单片机应用技术选编(9)

何立民　主编

北京航空航天大学出版社

内 容 简 介

《单片机应用技术选编》(9)选用了2000年、2001年国内30多种电子技术类、计算机应用类期刊中有关单片机与嵌入式系统文章共443篇,其中120篇全文刊登,其余文章均有摘要,所有文章都注明出处,以便读者查找。全书共分9章,计有专题论述、综合应用技术、操作系统与软件技术、网络、通信与数据传送、新器件及其应用技术、总线及其应用技术、可靠性及安全性技术、典型应用实例及文章摘要。本书收集了当今单片机与嵌入式系统最新的应用资料,对读者提高单片机及嵌入式应用水平有重要参考价值。

图书在版编目(CIP)数据

单片机应用技术选编(9) / 何立民主编 .—北京:北京航空航天大学出版社,2004 .1

ISBN 7-81077-313-5

Ⅰ.单… Ⅱ.何… Ⅲ.单片微型计算机—文集

Ⅳ.TP368 .1-53

中国版本图书馆CIP数据核字(2003)第035817号

单片机应用技术选编(9)

主　　编　何立民

责任编辑　曾昭奇

北京航空航天大学出版社出版发行

北京市海淀区学院路37号,邮编100083　发行部电话(010)82317024

http://www.buaapress.com.cn　E-mail:bhpress@263.net

河北省涿州市新华印刷厂印装　各地书店经销

*

开本:787×1092　1/16　印张:47.25　字数:1210千字

2004年1月第1版　2004年1月第1次印刷　印数:5 000册

ISBN 7-81077-313-5　定价:69.00元

序　　言

由于忙于《单片机与嵌入式系统应用》期刊创建,又由于国内单片机处于转型时期,《单片机应用技术选编》(9)(后文简称《选编》(9))拖至今天才与读者见面。由于文章的滞后,《选编》(9)中大部分文章都带有转型前的印记。其中有一些,如嵌入式操作系统,则代表了未来的一些方向。从目前(2002～2003 年)单片机与嵌入式系统类期刊上,则可以较清晰地看出单片机与嵌入式系统当前与未来的趋势,希望读者关注。

1 . 关于后 PC 时代

早在 1999 年 1 月,美国著名未来学家尼葛洛庞蒂访华时,曾预言:“ 4～5 年后,嵌入式智能(电脑)工具将是 PC 和因特网之后最伟大的发明。”他给后 PC 时代的争论提出了自己鲜明的观点,明确地指出后 PC 时代就是嵌入式系统时代。如今,嵌入式系统带来的工业年产值已超过 1 万亿美元。后 PC 时代并不意味着 PC 机技术的尽头,而是意味着 PC 机技术发展模式的成熟,形成了大产业的专业化发展状态。在后 PC 时代,大规模群众性、学院式研究则从 PC 技术转向嵌入式系统技术,特别是计算机专业人士的介入必将提升嵌入式系统应用水平。与通用计算机系统相比,嵌入式系统的普遍性及无所不在,几乎将所有从事计算机应用、电子技术工程、涉电工程技术人员卷入其中;因此,在今后相当长的时间里,嵌入式系统会成为持续的热点技术。

2 . 单片机应用的转型

作为专门为嵌入式应用推出的单片机,开创了嵌入式系统应用的先河。单片机从早期的单片微型计算机(single chip microcomputer)到微控制器(microcontroller unit)到 SoC,技术上有了很大的发展,满足了最大多数的嵌入式系统的底层应用,也为嵌入式应用积累了丰富的经验,并培养了大批人才。但长期以来,由于从事单片机应用开发人员以嵌入对象领域人员为主,主要精力致力于以对象环境特征为主的传统电子系统应用研究方法,缺少计算机的工程方法指导。因此近年来,随着微处理器技术的发展,计算机界对嵌入式系统的大力介入,单片机及其相关技术有了很大的发展。所以,我国单片机界应注意在新形势下的技术转型问题。

对于单片机本身,将从基于裸片、指令系统的随意性设计转为基于高级语言、操作系统、集成开发环境、软硬件平台,在计算机工程方法指导下新型的嵌入式系统设计。随着 32 位嵌入式微处理器价格的下降,会出现一些以通用 32 位嵌入式微处理器为内核的单片机。厂家推出新型单片机时,也将一改过去的裸片方式而为用户提供完善的软硬件平台及成套的解决方案。

对于嵌入式应用的解决方案,单片机将从唯一方案转向多种方案选择,如基于 VHDL 的 CPLD 的系统解决方案、以 DSP 为核心的系统方案以及基于 IP 核的 SoC 设计。当前,由于 PLD 技术已进一步向 SoPC 方向发展,许多 PLD 厂家在原先的 CPLD/ FPGA 中提供了许多通用 IP 核,为用户构建嵌入式应用系统带来极大方便,对于大批量的产品需求,也提供了转向 ASIC 设计的方便界面,因此,这种方案有上升趋势。

3．嵌入式系统概念的统一

长期以来,我国嵌入式应用一直以单片机为主。其从业人员主要是非计算机专业人员,单片机主要用于非计算机专业的对象领域,没有源自计算机专业的通用计算机系统/ 嵌入式计算机系统的参照概念。当计算机专业队伍大量转入嵌入式计算机系统应用时,“ 嵌入式系统 ”一词便开始流行,而作为以嵌入式应用为唯一目的的单片机这一典型的嵌入式系统业界,反而对“ 嵌入式系统 ”一词感到陌生;而计算机专业人士对单片机也不以为然。实则是体现出客观历史与现实上单片机与嵌入式计算机系统应用的差异性。这些差异性表现为:单片机应用面对最底层的经典电子系统领域,从业人员主要是对象领域的专业人员,受传统电子系统设计方法的影响较大,以 8 位机为主流机型;嵌入式计算机系统从业人员以计算机界为主,离对象领域较远(目前从事网络、通信、多媒体、人工智能较多),习惯采用基于软、硬件平台的计算机工程方法与集成开发环境,并以 32 位机为主流机型。无论是单片机,还是嵌入式计算机系统,都是基于嵌入式应用的专用计算机系统,最终都要向 SoC 发展,都应统一成‘ 嵌入式系统 ”。考虑到历史、现实与未来的发展,传统的单片机领域与嵌入式计算机系统领域的上述差异还会存在。在统一成嵌入式系统后,可以将传统的单片机领域归为嵌入式系统的低端,而将原先嵌入式计算机系统作为嵌入式系统的高端。

4．嵌入式系统热点技术的转移

由于网络、通信、数字视频音像技术的发展,嵌入式系统已从过去以单片机为主发展到多种形式的系统解决模式,以下一些技术热点值得注意。

(1) 32 位嵌入式系统的发展

在网络、通信、数字视频音像系统中要求大量的数据处理、控制,要求嵌入式操作系统支持,8 位处理器无法满足要求,因而寻求 32 位处理器解决。32 位嵌入式系统的发展不会遵循 8 位单片机百花齐放的模式,而会走通用嵌入式处理内核的模式,基于通用内核来构成各种专用、准专用 32 位单片机。32 位 ARM 嵌入式处理器有可能成为 32 位嵌入式系统事实上的标准内核。32 位通用内核的大量使用所形成的价格优势,可能会对嵌入式系统低端应用产生冲击。

(2) 基于 DSP 的嵌入式系统

近年来 DSP 有了很大发展,除了完善数字信号处理功能外,还增加了控制管理能力。随着网络通信、数字视频音像技术的发展,许多依靠单片机解决的嵌入式应用系统转向寻求 DSP 器件解决,形成了当前 DSP 热点的持续发展。DSP 技术的发展带动了人工智能、识别技术、编/ 解码技术的发展。

(3) 基于 VHDL 的 CPLD 技术

近年来 PLD 得到了飞速发展,几个著名的 PLD 厂家都发展了各种大规模的、各具特色的 CPLD,如 PSoC、SoPC 以及各种可配置的 PLD 器件,并提供各种方便、价廉的 EDA 软件工具,提供了用 CPLD 来解决嵌入式应用系统的一种快捷途径,形成了当前又一个技术热点。基于 CPLD 的嵌入式应用系统设计,通常都采用各种可配置的 CPLD,这样来构成系统方便、易行。基于 CPLD 的嵌入式应用系统设计可使用厂家提供的各种 IP 硬核或软核,可大大加快研发周期,适合小批量应用。另外,基于 VHDL 的 CPLD 系统设计易实现微电子的 ASIC 设

计,对于大批量的用户,这是一条新产品研发的合理路径。

(4) 基于 SoC 的 ASIC 系统设计

以往,将大批量的单片机应用系统转成专用集成器件时,都是依靠半导体厂家的 ASIC 专家进行。随着 IC 设计技术的普及、多项目晶圆服务(MPW)平台的支持、IC 设计软件工具的完善与普及、半导体厂家可提供更多的 IP 软核、硬核服务,大批量的单片机应用系统研发人员会转向基于 SoC 的 ASIC 系统设计。通常采用的途径是直接的 ASIC 设计,然后通过多项目晶圆流片,或通过 CPLD 设计验证,转而进入 ASIC 设计。无论哪种方法,都要求有强大的 IP 库的支持,因此,今后 IP 库的建立与发展是嵌入式系统应用设计中的一项基础性工作。传统的单片机应用系统设计方法会逐渐减少。

(5) 基于集成化开发环境的产品研发

长期以来,单片机产品研发都是基于裸片、初级开发装置、汇编语言的随意性方式。这是由于客观的历史因素形成的。当时,对于传统的电子技术人员,这种方式容易上手,但研制周期长,受个人因素影响大,不易移植与交流,品质不易保证。为了改变这种情况,目前芯片厂家在推出新的系列时,会给用户提供完善的集成开发环境。用户可在厂家提供的软、硬件平台上进行产品开发,大大缩短了产品开发周期。随着单片机逐渐向 SoC 化发展,用户产品系统的软硬件设计也趋向于在厂家提供的试验模块的平台基础上进行,不再是从零开始,既缩短了产品开发周期,又有平台软硬件的品质保证。

上述的这些嵌入式系统转型与技术热点的转移都可在科技期刊中反映出来,本《选编》(9)从 2000 年、2001 年各类期刊中共收集了 433 篇文章,全文部分 120 篇,摘要部分 313 篇。与以往相比,突出了操作系统与软件技术、网络、通信与数据传输以及总线技术。

《选编》(9)在收集文章时得到了胡敏编辑的帮助,马海珍承担了与文章作者的全部联系工作,邬宽明老师则完成了全部文章摘要工作,提高了文章摘要的内容深度。在此深表感谢。对于全文发表的文章作者,我们都要发函,与作者联系,并在本书出版后付给相应的稿酬。对于个别未能取得联系的作者,请见到本书后迅速与我们联系。

联系人:马海珍

通信地址:(100083)北京航空航天大学出版社

联系电话:(010)82317022

主编　何立民

目　　录

第一章　专题论述

第二章　综合应用技术

第三章 操作系统与软件技术

第四章 网络、通信与数据传送

第五章　新器件及其应用技术

第六章　总线及其应用技术

第七章　可靠性及安全性技术

第一章

专题论述

1 .1 集成电路进入片上系统时代

沈阳东北微电子研究所(110032) 王效平 刘捷臣

一、系统级集成电路的发展概况

随着集成电路的深亚微米制造技术、设计技术的迅速发展,集成电路已进入片上系统时代。所谓片上系统,又叫系统级芯片,也就是系统级集成电路,其英文简写为 SOC(System On a Chip)或者 SLI(System Level IC)。系统级集成电路(SOC)是指在单一硅芯片上实现信号采集、转换、存储、处理和 I/O 等功能,或者说在单一硅芯片上集成了数字电路、模拟电路、信号采集和转换电路、存储器、MPU、MCU、DSP、MPEG 等,实现一个系统的功能。SOC 是在 ASIC 的基础上发展起来的电路,与 ASIC 完全不同,具有很多独特的优点。

(1) SOC 增加了功能:从单一功能增加到多功能,实现一个系统的功能,实现了高速、高集成度和低功耗。

(2) SOC 大大降低整机的成本:由过去用多块 IC 构成系统,变成现在的一块 SOC。

(3) SOC 大大降低整机的体积:这是系统制造商进一步发展的方向,尤其对便携式的电脑、通信及多媒体产品的生产厂家更具有吸引力。

(4) SOC 促进了整机系统更新换代的速度:它缩短了供需双方的差距,整机更受用户的欢迎,易于占领市场。

由于 SOC 的这些优点,正好满足了通信、电脑、消费产品向轻、薄、短和耗电少的方向发展,因此市场对 SOC 产品有强烈的需求。目前 SOC 是微电子产业界最热门的话题。如果说 VLSI 促进 PC 广泛的应用而带来了信息产业的第一次革命,那么 SOC 的发展正在带来信息产业的第二次革命。美国、日本和欧洲等国各大半导体公司纷纷加大 SOC 生产线的投资,建立 0 .15～0 .20 μm 加工线,目标是生产 SOC。例如 NEC 公司为生产 SOC 建立 2 条 IC 芯片加工线,一条为低功耗 SOC 工艺线(UC3),最细线宽为 0 .18 μm,功耗达到 19 nW/ MHz/ 门;另一条高性能 SOC 工艺线,最细线宽为 0 .15 μm,速度达到 500 MHz。这二条工艺线加工 SOC 的集成度可达 3400 万门/ 20 mm×20 mm,BGA 封装引脚达 3000 pins, NEC 计划于 1999 年 2 季度投产。SGS-Thomson 公司正在建一条开发生产 SOC 芯片的加工线, 采用 0 .12 μm、300 mm 技术。

世界各大半导体公司纷纷宣布了他们开发的 SOC 产品,有的已进入市场,有的即将进入市场。美国 NS 公司于 1999 年 1 月 15 日宣布首枚 DVD SOC 芯片进入市场,全部核心功能集成在一个 SOC 芯片上,包括 32 位 RISC、DSP、MPEG 等。该公司 1999 年内推出单片 SOC 电脑系统(PC-on-a-chip),整台电脑功能将被置于这个芯片之内。美国 IBM 公司把逻辑电路与存储器集成在 SOC 上的技术,可提高个人电脑、移动电话、游戏机的能力,提高处理速度和存储容量,第一个嵌入 DRAM 设计到逻辑电路的 SOC SA - 27E 已完成。日本 EPSON 公司在打印机和 LCD 显示器的 SOC 方面也有卓越表现,已把多种 IC 模块如 32 位 RISC、SH - 3、

ARM-7、DRAM、USR/ IEEE1394 接口集成在 SOC 上;同时还把移动手机、传呼机、便携式 PC 和 PDA 等整机作为发展 SOC 的重点。ST 微电子公司已经解决了在目前模拟电视向全数字电视(DTV)过渡中的数字机顶盒的 SOC 的解决方案。LSI logic 公司和 VLSI 公司利用 0.18 μm CMOS 工艺为 GSM 移动通信系统电话手机开发了 SOC 芯片,除 RF 部分以外的全部功能都包含在内(包括基带线路、ADC、DAC、RISC、2 个 DSP 及逻辑控制线路等),于 1999 年上半年上市。预计近期可以将 1.9 GHz 以下频率的 RF 模拟线路也集成到主流硅工艺的 SOC 芯片上;三年内可能解决双频移动电话、无线局域网以及无线数据通信所需要的、5 GHz 以下的 RF 模拟线路集成在 SOC 芯片上;五年以内可把 10 GHz 的 RF 模拟线路集成在 SOC 芯片上。

随着 SOC 成为市场的热点,迫使可编程器件 CPLD 和 FPGA 转向 SOC。Xilinx 公司将其 FPGA 产品转向 SOC。该公司首推出百万门 Virter 系列 FPGA,为解决系统级设计问题提供了新的 FPGA 平台。Quick logic 公司宣布将利用该公司标准产品 FPGA 和硬核模块向客户提供可编程 SOC 产品。Altera 公司宣布新型可编程逻辑器件 APEX 20K 系列已于 1999 年第一季度正式面世。它是第一个成功地将乘积项、查询表功能及内嵌式集成的体系结构。APEX 20K 也是第一个器件密度达到百万门、系统性能可支持 64 位/ 68 MHz PCI 标准的产品系列,将整个复杂的系统集成一片可编程芯片内,实现 SOC。其第一个产品 EP20K 400 已供应市场。

