及嵌入式系统研究。

2 μC/OS–II 操作系统简介

μC/OS-II 是一种可剥夺型多任务实时操作系统内核。它提供了任务调度、多任务管理、任务通信交互、时间管理和内存管理等基本的系统功能，而文件系统、通信协议栈等则需要用户额外实现。μC/OS-II 具有可移植性强、可固化、可裁剪、稳定可靠和执行效率高等特点，特别适合嵌入式应用场景，在微控制器中得到了广泛使用。

在 μC/OS-II 等 RTOS 中编写应用程序与在裸机环境中不同的是，应用程序是由一个个的任务组成的，每一个任务都可以看作一个无限循环的函数。在 μC/OS-II 中，各个任务可处于等待状态、睡眠态、就绪态、运行态和被中断态五种状态，在内核的调度和管理下进行状态切换，并按照设定的优先级占用微处理器进行工作。各任务是独立运行的，通过中断来响应用户的外部异步事件，任务之间可通过信号量、消息队列、邮箱等方式进行数据交互和同步。

3 AT91RM9200 及串口外设资源简介

AT91RM9200 为 ATMEL 公司生产的一款基于 ARM920T™ARM®Thumb® 处理器构建的微处理器，主频最

大为 180MHz，处理速度可达 200MIPS，含有丰富的应用外设及标准接口。论文使用其中的通用同步 / 异步收发器（USART）和外设数据控制器（PDC）进行设计。

AT91RM9200 包括 4 个 USART，提供全双工通用同步 / 异步串行连接，可以进行数据帧格式编程以适应多种串行标准，接收器支持奇偶错误、溢出错误、接收超时等检测。USART 支持连接外设数据控制器（PDC），使用 PDC 可在 UART、SPI 等片上串行外设与存储器之间直接读写数据，数据传输过程可以不需要处理器寄存器参与，使其有更多资源去进行计算和控制工作或者响应其它异步事件，从而改善处理器性能。PDC 通常是成对构建的，一个负责接收，一个负责发送，在 AT91RM9200 中，其用户接口位于相应的外设存储空间中。

4 ModBus–RTU 通信协议简介

RS485 本身没有规定应用层的通信协议，可以按照实际需求自行设计。工业上常用 RS485 通讯协议主要是 Modbus-RTU 和 Modbus-ASCII，其中 Modbus-RTU 应用范围更广，已经成为了一种基本的标准通信协议栈。论文采用 Modbus-RTU 协议栈进行数据的解析处理。

4.1 Modbus–RTU 协议数据帧及帧间隔时间简介

Modbus-RTU 的数据帧格式如表 1 所示，数据帧最大长度为 256 字节，除去地址码和校验码，协议处理单元的最大长度为 253 字节，有 256 个不同地址的寻址空间，其中 0 为广播地址。Modbus-RTU 规定相邻两帧的时间间隔至少大于 3.5 个字符时间。实际使用中，当波特率大于 19200bps 时，时间间隔通常采用固定值 0.175ms。

表 1 ModBus–RTU 数据帧格式

地址码	功能码	数据	CRC16 校验
1 Byte	1 Byte	N Bytes	2 Bytes

4.2 功能码报文格式简介

功能码用于表示数据帧的功能。Modbus 主要有 1 到 16 总共 16 个功能码。在论文的设计中，RS485 主要用到其中的两个字操作指令：读保持寄存器功能码 03 和写多个保持寄存器功能码 16。其主站查询报文格式和从站响应报文格式分别如表 2~ 表 5 所示。需要注意的是，Modbus-RTU 协议中寄存器值含有两个字节，字节序为高位在前。

表 2 功能码 03 主站查询报文格式

地址码	功能码	起始地址	寄存器数	CRC16 校验
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes

表 3 功能码 03 从站响应报文格式

地址码	功能码	数据域字节数	数据	CRC16 校验
1 Bytes	1 Bytes	1 Bytes	2N Bytes	2 Bytes

表 4 功能码 16 主站查询报文格式

地址码	功能码	起始地址	寄存器数	字节数	数据	CRC16 校验
1 Byte	1 Byte	2 Bytes	2 Bytes	1 Byte	2N Bytes	2 Bytes

