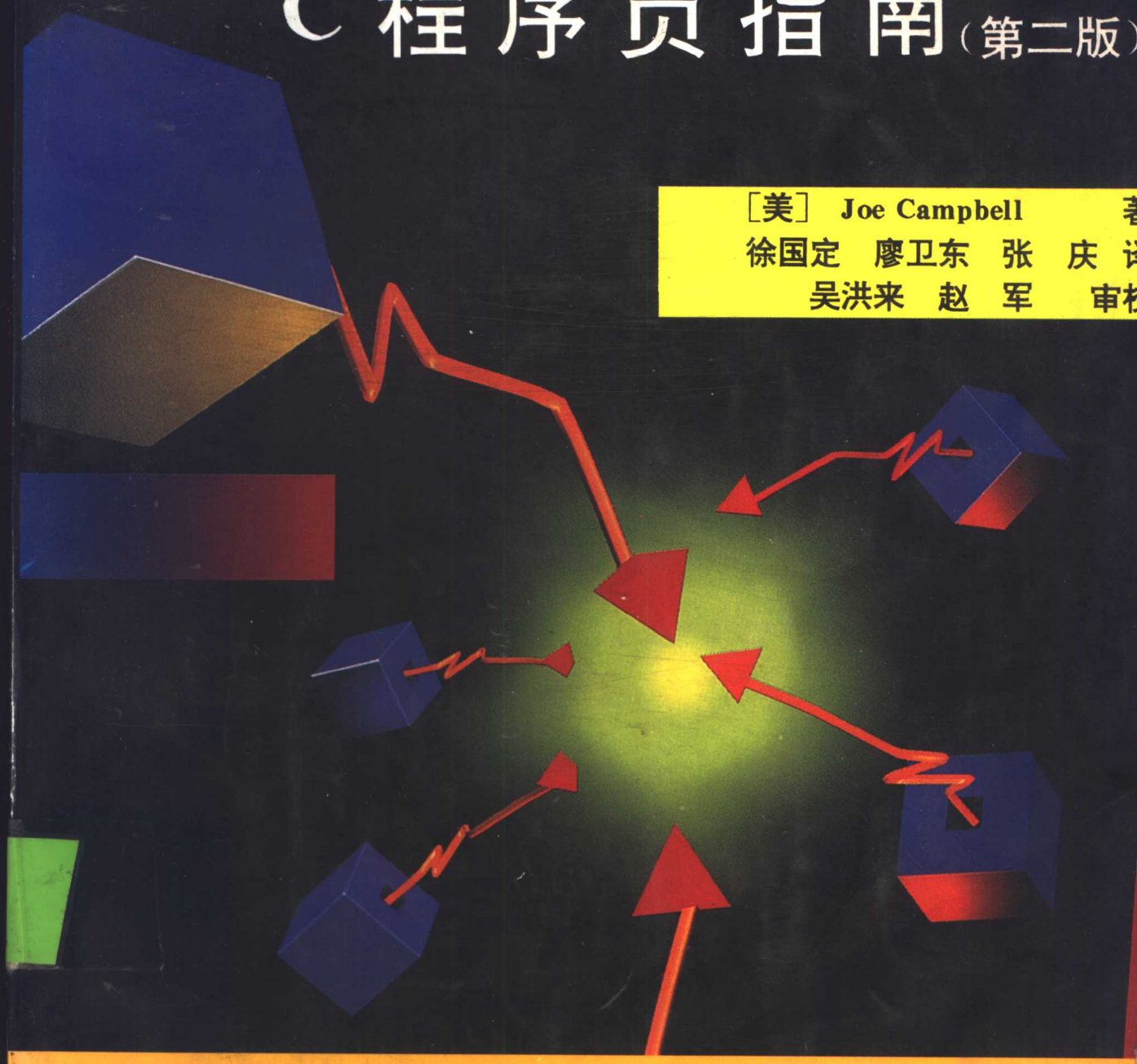
SAMS

北京科海培训中心

串行通信
C程序员指南（第二版）

[美] Joe Campbell 著
徐国定 廖卫东 张 庆 译
吴洪来 赵 军 审校



清华大学出版社

SAMS 北京科海培训中心

串行通信C程序员指南

（第二版）

[美] Joe Campbell 著

徐国定 廖卫东 张 庆 译

吴洪来 赵 军 审校

清华大学出版社

(京)新登字 158 号

C Programmer's Guide to Serial Communications

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:门头沟胶印厂
发　行:新华书店总店北京科技发行所
开　本:16　印张:45.5　字数:1107 千字
版　次:1995 年 6 月第 1 版　　1995 年 6 月第 1 次印刷
印　数:0001～8000
书　号:ISBN 7-302-01900-2/TP・864
定　价:78.00 元

