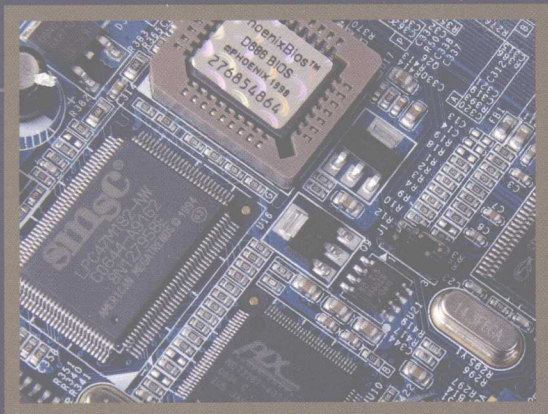




高等学校计算机科学与技术教材

计算机 高级接口技术

□ 周洪利 编著



- 原理与技术的完美结合
- 教学与科研的最新成果
- 语言精炼，实例丰富
- 可操作性强，实用性突出

本书相关资料网站：

www.zhouhongli.com

清华大学出版社

● 北京交通大学出版社

高等学校计算机科学与技术教材

计算机高级接口技术

周洪利 编著



清华大学出版社

北京交通大学出版社

·北京·

内 容 简 介

本书介绍现代 PC 机的接口技术, 内容包括串行接口、并行接口、USB、红外线接口和蓝牙接口。针对以上接口, 全面介绍了接口的原理、接口标准、接口硬件芯片和接口的程序设计方法。

本书在内容组织方面注重面向实际应用, 尤其是接口软件的编程方法。结合不同接口的特点, 由浅入深地介绍了不同的接口程序实现方法。包括通过 ActiveX 控件方式、直接 I/O 方式、API 方式等。书中所有的程序设计方法都面向 Windows 2000/XP 操作系统。采用的编程工具是 Visual C++ 6.0 和 Visual Basic。

本书可以作为大学计算机专业高年级本科生或研究生的教材。本课程的授课和实验学时数应不少于 32 学时, 但在具体的教学实践中可以根据需要选讲其中的不同部分。本书也可作为接口硬件和软件开发人员的参考用书。

配合本书的学习和实验, 作者专门设计了高级接口实验台, 该实验台具备书中介绍的全部接口, 可以完成本书介绍的全部实验内容。

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13501256678 13801310933

图书在版编目(CIP)数据

计算机高级接口技术/周洪利编著. — 北京: 清华大学出版社; 北京交通大学出版社, 2008. 11

(高等学校计算机科学与技术教材)

ISBN 978-7-81123-420-6

I. 计… II. 周… III. 微型计算机-接口-高等学校-教材 IV. TP364.7

中国版本图书馆 CIP 数据核字 (2008) 第 157989 号

责任编辑: 贾慧娟 特邀编辑: 陈 海

出版发行: 清华大学出版社 邮编: 100084 电话: 010-62776969

北京交通大学出版社 邮编: 100044 电话: 010-51686414

印刷者: 北京东光印刷厂

经 销: 全国新华书店

开 本: 185×260 印张: 25 字数: 622 千字

版 次: 2008 年 11 月第 1 版 2008 年 11 月第 1 次印刷

书 号: ISBN 978-7-81123-420-6/TP·444

印 数: 1~4000 册 定价: 37.00 元

本书如有质量问题, 请向北京交通大学出版社质监组反映。对您的意见和批评, 我们表示欢迎和感谢。

投诉电话: 010-51686043, 51686008; 传真: 010-62225406; E-mail: press@bjtu.edu.cn。

序

计算机接口技术是计算机应用的关键技术,随着计算机应用的日益广泛和深入,计算机接口技术不仅是大学计算机科学与技术专业的核心专业课程,而且早已成为自动控制、电子工程、信息工程、通信工程、光电工程、机电工程等许多非计算机专业的必修专业基础课程。

随着计算机硬件软件技术的高速发展和总线标准的不断推陈出新,计算机接口技术也有了很大发展。相应课程教学如何与时俱进,跟上接口技术的发展,使其兼具先进性和实用性,是各校有关教师共同关心的问题。北京交通大学周洪利老师《计算机高级接口技术》一书的出版对解决这一问题作了一个很好的尝试。

该书不同于一般接口教科书的是,在讲清接口基本原理、保持必要的传统接口技术内容如串行接口和并行接口的基础上,着重介绍了近年来在现代计算机系统中常用的几种新型接口,包括 USB 接口、红外接口和蓝牙接口,而且对每一种接口的介绍都包含了接口原理、常用接口硬件芯片、接口协议和接口软件编程技术等内容,体现了原理与技术并重、软件与硬件结合的接口技术特点。