二、系统级集成电路设计技术

1.系统级集成电路设计方法

SOC 芯片集成了一个复杂的系统,芯片的集成度大、速度快,还要求解决各种干扰问题。完成系统级设计,毫无疑问是一项十分艰难的任务。按照国际上对集成电路设计人员的设计能力评价,在 RTL 层次,当今职业设计人员的设计能力平均每天完成 200 门 IC 设计工作量,而片上系统的集成度大致在 1 000 万个等效门。如果从头设计起,完成上述 SOC 芯片设计需要相当多的人力和时日,还要花费大量 NRE(非重复性工程费)。现在电子产品的生命周期正在不断缩短,创新的产品、新结构的产品一般不超过一年,而要求完成芯片设计时间就更短些,因此必须找新的出路。让我们回忆一下平常做整机系统设计或电路板设计的情况:去买一些现成的元器件和 IC 芯片,然后把它们组合在一起,调试、查错,最终完成产品。现在我们把这种方式应用到 SOC 芯片设计上,把已经验证的 IC 电路以模块的形式去参加 SOC 芯片的设计。这样设计就变得容易了,也可大大缩短设计时间,从而解决了 SOC 芯片上市时间长和设计成本高的问题。因此人们在进行 SOC 芯片设计时,必须采用重复使用已验证的 IC 模块。这些已验证的、可重复利用的 IC 模块具有知识产权问题,通常称 IP(Intelligent Properity),也有人称它为系统宏单元(System - Level Macro),或虚拟部件(Virtual Component)、芯核(Core)。

SOC 芯片设计有三种不同方法。第一种方法称为专用系统设计方法。系统商定义出系统指标、SOC 芯片中应采用的 IP 模块,如通用 IP、专用的 IP 全由芯片厂商设计(可能是半导体厂商,也可能由专门设计芯片的公司)完成。这种设计方法的特点是片上系统成本可能最低,但需要较长的设计时间,灵活性较小,适于 SOC 芯片产量特别大的情况。第二种方法称为部分集成法。系统厂商设计 SOC 芯片中的专用电路部分,SOC 芯片中的 DSP IP、MPU IP 以

及存储器 IP 等由 IP 公司提供,由半导体厂商或专门设计芯片的公司完成整体设计。这种方法有一定的灵活性,新产品开发时间也较短。第三种方法称为桌面集成法。各种 IP 模块的供给公司提供给电子系统商 IP 模块,由电子系统商设计其专用电路部分,并与外购的各种 IP 相结合,完成 SOC 芯片设计。这种设计方法成本最低,设计灵活性最大。

2 . 系统级集成电路设计中的 IP 问题

IP 模块在 SOC 芯片中的再利用是一个非常复杂的问题,SOC 芯片设计业面临很多问题。IP 模块知识产权保护是在 SOC 芯片设计中再利用的关键,连美国的知识产权专家都认为:目前没有现成的法律可以保护这种知识产权,估计也很难批准通过这样的法律,因为这样法律执行起来也很难。除法律手段外,依靠技术手段来保护知识产权的工作正在进行当中。当前尚存在以下 2 个问题:① 从 IP 模块的提供者来看,如何设计商用 IP,如何进行恰当的描述,使得既能方便使用者的再利用又不暴露知识产权的秘密,如何对 IP 模块进行维护使它适应技术的发展; ② 从 IP 模块的使用方面来看,通过什么渠道可以找到所需要的 IP 模块,如何对它进行评估、验证,如何能够买到,如何正确使用及许多标准化的问题。

为了解决上述 IP 模块在 SOC 芯片设计中存在的问题,1996 年 9 月世界 35 个著名公司组成一个国际性企业联合组织——虚拟插座接口联盟(The Virtual Socket Interface Alliance, 简称 VSIA)。参加该组织的公司包括半导体制造公司、设计公司、EDA 公司、IP 提供者公司及整机系统公司。目前已发展到 170 多家厂商。VSIA 的目标是开发 IP 模块的接口,制定一系列的开放标准、IP 功能评价和验证方案。此外还有其他一些有关 IP 组织,如可再利用的专用 IP 开发者协会 RAIPD(Reusable Application - specific Intellectual Properity Developers), 该协会也在进行这方面的工作。为了更好地交易 IP 模块,东芝、Motorola、西门子等 9 家有关半导体公司宣布:他们和英国苏格兰发展局联合在英国成立了虚拟部件交易所(VCX), 会员公司可获得 VCX 提供在交易所登记的 IP 模块信息,协助买卖双方订立合同,并征收权利使用费,监督盗用,仲裁争议等。

IP 模块的再利用,除缩短 SOC 芯片设计时间外,还能降低设计和制造的成本,提高可靠性,因而将会给 IC 产业和电子工业带来巨大的商业利益,引起 IC 产业结构变革。现在出现一些与 IP 模块有关的业种,如 IP 模块提供者、IP 流通者、利用 IP 的设计公司、围绕 IP 的诉讼的调停者、IP 价值评价机构等。SOC 芯片设计不再限于单个设计部门,不再限于一个公司内部,而是整个 IC 产业的各种资源实现优化配置,促使 IC 产业更快地发展。现在 IP 模块提供者公司发展最快,已有 1 000 余家,其中有实力的近 100 家,表 1 .1 - 1 列出一些较大的 IP 模块供应商。

表 1 .1 - 1 世界主要 IP 供应商

公司名称	IP 代表的产品
Actel	FPGA IP 核
Advancel Logic	X
Altera	FPGA IP 核
Arcus Technology	通信电路核
Argonaut Risc Cores	RISC 处理器核
ARM	MPU 核
ARM Semiconductor(USA)	网络电路核

续表 1.1-1

公司名称	IP 代表的产品
Artisan	SRAM 核
ASPEC Technology	X
Atmel	FPGA IP 核
CAST	DSP 核
Comit Systems	FPGA IP 核库
CoreEL Microsystems	ATM 等核
Corelogic	通信电路核
CRISC	MPEG2 编码器核
Denali Software	存储器、编码器核
Digital Design & Development(ddd)	X
Digital Objects Corp	标准总线接口电路核
Dolphin Integration	X
Eureka Technology	功率 PC、PCI 核
Evolution(Richard Watts Associates)	MCU 等核
エクセレント・デザイン	MPU、MPEG、解码器等核
FASTMAN .Inc	X
HCL America	IEEE 1394 核
IMODL	检测电路核
Innovative Semiconductors	多媒体、通信电路等核
Integrated Silicon Systems	DSP 核
iREADY	网络电路核
KTech Telecommunications .Inc	X
Logic Innovations	PC 总线等核
Logic Vision	BIST 等测试电路核
メカチツフス	声音或图像电路核
Memec Design Services	82XX 电路、通信电路等核
Mentor Graphics(Inventra)	多种核
Macro Designs	IEEE 1394 核
National Semiconductor(Mediamatics)	MPEG 等核
NEuW Intellectual Property	通信、图像电路有关的核
Nextwave	MPEG 编码器核
NMI Electronics	存储器、控制器等核
Nova Engineering	通信电路有关核
Packet Engines	网络电路核
Palmchip	磁测试电路核
Phoemx Technologies(Virtual Chips)	PCI、PCMCIA、USB 等核
QuickLogic	FPGA IP 核

续表 1 .1 - 1

公司名称	IP 代表的产品
Rambus	DRAM 周边电路核
リアルヒション	X
Rice Electronics	DSP 核
Sand Microelectronics	PCI、USB 等总线系列核
三精システム	时序电路、DMA、计算器等核
Sapien Design	USB 等核
SICAN Microelectronics	ATM、QPSK 等通信系列核
Sierra Research and Technology	高速网络电路核
Silicon Engineering	X
Synova	MPEG 2 解码器等核
Technology Rendezvous	IEEE 1394 核
The Western Design Center	8 位、16 位 MCU 核
ザィン・スィクロシステム	LVDS、IEEE1394
研究所	存储器等核
VAutomation Inc .	MCU 等核
VIRAGE Logic	存储器、测试功能等核
Xilinx	FPGA IP 核
Zoran(Compcore)	X

通常 IP 模块分三类:硬 IP、软 IP 和固 IP,也称硬核(Hard Core)、软核(Soft Core)和固核(Firm Core)。硬核的电路布局和工艺是固定的,不能更改。硬核已完成了全部的前端和后端设计,制造工艺也已确定。它的特点是灵活性最小,知识产权的保护比较简单。IP 模块提供者给用户的是封装好的行为模型,用户只能从外部测试硬核的性能,却无法得到厂商真正的电路设计。

软核是包括逻辑描述(RTL 和门级 Verilog HDL 或 VHDL 代码)、网表和不能物理实现的用于测试的文档(test bench tile)。与硬核相比较,软核有最大的灵活性。用户能把 RTL 和门级 HDL 表达的软核修改为自己所需要的设计,综合到选定的厂商工艺上,并通过布局布线实现具体电路。

固核是一种介于软核和硬核之间的 IP,通常以 RTL 代码和对应具体工艺的网单混合形式提供。固核既不是独立的,也不是固定的,它可根据用户要求进行修改,使它适用于某种可实现的工艺过程。固核允许用户重新定义关键的性能参数。从完成 IP 模块设计所花费的代价来看,硬核代价最高;从 IP 模块的使用灵活性来讲,软核的可重复使用性最高;从期望 IP 模块的价值最高的角度出发,人们期望 IP 完成物理设计,但这会使 IP 模块的可重复性降低。固核可根据系统设计需求修改,但知识产权不易保护,虽然用户乐于接受,但缺少固核的提供者。

3 . 深亚微米设计技术

系统级集成电路实现的必要条件之一是其线宽达到深亚微米水平。当代 SOC 芯片多数为 0 .25 μm～0 .18 μm 设计规则,这与传统 IC 设计技术完全不同,因此给 SOC 芯片设计带来

新的困难。集成电路设计进入深亚微米阶段后,特征尺寸缩小,其横向和纵向尺寸都大大缩小,芯处内的互连线长度也急骤增大。互连线和线间的电阻和电容对信号传输的影响非常显著。这一变化引入了许多新问题,给 SOC 芯片 EDA 设计提出了许多的挑战。

器件的特征尺寸减小到深亚微米,器件的物理特性和电学特性就会发生很大的变化。原来的模型已不适合,必须考虑深亚微米尺寸引起的物理效应。单元本身的固有延迟相对大大减小,互连线引起的延迟在整个单元延迟中的比例越来越大;深亚微米连线变细,连线间距变小,连线变长,这就使连线分布电容增加。由于信号频率很高,所以会引入串扰的影响和噪声影响;由于互连线变细,易于引起电迁徙和热载流子效应。因此在集成电路的设计策略上需做较大的调整,从原来面向电路单元——先安排电路模块,后考虑互连引线,改为面向电路互连线——先安排电路互连线网,再挂电路模块。这样可以从总体设计上保证芯片的高速工作。特别是深亚微米芯片的速度较快,对时序要求更严格,因此前端的逻辑设计与后端的物理设计间很难保持一致。对于逻辑设计中仿真分析后的功能和时序都正确的网表,在布线设计后由于芯片空间和连线的限制,互连引线的延迟与逻辑设计使用的模型不一致,因而时序变得不再满足约束的要求了。这时就得返过头来回到逻辑设计进行修改,然后再进行仿真分析。由于逻辑设计和布局布线之间的控制因素不统一,会导致逻辑设计和物理设计的设计循环不收敛,因而会导致设计周期大大加长。下表列出由于特征尺寸缩小,电路的复杂性和时序问题引起的逻辑设计和物理设计之间重复修正的数据。

特征尺寸	0 .8 μm	0 5 μm	0 .35 μm
平均重复次数	1 次	5 次	m 10 次

为了在设计的初期,获得有关互连线的信息,目前常用的一种设计方法是在设计流程中加入布局规划,通过布局规划对电路进行预布局并得出电路互连延迟估计。然后这些估计被用来指导后续的设计过程。这种方法的最大困难在于,在布局规划中对电路之间最大互连延迟估计的准确度很难保证。另外,由于在设计初期即对电路的物理位置进行了约束,势必影响电路的优化程度。目前用高层次设计方法试图在设计初期获得一些深亚微米集成电路连线的物理信息,并以此来指导后续的设计过程,但目前仍不能十分有效地避免设计过程的迭代。高层次设计方法现在仍不能适应深亚微米高性能系统设计的需要,应从设计流程、电路结构、算法等多方面综合考虑,力图把几方面的方法有效地结合起来探索全面的解决方案。

三、系统级集成电路测试技术

SOC 芯片的测试技术难度较大,因为它涉及几种技术集成一块硅片上。过去测试设备主要是针对专用优化的自动测试仪,一台 VLSI 测试仪可能会有高功效的数字测试能力,但对检测混合信号 IC 未必能行;另一方面混合信号测试仪可能在模拟功能检测方面十分优秀,但对高性能的数字 IC 却无能为力。SOC 芯片测试设备则必须能够精确地检测模拟和数字两种电路,并支持扫描检测和嵌套式存储器检测。检测工程师还必须面对缺少检测触点的现状。总之,SOC 芯片集成度和功能已太大,对输入引脚加测试向量,再从输出引脚观察结果的传统的检测方法已不适用。因传统方法测试向量集就会过分庞大,执行时间也会长得惊人。

目前开发 SOC 芯片测试仪的供应商只有几家,如 Schlumberger、Advantest、Teradyne 和

LTX。它们的 SOC 芯片测试仪器频率最高达 1 GHz,引脚数最多为 2048 pins,通道数最高达 1 024。大部分产品是在原逻辑和存储器测试仪器上添加硬件和软件,特别在测试方法上需要改进。目前 SOC 芯片测试现状不能令人满意,SOC 芯片测试并无统一标准,美国 IEEE 学会专业组正在注意该标准的制订。SOC 芯片商需要拥有成套的 SOC 芯片测试方法,需要满足各个生产阶段的测试技术要求和降低测试成本。扫描检测法、内建自测试电路技术及 IDDQ 检测等新技术将在 SOC 芯片测试中发挥更大作用。

四、系统级集成电路芯片加工技术

半导体加工技术是实现 SOC 芯片的基础,深亚微米加工技术是实现 SOC 芯片的关键条件,目前的 SOC 芯片特征尺寸为 0 .18 μm～0 .25 μm,集成度达数百万门/ 芯片。为了继续提高 SOC 芯片的性能和增加功能,扩大品种,降低成本,必须继续缩小 IC 的特征尺寸,增大集成度。按 Moore 定律,半导体技术,每隔 3 年,IC 特征尺寸缩小 30%,集成度提高 4 倍。人们积极开发新的加工技术,进行专项工艺研究,其中包括远紫外、电子束和 X 射线曝光光刻技术;芯片多层布线技术(增加布线层数);提高互连线电导率(采用 Cu 导线);减少布线间隔离层介电常数;利用掺杂和源漏工程扼制“ 二次效应 ”;增大 IC 加工线的晶圆片直径等。

SOC 芯片加工技术另一个问题是在一条加工线上能兼容不同类 IC 的生产工艺,也就是在同一 IC 芯片上比较满意地做好不同门类的 IC。SOC 芯片除含大量数字逻辑电路外,还有精密模拟电路、专用存储器,在某些 SOC 芯片上还包括射频器件。目前 SOC 芯片加工线多是可修改的高速数字逻辑电路工艺加工线。这种加工线能方便地增加工艺模块,适应各种不同工艺电路的加工需要。可以增加的工艺模块包括精密模拟电路工艺模块、高密度存储器工艺模块(含 SRAM、DRAM、闪速存储器)、射频电路工艺模块等。例如 Luncent Technologiec 公司的 SOC 芯片加工线就是采用 0 25 μm、5 层金属互连的 CMOS 数字电路工艺,其中包括可修改的精密模拟电路工艺模块、4 Mb 高速 SRAM 模块、8 Mb 5V 闪速存储器工艺模块及 1 .9 GHz 无线射频器件工艺模块、激光编程门阵工艺模块等。在数字逻辑工艺基础上增加工艺模块,将导致加工电路成本增加。如基于 17 次掩膜的 0 .25 μm 逻辑电路工艺,SRAM 工艺模块需增加 2 道掩膜步骤,闪速存储器工艺模块需增加 5 道掩膜步骤。每增加一道掩膜步骤,就增加晶片加工成本 2%～5%,这就相当于把含有高密度 SRAM 模块和闪速存储器模块的 SOC 芯片的成本分别提高 7%～18%,但 SOC 芯片系统在成本上,由于功耗和速度上的优势,还是优于多芯片系统。