作者介绍

Joe Campbell 是多部计算机图书的著作者，这些图书中包括经典之作《RS-232 解决方案》(RS-232 Solution)。Joe 曾经是 Everex Systems 公司调制解调器分部的软件开发经理，并在那个岗位上工作了五年。与此同时，他不仅编写了 TIFF Class F(一种用以存储传真图象的标准文件格式)，而且创建了 FaxBios(一个业已被 Word Perfect 以及其他公司所采纳的传真 API)。此外，Joe 还是“2 类”(Class 2)传真调制解调器(即 EIA/TIA-592 标准)的一名技术贡献。

第一版 序 言

考虑到每位编程人员都不可避免地要和串行端口打交道，本书至今尚未问世很是不可思议。在这方面，一个显而易见的因素是串行端口所包含内容的纷繁复杂。的确，在写作某方面专著的过程中，作者常常会希望毫无遗漏地覆盖该主题的所有方面，但是，鉴于其几近无限的涵盖范围，要完成一本有关数据通信方面的著作就犹如生活本身一样需要久经磨难。同时因为作者认为大家对这种史诗般的主题有着迫切需要，所以本书将专注于其中一个小小的子话题，即微机上的异步串行通信，而其他重要的主题——同步通信和网络等等——将不得不另辟它作。

我们所选的主题是数字电子通信的最简单形式。扼要地说，在异步串行通信中，数据先被分解成位，并逐位地作为二进制电子脉冲传送，然后再由接收方重新组装。事实上，在领会这种通信形式所隐含的思想方面几乎不需要任何技术性的知识，因为它类似于日常生活中的许多事情。比方说登山缆车，在等候缆车时，一群登山者（数据位）并行地站立着（字节）；而一旦坐入吊篮里，他们将被一个接一个地（串行地）输送（发送）；最后，在山顶上，他们将走出吊篮（被接收），并且又开始成群结队（字节）。

尽管看似亲切，可在为高级技术性用户所编写的著作中，类似这种直观比喻通常是不必要的，上面这个比喻的提出是为了照顾那些非专业人员以及一些抱有串行通信深奥莫测看法的读者。在一些面向编程人员的出版物中，我们常常也会看到一些原本很有智慧的作者在这种“黑色艺术”面前是那么的手足无措！这个观点的论据俯拾即是，甚至包括在串行通信硬件供应商的名字 Syzygy 和“黑盒目录”（The Black Box Catalog）中，其中 Syzygy 使人不由想起星相学和招魂术，而后者的含义则更是不言而喻！甚至连我在《The RS-232 Solution》中也不得不屈从于这种态度，命名一个题为“地下室里的小精灵”（Elves in the Basement）的章节。

对于这些被人们视为“神秘主义科学”的现象来说，其中有许多都可归咎于 19 世纪中后期本质上从机械主义的领域中演化而来的技术。也就是说，那些按现代标准看似乎不可理喻的设计决策，实际上应作为一个时代的产物。在那个时代里，AC 电流对人们来说还如同神话，DC 马达仍然处于雏形阶段，而通信设备则依靠一些发条和齿轮的机制来驱动。在这方面，奇迹并不在于这种技术运作得有多么良好，而在于它的确可以工作！可是，即使是在一个高容量计算机的时代，这种有百年历史的技术也依然存在，甚至日见繁荣，这只能作为其生命力和可靠性的有力见证。为此，我们的首要任务应是从其发展并得以形成的角度来领会当今的技术。

阅读本书的预备知识

这是一本有关串行通信的中高级专著，它假定读者已掌握了基本的计算机概念和术语。你如果不熟悉“FIFO”、“轮询”、“RAM”和“矢量”之类的术语，手边一定要有本好的计算机词典，而且头脑要灵活，可能还得需要大量的时间。