像 USB、红外和蓝牙这些“高级接口”,在硬件构成原理、接口协议及其实现等方面都较一般接口复杂,加之在现代计算机中接口软件的设计多由基于 DOS 操作系统的设计变为基于 Windows 操作系统的设计,这就决定了,没有对数字电路硬件知识和基于 Windows 的系统编程技术的熟练掌握,就很难从底层开始自行设计开发这类接口。在这类接口的实际设计开发中,一般接口硬件多是采用专业公司提供的现成的通用或专用大规模集成电路接口芯片,并不需要由用户从底层开始进行设计,因此,技术人员在接口软件设计方面花的时间和精力往往远多于接口硬件电路的设计。接口软件设计时,由于高级接口的协议复杂性,不同的接口采用的设计方法也不尽相同。理论上讲,采用驱动程序的编程方法对任何接口都是适用的,但这种设计方法比较复杂,需要深入了解接口协议和操作系统方面的知识,开发周期长,技术要求高,因此在实际的工程开发中常利用一些现有的软件接口模块,包括控件、DLL 和特定类等,在这些软件接口模块的基础上进行接口软件开发将使开发工作变得简单快捷。该书充分考虑到上述特点,在编写时一改传统的计算机接口技术教材把重点放在接口硬件设计方面的惯例,而着重于接口软件的设计、接口协议的编程实现。这也是该书区别于一般接口教科书的第二个特点。

该书第三个重要特点,是较好地贯彻了理论与实际紧密结合的原则。讲各种接口时,都引入了相应的实际案例或实验案例,而且在不同的接口软件设计中采用了不同的编程方法和编

程语言。涉及的编程方法有 API 方法、Activate X 控件方法、DLL 调用方法、Socket 方法、基于特定类的方法和采用驱动程序的编程方法等；涉及的编程语言有 VB、VC++ 等。读者通过学习，可以获得多种实用接口编程方法和技巧，增强接口设计开发能力。

总之,《计算机高级接口技术》一书顺应了计算机接口技术的发展,介绍的内容面向实际开发需求,有很好的先进性和实用性,是一本难得的好书。我相信,它的出版对于促进计算机教学与实际应用需求的结合,将发挥很好的作用。该书可以作为各工科专业特别是计算机专业和其他以电为主专业的高年级本科生/研究生课程的教材或教学参考书,也适于从事实际计算机应用系统开发工作的工程技术人员参考。

第一届国家级教学名师 邹逢兴

目 录

第1章 接口概述	1
1.1 接口的硬件构成	1
1.1.1 数据寄存器和数据缓冲存储器	1
1.1.2 状态寄存器和控制寄存器	2
1.1.3 地址译码单元	3
1.1.4 信号的波形变换	3
1.1.5 差错控制	5
1.1.6 中断和 DMA 机制	7
1.2 接口的分类和性能	7
1.2.1 接口的分类	7
1.2.2 接口的性能	8
1.3 接口软件的控制方式	9
1.3.1 无条件传送方式	9
1.3.2 查询控制方式	9
1.3.3 定时查询控制方式	10
1.3.4 中断控制方式	11
1.3.5 DMA 控制方式	11
第2章 串行接口原理	13
2.1 串行接口标准	13
2.2 RS-232C	14
2.2.1 RS-232C 的基本特征	14
2.2.2 RS-232C 信号	15
2.2.3 RS-232C 连接	17
2.3 RS-422/485	18
2.3.1 RS-422/485 的平衡传输	18
2.3.2 RS-422 连接	18
2.3.3 RS-485 连接	19
2.3.4 RS-422/485 的传输距离	20
2.4 PC 机中的 UART	21
2.4.1 UART 的功能	21
2.4.2 UART 寄存器	23

第3章 串行接口程序设计	30
3.1 MSComm 控件	30
3.1.1 MSComm 控件的属性	30
3.1.2 VB 实现的 MSComm 测试程序	37
3.2 串行接口 GPS 接收机编程实验	42
3.2.1 NMEA-0183 数据格式	43
3.2.2 一个 GPS 接收程序的实现	45
3.3 Win32 API 实现串行接口编程	53
3.3.1 用于串行接口编程的 API 函数	53
3.3.2 Windows API 串行接口编程方法	66
第4章 并行接口	72
4.1 并行接口概述	72
4.1.1 初始的并行接口	72
4.1.2 并行接口的标准	74
4.2 并行接口结构	78
4.2.1 SPP	78
4.2.2 双向(PS/2)模式	80
4.2.3 EPP	82
4.2.4 ECP	84
第5章 并行接口程序设计	90
5.1 实现直接 I/O 方式的软件	90
5.1.1 Inout32	91
5.1.2 TVicPort	91
5.1.3 TVicLPT	96
5.2 一个 PC 对等连接的并行接口测试实验	100
5.2.1 对等连接	101
5.2.2 软件设计	102
第6章 USB 接口标准	111
6.1 USB 接口特点	111
6.2 USB 电器规范	113
6.2.1 USB 接口	113
6.2.2 USB NRZI 编码	117
6.3 USB 传输	117
6.3.1 USB 传输类型	118
6.3.2 USB 传输的要件	119
6.3.3 事务	121
6.3.4 USB 传输	125
6.4 USB 描述符	130
6.4.1 USB 描述符概述	130