五、系统级集成电路未来的发展

目前集成电路已进入片上系统时代。1999 年是 SOC 芯片技术最引人关注的一年,SOC 芯片已是芯片业发展的主流;但总的来讲,SOC 芯片产业还处于起步阶段,还有许多技术瓶颈亟待解决,小结如下:

(1) EDA 工具还不够成熟,其处理能力跟不上 SOC 芯片工艺技术的发展,使得 SOC 芯片设计不能充分发挥其全部性能。

(2) IP 再利用还不成熟,还存在如下问题,如实现现行标准兼容以前设计的 IP,制定 IP 的国际标准,实现模拟和数字 IP 优化设计及 IP 知识产权保护等问题。

(3) SOC 芯片的测试是困难的,涉及数字和模拟综合测试,需要找到一种成熟的测试和故

障诊断的策略。

(4) 在深亚微米 SOC 芯片设计中要解决互连线等寄生效应及引起的干扰、门锁效应、短沟效应、电迁徙效应及 SOC 设计中带来的其他问题。

(5) SOC 芯片封装因 SOC 芯片集成度高、功能复杂、引脚多、间距窄、功耗大,必须采用多引脚高密度封装技术,并要解决封装中的散热问题。

(6) SOC 芯片最终要集成功率器件,这依赖于新的有效隔离寄生参数技术和高导热性的新工艺技术。

(7) SOC 芯片设计现缺乏复合型人才,SOC 芯片设计实际是一个系统设计,要求设计者具有宽广和专门知识,既要有数字电路的特长,又要有模拟电路专长;既要有硬件知识,又要具有软件本领。

随着 SOC 芯片技术的发展,这些技术瓶颈必将逐步解决,在通信技术、计算技术、多媒体技术的驱动下,SOC 芯片必将在 IC 市场占重要地位。1999 年世界 SOC 芯片市场达 90 亿美元,2001 年为 300 亿美元。在系统集成的大趋势下,集成电路制造和设计公司、系统整机厂家将进行前所未有的紧密合作。它们将再与 EDA 公司、IP 供应公司、有关学术研究机构紧密合作,在 21 世纪创造 IC 领域的新奇迹,SOC 芯片必将成为 IC 市场的主流。

参 考 文 献

1 oseph Desposito . Wesco/ IC Expo 98 Delves Into System-on-a-Chip lssues . Electronic Design, 1998(1):59～64

2 Dave Bursly . The System-on-a-Chip It s Not Just A Dream Anymore . Electronic Design, 1997, 13(10):105～118

3 Bill Salefski etc . Reuse - Driven Methods Can Help Optimize Systems . Electronic Design, 1998, 22(6):82～88

4 饭冢哉 . IP 时代システムLSI 开发とデザィンハウスの役割 .电子材料,1998(1):59～63

5 王志华 基于核心模块的片上系统技术 .电子产品世界,1999(3):56～59

6 鲁瑞兵,魏小军 深亚微米集成电路高层次设计方法进展 .电子科技导报,1999(4):7～11

选自《微处理机》季刊,2000 年第 1 期

1.2 系统集成芯片综述

杭州浙江大学信息与通信工程研究所(310027)
姚庆栋 张朝阳 刘 鹏 张 明 章 勇

一、引 言

集成电路芯片工艺的发展,每隔一年半芯片规模增长一倍,工作速度翻一番,这是从20世纪后半期集成电路发展趋势总结出来的经验规则,通常称为摩尔定律。芯片工艺的发展给硬件带来了根本的变化,一个测控、通信、多媒体、消费类电子系统或子系统可以集成在一个芯片上,标志了一个系统集成芯片时代的来临。从表1.2-1列出的超大规模集成电路工艺、晶体管数、片上RAM/CACHE容量和典型PC机微处理器(MPU),可以预见今后几年仍会保持这样的发展趋势。

表1.2-1 集成电路的发展趋势

年份	工艺/μ	晶体管数/M	片上RAM/CACHE/KB	时钟/MHz	典型CPU芯片
1990	1	0.2	4	10	386
1994	0.5	2	16	100	Pentium
1999	0.25	20	256	800	Pentium Ⅲ新型

二、系统集成芯片结构

目前,一个计算量大的数字系统还不能用一个微处理器(MPU)实现,而要依靠多个计算功能部件的并行计算,加上输入/输出(I/O)、各种存储、缓存和寄存器部件构成。并行计算结构是组成的关键,所以系统集成芯片的一种分类方法就以并行计算结构作为依据。同时,PC机和工作站MPU采用超标量、超长指令(VLIW)在指令级上并行调度和协调芯片上计算功能部件,发挥并行计算效率的成熟经验;采用高速存储(片上的RAM、CACHE和寄存器与片外内存多级存储)构造克服存储器速度跟不上计算功能部件速度[1]的经验,也都在系统集成芯片上得到应用。

Pirsch讨论了视频处理多媒体系统,文献[2]把结构分成两种主要类别,一类是专用结构(dedicated architecture),另一类是可编程结构(programmable architecture)。这对其他系统集成芯片也是适用的。专用结构主要是硬连接电路类型结构,其中小部分也可能由程序控制。这种结构专门针对某一系统某一处理算法设计,在工艺制造上容易做到省芯片面积、省功耗、低成本。可编程结构原则上是一种专用的微处理器,能适应系统结构变动和改进引起算法变动的需求,为今后系统性能提高升级留有余地,为今后芯片的升级节省研究设计时间、节省成本、提高效率。通用的MPU和DSP能实现系统的各种功能和实时需求,可以采用。一般的

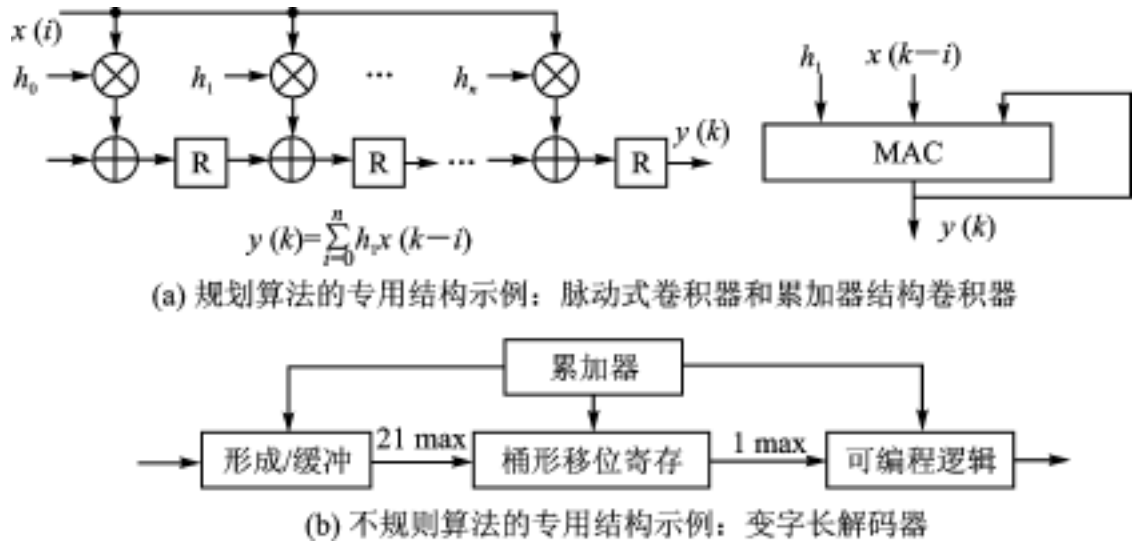
MPU 和 DSP 是以办公文件处理、数据库调用、科学和工程计算或 CAD 为目的设计的,并不考虑实时操作,而一般工业、国防、通信的嵌入式系统必须做到实时运行,计算字长只要 8～16 位,因此大多数 MPU 和一部分 DSP 的许多功能因用不上而浪费掉。为了既消除浪费又满足实时要求,有必要根据具体应用设计专用指令集处理器(ASIP——Application Specific Instruction Set Processor)[3]。这是一种在硬件上加强并行计算功能,在指令集上抛弃与专用无关的指令,在软硬件构造上都优化了的处理器,可降低芯片面积和生产成本。目前,一般结构与专用结构相比面积和成本都稍大一些,但在更新开发下一代芯片时可以节省开发成本,从使用灵活、便于修改算法程序方面来看也是合算的,因此成为当前的发展趋势。在系统集成芯片中也有以可编程处理器为主与部分专用结构相结合的混合结构。

1.专用结构

20 世纪 80 年代中期和 90 年代初期,芯片的时钟频率做到 10 MHz 左右,规模在 20 万晶体管或 5 万门左右。通常系统中一段算法可用一片芯片实现,如 64 级卷积器、BCH、RS 解码器可以集成在一个芯片上;一个 H.261 解码系统由 7 片芯片构成。在数据率和时钟频率相当时,具有规则算法的系统流行脉动式结构。所谓规则算法是指可化成代数的矢量、矩阵运算等,例如位移估值器[4]、Viterbi 译码器等[5],[6]。

图 1.2-1(a)的前部分是一个 n 级卷积器的脉动式结构,可用来做卷积变换。一个 n 级卷积器由 n 个乘法器、n 个加法器和 n 个寄存器构成。这种结构把计算速度不高的多个运算器排成流水并行,利用速度较快的寄存器存储数据,避免数据存储到速度较慢的存储器。这种结构的时钟每一节拍可从流水级输出计算结果,满足实时运算的要求,即以较多运算器作流水并行换取计算速度。20 世纪 80 年代末、90 年代初,LSILogic 公司和 Inmos 公司的信号处理卷积芯片都是这种结构。今天芯片时钟频率可以高达 1 GHz,一个 64 级卷积计算部件可用一个或几个乘加累积器(MAC)实现,如图 1.2-1(a)后部分所示。由此可见,集成电路工艺的进步促进了结构的变化。

不规则算法也可设计专用结构与算法匹配,例如图 1.2-1(b)是一种变字长解码结构[1],先把输入编码数据按前缀码规则分成不等长码字,送到可编程逻辑做成的查找表解码。根据查找表送出码字长度控制形成/缓冲和桶形移位寄存,把下一个码字送到查找表。



(a) 规划算法的专用结构示例:脉动式卷积器和累加器结构卷积器

(b) 不规则算法的专用结构示例:变字长解码器

图 1.2-1 专用结构图

专门针对各子算法、子系统设计的结构组合在一起,就如同把原来在一个印制板上的电路

集成到芯片上,生成一个专用结构的系统集成芯片。BroadCom 的 BCM3115 便是这种芯片的典型例子。

2 . 可编程结构

前面已经谈到,系统集成芯片的可编程处理器大多是面向应用系统的 ASIP,可编程结构的风格如同一个计算机系统,并且是并行计算系统,用嵌入软件实现系统功能。

(1) 分裂运算器和专用指令系统

多媒体和通信系统的运算字长主要是 8 位、16 位,PC 机和工作站的 MPU 与部分 DSP 设计成既可在 32 位字长上运算,又可分裂成 4 个 8 位或 2 个 16 位字长的运算器工作,并且配有专用寄存器和使分裂运算器并行工作的专用单指令多数据流(SIMD)指令。属于这类指令的有 Ultra Sparc 芯片的 VIS 指令、Pentium 的 MMX 指令、AMD 的 3DNOW 等,可方便地做多媒体视频处理。TI 公司的 TMS320C62DSP 是一种不完全的分裂运算器结构,它的 32 位加法器可分裂成 2 个能同时并行做 16 位字长的加法,其乘法器只能做 16 位乘 16 位字长的乘法,32 位乘法则只能用微程序来做。

通用的 MPU 和 DSP 采用分裂运算器和专用指令系统的结构,其目的是扩充芯片能力,提高处理效率。例如 Pentium 芯片做成的 PC 机就可扩展成为多媒体计算机,主要为办公文字处理设计的功能扩充了,实现了看 VCD、听 CD 碟片的多媒体功能。在 TMS320C62 芯片上处理 16 位字长运算,效率高,适于通信、雷达、软件无线电等一维信号处理。这种结构的系统往往是在通用化的基础上扩展某类应用能力的。

(2) 超长指令结构

为了发挥指令级并行处理能力,多数 PC 机和工作站的 MPU 芯片采用超标量结构。MPU 同时读取几条指令,由片上的调度器分配、发放到各个计算单元并行执行。调度器要能最大限度地利用片上运算和存储资源,防止和解决冲突。超标量结构实行的是一种芯片上动态调度发放指令方法,适用于 PC 机和工作站的用户使用不同应用程序和经常改变应用程序的情况。专用系统集成芯片虽然有的也用超标量结构,但由于功能和计算单元比通用的 MPU 多,并行度要高一些。为了避免片上调度器设计的困难,往往采用超长指令结构(VLIW),见图 1 .2 - 2(a)。它是把若干个功能计算单元(FU)的指令联成超长指令字(VLIW)对各计算单元(FU)进行控制。各 FU 使用的寄存器栈是公共的,称为全局寄存器部件,用读/写交叉开关方便地和各 FU 互联。超长指令由编译器编成,因此是一种静态调度。它把应用程序中的计算分配给各 FU 执行。由于专用系统集成芯片嵌入的应用程序已在研制单元设计好,系统工作时用户无需更换应用程序,也不会经常更改,因此避免使用片上调度可以简化芯片的结构。这是许多系统集成芯片采用超长指令的原因。这种结构的数据并行和指令并行结合得比较好,可以发挥细颗粒并行性能,也可做一些任务并行。

TI 公司的 TMS320C62 属于这种结构。它有 32 个寄存器栈,8 个功能部件,时钟频率为 200 MHz,峰值处理能力可达 1600 MIPS(兆指令每秒)。它适宜移动通信基站、声码器、不对称数字用户环路(ADSL)、电缆调制解调器(Cable Modem)等通信系统使用。Philips 的 Trimedia TM - 1000 有 1 个 VLIW 核、128 个寄存器栈、27 个功能单元,时钟频率为 100 MHz,同时发放 5 条指令。芯片上还有专用结构的变字长解码、图像处理器、通信接口等,峰值处理能力达到 4 GOPS(吉次运算每秒),依靠编程可以作为 H .261, H .263 编解码器双工工作,也可作为 MPEG - 2 解码器工作。

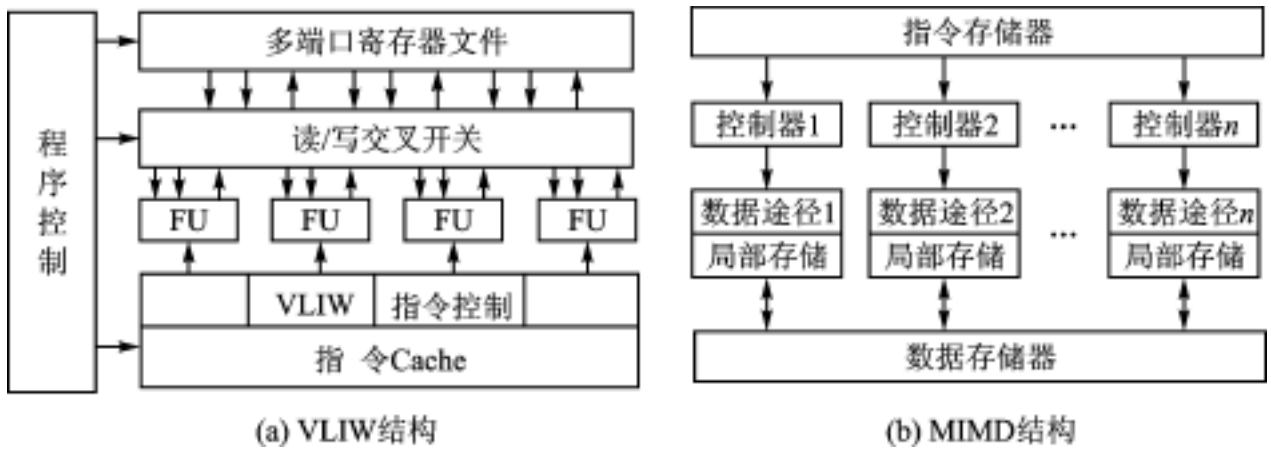


图1.2-2 可编程结构图

(3) 多指令数据流结构

多指令数据流(MIMD)结构(见图1.2-2(b)),主要可以利用任务并行的优点,也可利用数据并行的优点。和超长指令结构不同,其每一数据途径有1个控制器,可提供给各个数据途径的各自程序控制流。也可在数据并行的时候执行单指令多数据流的程序。这种结构的代表是TMS320C80多媒体处理器,其优点是每一处理途径的处理有着高度的灵活性,适合多媒体进行各个层次的处理,满足低、中、高层次处理的不同要求;但编程比较困难,只能由程序人员人工确定处理功能器件的同步。由于每一数据的途径都要有控制电路,硬件复杂度增加,所以使用不如其他结构广泛。