本书的第一部分不包含源代码，所以它是所有编程人员普遍感兴趣的。其中，第 1 章和第 2 章对高级用户有一定的吸引力，第 3 章主要面向程序设计人员，因为它用大量篇幅介绍“循环冗余校验(CRC)”这一深奥主题。第 4 章进度放慢，每一个理论都伴随一个例子说明。第 5 章介绍 modem 理论，但对两个流行灵巧型调制解调器的详尽剖析却留待第 8、9 两章进行。第 6 章介绍的是理想串行 I/O 硬件，接着在第 7 章考察三种实际的 IC(NS8250，NS16550 和 Z80SIO)与理想化硬件的符合程度。

而本书第二部分则是专门针对编程而写的，其中列举了大量 C 实例。尽管这些程序设计例子的设计和目标适合于所有语言，但如果你不具备 C 的必备背景知识，将很快发觉自己已经陷入了沼泽地中；毫无偏颇地说，C 的初学者也可能会很快遭受挫折。如果你不幸成为这些人当中的一员，请不要放弃希望，你可以“循序渐进”，不断地增进你的熟悉程度。最重要的是，在学习 C 语言时，最佳方法就是读那些为实际应用而编写的源代码。

第二版 序 言

本书第一版发表于 1987 年夏天;此后六年多来,串行通信技术及其应用领域有了高速发展和显著的变化。根据最新发展的技术和开拓新应用领域的实际需要,大幅度地更新、充实,特别是为串行通信的应用开发人员提供编程指南成为出版商和广大读者的共同需要,也是本书第二版的核心思想。

80 年代末至 90 年代初,串行通信技术在多个方面有了较大发展。首先是高速调制解调器的问世,在本书初版时,2400 bps 调制解调器的技术刚刚问世,到 1993 年,本书第二版付印前,14400 bps(V. 32 bis)的调制解调器已经商品化,速率在 32000 bps 以上的,更高速度的调制解调器已经出现。其次,在物理速度大大增加的情况下,modem 通道采用了网络风格的协议(MNP 2-4,V. 42)以实现无差错通信。此外,为了增加吞吐量,modem 中采用了通信中的数据压缩技术(MNP-5,V. 42 bis)。与此同时,计算机化的传真以及普通传真技术也有了惊人的发展,其应用也日益普及。如果说 1987 年时,只要求串行端口的吞吐量为 2400 bps,如今则已要求能运行 57600 bps,甚至更高。因此,慢速的串行端口已很难适应这种需要,也就是说,计算机的硬件需要更新,必须引入新的串行控制器。

作者努力把近年来的发展体现在全书的改写之中,这些发展是:高速 modem、modem 协议、数据压缩、改进了的串行控制器以及快速发展的传真技术。当然,为了更好地应用这些发展的技术,还必须加入通用的 Microsoft Windows。但在本书中并没有涉及这一题目,因为读者不会用本书所学知识来编写 Windows 的串行通信程序,在这里仅向读者提供一个初步的程序设计——一个不直接讨论串行硬件、而针对系统的串行设备驱动程序。Windows 串行驱动程序至今为止被证明是慢而笨的。如果确有需要编写用户的 Windows 串行 I/O 驱动程序,程序员可参考两本书:(1)Windows 设备驱动程序;(2)本书中提供的设备驱动程序模型。

下面是本书内容的一个简述,它由两部分组成。

第一部分共 11 章,介绍串行通信基本内容:ASCII 码、异步通信、错误检测(包括 CRC)、信息传输(XMODEM 和 Kermit 文件传送协议)、调制解调器的原理及其局限性、RS-232、UART(包括 NS16550)、产业标准的调制解调器的详细讨论、数据压缩技术和传真。

第二部分专门讨论和串行通信有关的 C 程序设计,共 13 章。在设计中,作者把与硬件有关的部分都安置在一、两个硬件指定的配置文件中,这使得所设计的函数库具有良好的可移植性和通用性。例如,设置波特率、数据格式及其他参数、控制和监控 RS-232 接口、"打开"和"关闭"串行端口、执行串行输入/输出格式化、执行虚拟流控制等等的可移植函数,执行串行 I/O 精确计时需要的函数,高性能中断 I/O 的完整代码,像协议 modem 样控制常规 modem 的一个完整的函数库。高性能的 XMODEM 文件传送模型,使用快速查表算法的 CRC 函数,高效率传真(G3)数据压缩的完整代码等等都是这方面的极好例子。

