基于 Visual Studio 的串口通信*

周 舒**

(华为技术有限公司 深圳 518057)

【摘要】 介绍了一种基于 Visual Studio 的 PC 对单片机的串口通信方案，该方案提供了 PC 和单片机端的通信系统模型，该模型从网络的 OSI 层次系统结构中演化而来。研究了该模块的接口、结构、数据结构和相互调度。

关 键 词 串口； 通信； 层次结构模型； 单片机

中图分类号 TN915； TP393

在计算机和单片机的通信中，串口是很常见的，价廉且没有过于复杂的编解码，硬件上的控制和实现简单，实现自己定义的协议不复杂。另外，对于实现控制流的数据量，其通信带宽也足够，因而在工业、通信、军事控制上的应用较广泛，特别是 PC 机(作为控制端)和单板(作为终端)的串口通信最为常见。本文对 PC 机和单板的串口通信方案进行讨论。

1 设计思路

本文介绍一种借鉴网络上分层结构的实现模式。网络协议中，为了便于分隔、简化功能模块，采用了层次结构这种系统分解方法，图1所示是国际标准化组织(ISO)定义的开放性网络层次结构模

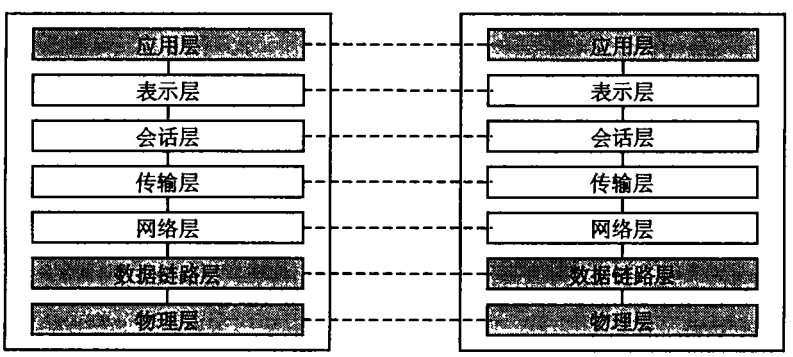

图1 ISO 网络层次结构

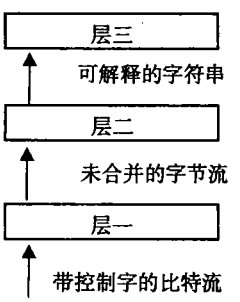

图2 串口通信层次结构

型(OSI)[1]。每一层负责直接使用下层向它提供的服务，并完成自身的功能，然后向上层提供“增值”后的功能，即数据和控制信息从发送端各层由上至下传送至物理媒介，然后接收方再从下到上经各个层传递后到达目的地点，实现串口通信的层次结构如图2所示。串口通信数据量小，不需要很复杂的语法和语意表达，在端对端的情况下，路由选择的问题比较简单，因此仅仅选择实现了物理层、数据链路层和应用层。接收时，层一负责从串口的硬件缓冲区中读取 PC 或单片机端发送的数据，层二把层一读取出来的数据根据一定的逻辑关系拼装成完整的命令，最后是层三来解释执行由层二拼装出来的命令，并作出响应。发送时，因为层一不关心数据的语法和语义，因此不需要层二的拼装功能，层三把数据直接发送给层一。由此可见，这种实现模式中，层一实现了

2000年10月13日收稿
* 电子部生产发展基金资助项目
** 男 25岁 硕士 工程师

物理层的作用，层二相当于数据链路层，而层三相当于应用层。这样采用分层处理的方法还有个好处是底层模块独立于高层消息处理，最大限度地降低软件对硬件的依赖性，有利于软件的重用和移植。另外，与不同硬件相关的 I/O 驱动程序各自独立，可单独修改而互不影响。

2 计算机上的实现

在 PC 端，如果采用 Windows 9x/NT 操作系统，其许多编程界面，如 Visual C++、Visual Basic、C++ Builder、Delphi 等已经提供了封装的串口功能模块，实现起来非常容易，并且可以采用消息触发机制。这里介绍常见的 Visual Studio 下的实现方法。在 Visual Studio 上串口的实现比较容易，因为它已经封装了针对串口操作的一系列函数，包括底层针对硬件的 inp/outp 系列函数，以及可实现覆盖式和非覆盖式的文件操作函数集合，常用的是文件操作函数[2]。