Pirsch等[7]研制了一种由RISC核和SIMD协处理器组成的异构性MIMD结构AxPe芯片。PISC核做高层处理,如I/O的管理、变字长解码。协处理器做余弦变换和逆变换、位移估值和补偿。把4个AxPe集成为一个系统芯片,可用作H.263和MPEG编解码器。

三、0.25μ工艺的VLIW结构

VLIW结构很受重视,例如Trimedia和MPact就采用了VLIW结构。文献[8]研究了0.25μ工艺VLIW结构可以达到的并行度和工作速度,提出了一种方案,做视频处理器类型的系统芯片时,时钟频率达到600 MHz,每周期执行64个操作,可以达到38 GOPS的峰值操作速度。下面介绍该文的一些思想,作为系统集成芯片研究的参考。通常的MPU常用每秒执行多少条指令作为工作速度的指标,但系统算法设计后,归结出的结果是希望每秒执行多少次算术运算(加法、乘法)。每秒执行多少条指令和每秒执行多少次算术运算并无对应的关系。一般做一次算术运算需要取两个变量进行运算,并把结果写回。对于运算量大的实时系统芯片,希望能在一个时钟节拍内完成这些操作;因此结构上采用层次存储,由寄存器文件(RF)与算术运算和功能单元相配合,只有RF速度快,读写时无耽搁,才能在一个时钟节拍完成读数操作和把操作结果写回。VLIW结构的设计是以一组RF与算术运算和功能单位组(简称功能单元FU)配置在一起的,能做最佳的互相联结,保证计算最快,数据交换最快。按照这种思想设计的VLIW结构模型就是如图1.2-2(a)所示的理想的全局寄存器栈VLIW结构。

当希望片上尽量多安置一些FU,并为每个FU配上3个RF时,设计就会变得很复杂。如果片上要安排64个FU,按全局寄存器栈VLIW结构模型设计,就要配置192个RF,并要有192个端口才能和各FU联结。这种结构虽然执行程序算法十分灵活,但实际上结构互联十分复杂,由于芯片工艺参数和布局结构造成工作速度下降,实际上就做不到高速运算,用

0.25 μ工艺的一个 24 端口 RF 执行时延超过 2 ns,时钟频率达不到 500 MHz,因此图 1.2-2(a)的全局寄存器栈 VLIW 结构模型只适用于 FU 不多的情况。

要解决 FU 多、全局寄存器栈复杂和工作速度不高的矛盾,可采用分布数据途径结构(见图 1.2-3),把 FU 分成组,每组把 RF 分散给各数据途径,但可通过全局互联网络连接各条数据处理途径。该图的一个 FU 组为 2 个 FU,每个 FU 含 1 个算术逻辑运算器(ALU)、1 个乘法器(MPY)和 1 个桶形移位寄存器(BS)。实际上数据处理途径可以根据系统的需要设置。TMS320C62 就只含有 2 条数据处理途径。每条数据处理途径有 3 个加法器、1 个乘法器。在芯片中 FU 多时,可如图 1.2-3 布局,把 RF 放在交叉开关全局互联网络的两边,使各数据路径的 RF 可方便地交换数据。FU 和 Cache IC 靠着 RF 布置,使每一数据处理途径都非常紧凑。每一条数据处理路径都含有自己的局部存储器(LM),即一种数据缓冲器。数据途径经 DMA BUS 和外存储器沟通(见图 1.2-3(b))。

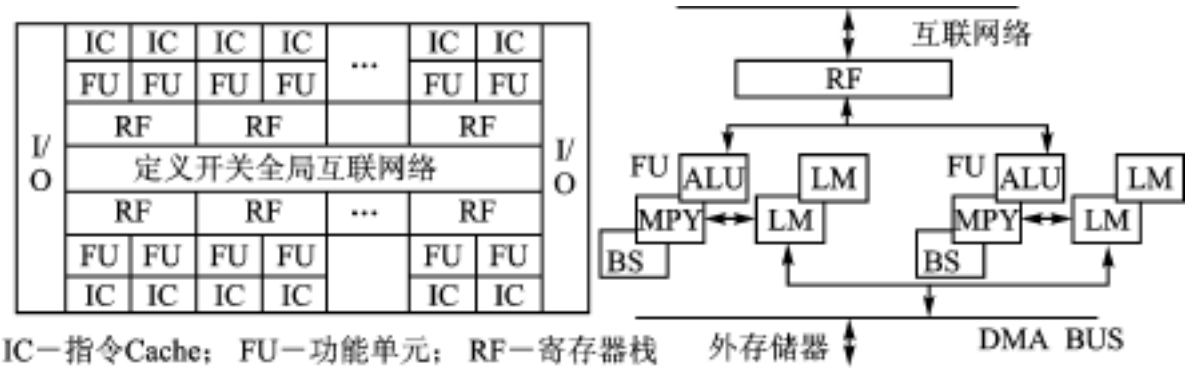


图 1.2-3 一种 VLIW 结构的数据途径模型

交叉开关网络可用 2-1 传输门复接器(见图 1.2-4(a))构成,图 1.2-4(b)示出的是一个 1 位 8 端口交叉开关网络的例子。文献[8]在 0.25 μ工艺条件对这种结构作了仿真,对于 16 位数据线,这种交叉开关在 16,32 和 64 端口时的时延分别为 1 ns,1.5～2 ns,2.5～5 ns。如希望时钟频率达到 600 MHz 以上,这种交叉开关结构只能做到 16 端口。

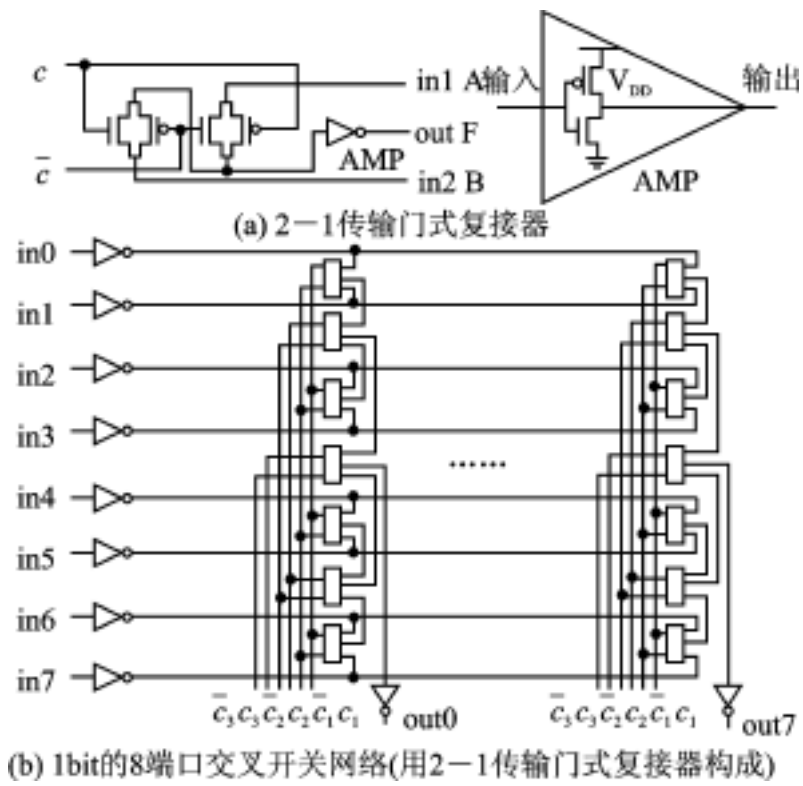


图 1.2-4 交叉开关网络

文献[8]考虑芯片最大尺寸可做到 20 mm×20 mm,用一半面积作数据处理途径,即包括互联通信网络、局部存储器、寄存器栈及计算功能单元等;用一半作指令存储器、分支单元、程序计数器、旁路单元、片上控制及内部联络等,则可构成 16 条数据途径,每条途径有 2 个 FU,每个 FU 由 1 个 ALU、1 个 MPY 和 1 个 BS 构成。每条数据途径占用面积见表 1.2-2。

表 1.2-2 每条数据途径占用面积 单位 mm^2

2 个 16 位 ALU	2 个 8×8MPY	2 个 16 位 BS	64 个 6 端口 RF	16 个 128BLM	FU 内部联络	总计
0.8	4	2	0.61	4.5	0.43	12.34

这个结构中,RF 与 LM 时延都在 1 ns 以内。16 条数据途径总需面积 197.4 mm^2,加上 16 端口交叉面积 2.6 mm^2,数据途径总面积 200 mm^2。由于数据途径做到 16 条并行,时钟频率可达到 600 MHz,这种 VLIW 结构的视频处理系统的计算能力可达到 38 GOPS。

文献[8]给出了 0.25 μ 工艺的一种 VLIW 结构及其所能达到的工作能力,对系统集成芯片研究有很大参考价值。由此可以看出,系统集成芯片要把很大的精力放在芯片内部通信互联、片上寄存器和存储器的设计上。

四、芯片嵌入软件和软硬件协同设计

和硬联结电路芯片不同,芯片嵌入软件和开发工具是系统集成芯片的一个关键,据统计这方面的工作量占全部工作量的 60%。

目前,芯片嵌入软件多数用汇编语言、少数用标准 C 语言编写,原因是系统集成芯片用于实时运算,要求代码执行效率高,额外开销少,至少在循环程序内部要没有额外开销。一般 DSP 的 C 语言编译器效率低,代码执行速度比汇编代码慢 3～10 倍。随着系统越做越大,用汇编语言人工编程效率也较低,因此希望能随着系统集成芯片设计同时开发出一个好的 C 编译器,并且最好能在同类 ASIP 中互相转用,以节省开发人工。同时要有一个汇编指令仿真器,具有每秒运行 1 万到 1 百万条指令的速度,最好还有步进准确跟踪的执行能力。把 C 编译和仿真器或芯片结合在一起,形成能进行原程序查错的软件。这样对编制应用软件和嵌入软件就会有较好的效力。对于复杂的系统芯片,其中的 RISC 或 VLIW 核需要配上实时操作系统(RTOS)[9],用来和实时环境进行交互,以及调度管理芯片上各种任务的执行,因此一个芯片的设计,必须硬件/软件协同设计,图 1.2-5 示出了一个理想的硬件/软件协同设计环境[3]。

文献[10]以一个媒体处理器为例,介绍了一个处理器核的硬件/软件协同设计方法(见图 1.2-6)。这个例子从处理器流水级布置上研究结构的优化。

另一例是我们作了一个 REED-SOLOMON 解码器预设计,选用一个 32 条指令的 RISC 类型指令集,整个解码程序可编成 188 条(24 位字长)指令。经过并行度分析,化成 38 条长指令(48 位字长),节省了指令存储空间(336 B)。优化前计算定位多项式和误差多项式需 99 个码符时间,优化后缩减为 59 个码符时间。数据途径用虚拟机结构需 34 000 门,优化后缩减成 25 000 门。可见硬件/软件协同设计方法确实是系统集成芯片设计的好方法。

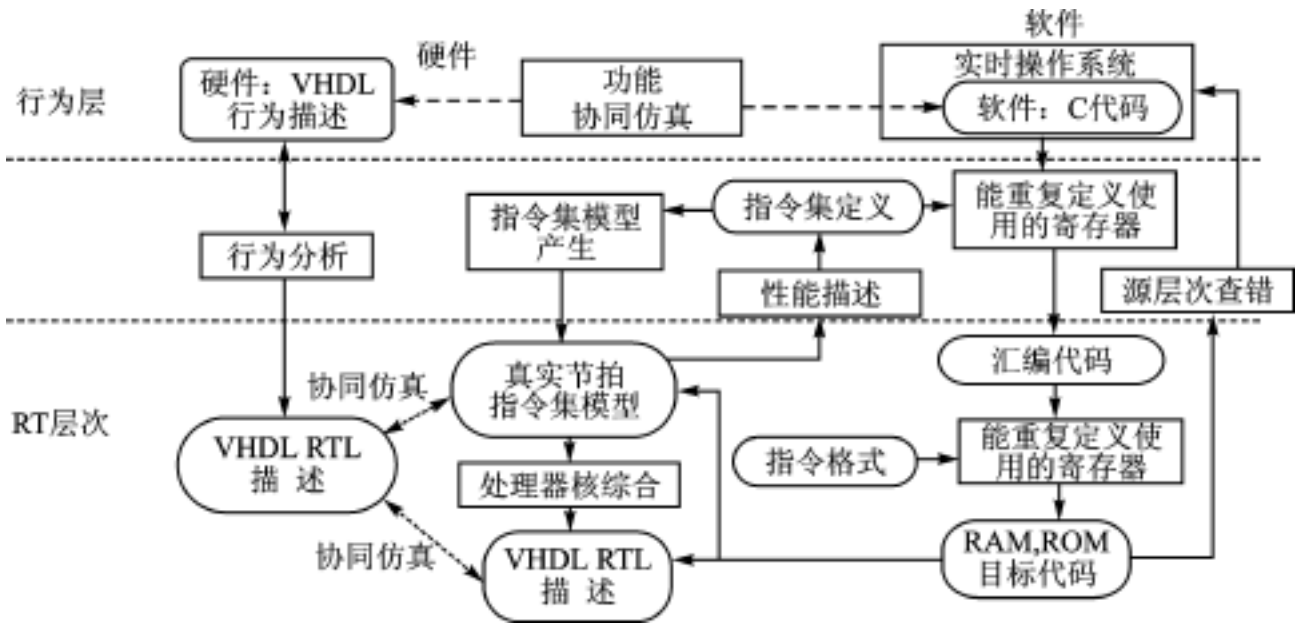


图 1 2-5 理想硬件/ 软件协同设计环境

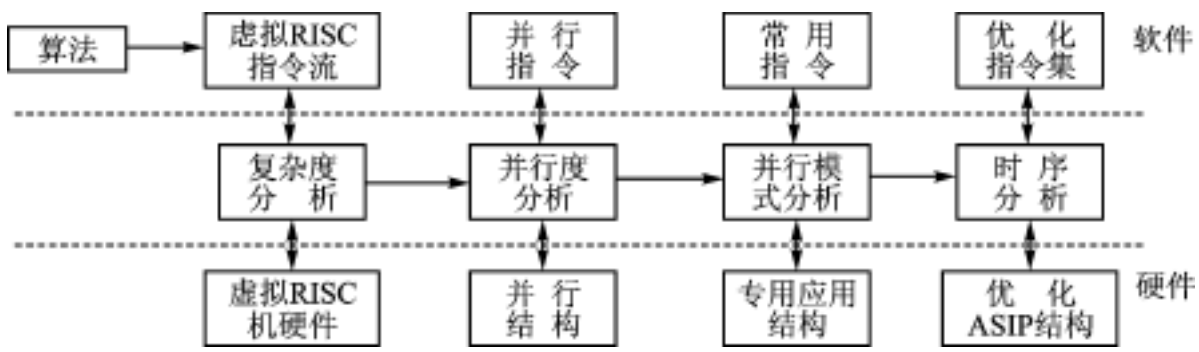


图 1 2-6 一个硬件/ 软件协同设计流程图

五、结 论

本文对系统集成芯片作了综述介绍。随着集成电路的发展,系统集成芯片必将会以很快速度发展。系统集成芯片无疑会向可编程化方向发展。过去研究的并行技术和计算机指令级并行以及微处理器的成果与经验,多已应用到系统集成芯片上。硬件的成熟程度相对好一些。嵌入软件和发展软件的编译器、仿真器与源码查错软件,在芯片设计一开始就应受到重视。

参 考 文 献

1 ennesey J, Patterson D. Computer Architecture: A Quantitative Approach[M]. San Francisco: Morgan Kaufmann, 1990

2 Pirsch P, Stolberg H-J. VLSI Implementations of Image and Video Multimedia Processing Systems[J]. IEEE Transactions on Circuits and Systems for Video Technology, 1998, 8(7):878～891

3 Paulin P G, Liem C, Cornero M, et al. Embedded Software in Real-Time Signal Processing Systems: Application and Architecture Trends[J]. Proceeding of the IEEE, 1997,85(3):419～435