181	6.4.2	设备描述符	132
281	6.4.3	配置描述符	134
281	6.4.4	接口描述符	135
181	6.4.5	端点描述符	136
181	6.4.6	字符串描述符	137
181	6.5	USB 标准请求	138
281	6.5.1	控制传输的设置阶段	138
281	6.5.2	标准请求	140
81	第 7 章	USB 接口芯片	144
281	7.1	USB 接口芯片分类	144
181	7.2	微处理器接口的 USB 专用接口芯片	144
181	7.2.1	Philips 的 PDIUSBD12	145
281	7.2.2	NS 的 USBN9603/9604	146
281	7.2.3	MAXIM 的 MAX3420E	148
281	7.3	带有 USB 接口的微处理器	150
281	7.3.1	Cypress 的 CY7C63000A 系列	150
281	7.3.2	Cypress 的 CY7C63612 和 CY7C63613	151
281	7.3.3	Cypress 的 EZ-USB 系列	151
281	7.3.4	Atmel 的 AT89C5131	153
281	7.3.5	Microchip 的 PIC 16C745/765	154
281	7.3.6	Signal 的 C8051F320/321	156
281	7.4	USB 桥芯片	157
281	7.4.1	FTDI 的 USB-UART 桥 FT232R	157
281	7.4.2	FTDI 的 USB-并行桥 FT245BM	159
281	7.4.3	SigmaTel 的 USB-IrDA 桥 STIr4200	161
81	第 8 章	USB 接口 HID 设备	163
281	8.1	HID 设备简介	163
281	8.1.1	HID 设备的特点	163
281	8.1.2	HID 设备的硬件要求	164
281	8.1.3	HID 固件的要求	164
281	8.2	HID 设备描述符	165
281	8.2.1	HID 设备的描述符	165
281	8.2.2	HID 描述符	166
281	8.3	USB 接口的键盘描述符范例	179
281	8.3.1	设备的描述符	179
281	8.3.2	配置描述符	179
281	8.3.3	接口描述符	180
281	8.3.4	HID 描述符	180
281	8.3.5	端点描述符	181

8.3.6	字符串描述符	181
8.3.7	报表描述符	182
8.4	HID 的特定请求	183
8.4.1	Get_Report 请求	184
8.4.2	Set_Report 请求	184
8.4.3	Set_Idle 请求	184
8.4.4	Get_Idle 请求	185
8.4.5	Get_Protocol 请求	185
8.4.6	Set_Protocol 请求	185
8.5	HID 程序设计	185
8.5.1	HID 访问使用的 API 函数	187
8.5.2	查找 HID 的过程	187
8.5.3	获得 HID 的能力	193
8.6	HID 实验	203
8.6.1	实验电路	203
8.6.2	固件程序设计	205
8.6.3	获得 HID 设备描述符	252
8.6.4	设备的初始化测试	254
8.6.5	HID 测试程序的实现	256
第 9 章	红外线接口	270
9.1	IrDA 版本标准	270
9.1.1	IrDA Data 标准	270
9.1.2	AIR 标准	271
9.1.3	IrDA 标准比较	271
9.2	IrDA 物理层接口标准	271
9.2.1	IrDA 编码方式	272
9.2.2	IrDA 调制方式	273
9.3	IrDA 协议	275
9.3.1	IrDA 协议结构	275
9.3.2	IrDA 低层协议	275
9.3.3	IrDA 高层协议	276
9.4	红外接口器件	276
9.4.1	MCP2120 红外编解码器	277
9.4.2	HSDL-1001 红外收发器	279
9.4.3	MCP2150/2155 红外协议控制器	280
第 10 章	红外接口程序设计	290
10.1	IrDA 程序概述	290
10.1.1	Winsock 与红外协议	290
10.1.2	红外设备的寻址	292