本书提供的七个附录,在逻辑上不属于任何一章,但这些信息对读者来说也是十分重要的。

目 录

序 言 …… (1)

第一部分 串行通信基础

第 1 章 ASCII 字符集 …… (3)

1.1 上下文意义 …… (3)
1.1.1 机器的上下文意义:指令集 …… (4)
1.1.2 人类上下文意义:字符集 …… (4)
1.2 ASCII 字符集的位 …… (4)
1.2.1 "ASCII"的歧义性 …… (5)
1.2.2 ASCII 码表 …… (5)
1.3 图形字符 …… (7)
1.3.1 数字字符 …… (8)
1.3.2 拉丁字母表 …… (9)
1.3.3 特殊字符 …… (10)
1.3.4 ASCII 码排序序列 …… (13)
1.4 控制字符 …… (13)
1.4.1 物理设备控制字符 …… (15)
1.4.2 逻辑通信控制字符 …… (17)
1.4.3 物理通信控制字符 …… (18)
1.4.4 信息分隔符 …… (19)
1.4.5 用于代码扩展的控制字符 …… (20)
1.4.6 控制字符的繁难 …… (21)
1.4.7 控制字符的图案表示 …… (22)
1.4.8 ANSI X3.64:控制代码扩展 …… (22)
1.4.9 ANSI X3.64 控制代码格式 …… (26)
1.5 控制序列前导符 …… (26)
1.5.1 用 ANSI X3.64 编程 …… (27)
1.6 换 行 …… (28)
1.7 一个合理的建议 …… (29)
1.7.1 与行结束相关的词汇表 …… (29)

第 2 章 异步通信技术基础 …… (30)

2.1 电子通信的历史 …… (30)
2.1.1 早期的并行系统 …… (30)
2.1.2 串行二进制系统 …… (31)

2.1.3 早期的打印电报 …… (32)
2.1.4 五位代码 …… (33)
2.1.5 机器自动编码和解码 …… (34)
2.1.6 同步化 …… (36)
2.1.7 为什么要用五位代码? …… (40)
2.1.8 ASCII码的传送 …… (41)
2.1.9 串行术语 …… (41)
2.2 通信线路的用法 …… (43)
2.2.1 同步与异步串行通信的比较 …… (43)
2.3 小 结 …… (44)

第3章 错误及错误检测 …… (45)

3.1 错误起源 …… (45)
3.2 错误检测 …… (45)
3.2.1 冗余位 …… (46)
3.2.2 块冗余:奇偶校验 …… (47)
3.3 循环冗余校验(CRC) …… (49)
3.3.1 模-2算术运算 …… (49)
3.3.2 普通写法的模-2除法 …… (50)
3.3.3 模-2除法与硬件 …… (51)
3.3.4 清除余数寄存器 …… (54)
3.3.5 典型的CRC电路 …… (55)
3.3.6 CRC和多项式 …… (55)
3.3.7 选择生成器多项式(除数) …… (56)
3.3.8 获得零余数 …… (57)
3.3.9 对累加器清零的另一种考虑 …… (57)
3.4 CRC的各种变形 …… (59)
3.4.1 典型CRC …… (60)
3.4.2 前导零 …… (60)
3.4.3 一字节数据的CRC …… (60)
3.5 小 结 …… (62)

第4章 信息传输 …… (63)

4.1 流控制 …… (63)
4.2 软件流控制过程 …… (63)
4.2.1 逐个字符的流控制过程 …… (64)
4.2.2 行流控制 …… (66)
4.3 流控制协议 …… (67)
4.3.1 用于硬件设备的流控制协议 …… (67)
4.3.2 字符协议 …… (67)
4.3.3 整行协议 …… (69)
4.3.4 文件传送协议 …… (69)