4 Moor W, McCabe A, Urquahart R. Systolic Arrays[M]. Adam Hilger. 1986(2.5 An efficient systolic array for distance computation required in a video-code based on motion-detection)

5 Fattweis G, Meyr H. Parallel Viterbi Algorithm Implementation: Breaking the ACS-Bottleneck[J]. IEEE Transactions on Communications, 1989, 37(8):785～789

6 宣建华,姚庆栋. 高速 Viterbi 处理器:流水式快速处理结构[J]. 通信学报,1995,16(1):94～100

7 Gaedke K, Jeschke H, Persch P. A VLSI-based MIMD architecture of a multiprocessor system for real-time Video Processing Application[J]. Journal of VLSI Signal Processing, 1993, 5:159～169

8 Dutta S, óconnor K J, Wolf E, Wolfe A. A Design study of a 0.25um Video Siganl Processor[J]. IEEE Transactions on Circuits and Systems for Video Technology, 1988, 8(4):501～519

9 刘鹏,姚庆栋. 实时图像处理并行操作系统的微内核设计和实现[J]. 电子学报,1999, 27(7):42～46

10 hang Y, Ma K K, Yao Q. A Software/ Hardware Co-Design Methodology for Embedded Microprocessor Core Design[J]. IEEE Transaction on Consumer Electronics, 1999, 445(4):1241～1246

选自《计算机自动测量与控制》双月刊,2000 年第 3 期

1.3 Java嵌入技术综述

昆明理工大学信息与自动化学院计算机系(650093)
王海兵 吕 杨

一、概 述

计算机技术和网络技术的发展将人类引入了网络化信息时代。Java语言的面向对象和跨平台特性以及分布式、可靠性、安全性、可移植性、动态性、多线程性等特性不仅为Internet的使用提供了良好的开发和运行环境,而且对嵌入式系统的发展产生了较大的影响。Java嵌入技术是将Java语言编写的应用程序嵌入到电视、电话、洗衣机以及微波炉等各种消费类电子产品中,使这些设备具有计算机的特性。通过Java嵌入技术,各种嵌入设备可以在网络环境下安全地发送和执行可移植代码。

嵌入计算技术在20世纪90年代取得了飞速的发展。Java正是为了在嵌入技术领域得到发展而开发的一种完全面向对象并独立于硬件平台的编程语言。Java嵌入技术自从Java语言的公布以来就一直在不断发展。早在1996年初,Sun公司就公布了Java芯片的初步资料,同年底公布了JavaCard 1.0;1997年底又正式公布了Personal Java和JavaCard 2.0,并兼并了从事嵌入式实时操作系统开发的Chorus公司;1998年初公布了Embedded Java的初步资料;1999年7月公布了Embedded Java的应用环境规范,同时还公布了Java实时操作系统规范。各种Java嵌入技术的应用规范的发布,使Java嵌入设备的开发得到了快速发展,也为刚发布不久的Jini分布式嵌入网络体系提供了大量的非PC联网对象。

二、Java嵌入系统构成

Java嵌入系统包括硬件构成和软件构成。为使嵌入设备具有计算和可编程的特性,在硬件上必须添加微处理器以及存储设备;软件上必须添加系统软件和应用软件。系统软件包括Java虚拟机JVM(Java Virtual Machine)和Java操作系统Java OS(Java Operation System),且操作系统为实时操作系统RTOS(Real Time Operation System)。应用软件包括在各种Java嵌入开发平台上编写的应用程序。

1.硬件构成

Java嵌入总是基于一个实际的物理设备,目前的各种家用电器和消费类电子产品,小到个人数字助理PAD,大到冰箱或汽车,都可以成为Java嵌入设备。将Java应用程序嵌入到这些设备中,必然要改变设备的底层硬件结构和工作规范。现在的电器设备大多以中大规模集成电路为主要核心,要实现嵌入设备,就必须在硬件上增加CPU、ROM或RAM等智能处理设备。在此基础上才能存储和运行JavaOS、Java虚拟机和各种嵌入的Java应用程序。这样,电子设备变成了智能化的设备。Java嵌入设备的成功范例已经很多,如网络汽车、Web电话、Java智能卡、Java戒指等。

2. Java 虚拟机

Java 虚拟机是用来运行经过编译的 Java 字节码的计算机实现的。从嵌入设备角度来看，Java 虚拟机是建立在处理器等硬件基础之上的，在本设备上运行的所有 Java 应用都要通过它执行。Java 虚拟机可由两种方法实现：纯软件仿真和用 Java 芯片。目前现有的 JVM 一般是在特定宿主机上的软件产品，随着 Java 应用的迅速发展，传统用软件实现的 JVM 已经难以满足速度上的要求，用硬件实现 JVM 已经成为 Java 发展的一个重要方向。

Java 虚拟机可以说是上层 Java 应用程序与底层异构硬件的中间接口，在 Java 平台结构中 Java 虚拟机起着承上启下的作用。在 Java 虚拟机上方是 Java 应用程序接口，由 Java 基本类和 Java 标准扩展类组成。在 Java 虚拟机下方是移植接口，由依赖于平台的和不依赖于平台的两部分组成，其中依赖于平台的部分为适配器，Java 虚拟机通过移植接口在具体的操作系统上实现。而如果直接在 Java 操作系统上实现，则不需要依赖于平台的适配器。Java 虚拟机与 Java 平台的关系如图 1.3-1 所示。

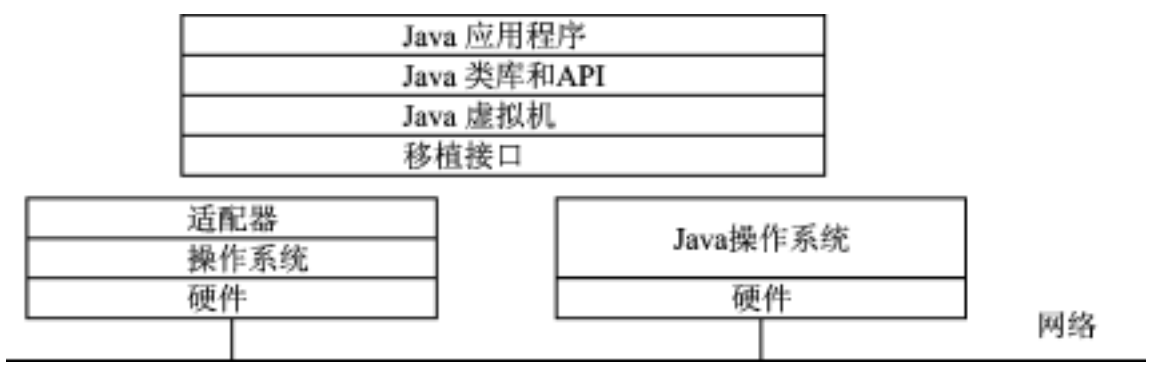


图 1.3-1 Java 虚拟机与 Java 平台

在嵌入计算环境下，安全性是至关重要的。Java 虚拟机在 Java 的安全体系中占有重要地位。在 Java 虚拟机中可完成内存的布局、字节码的检查和字节码装载器的安全检查。

3. Java 操作系统

Java 操作系统是指为运行 Java 应用程序而开发的操作系统。Java 操作系统有许多产品，其中 Java OS 是目前最流行的一种 Java 操作系统。Java OS 是为使 Java 能够在各种计算机和嵌入设备平台上运行优化而设计的一种新的平台。它既提供 Java 平台来自动运行各种 Java应用程序，同时也提供 Java 虚拟机和底层功能以实现窗口、网络以及文件系统功能，而无需任何主操作系统的支持。Java OS 由两部分组成：一是本地代码部分，具有指定的指令集和硬件平台；二是 Java 代码部分。其中 Java 代码部分具有平台独立性。

Java OS 中还包括了许多用 Java 编写的设备驱动程序和一套网络协议组。这些协议包括符合 TCP、UDP、IP 和 ICMP 标准的基本传输和路由选择机制。由于 Java OS 不包括其他操作系统中所存在的外来特性，它允许建立更小和更为简单的设备。Java OS 只需用 4 MB 的 ROM 和 4 MB 的 RAM 就可建立一个完整的系统。Java OS 具有高度的可配置性，可根据嵌入设备的硬件设置和要求删除其中不需要的部分。针对 Java 嵌入设备的多样性，每一个嵌入设备都需要一个 Java 实时操作系统，因此在 RTOS 上开发 JVM 和它的支持环境成为目前最广泛的应用。

三、嵌入式 Java 的应用平台和接口

1 . Personal Java

Personal Java 是专门为构建可连接网络的消费类电子设备而设计的 Java 应用环境 JAE (Java Applicatioin Envirment)。该环境主要用于家庭、办公室和移动性应用及产品的设计与实现。它是 Sun 公司推出的一系列 Java 产品中专门面向消费类电子设备的一个平台,由 Java 虚拟机(JVM)和一系列 Java API 组成。其中 Java API 包括核心、选项和类库。另外, Personal Java 还必须在资源有限的环境中包括消费类电子产品及应用需要的一些特定的特征。Personal Java API 与 Embedded Java API 共享一套核心 API,具有运行 Applet 支持对网络资源的更直观的显示和网络连接功能。适合 Personal Java 应用环境的消费类电子设备包括手持移动设备(如 PDA、HPC 等)、数字机顶盒、游戏操作台等。

Personal Java 对 Java 保持向上兼容,即一个标准的 Java 运行环境可运行 Personal Java 程序,但 Personal Java 则不一定可以运行任何 Java 程序。Personal Java 的系统结构如图 1 .3 -2 所示。Personal Java 技术在汽车自动导航设备、医疗电子设备、游戏等领域都具备广阔的应用潜力。目前,在机顶盒(Set-top Box)、智能能信设备等领域已出现了大量基于 Personal Java 平台的产品,例如 Toshiba、TCI、Philips、Panasonic、Web TV、Microsoft 等多家公司的 Set-top Box 均采用 Personal Java 作为产品的软件平台。

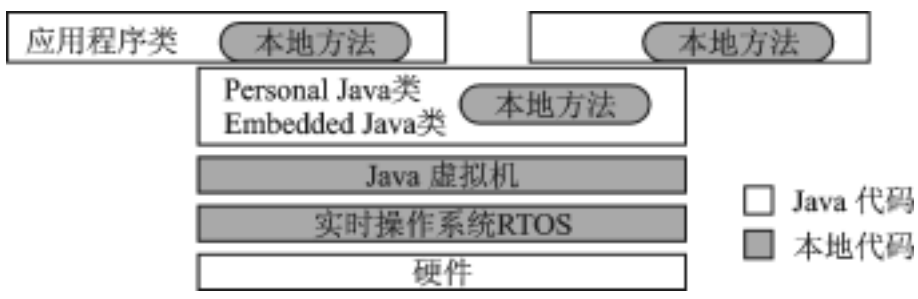


图 1 .3 -2 Personal Java 或 Embedded Java 的系统结构

2 . Embedded Java

Embedded Java 和 Personal Java 一样也是一个新的 Java 程序运行环境,也是由 Java 语言核心和扩展 API 组成的;但它比 Personal Java 更小,对 Personal Java 和 Java 保持向上兼容。Embedded Java 的系统结构如图 1 .3 -2 所示。Embedded Java 也是用于各种嵌入设备,是为那些比 Personal Java 的硬件环境要求还严格的嵌入设备而设计的。它与 Personal Java 一低一高,从而为各种不同的嵌入设备环境提供了相应的 Java 解决方案。

Embedded Java 主要适用于专用的工业或低端的电子设备,如打印机、低端电话机等。它是 Personal Java API 的子集,主要差别是缺少 AWT 类(因此不支持复杂的图形显示),缺少 Applet,仅支持 Application,并且 Networking 类是可选的。在 Embedded Java 应用程序环境规范中对 Embedded Java 的 API 做了明确规定。

3 . Java Card

Java Card 也是一个软件规范,用于智能卡编程。当前的标准是 Java Card 2 .1,遵从 ISO7816 -4 规范,可在绝大多数智能卡上运行。目前已取得各大智能卡生产商的广泛支持。Java Card 是标准 Java 的一个子集。Java Card 2 .1 技术规范包括有关在智能卡上创建 Java Card 虚拟机和应用编程接口 API(Application Program Interface)的详细信息。当前开发Java

Card 的最新工具是 Development Kit 2.1.1。由于智能卡的硬件功能远不如 PC 强大,所以并不是所有的 Java 语言规范和 Java 虚拟机规范都被 Java 智能卡支持。智能卡的 RAM 一般不超过 1 KB,ROM 一般不超过 16 KB。最低系统要求为 16 Kbps ROM、8 Kbps EEPROM 和 256 B RAM。

Java Card 虚拟机是建立在特定集成电路和本机操作系统执行程序上的。JVM 层利用一般语言和系统接口隐藏了制造商的专利技术。Java Card 框架定义了一系列用来开发 Java Card 的应用程序和为这些应用程序提供系统服务的 API。某些特定行业或特殊的商务应用可提供添加的库,以提供服务或优化安全性和系统模型。Java Card 应用程序称为 Applet。一个卡上可驻留多个 Applet。每个 Applet 均被其应用程序标识符惟一标识,如 ISO7816 第 5 部分的规定。

4. Java 芯片

Java 芯片是指可直接执行 Java 字节码的处理器。由于目前的 Java 应用程序都基于一个用软件实现的 Java 虚拟机来解释或编译执行,这难免影响了执行的速度并多占用内存。而 Java 芯片用硬件实现了 Java 虚拟机规范。目前 Java 芯片的规范是 PicoJava I 核心规范。该规范也是目前惟一已公布的 Java 芯片内部结构规范。该规范为优化运行 Java 代码做了许多创新设计。

PicoJava 在芯片一级上实现了字节码指令,并且还采用了其他方法加速了 Java 字节码的执行效率;在芯片一级上实现了 Java 中常用的线程同步和各种内存垃圾回收方法;它还实现了方法的调用和局部变量载入的屏蔽。因此用 Java 芯片运行 Java 程序比 Java 解释器快 10 倍。同等时钟频率下的 Java 芯片比采用解释器的 Pentium 机快 10 倍。

PicoJava I 芯片体系结构是 MicroJava 芯片的基础。该芯片将广泛应用于嵌入式应用系统,如机顶盒、工业数据采集设备、蜂窝电话、网络计算机等。

5. Java Phone API

Java Phone API 是一个专门对直接电话控制、数据报信息和电源管理等电话设备进行功能访问的应用程序接口。通过 Java Phone API,能够在无线蜂窝电话和 Internet 屏幕电话上安全地传递动态的信息服务。

6. Java TV API

Java TV 应用程序接口(API)是 Java 平台的扩展。它是 Sun 公司和数字电视工业界的关键领导者通过开放的程式共同开发的。目前,众多电器制造商已经公开支持采用这一 API 作为全球的数字电视标准。

四、应用前景

计算机技术的发展已经从以 PC 为主的单机模式转为目前的网络模式,在后 PC 时代,计算模式的发展正转变为计算的网络化和计算的嵌入化。

计算网络化已为人们所熟悉。网络世界给人类生活带来了巨大的变化,人们可以在网上收发电子邮件、网上聊天、浏览 Web 上的多媒体。这一切仍不能满足人们的要求,于是有了信息家电联网。其宗旨是把与人们生活息息相关的各种家用电器像使用电话一样连接成一个大网络,我们可以在办公室直接控制家里的洗衣机或微波炉,使其不需要人的直接操作就能为我们服务。Jini 分布式网络体系就是将各种各样的电器设备联网,使这些设备插上插头就能自

动识别并提供或接受服务。这种网络体系实现了真正的“网络计算机”或“分布式计算网络”。

计算的嵌入化广义上讲是将计算机制引入到非计算机设备,如电话、电视、吸尘器、洗衣机和汽车等等,使这些设备具有计算的功能。计算的嵌入化将大大地扩大“计算机”的含义,我们甚至可以将这些嵌入设备称之为“计算洗衣机”或“计算微波炉”等等。计算的嵌入化是随计算的网络化发展而发展的,而 Java 嵌入技术只是嵌入领域的一部分。目前微软的 Windows CE、Acer 的 XC 等都是针对计算的嵌入化而开发的软件,但 Java 嵌入技术在这一领域有着更广阔的应用前景。这一优势是由 Java 语言的天然特性决定的。用 Java 语言开发嵌入设备可以大大降低软件开发的复杂程度,从而减少软件开发费用,降低系统成本。Java 语言的可移植性可使开发的程序做到“一次编写,到处运行”,同时 Java 语言是高度安全的,这对嵌入设备是非常重要的。Java 的动态编联可以在分布环境中动态地维护应用程序及其支持类库之间的一致性。而且,Java 字节码短小,有利于节约嵌入设备宝贵的硬件要求。另外,Jini 是建立在 Java 之上的,Jini 针对 Java 虚拟机联合体;而 Java 是针对单个 Java 虚拟机。它们一个整体,一个局部,两者结合在一起,构成了未来的家庭网络。