00E	10.2 IrSock 的使用	292
40E	10.2.1 WSAStartup() 和 WSACleanup() 函数	293
80E	10.2.2 socket() 和 closesocket() 函数	294
17E	10.2.3 SOCKADDR_IRDA 数据结构	294
57E	10.2.4 bind() 函数	295
87E	10.2.5 listen() 函数	295
07E	10.2.6 accept() 函数	296
07E	10.2.7 send() 和 recv() 函数	296
77E	10.2.8 采用 IRLMP_ENUMDEVICES 参数的 getsockopt() 函数	297
88E	10.2.9 采用 IRLMP_IAS_QUERY 参数的 getsockopt() 函数	299
	10.3 一个简单的 IrSock 编程实例	302
	10.3.1 外部设备的硬件构成	302
	10.3.2 通过 Winsock 实现的 IrDA 应用程序的方法	303
	10.4 通过 IrDA 实现与手机连接的编程实验	310
	10.4.1 AT 指令集概述	310
	10.4.2 手机控制主要 AT 指令	311
	10.4.3 实验程序的功能	317
	第 11 章 蓝牙接口原理	319
	11.1 蓝牙技术	319
	11.1.1 蓝牙技术概述	319
	11.1.2 蓝牙与其他无线射频标准的比较	321
	11.1.3 蓝牙标准的发展	322
	11.2 蓝牙协议	324
	11.2.1 蓝牙协议结构	324
	11.2.2 蓝牙协议应用举例	326
	11.2.3 无线信道和基带	327
	11.2.4 链路管理器协议	335
	11.3 蓝牙接口硬件	345
	11.3.1 CSR 的 BlueCore 系列蓝牙芯片	345
	11.3.2 Broadcom 的蓝牙芯片	348
	11.3.3 蓝牙模块	349
	第 12 章 基于 HCI 的蓝牙程序设计	352
	12.1 蓝牙 HCI 概况	352
	12.1.1 蓝牙 HCI 的构成	352
	12.1.2 HCI 分组及其格式	353
	12.1.3 HCI 传输	355
	12.1.4 HCI 流控机制	357
	12.2 蓝牙 HCI 命令和事件	358
	12.2.1 用于蓝牙设备查询的 HCI 命令	358

305	12.2.2	用于 ACL 连接的 HCI 命令	360
306	12.2.3	用于同步连接的 HCI 命令	364
307	12.2.4	HCI 链路策略命令	368
308	12.2.5	常用的控制器与基带命令	371
309	12.2.6	控制器与基带参数及其 HCI 命令	372
310	12.2.7	设备信息参数	373
311	12.3	蓝牙 HCI 编程实验	376
312	12.3.1	实验环境	376
313	12.3.2	SCR 的一个基于 USB 传输的 HCI 类	377
314	参考文献		388

305	10.3	一个简单蓝牙设备	10.3
306	10.3.1	设备描述符	10.3.1
307	10.3.2	通过 Windows 实现 HCI 设备	10.3.2
308	10.4	通过 HCI 设备实现蓝牙设备	10.4
309	10.4.1	AT 命令	10.4.1
310	10.4.2	蓝牙设备 AT 命令	10.4.2
311	10.4.3	蓝牙设备 AT 命令	10.4.3
312	11	蓝牙设备 AT 命令	11
313	11.1	蓝牙设备 AT 命令	11.1
314	11.1.1	蓝牙设备 AT 命令	11.1.1
315	11.1.2	蓝牙设备 AT 命令	11.1.2
316	11.1.3	蓝牙设备 AT 命令	11.1.3
317	11.2	蓝牙设备 AT 命令	11.2
318	11.2.1	蓝牙设备 AT 命令	11.2.1
319	11.2.2	蓝牙设备 AT 命令	11.2.2
320	11.2.3	蓝牙设备 AT 命令	11.2.3
321	11.2.4	蓝牙设备 AT 命令	11.2.4
322	11.3	蓝牙设备 AT 命令	11.3
323	11.3.1	蓝牙设备 AT 命令	11.3.1
324	11.3.2	蓝牙设备 AT 命令	11.3.2
325	11.3.3	蓝牙设备 AT 命令	11.3.3
326	12	基于 HCI 的蓝牙设备	12
327	12.1	蓝牙 HCI 设备	12.1
328	12.1.1	蓝牙 HCI 设备	12.1.1
329	12.1.2	蓝牙 HCI 设备	12.1.2
330	12.1.3	蓝牙 HCI 设备	12.1.3
331	12.1.4	蓝牙 HCI 设备	12.1.4
332	12.2	蓝牙 HCI 设备	12.2
333	12.2.1	用于蓝牙设备的 HCI 命令	12.2.1

第1章 接口概述

1.1 接口的硬件构成

计算机系统所连接的外部设备是多种多样的,它们不但在工作速度上与中央处理机差别极大,而且在数据表示的形式上与计算机内部形式也不一致。因此,要实现外部设备与主机之间的连接和信息交换,必须经过一个数据转换和传输的机制。这种机制叫作 I/O 接口。

图 1.1 说明了主机、外部设备和 I/O 接口之间的连接关系,主机和任何外部设备的连接都是通过接口电路实现的,接口电路连接在计算机总线和外部设备之间,是接口的硬件部分,接口的另一组成部分是软件部分,接口电路的工作是在接口软件的控制下完成的。