4.4 自动重复请求(ARQ)协议 …… (70)
4.4.1 发送并等待 ARQ …… (70)
4.4.2 连续 ARQ …… (71)
4.5 信息包 …… (71)
4.5.1 分隔信息包的控制字符 …… (71)
4.5.2 限定信息包的区段长度 …… (72)
4.5.3 数据区段固定长度的信息包 …… (72)
4.6 XMODEM 协议 …… (73)
4.6.1 XMODEM 的技术说明 …… (73)
4.6.2 XMODEM 发送 …… (73)
4.6.3 XMODEM 接收 …… (75)
4.6.4 XMODEM-CRC …… (77)
4.6.5 超时处理 …… (78)
4.6.6 XMODEM 的有关问题 …… (79)
4.6.7 多文件 XMODEM …… (80)
4.7 Kermit 简介 …… (81)
4.7.1 Kermit 协议 …… (83)
4.7.2 在 Kermit 信息包内控制字符的编码 …… (85)
4.7.3 在 Kermit DATA 区段中数据字节高阶位的编码 …… (87)
4.7.4 在 Kermit DATA 区段中重复计数的编码 …… (87)
4.7.5 Kermit 信息包中的区段 …… (88)
4.7.6 Kermit 信息包的类型 …… (89)
4.7.7 一个 Kermit 通信会话范例 …… (91)
4.7.8 终止一个传送过程 …… (91)
4.7.9 Kermit 的扩充 …… (93)
4.7.10 连续 ARQ …… (94)
4.7.11 加长信息包 …… (94)
4.7.12 如何选择 …… (96)
4.8 局部链路协议 …… (96)
4.9 小 结 …… (97)

第 5 章 调制解调器及其控制 …… (99)

5.1 调制解调器 …… (99)
5.1.1 调制解调器基础 …… (99)
5.1.2 调制 …… (100)
5.1.3 通信方式或带宽用法 …… (102)
5.1.4 频率调制 …… (104)
5.1.5 带宽限制 …… (106)
5.1.6 相位调制 …… (107)
5.1.7 微分相移键控 …… (109)
5.1.8 正交调幅 …… (110)
5.1.9 握手:建立数据链路 …… (110)
5.1.10 带宽增加 …… (114)

5.1.11 ECM:回波抵消多工系统 …… (114)
5.1.12 网格编码 …… (115)
5.1.13 V.32 …… (115)
5.1.14 V.32bis …… (117)
5.1.15 V.FAST …… (117)
5.2 调制解调器控制 …… (117)
5.3 RS-232标准 …… (118)
5.3.1 交换电路的法定功能说明 …… (120)
5.3.2 电信号特征 …… (122)
5.3.3 接口电路的机械描述 …… (123)
5.4 实际的RS-232 …… (124)
5.4.1 调制解调器和RS-232接口 …… (125)
5.4.2 微机接口 …… (129)
5.5 灵巧型调制解调器 …… (130)
5.5.1 调制解调器和流控制 …… (131)
5.6 RS-232的非标准用法 …… (134)
5.6.1 空(Null)调制解调器 …… (134)
5.6.2 连接非调制解调器设备 …… (135)
5.7 小 结 …… (136)

第6章 UART:一个概念上的模型 …… (138)

6.1 软件异步I/O …… (138)
6.1.1 软件异步输出 …… (138)
6.1.2 软件异步输入 …… (139)
6.2 UART的介绍 …… (142)
6.2.1 串行数据时钟 …… (142)
6.2.2 UART发送器 …… (143)
6.2.3 UART接收器 …… (145)
6.2.4 错误检测 …… (145)
6.2.5 接收器同步 …… (147)
6.3 数据格式 …… (150)
6.3.1 奇偶校验 …… (150)
6.3.2 数据位个数 …… (151)
6.3.3 停止位个数 …… (152)
6.3.4 “送BREAK”位 …… (152)
6.4 RS-232接口 …… (152)
6.4.1 RS-232输出寄存器 …… (153)
6.4.2 RS-232输入 …… (153)
6.4.3 握手信号 …… (154)
6.4.4 RS-232状态寄存器 …… (154)
6.4.5 RS-232输出控制寄存器 …… (154)
6.4.6 RS-232反相逻辑 …… (155)
6.5 UART中断 …… (155)

6.5.1 生成中断 …… (156)
6.5.2 中断向量的确定 …… (156)
6.6 FIFO 式 UART …… (157)
6.6.1 过速/欠载 …… (157)
6.7 中断等待 …… (158)
6.7.1 中断饱和 …… (158)
6.8 块结束问题 …… (159)
6.9 一个理想 FIFO 的 UART …… (159)
6.10 小结 …… (159)