五、结束语

Java 嵌入技术是面向信息家电联网为主的,以 Jini 分布式嵌入网络为联网体系的综合技术,这一技术目前正处于开发和推广阶段。相信随着嵌入计算技术和网络技术的发展,Java 嵌入技术必将给人类带来一个全新的生活方式。

参 考 文 献

1 王克宏,等. Java 嵌入技术[M]. 北京:清华大学出版社,1998

2 徐剑军,等. Jini-Java 嵌入技术的新突破[J]. 计算机工程与应用,1999,(12):30～32

3 陈虎,戴蔡,等. JVM 指令系统特点和它对 Java 芯片设计的影响[J]. 计算机应用研究,2000,17(6):107～110

4 刘宏. 解读 Personal Java[N]. 计算机世界,1999-03-08

5 杜飞龙. 智能卡之魂 Java Card 2.0[N]. 计算机世界,1999-09-13

6 李建平,杜克明. 嵌入式代理的开发[J]. 计算机应用研究,2000,17(6):50～51

选自《计算机应用研究》月刊,2001 年第 6 期

1.4 Java 的线程机制

清华大学计算机系(100084) 戴梅萼

线程是指程序中能顺序执行的一个序列。一个线程只有一个入口点,但可能有几个出口点,不过,每个时刻的执行点总是只有一个。线程是不能独立运行的程序,而只是某个整体程序内部的一个顺序执行流。

多线程是 Java 的一个重要特点。如果一个程序是单线程的,那么,任何时刻都只有一个执行点。这种单线程执行方法使系统运行效率低,而且,由于必须依靠中断来处理输入/输出,所以,当出现频繁输入/输出或者有优先级较低的中断请求时,实时性就变得很差。多线程系统可以避免这个缺点。所谓多线程,就是通过系统的调度使几个具有不同功能的程序流即线程同时并行地运行。

在单处理器计算机系统中,实际上是不可能使多个线程真正并行运行的,而要通过系统用极短的时间、极快的速度对多个线程进行切换,宏观上形成多个线程并发执行的效果。

一、线程和进程机制上的差别

线程和进程很相像,都是程序的一个顺序执行序列,但两者又有区别。进程是一个实体,每个进程有自己独立的状态,并有自己的专用数据段,创建进程时,必须建立和复制其专用数据段;线程则互相共享数据段。同一个程序中的所有线程只有一个数据段,所以,创建线程时不必重新建立和复制数据段。由于数据段建立和复制这方面的差异,使线程的建立和线程间的切换速度大大优于进程;另一方面,线程又具备进程的大多数优点。

假设银行系统办理存款和取款手续,将账本看成数据段。如果按进程这种机制,那么,当储户去存/取款时,银行应先把账本复制一遍,为储户建立一个独立的账本再结算。如果按线程机制,那么,银行里所有的出纳员都用同一个账本。储户来办存/取款时,也从这个账本直接结算。用线程机制省去了数据段复制这一步,这显然是线程独具的特点。

由于多个线程共享一个数据段,所以,也出现了数据访问过程的互斥和同步问题。这使系统管理功能变得相对复杂。

总的来说,一个多线程系统在提高系统的输入/输出速度、有效利用系统资源、改善计算机通信功能以及发挥多处理器硬件功能方面显示了很大优势,因此,一些最新的操作系统如 Windows 95、Windows 98、Windows NT 等都提供了对多线程的支持。但是,在多线程操作系统下设计多线程的程序仍然是一个比较复杂和困难的工作。由于需要解决对数据段的共享,所以,原则上应该从程序设计角度采用加锁和释放措施,稍有不慎,便会使系统产生管理上的混乱。而 Java 从语言一级提供对多线程的支持,这样,可由语言和运行系统联合提供对共享数据段的管理功能和同步机制,使得多线程并行程序设计相对比较容易。

二、Java 线程的生命周期

每个线程都是和生命周期相联系的。一个生命周期含有多个状态,这些状态间可以互相转化。

Java 线程的生命周期可以分为 4 个状态:创建(new)状态、可运行(runnable)状态、不执行(not runnable)状态、消亡(dead)状态。

创建状态是指创建一个线程对应的对象的过程。Java 系统中,这些对象都是从 java .lang 包内一个称为 Thread 的类用关键字 new 创建的。刚创建的线程不能执行,必须向系统进行注册,分配必要的资源后才能进入可运行状态。这个步骤是由 start 操作完成的。而处于可运行状态的线程也未必一定处于运行中,有可能由于外部的 I/O 请求而处于不运行状态。进入消亡状态后,此线程就不再存在了。

一个线程创建之后,总是处于其生命周期的 4 个状态之一。线程的状态表明此线程当前正在进行的活动,而线程的状态是可以通过程序来进行控制的,就是说,可以对线程进行操作来改变状态。这些操作包括启动(start)、终止(stop)、睡眠(sleep)、挂起(suspend)、恢复(resume)、等待(wait)和通知(notify)。每一个操作都对应了一个方法,这些方法是由软件包 java .lang 提供的。通过各种操作,线程的 4 个状态之间可按图 1 .4 -1 所示进行转换。

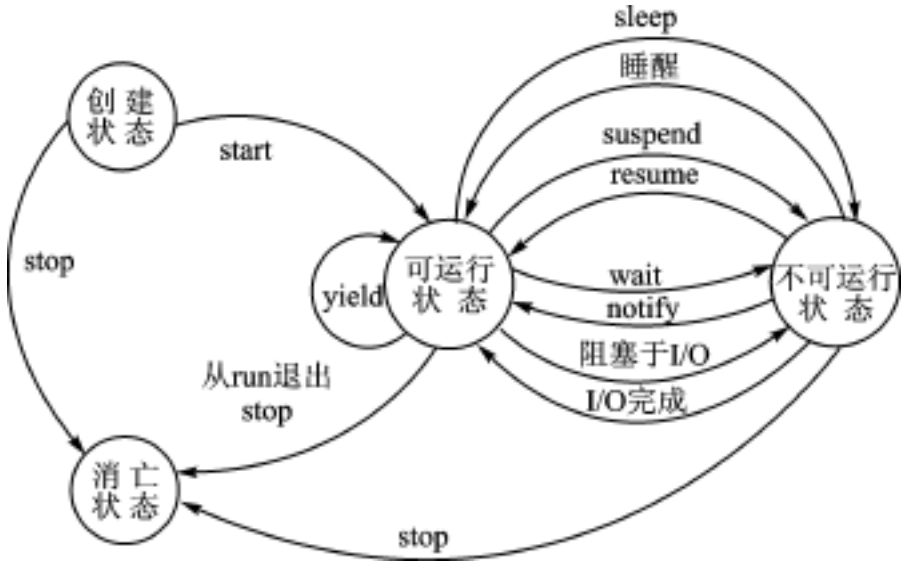


图 1 .4 -1 线程的状态转换

1 .创建状态

如果创建了一个线程而没有启动它,那么,此线程就处于创建(new)状态。比如,下述语句执行以后,使系统有了一个处于创建状态的线程 myThread:

```
Thread myThread = new MyThreadClass( )
```

其中,MyThreadClass()是 Thread 的子类,而 Thread 是由 Java 系统的 java .lang 软件包提供的。处于创建状态的线程还没有获得应有的资源,所以,这是一个空的线程。线程只有通过启动后,系统才会为它分配资源。对处于创建状态的线程可以进行两种操作:一是启动(start)操作,使其进入可运行状态;二是终止(stop)操作,使其进入消亡状态。如果进入消亡状态,那么,此后这个线程就不能进入其他状态,也就是说,它不复存在了。

start 方法是对应启动操作的方法,其具体功能是为线程分配必要的系统资源,将线程设置为可运行状态,从而可以使系统调度这个线程。

2. 可运行状态

如果对一个处于创建状态的线程进行启动操作,则此线程便进入可运行(runnable)状态。比如,用下列语句:

```
myThread .start( ) ;
```

则使线程 myThread 进入可运行状态。上述语句实质上是调用了线程体即 run( )方法。注意,run( )方法包含在 myThread 线程中,也就是先由 java .lang 包的 Thread 类将 run( )方法传递给子类 MyThreadClass( ),再通过创建线程由子类 MyThreadClass( )传递给线程 myThread。

线程处于可运行状态只说明它具备了运行条件,但可运行状态并不一定是运行状态。因为在单处理器系统中运行多线程程序,实际上在一个时间点只有一个线程在运行,而系统中往往有多个线程同时处于可运行状态,系统通过快速切换和调度使所有可运行线程共享处理器,造成宏观上的多线程并发运行。可见,一个线程是否处于运行状态,除了必须处于可运行状态外,还取决于系统的调度。

在可运行状态可以进行多种操作,最通常的是从 run( )方法正常退出而使线程结束,进入消亡状态。此外,还可以有如下操作:

挂起操作:通过调用 suspend 方法来实现;

睡眠操作:通过调用 sleep 方法来实现;

等待操作:通过调用 wait 方法来实现;

退让操作:通过调用 yield 方法来实现;

终止操作:通过调用 stop 方法来实现。

前面三种操作都会使一个处于可运行状态的线程进入不可运行状态。比如,仍以 myThread 线程为例,当其处于可运行状态后,再用如下语句:

```
myThred .sleep(5000) ;
```

则调用 sleep 方法使 myThread 线程睡眠 5 s(5 000 ms)。这 5 s 内,此线程不能被系统调度运行,只有过 5 s 后,myThread 线程才会醒来并自动回到可运行状态。

如果一个线程被执行挂起操作而转到不可运行状态,则必须通过调用恢复(resume)操作,才能使这个线程再回到可运行状态。

退让操作是使某个线程把 CPU 控制权提前转交给同级优先权的其他线程。

对可运行状态的线程也可以通过调用 stop 方法使其进入消亡状态。

3. 不可运行状态

不可运行(not runnable)状态都是由可运行状态转变来的。一个处于可运行状态的线程,如果遇到挂起(suspend)操作、睡眠(sleep)操作或者等待(wait)操作,就会进入不可运行状态。另外,如果一个线程是和 I/O 操作有关的,那么,在执行 I/O 指令时,由于外设速度远远低于处理器速度而使线程受到阻塞,从而进入不可运行状态。只有外设完成输入/输出之后,才会自动回到可运行状态。线程进入不可运行状态后,还可以再回到可运行状态,通常有三种途径使其恢复到可运行状态。

一是自动恢复。通过睡眠(sleep)操作进入不可运行状态的线程会在过了指定睡眠时间以后自动恢复到可运行状态;由于 I/O 阻塞而进入不可运行状态的线程在外设完成 I/O 操作后,自动恢复到可运行状态。

二是用恢复(resume)方法使其恢复。如果一个线程由于挂起(suspend)操作而从可运行状态进入不可运行状态,那么,必须用恢复(resume)操作使其再恢复到可运行状态。

三是用通知(notify 或 notifyAll)方法使其恢复。如果一个处于可运行状态的线程由于等待(wait)操作而转入不可运行状态,那么,必须通过调用 notify 方法或 notifyAll 方法才能使其恢复到可运行状态。采用等待操作往往是由于线程需要等待某个条件变量,当获得此条件变量后,便可由 notify 或 notifyAll 方法使线程恢复到可运行状态。

恢复到可运行状态的每一种途径都是有针对性的,不能交叉。比如,对由于阻塞而进入不可运行状态的线程采用恢复操作将是无效的。

在不可运行状态,也可由终止(stop)操作使其进入消亡状态。

4 . 消亡状态

一个线程可以由其他任何一个状态通过终止(stop)操作而进入消亡(dead)状态。线程一旦进入消亡状态,那它就不再存在了,所以也不可能再转到其他状态。

通常,在一个应用程序运行时,如果通过其他外部命令终止当前应用程序,那么就会调用 stop 方法终止线程;但是,最正常、最常见的途径是由于线程在可运行状态正常完成自身的任务而“寿终正寝”,从而进入消亡状态。这个完成任务的动作是由 run 方法实现的。

三、Java 线程的两种创建途径

一种途径是通过对 Thread 的继承来派生一个子类,再由此子类生成一个对象来实现线程的创建。这是比较简单直接的办法。Thread 类包含在系统 API 提供的 8 个软件包之一 java .lang中。Thread 类中包含了很多与线程有关的方法。其中,一个名为 run 的方法就是用来实现线程行为的。比如:

```
1  import java .lang .*                    // 引用 lang 包
2  class Mango extends Thread {
3     public void run( ) {
4              ……
5     }
6  }
```

上述程序段中,第 1 行语句引用软件包 lang。这样做是为了给编译器一个信息,从而使后面程序中有关 lang 包中的方法可直接用方法名,而不必带前缀“java .lang”。第 2 行语句是从 lang 包 Thread 派生一个子类 Mango,而这个子类中提供了 run 方法的实现。这样,运行时,将由子类 Mango 的 run 方法置换父类 Thread 的 run 方法。

不过这一步还没有创建线程,必须由子类生成一个对象,并且进行启动操作,这样才能得到一个处于可运行状态的线程。生成对象其实就是完成线程的创建,而启动是对已创建的线程进行操作。具体语句如下:

```
Mango t = new Mango( );
t .start( );
```

上面先用关键字 new 使线程进入创建状态,又调用 start( )方法使线程进入可运行状态。注意,start( )方法是由 Thread 继承给子类 Mango,然后又在生成对象时由对象 t 从类 Mango

得到的。

另一种途径是通过一个类去继承接口 Runnable 来实现线程的创建,而这个类必须提供 Runnable 接口中定义的方法 run()的实现。Runnable 是 java.lang 包中的一个接口,在 Runnable 接口中,只定义了一个抽象方法 run()。所以,如用这种途径来创建线程,则应先由一个类连接接口 Runnable,并且提供 run()方法的实现。比如,下面的程序段实现了与接口的连接。

```
1  public class xyz implements Runnable {
2  int i;
3  public void run() {
4          while (true) {
5             System.out.println("Hello"+   i++);
6          }
7      }
8  }
```

然后再创建一个线程:

```
Runnable r = new xyz();
Thread t = new Thread(r);
```

这种途径创建线程比第一种途径灵活。当一个类既需要继承一个父类,又要由此创建一个线程时,由于 Java 不支持多重继承,这样,用第一种途径将行不通,因为,按此思路创建线程也是以继承的方法实现的。于是,就需要一个类既继承 Thread 类,又继承另一个父类,而用接口方法却能实现这个目标。

四、线程的启动和终止

Thread 的 start()方法对应于启动操作,完成两方面的功能:一方面是为线程分配必要的资源,使线程处于可运行状态;另一方面是调用线程的 run()方法置换 Thread 中的 run()方法,或者置换 Runnable 中的 run()方法来运行线程。

使用 start()方法的语句很简单,即:

```
ThreadName.start();
```

下面的程序段先创建并启动线程 myThread,然后使用 sleep()方法让其睡眠 20 000 ms 即 20 s,使其处于不可运行状态;20 s 后,线程又自动恢复到可运行状态。

```
Thread MyThread = new MyThreadClass();
MyThread.start();
try {
   MyThread.sleep(20000);
 }
catch (InterruptedException e) { }
```

参 考 文 献

1 eter Van der Linden . Just Java-Java Series . 1996

2 Jerry R Jackson . Java by Example-Java Series . 1996

3 Gary Cornell . Core Java-Java Series . 1996

4 John A Pew . Instant Java-Java Series . 1996

选自《电子技术应用》月刊,2000 年第 4 期

1.5 嵌入式系统中的 JTAG 接口编程技术

中国科学院声学研究所(100080) 何希顺
清华大学自动化系(100084) 张 跃 何荣森

随着手持式电子设备的迅猛发展,手机、PDA、掌上电脑、电子书和数码相机等正在快步走进人们的日常生活。这一类电子产品同属于嵌入式系统的范畴,都是以高性能的微处理器为核心扩展相应的存储器和功能电路,运行小巧的操作系统和相应的应用程序,实现电子设备的各种功能。由于是手持设备,因而要求体积小、质量轻、耗电少。这些特点决定了设备内部的印制板的尺寸须比常规电子设备小得多,而且元器件密度大,双面贴装。这给设计人员带来了若干问题,如操作系统代码和应用程序的写入、板上芯片的测试等。本文作者结合实际工作,就嵌入式系统中如何通过 JTAG 接口进行 FLASH 芯片编程作了探讨。