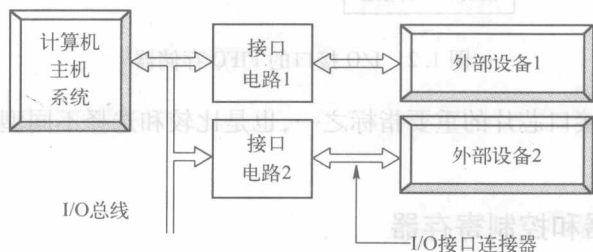


图 1.1 主机、外部设备和 I/O 接口间的关系

接口电路硬件部分主要包括数据寄存器、数据缓冲存储器、状态寄存器和控制寄存器等,后面各节将进行详细介绍。

1.1.1 数据寄存器和数据缓冲存储器

数据寄存器,用于暂存输入输出数据。由于 CPU 和接口的传送是异步的,需要输出的数据必须先送到数据输出寄存器,然后在接口电路的同步控制之下传送到接口输出端;输入数据时,在接口电路的同步控制下将数据传送到数据输入寄存器,CPU 通过中断或查询机制得到数据输入寄存器的状态,再执行输入指令将数据传送到主机。

如果接口电路中只有一个或几个数据寄存器,在高速的接口传送中会频繁地进行主机和接口电路之间的数据交换,大量占用 CPU 的时间,尤其在 Windows 这样的非实时多任务操作系统下可能严重影响接口的传输速度,甚至造成数据的丢失。

解决上述问题的办法是在接口电路中设立存储器,这就是接口的缓冲存储器。接口的缓冲存储器一般是一定容量的 FIFO(First In First Out)存储队列,数据输出时,主机先将数据传送到发送 FIFO 存储器,从发送 FIFO 存储器的传送输出是在接口电路的控制下完成的。数据输入时,输入的数据先暂存在输入 FIFO 存储器,当达到一定的数量时,再通知主机传送到主

存储器。

FIFO 存储器的容量可以根据接口的传输速度和接口软件的具体要求确定,现在接口的传输速度日益提高,FIFO 的容量也随之增大,但增大 FIFO 容量会增加接口电路的成本,一般是几十到几百个字节。

FIFO 存储器构成了先入先出存储队列,如图 1.2 所示。在接收和发送电路中都可以有 FIFO,FIFO 队列有一个 FIFO 计数器,以记录当前 FIFO 的存储状态,在连续地发送和接收过程中 FIFO 机制特别有用,一般在发送时可以设定当发送 FIFO 中的数据接近全部发送出去时触发中断,通知主机软件,主机软件再次填满发送 FIFO;同样在接收时,当接收的数据接近填满接收 FIFO 时触发中断,通知主机软件将接收 FIFO 中的数据传送到软件的数据缓冲区。

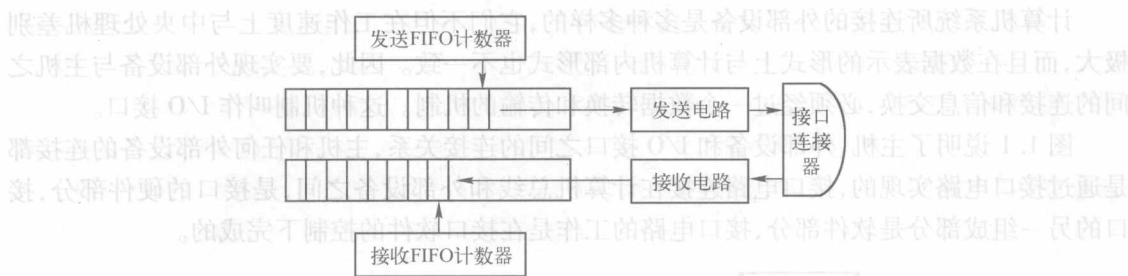


图 1.2 L/O 接口的 FIFO 存储器

FIFO 容量是衡量接口芯片的重要指标之一,也是比较和选择不同型号的接口芯片的依据之一。

1.1.2 状态寄存器和控制寄存器

为了记录接口电路的状态,在接口电路中一般有多个状态寄存器,这些寄存器存放的信息包括接收/发送数据的情况、接口所连接的外部设备的状态、中断状态及接口的错误信息等。

接口的状态信息必须满足接口的控制要求,接口软件可以通过读接口状态寄存器掌握接口的状态信息,一般接口软件需要了解的接口状态信息包括:

- 是否需要传送数据到接口发送寄存器或发送缓冲存储器;
- 是否需要从接口的数据接收寄存器或接收缓冲存储器读取数据;
- 通过接口所连接的设备的当前状态;
- 如果产生中断,是什么类型的中断;
- 接口电路是否产生数据传输错误。