Visual Studio 中串口被虚拟成文件，所有的操作都虚拟成对文件的读写。这样，给开发者一个统一的编程界面，简化了编程接口。同时文件的处理屏蔽了硬件上的端口、地址等特征。虚拟的缓冲区代替了实际的硬件缓冲区，让开发者更专注于逻辑上 IN/OUT 的考虑。读写串口首先要创建一个和串口相联系的文件，相应的函数是：

CreateFile (lpFileName, dwAccess, dwShareMode, lpSecurity, dwCreation, dwFlags, hTemplateFile) ;

参数说明(按入口顺序排列)	备注
LPCTSTR lpFileName	文件名
DWORD dwAccess	文件读写权限
DOWRD dwShareMode	文件共享模式
LPSECURITY_ATTRIBUTES lpSecurity	文件句柄可否被子进程使用
DWORD dwCreation	文件创建模式
DWORD dwFlags	文件属性
HANDLEh TemplateFile	Windows 95不支持

需要注意的是，当把文件名设定为 COM1、COM2等字符串时，系统将把对应的硬件设备虚拟成程序试图创建的文件。建立了串口文件后，就可以对其进行读写。实际上，读文件对应的是从串口接收数据，写文件对应从串口发送数据，相应的函数为：

ReadFile (hFile, lpBuffer, nBytesToRead, lpBytesRead, lpOverlapped) ;

参数说明(按入口顺序排列)	备注
HANDLE hFile	文件句柄
LPVOID lpBuffer	数据接收区地址
DWORD nBytesToRead	读取多少字节
LPDWORD lpBytesRead	实际读取了多少字节
LPOVERLAPPED lpOverlapped	覆盖模式属性区指针

WriteFile (hFile, lpBuffer, nBytesToWrite, nBytesWritten, lpOverlapped) ;

参数说明(按入口顺序排列)	备注
HANDLE hFile	文件句柄
LPVOID lpBuffer	数据发送区地址
DWORD nBytesToWrite	发送多少字节
LPDWORDlpBytesWritten	实际发送了多少字节
LPOVERLAPPED lpOverlapped	覆盖模式属性区指针

串口文件不是普通的文件，串口不停地接收到数据和发送出数据。WINDOWS 系统提供了相应的一种特殊的函数，状态等待函数如下：

WaitCommEvent (hFile, lpEvtMask, lpOverlapped) ;

参数说明(按入口顺序排列)	备注
HANDLE hFile	文件句柄
LPDWORD lpEvtMask	等待的事件标志组合
LPOVERLAPPED lpOverlapped	覆盖模式属性区指针

该函数的作用是等待 lpEvtMask 参数所指定的事件或多个事件的发生。一旦调用，除非所有指定的事件都发生，该函数不会返回。在调用 RedFile 和 WriteFile 之前，先调用 WaitCommEvent，并把读或写的条件填入 WaitCommEvent 的参数 lpEvtMask 中，结果是实际执行读写操作的时候，可以保证读写的条件都已经满足。这里的关键是 WaitCommEvent 的参数 lpEvtMask，其中有很多标志位代表不同的事件(串口硬件的状态)，常用的有：

标志位	代表的事件
EV_BREAK	接收到输入的中断信号
EV_ERR	接收到错误的数据，包括校验错、帧错和溢出错
EV_RXCHAR	接收到一个字符
EV_TXEMPTY	上次要求发送的字节已经发送完成

用在 ReadFile 前面的一般是 EV_RXCHAR，用在 WriteFile 前面的是 EV_TXEMPTY，也可以在任何时候设定串口的各种属性，相应的函数有：

GetCommState (hFile, lpDCB) ； SetCommState (hFile, lpDCB) ；

lpDCB 是指向一个 DCB 结构的指针。DCB 结构中包含了很多有关串口的属性，常用的有：

标志位	含义
StopBits	设定停止位位数：1、1.5、2
Fparity	是否采用校验位
Parity	采用校验位的类型：奇、偶
Fbinary	是否采用二进制流，不带结束位
BaudRate	波特率：110、600、9 600、19 200、115 200、256 000…