一、几种常用的芯片编程方法

在嵌入式系统开发和产品生产过程中,对系统程序存储器编程主要使用三种编程方法:通过编程器编程、使用板上编程器编程和在系统编程。

1.通过编程器编程

这是在 PROM、EPROM、PAL 等芯片流行时常用的传统编程方法,即在可编程芯片焊装到电路板之前,使用专门的编程器对芯片进行代码或数据的写入,然后将已编程的芯片安装到电路板上。

使用编程器进行编程特别适合于 DIP 封装的芯片。如果是其他类型的封装,则必须使用相应的适配器。这种方法的缺点是需要手工进行待编程芯片的插入、锁定等工作,容易造成芯片方向错误、引脚错位等,导致编程效率降低。

2.使用板上编程器编程(OBP)

这种方法是在电路板上所有芯片已经焊装完毕后,再对板上的可编程芯片进行编程。通过专用电缆将电路板与外部计算机连接,由计算机的应用程序进行板上可编程芯片的代码或数据写入。芯片擦除和编程所需要的电源、控制信号、地址、数据和相关命令都由板外的编程控制器提供。在进行板上编程时,需要通过专门的辅助电路关断目标板上 CPU 的电源或将其外部接口信号设置为高阻状态,以免与编程时的地址、数据和控制信号发生冲突。

在板上编程可以克服芯片引脚错位、方向插反等问题,避免烧毁芯片、编程错误,保证了芯片编程的高成功率和可靠性。另一个优点就是及时软件升级,可以做到在产品出厂时系统使用最新版本的固化软件。这对于日新月异的手持电子设备而言是必须的。

这种方法的缺点是需要在电路板上设计编程用的接口、隔离等辅助电路,在编程时通过跳线或 FET 开关进行编程与正常工作的状态转换。这样会增加每个电路板芯片的数量,造成产品成本的增加。

3 . 在系统编程(ISP、ISW)

这种方法直接利用系统中带有 JTAG 接口的器件如 CPU、CPLD、FPGA 等,执行对系统程序存储器芯片内容的擦除和编程操作。一般而言,高档微处理器均带有 JTAG 接口,系统程序存储器的数据总线、地址总线和控制信号直接接在微处理器上。编程时,使用 PC 机内插卡或并行接口,通过专用电缆将系统电路板与 PC 机联系起来。在 PC 机上运行相关程序,将编程数据及控制信号传送到 JTAG 接口的芯片,利用相应指令从微处理器的引脚按照 FLASH 芯片的编程时序输出到 FLASH 存储器。

这种编程方法的条件是系统中必须存在带有 JTAG 接口或与之兼容的芯片如微处理器。优点是系统板上不需要增加其他与编程有关的辅助电路,减小了电路板的尺寸,避免了对微小封装芯片的手工处理,特别适用于电路板尺寸有严格限制的手持设备。

二、JTAG 接口介绍

面对复杂电路的设计、整板测试的难度及表面贴装技术带来的有限测试引脚等问题,业界不得不寻找一个标准加以解决。JTAG 边界扫描即 IEEE1149 .1 标准,定义了用于解决上述问题的硬件结构和工作机制。其优点在于将极其复杂的印刷电路板测试转变成具有良好结构性、可以通过软件简单而灵活处理。它虽然是一个主要用于片上电路的测试标准,但却打开了各种相关应用的大门。这个标准定义了可用于完成功能和互联测试以及内建自测过程的各种指令。芯片生产厂商如 ALTERA、XILINX、ATMEL、AMD、TI 等对标准进行了扩充,使用扩展的专用指令执行维护和诊断应用及对可配置器件的可编程算法,使 JTAG 接口广泛用于 FLASH 系列芯片的编程。概括起来,JTAG 接口主要应用于电路的边界扫描测试和可编程芯片的在系统编程。

1 . JTAG 接口的结构

在硬件结构上,JTAG 接口包括两部分:JTAG 端口和控制器。与 JTAG 接口兼容的器件可以是微处理器(MPU)、微控制器(MCU)、PLD、CPL、FPGA、ASIC 或其他符合 IEEE1149 .1 规范的芯片。IEEE1149 .1 标准中规定对应于数字集成电路芯片的每个引脚都设有一个移位寄存单元,称为边界扫描单元 BSC。它将 JTAG 电路与内核逻辑电路联系起来,同时隔离内核逻辑电路和芯片引脚。由集成电路的所有边界扫描单元构成边界扫描寄存器 BSR。边界扫描寄存器电路仅在进行 JTAG 测试时有效,在集成电路正常工作时无效,不影响集成电路的功能。具有 JTAG 接口的芯片内部结构如图 1 .5 - 1 所示。

测试逻辑的最高级电路包括以下 3 个主要模块。

(1) 测试访问端口(TAP)控制器

TAP 控制器提供对嵌入在 JTAG 兼容器件内部的测试功能电路的访问控制,是一个同步状态机。每个 JTAG 兼容的器件都有自己的 TAP 控制器。通过测试模式选择 TMS 和时钟信号 TCK 控制其状态转移,实现由 IEEE1149 .1 标准确定的测试逻辑电路的工作时序。

(2) 指令寄存器

指令寄存器是基于电路的移位寄存器,通过它可以串行输入执行各种操作的指令。

(3) 数据寄存器组

数据寄存器组是一组基于电路的移位寄存器。操作指令被串行装入由当前指令所选择的数据寄存器。随着操作的执行,测试结果被移出。

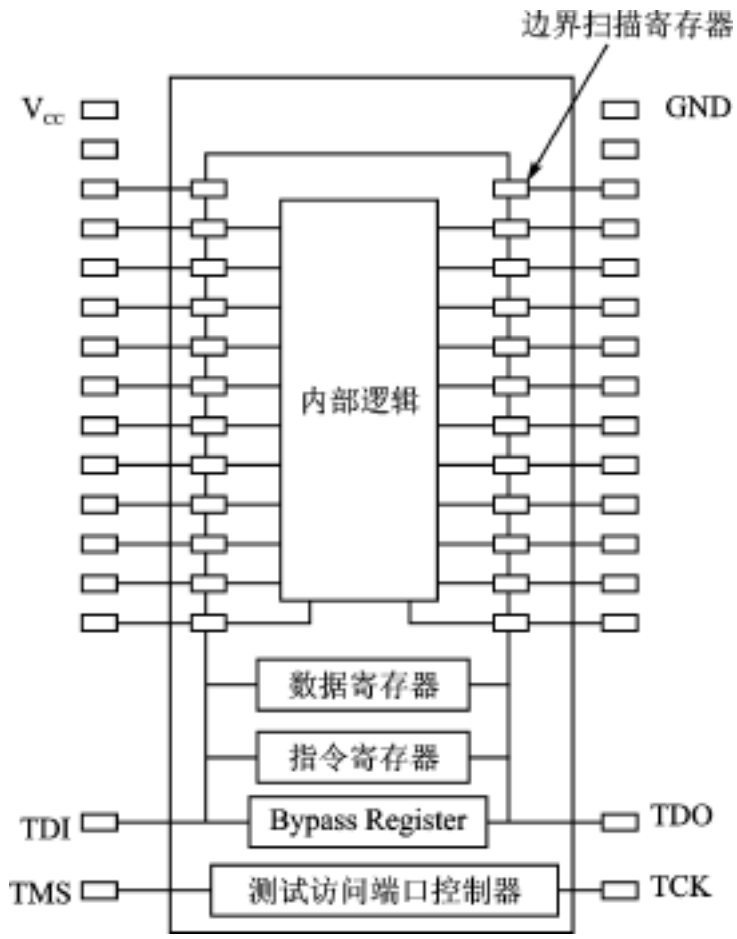


图 1.5-1 带有 JTAG 接口的芯片内部结构图

2. JTAG 引脚定义

JTAG 接口主要包括四个引脚:TMS、TCK、TDI 和 TDO 及一个可选配的引脚 TRST,用于驱动电路模块和控制执行规定的操作。各引脚的功能如下:

· TCK:JTAG 测试时钟,为 TAP 控制器和寄存器提供测试参考。在 TCK 的同步作用下,通过 TDI 和 TDO 引脚串行移入或移出数据及指令。同时,TCK 为 TAP 控制器状态机提供时钟。

· TMS:TAP 控制器的模式输入信号。TCK 的上升沿时刻 TMS 的状态确定 TAP 控制器即将进入的工作状态。通常 TMS 引脚具有内部上拉电阻,以保证该引脚在没有驱动时处于逻辑 1 状态。

· TDI:JTAG 指令和数据寄存器的串行数据输入端。TAP 控制器的当前状态以及保存在指令寄存器中的具体指令决定对于一个特定的操作由 TDI 装入哪个寄存器。在 TCK 的上升沿时刻,TDI 引脚状态被采样,结果送到 JTAG 寄存器组。

· TDO:JTAG 指令和数据寄存器的串行数据输出端。TAP 控制器的当前状态以及保持在指令寄存器中的具体指令决定对于一个特定的操作哪个寄存器的内容送到 TDO 输出。对于任何已知的操作,在 TDI 和 TDO 之间只能有一个寄存器(指令或数据)处于有效连接状态。TDO 在 TCK 的下降沿改变状态,并且只在数据通过器件移动过程中有效。该引脚在其他时间处于三态状态。

· TRST:测试复位输入信号,低电平有效,为 TAP 控制器提供异步初始化信号。

3. JTAG 测试访问端口(TAP)控制器

TAP 控制器是一个 16 状态的有限状态机,为 JTAG 提供控制逻辑,控制进入到 JTAG 结构中各种寄存器内数据的扫描与操作。TAP 状态转移图如图 1.5-2 所示,由 TCK 同步时钟

上升沿时刻 TMS 引脚的逻辑电平决定状态转移的过程(高电平 TMS = 1,低电平 TMS = 0)。对于由 TDI 端输入到器件的扫描信号共有两个状态变化路径:一个用于移入指令到指令寄存器;另一个用于移入数据到有效的数据寄存器,该寄存器由当前指令确定。

状态图中的每个状态都是通过 TAP 控制器进行数据处理所需要的。这些处理包括给引脚施加激励信号,捕获输入的数据,装载指令,边界扫描寄存器中数据的移入或移出。图 1.5-2表示了 TAP 状态机的基本流程,描述了从一个状态到另一个状态 TMS 信号的变化,在芯片 JTAG 接口的 TRST 引脚上加一低脉冲信号可以使 TAP 控制器复位到测试逻辑复位(Test-Logic-Reset)主状态。

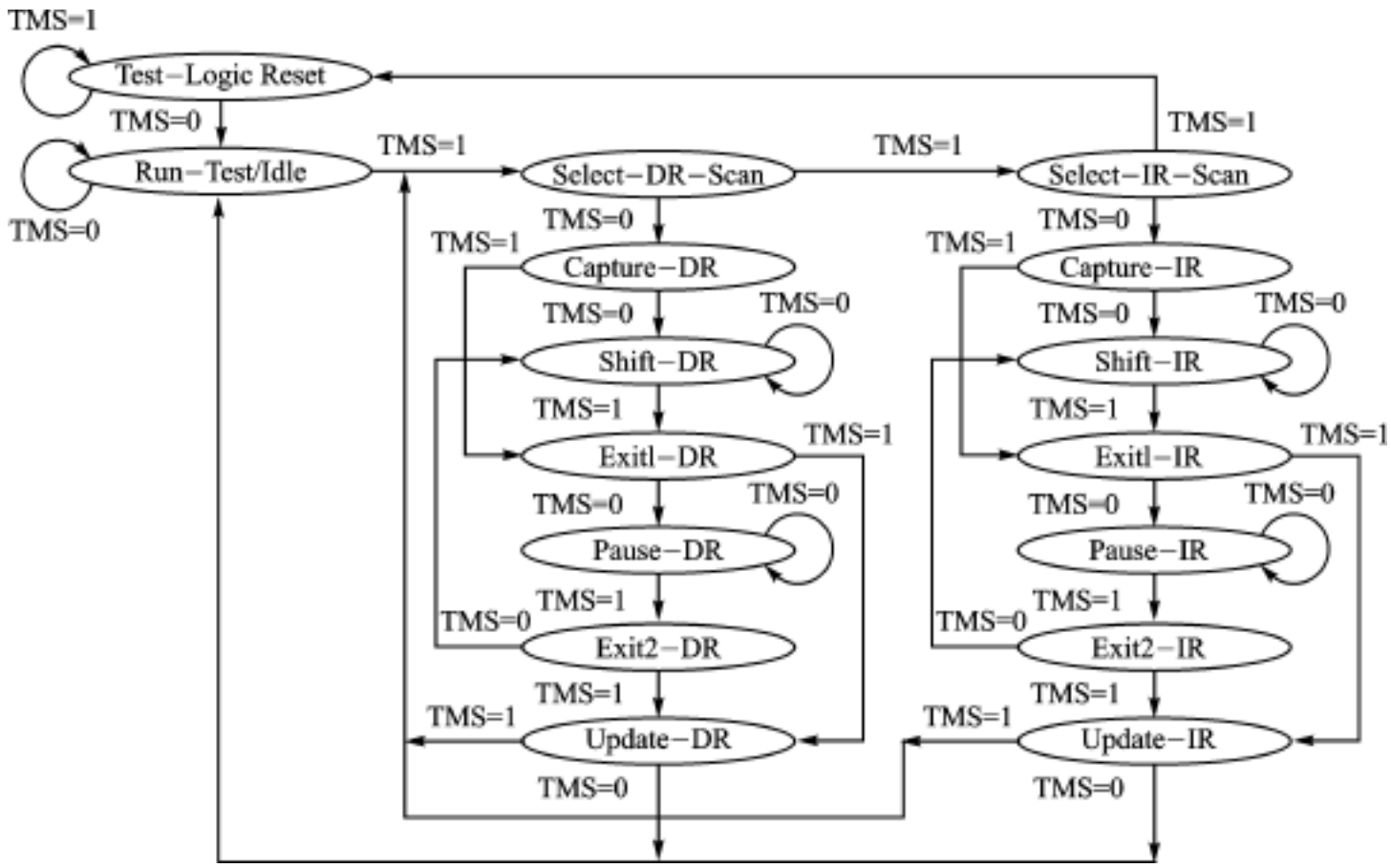


图 1.5-2 TAP 控制器状态转移图

4. JTAG 接口的控制指令

控制指令用于控制 JTAG 接口进行各种操作,控制指令包括基本指令和扩展指令。JTAG 接口标准要求芯片支持的基本指令有:EXTEST、INTEST、SAMPLE/ PRELOAD、BYPASS、IDCODE、HIGHZ。芯片厂商可以根据实际需要选择或添加扩展指令。

三、JTAG 接口的使用

通过 JTAG 接口可以进行电路板及芯片的测试,也可以实现对目标电路板上的程序存储器编程。本文仅讨论使用 JTAG 接口对板上 FLASH 存储器的编程。一般,可以利用专用的 PC 机内插卡式硬件控制器或独立的编程器访问 JTAG 器件,也可以直接由 PC 机的并行接口模拟 JTAG 时序,硬件控制器或编程器通过专用电缆连接到目标电路板。被编程的 FLASH 存储器芯片的地址线、数据线和控制信号线接到 JTAG 兼容芯片的相应引脚上。值得注意的是:采用这种编程方法,不要求 FLASH 器件具有 JTAG 接口,只要与其相连接的芯片具有 JTAG 接口即可。在编程 FLASH 芯片时,需要做的工作主要有: ① PC 机发送指令或数据到 JTAG 兼容芯片的边界扫描寄存器(BSR); ② 将保存在 BSR 中的指令或数据通过 JTAG 专