接口的一部分状态信息可能和中断有关,接口软件可以根据具体需要打开一些事件的中断请求,当这些事件发生,接口电路会产生中断信号到 CPU,当 CPU 接收到中断信号会启动中断处理程序,中断处理程序首先要读取相关的状态寄存器,以确定是什么事件引起的中断。

控制寄存器的作用是完成对接口的操作,软件可以通过向特定的控制寄存器写入特定的编码信息完成对接口的各种控制操作。这些操作包括:

- 通过接口控制接口连接的外部设备;
- 设置接口的参数,如速率、传送方式等;
- 设置接口的差错控制方式;

- 设置接口的中断,如是否打开中断、哪些事件可以产生中断及中断的优先级;
- 如果接口具有 DMA 机制,可以设置 DMA 方式和相关参数。

关于接口的状态寄存器和控制寄存器的描述是接口电路文档的主要内容,如果要编写接口控制软件,首先必须了解接口的状态寄存器和控制寄存器的定义。

1.1.3 地址译码单元

由于接口电路中有数据、状态和控制 3 类多个寄存器,这些寄存器在计算机系统中都有唯一的地址,CPU 通过执行对特定地址的端口读写命令来完成对寄存器的读写操作。在计算机系统中所有端口的地址都是唯一的。

当 CPU 执行读写命令时,在 I/O 总线上送出地址和读写控制命令,地址译码单元识别地址和读写控制信号。为了使译码电路简单,同一个接口电路使用的多个寄存器的地址一般是连续的,是端口地址空间中的一段,这一段地址中的最小的地址叫基地址。

如在 I/O 地址空间为 16 位的计算机系统中,串行接口 COM1 的基地址是 03F8h,占用的 I/O 地址空间是 03F8h ~ 03FFh 共 8 个端口地址,由于 8 个地址是连续的,而且只和最低 3 位地址(A2 ~ A0)有关,译码电路中一般只对 A15 ~ A3 地址段译码,A2 ~ A0 的译码器在接口芯片中。其译码电路部分如图 1.3 所示。

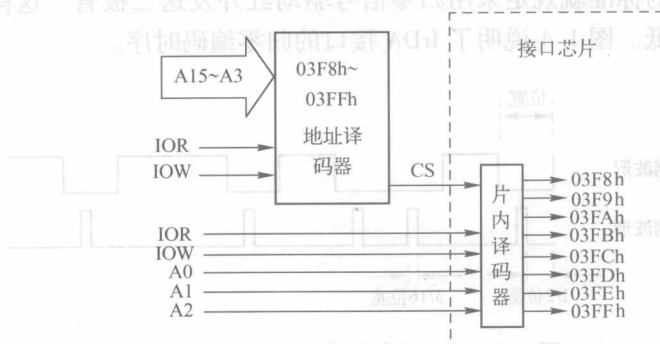


图 1.3 串行接口的译码器

接口电路的地址译码器必须把 I/O 读写信号也作为译码的条件,以区别于存储器地址空间。

这样的设计具有灵活性,可以将接口的 I/O 地址空间映射到系统的 I/O 地址空间的任何位置。需要指出的是,现在的带有即插即用功能的 I/O 总线中,芯片外部的译码器是可以通过系统的软件根据当前 I/O 地址空间的占用状态进行再配置,可以将其地址段映射到系统地址空间的任何位置。

1.1.4 信号的波形变换

接口信号的编译码是为了满足接口信号而设立的电路机制,不同的接口对信号波形的要求不同,尤其是串行方式的接口(如串行接口、USB 接口、红外接口等),需将数据在发送端变换成要求的波形,在接收端再变换回来。波形的编译码是在接口电路硬件中完成的。

接口信号的特征主要有单极性与双极性信号、归零与不归零信号和平衡与非平衡信号 3 类。

1. 单极性与双极性信号

单极性信号是指数字信号的 0 和 1 两种状态中有一个状态是 0 电平,而另一个状态是一个固定的电平。如果逻辑 0 对应 0 电平,逻辑 1 对应一个固定的正电平,就是单极性正逻辑信号。

双极性信号是指数字信号的 0 和 1 两种状态分别对应两个固定的非 0 电平,多数双极信号的两个电平是一正一负而且绝对值相等。双极性信号的两个电平差可能比单极性信号更大,所以可以提高信号在传输过程中的抗噪声干扰。

最常见串行接口 RS-232C 就是双极性信号方式,它的逻辑 0 和逻辑 1 信号电平分别是 $+3 \sim +12 \text{ V}$ 和 $-3 \sim -12 \text{ V}$,在不同的系统中常常采用 $+3 \text{ V}$ 、 $+5 \text{ V}$ 、 $+12 \text{ V}$ 和 -3 V 、 -5 V 、 -12 V 3 种电平。