第 7 章 实际的 UART …… (160)

7.1 National 8250/16450 …… (160)
7.2 8250 硬件基础 …… (161)
7.2.1 8250 时钟和同步 …… (162)
7.3 8250 内部结构 …… (162)
7.3.1 8250 内部寄存器寻址 …… (162)
7.3.2 8250 寄存器用法概要 …… (164)
7.3.3 8250 上的中断 …… (169)
7.3.4 中断服务例行程序 …… (170)
7.4 National 16550 UART …… (171)
7.4.1 16550 中断标识寄存器 …… (172)
7.4.2 线路状态寄存器 …… (173)
7.4.3 FIFO 控制寄存器 …… (173)
7.4.4 FIFO 中断模式操作 …… (173)
7.4.5 接收 FIFO …… (174)
7.4.6 发送中断操作 …… (174)
7.4.7 查询操作中 16550 的使用 …… (174)
7.5 Zilog Z80SIO 串行输入/输出控制器 …… (174)
7.6 Z80SIO 与 8250 比较 …… (175)
7.6.1 寄存器寻址 …… (175)
7.6.2 FIFO …… (175)
7.6.3 DELTA 状态位锁存 …… (176)
7.6.4 向量中断 …… (176)
7.6.5 Z80SIO 附加特征 …… (176)
7.7 Z80SIO 硬件基础 …… (176)
7.7.1 数据寄存器 …… (178)
7.7.2 控制/状态口寻址 …… (178)
7.7.3 Z80SIO 中断 …… (180)
7.7.4 其余寄存器概述 …… (184)
7.8 小结 …… (186)

第 8 章 baseline 灵巧型调制解调器 …… (187)

8.1 灵巧型调制解调器的灵巧之处何在? …… (188)

8.2 Hayes Smart 调制解调器的简史 …… (189)
8.3 调制解调器状态 …… (189)
8.3.1 命令状态 …… (189)
8.3.2 拨号状态 …… (190)
8.3.3 握手状态 …… (190)
8.3.4 在线状态 …… (190)
8.3.5 在线命令状态 …… (190)
8.3.6 哑终端方式 …… (190)
8.4 命令语法和调制解调器响应 …… (190)
8.4.1 命令语法 …… (190)
8.4.2 命令执行时间 …… (191)
8.4.3 按键退出 …… (192)
8.5 软挂起:在线命令状态 …… (192)
8.5.1 在线转义的危险 …… (193)
8.6 对命令的响应 …… (193)
8.7 硬件问题 …… (194)
8.7.1 自动波特率调整 …… (194)
8.7.2 速率和数据格式 …… (195)
8.7.3 非易失性存储器 …… (195)
8.7.4 调制解调器的 RS-232 接口 …… (196)
8.7.5 前面板 …… (197)
8.7.6 后面板 …… (197)
8.7.7 配置开关 …… (197)

第 9 章 智能调制解调器命令 …… (200)

9.1 存根命令 …… (200)
9.2 调制解调器命令 …… (200)
9.3 方式命令 …… (201)
9.3.1 用户接口命令组 …… (201)
9.3.2 基本拨号和应答命令组 …… (204)
9.3.3 拨号修改标志组 …… (205)
9.3.4 拨号变量命令组 …… (208)
9.3.5 其他命令组 …… (210)
9.3.6 Profile 管理命令组 …… (213)
9.3.7 电话硬件控制命令组 …… (213)
9.3.8 RS-232 命令组 …… (215)
9.3.9 连接性选择命令组 …… (216)
9.4 数字变量命令 …… (217)
9.4.1 S 寄存器字符变量 …… (218)
9.4.2 计数/定时用 S 寄存器变量 …… (219)

第 10 章 协议调制解调器 …… (226)

10.1 调制解调器协议和协议调制解调器 …… (226)