表 5 功能码 16 从站响应报文格式

地址码	功能码	起始地址	寄存器数	CRC16 校验
1 Bytes	1 Byte	2 Bytes	2 Bytes	2 Bytes

4.3 通信异常处理

当通信出现异常时，从站需要返回异常响应报文给主站，其中的功能码为正常功能码 +0x80。异常响应报文如表 6 所示。常用的异常码主要包括非法功能码（01）、非法数据地址（02）、非法数据（03）和从站故障（04）。

表 6 异常响应报文格式

地址码	功能码	异常码	CRC16 校验
1 Byte	1 Byte	1 Byte	2 Bytes

4.4 错误校验

Modbus-RTU 会对地址码、功能码和数据进行 CRC16 校验，检错失效概率在 0.0047% 以下，对于相对封闭的现场通信总线来说完全满足使用需求。

5 软件设计

论文的 RS485 从站是在 μC/OS-II 操作系统环境下实现的，从站作为一个独立任务与其他业务任务并发运行。而主站的数据帧到达属于外部异步事件，需要通过串口中断向处理器发出中断请求，再通过中断处理函数获取数据帧的内容。

为了处理不定长的主站数据帧，Modbus-RTU 协议中规定了报文间隔时间，并以此作为一帧数据结束的标准。在通常的设计中一般是通过串口接收中断接收数据，结合定时器的启停和定时器中断去判断数据帧是否接收完毕，这样处理器必然会频繁地进入定时器中断和串口中断，加重处理器负担，任务也会被频繁打断执行。特别是在波特率较高和通信比较频繁的应用场景下，可能会影响到系统的实时性和稳定性。

如图 1 所示，论文使用 AT91RM9200 的串口接收超时中断、发送空中断和 PDC 结合的方式，接收一帧数据帧只触发一次超时中断，也无需使用定时器，减少了中断次数。为减少中断处理时间，中断函数接收数据后通过消息队列将接收到的数据帧指针发送给从站任务，由从站任务对数据帧的完整性和合法性进行判定，数据解析成功后再使用 PDC 将响应数据通过串口返回给主站。

5.1 串口 PDC 配置

AT91RM9200 的 PDC 用户接口集成在每个外设存储器映射空间中，在完成常规的 USART 波特率等配置后，继续初始化 PDC 寄存器即可。表 7 列举了论文从站设计中需要

配置的寄存器。其中，偏移量为相对于所使用的 USART 寄存器存储空间地址的偏移地址。PTCR 用于使能或禁用 PDC 的发送和接收功能。RPR 用于存放数据接收缓存区地址，RCR 用于设定需接收的数据长度，在 PDC 接收数据过程中，该值则用于接收数据计数。TPR 则用于存放数据发送缓存区地址，TCR 值表示需发送的数据长度。

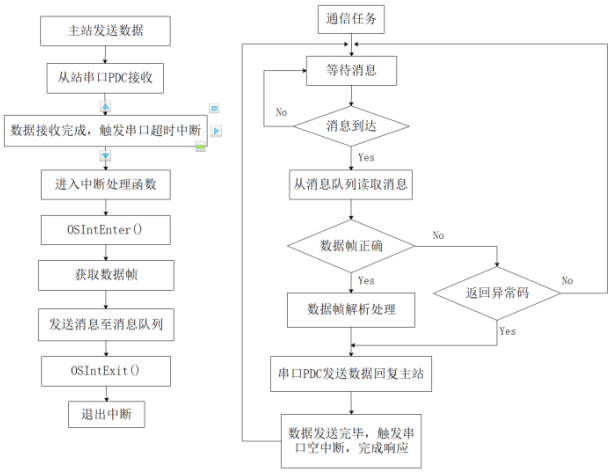

图 1 RS485 从站数据通信流程

表 7 需要配置的 PDC 寄存器

偏移量	寄存器
0x100	接收指针寄存器（RPR）
0x104	接收计数寄存器（RCR）
0x108	传输指针寄存器（TPR）
0x10C	传输计数寄存器（TCR）
0x120	传输控制寄存器（PTCR）