由此，可根据不同的协议流程，比较容易地写出 PC 机的串口通信程序。

3　单片机上的实现

在单片机端，没有现成的函数接口和虚拟数据缓冲区，必须通过汇编语言直接针对硬件编程，实现如图2所示的所有层次，即要解决从硬件接口的原始比特到应用层的逻辑数据结构的变换，并实现相应的操作。

在有一个可实现多任务调度的内核的条件下，实现层次结构有两个要点：1) 层次之间的数据结构；2) 各个层次之间的优先级和调度的同步问题。

3.1　单片机上的操作系统

在单片机上运行某个程序，需要操作系统可以是工业上通用的产品，如 pSOS、VRTX 或 VRWORKS 等，也可以是通过汇编语言实现的、面向特定应用的用户自定义模块。经比较，通用的工业产品功能齐全、性能稳定，提供整套现成的有关进程调度、任务通信等函数接口，使用起来非常方便。但是，齐全的功能配置也导致了体积庞大，因而很难根据具体的对象进行优化。而用户自己编写的操作系统有一定的优势，其短小、精练，只须实现需要的有限几个功能，还可以随时修改源代码，灵活性很好，可节省软件购买费用和使用许可证费用。两者各有优势，可以根据自己的需要选择。

3.2 层次之间的数据接口

从图3示出的实现模型中，包括了层一和层二，层二和层三两个接口数据结构。根据各个层次的功能划分，层一和层二之间的数据接口主要实现硬件的小容量缓冲区和逻辑上的大容量缓冲区的转换；层二和层三之间的数据接口则实现物理上的比特流到逻辑上的完整命令和数据结构的转换。在实现上，层一和层二的接口是一个字符串形式的缓冲区。层一把从硬件缓冲区中读取的数据填充到其中，其原因是硬件缓冲区太小了(仅一个字节)，而一般的单片机 CPU 处理速度不是很快，又要分时处理高层的数据拼装和相应的处理。如果整个过程的处理速度比串口的传输速度慢，则会出现第一个字节还没有处理完，缓冲区内的第二个字节就被第三个字节覆盖，所以需要一个大的缓冲区来满足速度上的差别，即提高串口的传输速度。这个缓冲区用如图4所示的一个循环队列来实现[3]。序号为 X 的单元的后续单元是$(X+1)\%M$，同时该数组维护两个指针，一个指向数组头，一个指向数组尾。这样，读取的时候从数组头开始取数据，并把头指针向后移动；填写的时候从数组尾开始添加数据，并把尾指针向后移动。循环数组的好处是整个数组的单元可以循环使用，省略了对数组单元是否有数据的判断。但是数组的可用空间比实际空间少一。假设数组有 M 个单元，则实际数组满了的情况下，总共可以存储 M–1个单元。扩展硬件缓冲区为更大的逻辑缓冲区主要是为了适应 CPU 处理和硬件端口接收的速度差异，提高系统应付大量突发数据流的能力。忽略中断切换的时间，假设 CPU 处理一个字节的时间为 t_1，串口上的数据流的速度为每秒 x 个字节，即串口的处理速度 $t_2=1/x$。在 $t_2>t_1$ 的情况下，系统仅能承受一个字节的数据。当缓冲区扩大到 M 字节的时候，系统能正确接收数据的时间为 $M\Big/\left[x-\frac{1}{t_1}\right]$，能承受的字节为 $Mx\Big/\left[x-\frac{1}{t_1}\right]$，是单纯使用硬件的 M 倍，即扩大的缓冲区能极大提高系统的性能。