10.1.1 调制解调器协议 …… (226)
10.1.2 背景 …… (226)
10.1.3 调制解调器协议的简短回顾 …… (227)
10.1.4 网络模型 …… (228)
10.1.5 OSI 七层网络模型 …… (228)
10.2 究竟什么是一个链路协议? …… (230)
10.2.1 同步还是异步? …… (230)
10.2.2 体系结构 …… (230)
10.2.3 数据透明性 …… (231)
10.2.4 数据等待时间 …… (231)
10.2.5 协议效率 …… (233)
10.2.6 BREAK 处理 …… (233)
10.3 MNP 协议 …… (234)
10.3.1 标题 …… (234)
10.3.2 信息 …… (234)
10.3.3 帧校验序列 …… (235)
10.3.4 BREAK 处理 …… (235)
10.4 MNP 异步协议:1 类和 2 类 MNP …… (235)
10.4.1 信息包引导符 …… (235)
10.4.2 标题字段 …… (235)
10.4.3 信息字段 …… (235)
10.4.4 信息包终结符序列 …… (235)
10.4.5 帧校验序列 …… (235)
10.4.6 协议效率 …… (236)
10.5 MNP 同步协议:3 类和 4 类 MNP …… (236)
10.5.1 信息包引导符/终结符序列 …… (236)
10.5.2 标题字段 …… (237)
10.5.3 帧校验序列 …… (237)
10.5.4 协议效率 …… (237)
10.6 LAPM(V.42) …… (237)
10.6.1 信息包引导符/终结符序列 …… (238)
10.6.2 地址字段 …… (238)
10.6.3 控制字段 …… (238)
10.6.4 信息字段 …… (238)
10.6.5 BREAK 处理 …… (238)
10.7 压缩理论 …… (239)
10.7.1 重复 …… (240)
10.7.2 统计重复:哈夫曼编码 …… (241)
10.7.3 历史重复:Lempel-Ziv …… (242)
10.7.4 5 类 MNP 压缩规程 …… (24[illegible])
10.7.5 V.42bis BTLZ 压缩规程 …… (246)
10.7.6 V.42bis 和 5 类 MNP 之间的比较 …… (247)
10.8 流控制 …… (247)

10.8.1 流控制与数据压缩 …… (248)
10.9 协议调制解调器及其命令 …… (250)
10.10 &Q 协议启动 …… (250)
10.11 S46:协议及压缩规程选择 …… (251)
10.12 S48:特性协商 …… (251)
10.13 S36:协商退却(Fallback) …… (252)
10.14 &K:流控制 …… (252)
10.15 线路速度控制 …… (253)
10.16 Wn 和 S95:扩展响应 …… (253)

第 11 章 传真机 …… (256)

11.1 T.30:传真通信协议 …… (258)
11.2 HDLC 信息包 …… (259)
11.3 传真字段 …… (261)
11.4 成串信息包 …… (262)
11.5 同步线路控制 …… (262)
11.5.1 轮询 …… (262)
11.5.2 X 位(X-Bit) …… (263)
11.6 传真的五个阶段 …… (263)
11.6.1 阶段 A:呼叫建立 …… (263)
11.6.2 阶段 B:识别和协商 …… (264)
11.6.3 阶段 C:数据("报文")传输 …… (265)
11.6.4 阶段 D:页后过程 …… (265)
11.6.5 阶段 E:呼叫释放 …… (266)
11.7 传真过程的实例描述 …… (266)
11.7.1 实例描述一:单页传真 …… (266)
11.7.2 实例描述二:规格相同的多页传真 …… (268)
11.7.3 实例描述三:不同规格的多页传真 …… (268)
11.7.4 实例描述四:轮询方式单页传真 …… (271)
11.7.5 实例描述五:在 G3 训练期间降低速度 …… (273)
11.7.6 实例描述六:在 G3 训练期间提高速度 …… (274)
11.8 DIS/DCS 位映象 …… (275)
11.8.1 向后兼容性和可扩展性 …… (275)
11.8.2 新的 FCF …… (275)
11.8.3 最小性能集合 …… (276)
11.8.4 DIS/DCS 信息包的逐位解释 …… (279)
11.9 T.4:传真图象协议 …… (282)
11.10 分辨率 …… (283)
11.10.1 1992 年的 T.4 建议 …… (284)
11.11 文件尺寸 …… (285)
11.11.1 页面尺寸 …… (285)
11.12 传真编码 …… (289)
11.12.1 一维编码(改进型哈夫曼编码) …… (289)