2. 归零与不归零信号

不归零(Non Return to Zero)信号是指在一个位传输时间内,信号的电平是恒定不变的,而归零信号的有效电平时间只占用位传输时间的一部分。

多数接口采用不归零信号,如串行接口 RS-232/422/485、并行接口、USB 等。

接口采用归零信号传输的目的主要有两个,一个是为了改变信号的频率特性,使接口可以从信号中获得同步时钟;另一个目的是为了降低信号的发送功率,有利于降低功耗。

如 IrDA 红外线接口的标准就规定采用归零信号驱动红外发送二极管。这样发光二极管的发光时间缩短,功耗降低。图 1.4 说明了 IrDA 接口的归零编码时序。

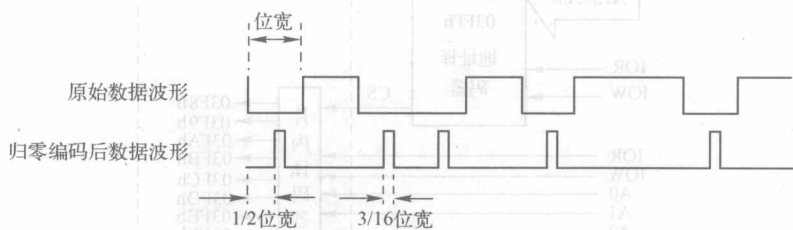


图 1.4 IrDA 的归零信号波形

3. 平衡与非平衡信号

非平衡信号是指所有的接口信号电平都是以地线作为参考点的,如果接口的每一个信号都具有自己的参考点,而且这个参考点是信号电平的反向信号,这就是平衡信号。非平衡和平衡信号的波形如图 1.5 所示。



图 1.5 非平衡和平衡信号波形

RS-485、RS-422、USB 等接口采用的都是平衡信号,在平衡传输中,每一个信号需要两条接口线传输。

和非平衡传输相比,平衡传输具有很好的抗干扰特性,适合在长距离传输中使用,如采用平衡传输的 RS-422、RS-485 的传输距离比采用非平衡传输的 RS-232 要远得多。

在短距离传输中采用平衡传输可以大大提高传输速度,新型的高速接口大多采用平衡传输,如 USB、IEEE-1394 接口的传输速度远远高于采用非平衡传输的传统接口。

现代接口的重要特征之一是平衡方式串行点对点传输,这种传输方式也被一些现代总线采用,目的是采用较少的传输线实现高速传输。采用平衡方式串行点对点传输方式的接口和总线包括 USB、IEEE-1394、SATA 和 PCI Express 等。

1.1.5 差错控制

由于噪声干扰的存在,在传输过程中不可避免地要产生传输错误,即误码。接口必须有差错控制功能,接口的差错控制可以在接口硬件电路实现,也可以在接口软件中实现。一般情况下,在硬件中实现全部的差错控制功能是不经济的,但接口电路中往往有一些简单的、辅助性的硬件差错控制功能。

实现差错控制首先要在接收端发现误码,再纠错。纠错的方法大致有两种,一是请求发送端重发,二是在接收端直接纠正。在接收端纠正需要在接收数据中有纠错信息编码。

差错控制的性能评估是误码率改进,即经过纠错控制的误码率与原始数据的误码率的比值。理论上讲任何差错控制方式的误码率改进都是有限的,也就是说经过任何差错控制的传输系统的误码率都不可能为零,但应小到对一个特定应用系统可以忽略的程度。

不同的数字传输系统对误码率的要求是不同的,即使在同一个接口中,不同类型的信息传输也可以采用不同的差错控制方案。典型的例子是 USB 接口,采用 USB 接口传输文件时采用批量传输模式,这种传输模式有性能很好的差错控制,如果用 USB 传输数字音频或视频数据,就完全没有差错控制,因为数字音频和视频信号对误码不敏感。

最简单的差错控制方式应该是奇偶校验,是以字符为校验单位,每一个字符增加一位校验数据位,可以是奇校验(加校验位后字符中 1 的个数是奇数)或者是偶校验(加校验位后字符中 1 的个数是偶数),经过校验的数据字符发送到接收方,在接收方通过硬件进行校验运算。奇偶校验的典型应用是 7 位的 ASCII 码传输,加一位校验位组成 8 位的字节。

普通的奇偶校验方式只用于以字符为单位的校验,以数据包为校验单位的奇偶校验被称为垂直奇偶校验,垂直奇偶校验将多个字符的相同序号位作奇偶校验,生成一个(单字节垂直奇偶校验)或多个(多字节垂直奇偶校验)校验字符。如图 1.6 所示。