5.2 数据帧结束判断实现

AT91RM9200 的 USART 外设具有接收器超时判断功能，通过检测 RXD 线空闲状态，当空闲超时，通道状态寄存器（US_CSR）TIMEOUT 位置高，并产生超时中断。超时时间可在接收器超时寄存器（US_RTOR）中的 TO 域设置。

5.3 中断处理函数

使用 PDC 进行收发时，主要是检测串口超时中断和发送空中断标志。检测到超时中断时，读取 PDC 的 RCR 寄存器中的值，每一时刻实际接收到的数据帧长度为设定数据长度与 RCR 值之差。并重置 RCR 寄存器值和 RPR 寄存器地址。随后将获取到的值放入消息队列。由通信任务从消息队列中取出数据帧进行处理，将需要回复主机的数据放入 TCR 寄存器值指向的发送缓存区并使能 PDC 发送，发送完成后，触发 USART 发送空中断，表示数据帧已发送完成。完成一次收发。

μC/OS-II 支持中断嵌套，在 μC/OS-II 环境下进行中断处理时，需要额外调用进入中断服务函数 OSIntEnter（）和退出中断服务函数 OSIntExit（）。进入中断时需要通过 OSIntEnter（）来更新中断嵌套层数。退出中断处理函数之前则需要通过 OSIntExit（）来判定是否还有嵌套的中断处理未完成。同时由于同时 μC/OS-II 属于可剥夺型内核，还需要在 OSIntExit（）中判断被中断的任务是否是就绪的最高优先级任务从而确定是否需要执行任务切换。

5.4 消息队列在设计中的应用

μC/OS-II 中的消息队列实质上是一个环形缓冲区，支持从任务或中断处理程序将一则消息的指针放入消息队列，其余任务则可以通过内核服务从消息队列中得到消息的指针，获取消息中包含的数据内容。

论文主要使用了内核提供的 OSQCreat（）、OSQPost（）和 OSQPend（）三个函数来实现功能。首先在系统初始化时建立串口收发数据的消息缓冲区，并通过 OSQCreat（）创建消息队列，获取消息队列指针。当串口检测到超时中断，接收到一帧完整数据时，即通过 OSQPost（）向消息队列发送一则消息，消息内容为串口数据收发缓冲区地址。当中断退出后从站任务重新获得运行时，即可通过 OSQPend（）从消息队列中获取到数据收发缓冲区地址，并读取数据帧内容进行解析。由此可以看出，负责接收数据的串口中断处理函数可以看作生产者，负责处理数据，响应主站的通信任务可以看作消费者，二者通过消息队列进行连接，可以实现中断数据接收和数据处理的解耦，也可以有效避免处理器在处理数据的过程中丢失主站发送过来的数据，降低丢包率。接收消息的任务通过 OSQPend（）等待消息到来， 在等待的过程中，任务将被阻塞，处于等待状态并触发一次任务调度，直至串口接收到主站发送过来的一帧数据后，由内核调度重新进入运行状态。

6 结语

论文主要介绍了一种 μC/OS-II 操作系统和 AT91RM9200 平台下 RS485 从站的实现方法。所述的 RS485 从站实现已在实际中长期稳定运行，表明这种实现方式的稳定可靠，能较好兼顾处理串口中断响应速度和通信任务的实时性需求，减少数据丢包率，并且可以降低设计难度。该实现方式并不局限于论文所述的平台，在含有 DMA 的 MCU、串口超时中断和其他 RTOS 系统中也可以使用，具备一定的通用性。

参考文献

[1] [美]Jean J.Labrossse.嵌入.式实时操作系统μC/OS-II[M].邵贝贝,译.第2版.北京:北京航空航天大学出版社,2003.

[2] AT91RM9200参考手册[Z].

[3] 方羽,梁广瑞,罗覃东.基于μCOS-II的MODBUS协议的实现[J].装备制造技术,2009(1):83-84.

[4] 杨更更.ModBus软件开发实战指南[M].北京:清华大学出版社,2017.