11.12.2 二维编码(READ 编码) …… (294)
11.12.3 编码方式综述 …… (297)
11.12.3 线终码 …… (297)
11.12.4 页编码 …… (299)
11.13 差 错 …… (300)
11.13.1 纠错 …… (301)
11.14 EIA 传真调制解调器 …… (302)
11.15 EIA 578(Class1) …… (304)
11.15.1 Clas-1 服务 …… (305)
11.15.2 命令综述 …… (306)
11.16 会话实例 …… (309)

第二部分 用 C 语言编写异步通信程序

第 12 章 设计一个基本的串行 I/O 库 …… (314)

12.1 Microsoft C 编译器 …… (314)
12.1.1 SIOLOCAL.H 文件 …… (314)
12.2 串行 I/O 库 …… (317)
12.2.1 函数库的层次结构——分级 …… (319
12.2.2 用于屏蔽状态寄存器的常量 …… (321
12.2.3 U16x50.LIB …… (322)
12.2.4 2 级库:BUOS.LIB …… (325)
12.2.5 3 级库:SIO.LIB …… (326)
12.3 终端模拟程序的第一个版本 TERM 0 …… (327)
12.3.1 控制台 I/O …… (328)
12.4 TERM0 …… (329)
12.5 连接 TERM0 …… (331)
12.5.1 0 级模块:IBMPC.C …… (331)
12.6 小结 …… (332)

第 13 章 程序的可移植性 …… (333)

13.1 1 级函数 …… (333)
13.1.1 结构中的函数指针 …… (334)
13.1.2 含有指向 UART 读/写函数的指针的结构 sio …… (335)
13.1.3 修改后的 UART.LIB 函数 …… (336)
13.1.4 内存映象系统中的指针 …… (337)
13.2 SIO 数据类型 …… (337)
13.2.1 管理多个 SIO …… (338)
13.3 修改 1 级函数 …… (339)
13.3.1 寄存器存储类型 register …… (341)
13.4 2 级函数 …… (341)

13.4.1 进一步讨论函数指针 …… (341)
13.4.2 BUOS.LIB中修改后的2级函数 …… (342)
13.4.3 声明和初始化SIO …… (343)
13.5 为内存映象UART提供的SIO结构 …… (346)
13.5.1 定时函数 …… (346)
13.6 定时函数的类型 …… (346)
13.6.1 延时 …… (346)
13.6.2 超时功能 …… (347)
13.7 系统定时器和时间保持器 …… (347)
13.7.1 系统节拍器 …… (347)
13.7.2 系统节拍器的软件接口 …… (348)
13.8 设计一个虚拟的定时系统 …… (348)
13.8.1 IBM PC机的定时程序 …… (349)
13.8.2 0级定时函数delay …… (350)
13.8.3 "等待字符"函数 …… (351)
13.9 为定时参数定义常量 …… (352)
13.10 UART清除器 …… (353)
13.10.1 测试定时函数的程序 …… (354)
13.11 小 结 …… (355)

第14章 波特率和数据格式函数 …… (356)

14.1 设计目标 …… (356)
14.1.1 用户准则 …… (357)
14.1.2 一般假设 …… (357)
14.1.3 虚拟寄存器 …… (358)
14.2 IBM PC机的数据格式 …… (361)
14.2.1 16X50的虚拟寄存器数组 …… (362)
14.2.2 位操作的通用结构(vregbits_) …… (363)
14.2.3 IBM PC机上的数据格式 …… (364)
14.2.4 通过SIO指针访问虚拟寄存器 …… (365)
14.2.5 为停止位和数据长度声明的结构 …… (366)
14.2.6 多寄存器操作 …… (367)
14.2.7 最终版的vsetbits函数 …… (369)
14.2.8 16X50的1级函数(_vsetbits) …… (371)
14.2.9 3级数据格式函数 …… (371)
14.2.10 结构定义的位置 …… (373)
14.3 波特率函数 …… (373)
14.3.1 vbaud_数据结构 …… (374)
14.3.2 IBM PC的波特率数据结构 …… (375)
14.3.3 设置波特率的2级函数 …… (376)
14.3.4 用于16X50的1级波特率函数(_vsetbr) …… (377)
14.3.5 设置波特率的3级函数 …… (378)
14.4 配置和恢复 …… 379)