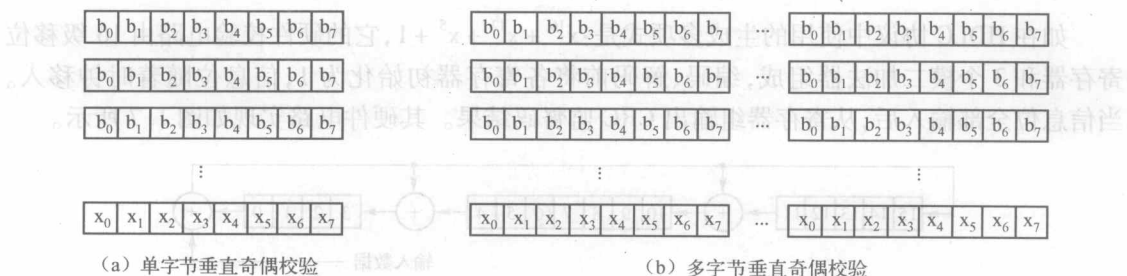


图 1.6 垂直奇偶校验字节的生成

CRC(Cyclical Redundance Checking) 校验是一种高效的校验方式,中文名称为循环冗余校验。编码和解码方法的硬件实现简单,检错和纠错能力强,在通信领域广泛地用于实现差错控制。CRC 校验可以在接口硬件中实现,也可以在软件中实现,如果参照网络层次协议的概念,CRC 校验一般在链路层实现。

利用 CRC 进行差错控制的过程可简单描述为:在发送端根据要传送的 k 位二进制数据序列,通过 CRC 校验多项式运算产生用于校验的 r 位监督码(CRC 码),附在原始信息后,构成一个新的二进制码序列数共 $k+r$ 位,然后发送出去。在接收端,根据与发送端相同的 CRC 校验多项式运算得到监督码 r' ,比较 r 与 r' ,以确定传送中是否出错。

在代数编码理论中,可以将一个二进制码组表示为一个多项式,码组中各码元当作多项式的系数。例如二进制序列 11100101 可表示为 $x^7+x^6+x^5+x^2+1$ 。

设编码前的原始信息为 k 位,信息多项式为 $P(x)$, $P(x)$ 的最高幂次加 1 等于 k 。CRC 校验多项式为 $G(x)$, $G(x)$ 的最高幂次等于 r 。经过 CRC 校验多项式运算得到的 r 位二进制监督码多项式为 $R(x)$ 。

r 位 CRC 监督码产生的规则是:先将要发送的二进制序列数左移 r 位(即多项式乘以 x^r)后,左移后的信息多项式为 $x^rP(x)$,再除以 CRC 校验多项式 $G(x)$,所得到的余数即为 CRC 监督码,其多项式为 $R(x)$ 。

经过研究和实用验证,一些比较成熟的 CRC 校验多项式已经应用在不同的标准中,包括各种长度的 CRC 校验多项式,表 1.1 所示为一些常用的 CRC 校验多项式。

表 1.1 常用的 CRC 校验多项式

名 称	生成多项式	应用举例
CRC-4	x^4+x+1	ITU G.704
CRC-5	x^5+x^2+1	USB Token
CRC-8	x^8+x^2+x+1	
CRC-12	$x^{12}+x^{11}+x^3+x+1$	
CRC-16	$x^{16}+x^{12}+x^2+1$	IBM SDLC
CRC-ITU **	$x^{16}+x^{12}+x^5+1$	ISO HDLC, ITU X.25, V.34/V.41/V.42
CRC-16	$x^{16}+x^{15}+x^2+1$	USB Data
CRC-32	$x^{32}+x^{26}+x^{23}+\cdots+x^2+x+1$	ZIP, RAR, IEEE 802 LAN/FDDI, IEEE 1394
CRC-32c	$x^{32}+x^{28}+x^{27}+\cdots+x^8+x^6+1$	SCTP

如在 HDLC 协议中使用的生成多项式是: $x^{16}+x^{12}+x^5+1$,它的硬件校验电路由 16 级移位寄存器和 3 个模二加法器组成,编码、解码前将各寄存器初始化为 1,信息位随着时钟移入。当信息位全部输入后,从寄存器组输出 CRC 监督码结果。其硬件电路实现如图 1.7 所示。

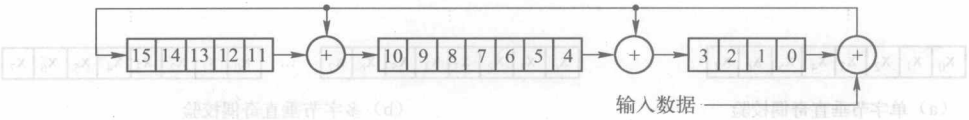


图 1.7 CRC 校验算法的硬件实现