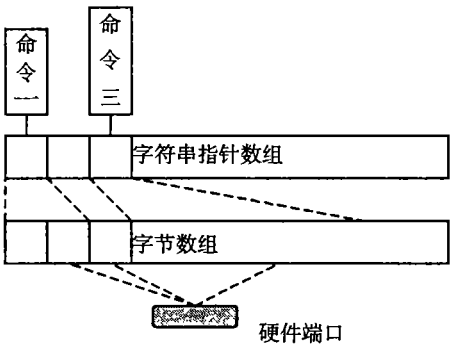

图3 层次间数据结构接口

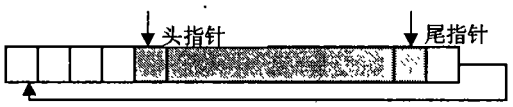

图4 循环数组

层二和层三之间的数据结构是为了实现通信协议的处理而设定的。层二把层一中的字节串拼装成一个个命令，然后存储在该结构中，以便层三来处理。在本模型中，我们把层二和层三的数据接口也设计成一个循环数组，但是数组的单元是指针，每个指针指向一条完整的消息。消息的填写和分离由层二控制，层二拼装完成一条消息之后，通知层三处理，层二和层三对消息指针数组的处理和上面所述的层一层二接口类似。

层二的功能和层三的功能可以合并在一个任务里面，这样可以减少系统的任务切换消耗。但在逻辑上，合并层二和层三使消息的确认和处理区分得不清晰，不利于以后的模块继承和修改。因此本文将其分成了两个层次，便于以后的修改和移植。

3.3 层次之间的调度

在多任务的条件下，每个任务的执行都需要掌握 CPU 计算时间，而 CPU 在一个时刻只能为一个任务服务，因此必然产生任务调度和优先级的确定问题[4]，即要避免死锁。一般需要使用硬件(互斥资源)的任务优先级应该较高，多个使用同一硬件的任务之间应该调整好等待机制和触发机制；低层任务的优先级应该高于高层任务，避免高层任务“等米下锅”；有逻辑联系的任务之间应该设定好触发条件，保证执行顺序的正确；相互间独立的任务、相对重要的任务应该有更高的优先级。

本文讨论的模型，在单片机中高层的两个层次对应两个任务。接收方面采用中断处理，用安装中断的方式监视和控制串口硬件资源的使用和状态，实际上是一个任务[5]；发送方面单独有一个任务。在优先级的安排方面，数据接收是硬件中断触发的，相当于最高的优先级；层二的优先级比层一低一点，层三的优先级又低于层二；发送任务是在层一和层三之间通信，所以优先级设定在这两个任务的优先级之间。各个层次之间有相互的联系：层一接收到字节以后触发层二，层二拼装完成消息以后触发层三，层三的发送消息处理触发发送任务，模型的层次调度关系如图5所示。

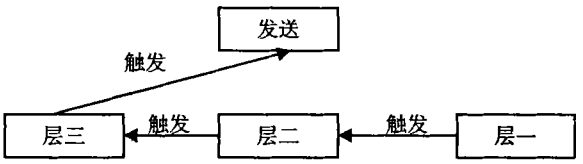

图5　任务调度关系

4　结 束 语

本文借鉴网络 OSI 分层结构的模式，实现了一种 PC 机和单板的串口通信方案。分层结构体现了逻辑上的模块化，有利于系统的移植和修改，最大限度地降低了软件对硬件的依赖性。且不同硬件的相关 I/O 驱动程序各自独立，保证了模块的内聚性和模块之间的独立性。本系统可适用于工业、通信、军事控制等领域的应用。

参 考 文 献

1　Tanenbaum A S. Computer networks. Seattle: Prentice Hall Inc , 1997

2　Microsoft Corp. Microsoft foundation class library. Seattle: Microsoft Corp, 1997

3　严蔚敏，吴伟民. 数据结构. 北京：清华大学出版社，1992

4　汤子瀛，杨成忠，哲凤屏. 计算机操作系统. 西安：西安电子科技大学出版社，1991

5　沈美明，温冬婵. IBM-PC 汇编语言程序设计. 北京：清华大学出版社，1991

Serial Communication in PC-Singlechip System

Zhou Shu

(Huawei Technology Company　Shenzhen　518057)

Abstract　Based on visual studio, this paper introduces a solution of PC-to-singlechip serial communication. It provides the modules of communication system in both PC and single chip, which are derived from the OSI network hierarchy module. This paper also provides the analysis of construction, data structure and scheduling of the modules. The hierarchy structure of this solution reflects the logical modularity of the whole system, whereby the system eliminates the reliability on hardware as much as possible. It provides the portability and flexibility to the system. The independence of each I/O driver on different hardware proves the coherence of subsystems.

Key words　serial;　communication;　hierarchy module;　singlechip system