用指令传送给 FLASH 存储器。这个过程是由运行在 PC 机上的软件进行控制的。

1. 硬件配置

在某个设计项目中,使用了 Intel 公司的 StrongARM 芯片 SA-1110 和该公司的 Strata 系列 FLASH 存储器芯片。SA-1110 芯片是一种高性能、低功耗、集成有多种常用接口的 SOC 微处理器芯片,特别适合于手持设备。而 Strata FLASH 是 Intel 公司使用独创的 1 个存储单元记录 2 位数据技术制造的闪速存储器芯片,其特点是体积小、容量大、成本低,特别适合于程序代码与数据的存储。选择的型号为: E28F128J3A,可以配置成 8 位或 16 位数据线方式。SA-1110 为 32 位芯片,其外扩的程序存储器和数据存储器也为 32 位,因此程序存储器需要 2 片 28F128J3A 配置为 32 位形式,如图 1.5-3 所示。

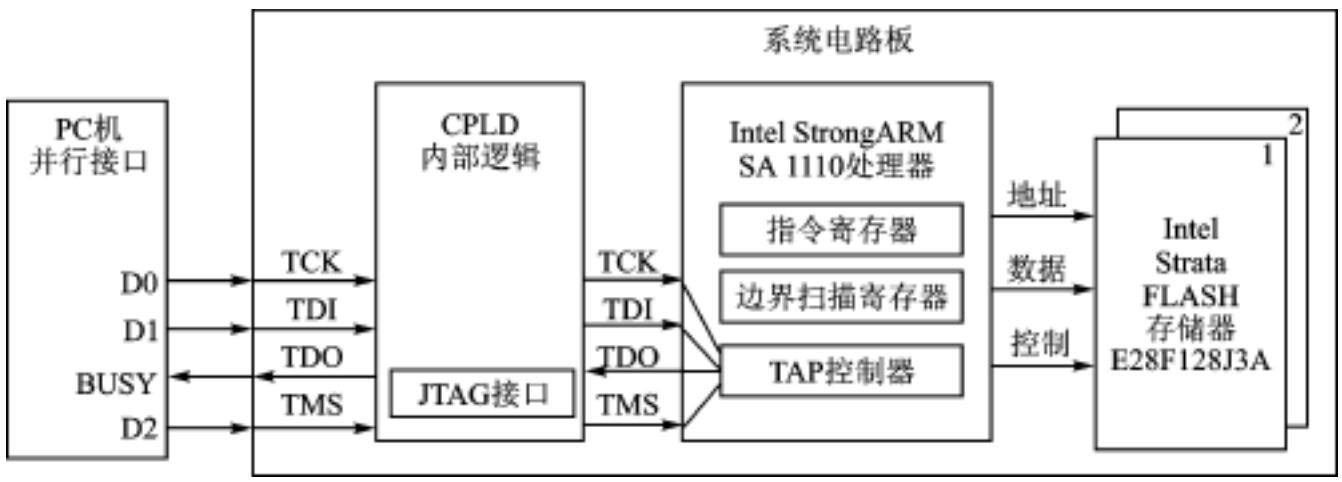


图 1.5-3 使用系统 CPU 的 JTAG 接口编程 FLASH 存储器

在本例中,JTAG 菊花链中包含两个 IEEE1149.1 兼容芯片,即 SA-1110 微处理器和 CPLD。由于 FLASH 的地址总线、数据总线和控制信号线接在 SA-1110 上,在利用 JTAG 接口编程 FLASH 存储器时,与 JTAG 链上的 CPLD 芯片无关,需要通过指令将 CPLD 芯片设为旁通模式。FLASH 芯片的控制信号如读信号($\overline{OE}$)、写信号($\overline{WR}$)和片选信号($\overline{CE}$)等直接由 SA-1110 产生。

从图 1.5-3 中可以看出,使用 PC 机并行接口的几个数据线和信号线来构成 JTAG 接口引脚信号,对应关系如表 1.5-1 所列。采用信号线直接连接的方法简便易行,只需要一条专用电缆即可操作 JTAG 接口。

表 1.5-1 PC 机并行接口与 JTAG 接口信号对应表

PC 机并行接口		对应 JTAG 接口	
引 脚	功 能	引 脚	功 能
2	D0	TCK	时钟
3	D1	TDI	数据输入
4	D2	TMS	模式选择
11	Busy	TDO	数据输出

2. 控制软件

SA-1110 芯片的 JTAG 接口实现了 IEEE1149.1 标准的部分功能,不能实现对芯片内部的测试及芯片仿真,但提供对芯片外的测试功能,可以用于对芯片外部电路的测试或编程。芯

片提供的 JTAG 指令包括:

· BYPASS(11111) 旁通片上系统逻辑指令,用于未被测试的芯片;

· EXTEST(00000) 片外电路测试指令,用于测试电路板上芯片之间的互联;

· SAMPLE/ PRELOAD(00001) 采样引脚/ 预加载数据指令,用于采样芯片引脚信号或通过加载数据控制引脚输出信号;

· IDCODE(00110) 读芯片识别码指令,用于识别电路板上的芯片;

· HIGHZ(00101) 设置高阻态指令,用于将芯片的引脚设为无效状态。

括号中的内容是指令的操作码,它们通过 TDI 引脚串行移入到指令寄存器。BYPASS 和 EXTEST 指令的操作码是 IEEE1149 .1 中规定的,因此对于所有的 JTAG 接口兼容芯片,这两个指令的操作码都是相同的。其他指令的操作码可以由芯片厂商根据实际定义。

结合待编程的 FLASH 存储器特征,利用上面提供的 JTAG 指令编写一个编程 FLASH 存储器的 PC 机应用程序,借助 SA-1110 芯片的 JTAG 接口将目标系统使用的操作系统和应用软件写入到 FLASH 存储器中。对目标板上的 FLASH 存储器进行编程时,在 PC 机上运行该程序来控制并行接口模拟 JTAG 时序并将编程代码传送到 SA-1110 的 JTAG 控制器。利用 JTAG 的边界扫描单元(BSC),把编程数据先移入到边界扫描寄存器(BSR);然后通过 JTAG 指令 EXTEST 按照 FLASH 芯片的编程时序将数据通过地址总线和数据总线写入 FLASH 存储器,实现芯片编程操作。在 FLASH 内容的写入过程中,程序对 2 片 FLASH 同时执行写操作,完成 32 位编程。在 PC 机上运行的编程操作程序框图如图 1 .5 - 4 所示,其中利用 PC 机并行接口实现 JTAG 接口信号的函数为:

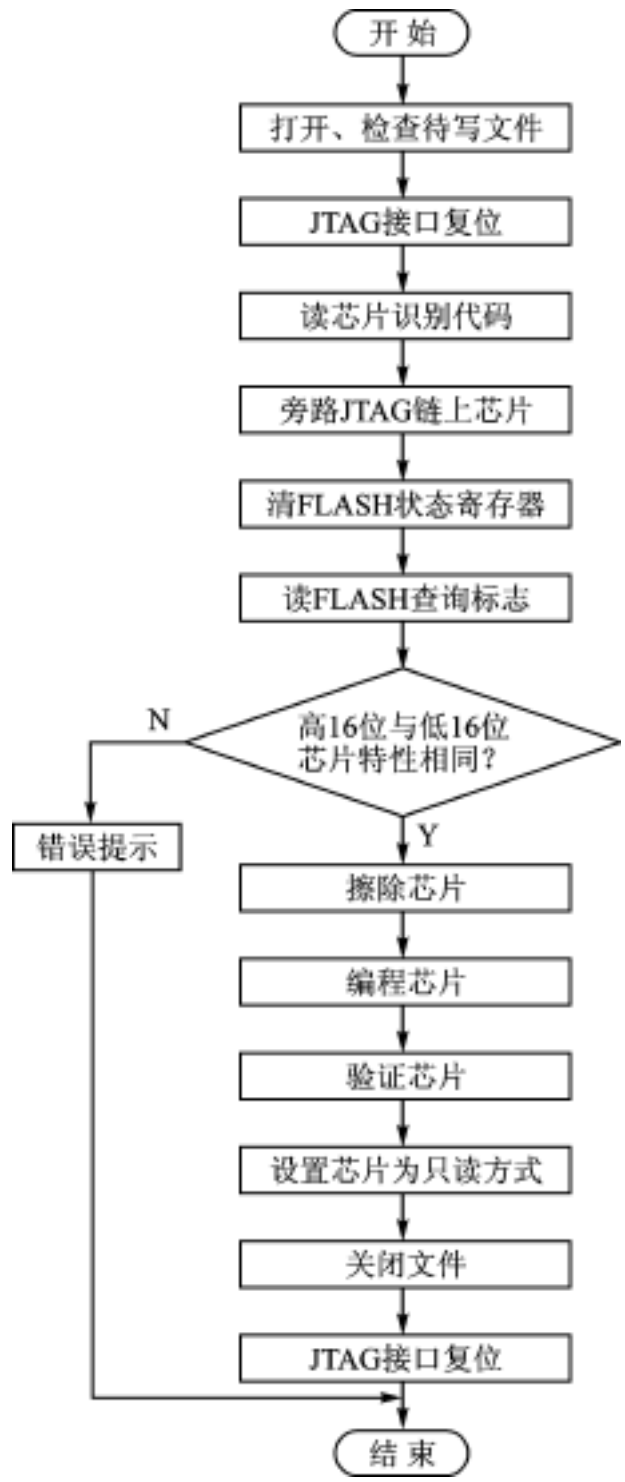


图 1 .5 - 4 编程 FLASH 存储器的 PC 机程序框图

```
int putp(int tdi, int tms, int rp)
{   / Output pins(LPT driving), LPT D0 Pin 2 and TCK,
   // LPT D1 Pin 3 and TDI, LPT D2 Pin 4 and TMS
   // Input pin(SA-1110 board drives), LPT Busy Pin 11 and TDO
      int tdo = - 1;
       outp(lpt_address, tms * 4 + tdi * 2);                    / TCK low
      _outp(lpt_address, tms * 4 + tdi * 2 + 1);                // TCK high
```

```
    if( rp = RP)_outp(lpt_address, tms * 4 + tdi * 2);        // TCK low
    if( rp = RP)tdo = !((int)_inp(lpt_address + 1)& 7);       // get TDO data
    return tdo;
}
```

通过 PC 机并行接口实现 SA-1110 的 JTAG 指令 EXTEST 的函数为:

```
void extest(void)
{   putp(1,0,IP);              / Run-Test/ Idle
    putp(1,0,IP);             // Run-Test/ Idle
    putp(1,0,IP);             // Run-Test/ Idle
    putp(1,0,IP);             // Run-Test/ Idle
    putp(1,1,IP);             // slect DR scan
    putp(1,1,IP);             // slect IR scan
    putp(1,0,IP);             // capture IR
    putp(1,0,IP);             // shift IR

    putp(0,0,IP);             // SA1110 Extest,指令长度为 5 位
    putp(0,0,IP);
    putp(0,0,IP);
    putp(0,0,IP);
    putp(0,0,IP);

    putp(1,0,IP);             // CPLD Bypass,指令长度为 4 位
    putp(1,0,IP);
    putp(1,0,IP);
    putp(1,1,IP);             //   xit 1-IR,操作码的最后一位必须通过时钟与下一状态 EXIT1_IR
                                有效处于同一时刻,由时钟控制 TMS 保持高电平时进入 EXIT1_IR
                                状态
    putp(1,1,IP);             // Update-IR
    putp(1,0,IP);             // Run-Test/ Idle
    putp(1,0,IP);             // Run-Test/ Idle
    putp(1,0,IP);             // Run-Test/ Idle
}
```

程序开始有四条语句:putp(1,0,IP)。**其意义在于无论** JTAG **状态机处于何种状态,经过这四条指令后,必将返回到** Run-Test/ Idle **状态,保证下面的操作从这一状态开始进入正常的操作状态。**

编程 FLASH **存储器使用的其他指令的实现方法与此类似。实践表明,使用** PC **机并行接口实现** JTAG **时序的方法是可行的,对** FLASH **存储器进行编程无需其他板卡设备支持,是一种较为简单的方法。**

使用 JTAG **接口对** FLASH **程序存储器进行编程的方法适合于系统中带有兼容** JTAG **接口的芯片。随着具有** JTAG **接口芯片应用的普及,需要对** JTAG **接口有深入的了解,以便更好地利用芯片的资源,设计出容易测试、便于维护与升级的高可靠性系统,延长产品的生命**

周期。同时,根据 FLASH 芯片及 JTAG 接口芯片的规范对使用 JTAG 接口进行编程的控制程序的优化,可以实现存储芯片的高速编程操作,对缩短产品的研发和生产周期,保证产品的上市时间非常有利。

参 考 文 献

1 ntel 公司 . Intel StrongARM SA1110 Microprocessor Advaced Developer s Manual . Dec . 1999

2 IEEE Std 1149 .1 Standard Test Access Port and Boundary-Scan Architecture, ISBN 1-55937-350-4(From IEEE, Inc, . 345 East 47th Street, New York, NY 100167, USA)

3 Kenneth P . Parker . The Boundary-Scan Handbook . pub lisher: Kluwer Academic Publishers

4 Intel 公司 . Designing for On-Board Programming Using the IEEE 1149 .1(JTAG) . November 1996

5 Xilinx 公司 . In-System Programming Using and Embedded Microcontroller . January 15 . 2001

选自《电子技术应用》月刊,2001 年第 12 期

1.6 EPAC 器件技术概述及应用

中国科技大学(230027) 朱东杰 张培仁 马 云

一、引 言

随着电子工业的快速发展,各类现场可编程逻辑器件大量出现。从 1985 年美国 Xilinx 公司生产出世界上第一片 FPGA(现场可编程逻辑阵列)器件开始,十几年来,这类可编程逻辑器件,无论在性能和集成度方面,还是在数量和种类方面都已取得了飞速的发展,极大地促进了电子及相关领域的发展。但模拟电路的设计,特别是涉及模拟集成电路和混合式/ 数字 ASIC(专用集成电路芯片)时,从来就是一项极具有挑战性的任务。

电子可编程模拟电路 EPAC(Electrically Programmable Analog Circuits)是美国 IMP 公司(国际微电子产品公司)研制的现场可编程门阵列器件 FPGA 的模拟对应产品。它通过提供功能互联以及电气特性方面的编程能力使得模拟电路的设计变得非常容易。

EPAC 具有集成度高、开发周期短及无风险等优点,能灵活实现模拟功能并可反复修改设计。在基于 Windows 的计算机上通过点击鼠标的方法,可在数小时内设计并实现模拟电路系统。EPAC 的三线串口可直接同 PC 或单片机接口,进行实时在线重新配置。由于 EPAC 器件在技术上、功能上、结构上具有突出的优点,芯片的设计具有决定意义的革新,因此从数千种产品中脱颖而出,在 1994 年被美国数家有权威的技术刊物评为年度最优秀产品。

二、EPAC 技术概述

1.工艺特征及技术特点

EPAC 器件是一种功能、性能和互联均可编程的模拟集成电路。

EPAC 技术的基础是混合信号 CMOS 工艺,其特点是使用片上 SRAM 和 E^2PROM 存储器。编程既可脱机又可实时进行,这意味着用户在保持原 E^2PROM 配置不变的同时进行高速在线变更(亦即在系统编程 ISP)。EPAC 器件是完全独立的,可以不需外部编程设备对其配置设定进行改变。第一型器件 IMP50E10 即形成了不使用附加外部元件的全功能模拟系统。

EPAC 技术将模拟设计任务从繁琐、易出错误的元件级提升到功能模块级。EPAC 开发与调试工具基于图形界面的 Analog Magic 开发软件,这一软件使用“结构校正”和所见即所得方法。

EPAC 器件具有一致的结构框架。框架中的关键部分包括串行接口、E^2PROM 存储器模块、“应用”部分和含有可编程模拟模块的功能电路部分。如果配置数据需要保密,那么可以通过设置保密位的方式进行锁定。

EPAC 器件与 FPGA 器件间的关键差别之一是积木式组件的特性。FPGA 的硬件只能获得许多相当低级的单元(标准宏单元、与或矩阵),再通过开发软件组成功能块;而 EPAC 则有高级单元(称为模块,器官级),以实现诸如 D/ A 变换、放大和比较等功能。这与 FPGA 是

大相径庭的。

EPAC 的可编程特性主要是指:每个功能模块内部功能的编程、各个功能模块之间连接的编程。通过上述工作,使 EPAC 组成系统整体功能。

图 1.6-1 所示为 IMP50E10 的简化框图。请注意芯片中输入模块(左边)、核心模块(中间)和输出模块(右边)的“信号流”。底部的“应用部分”包括串行接口、存储器模块、功率管理、振荡器、探测器及一个辅助运算放大器(AuxAmp)。

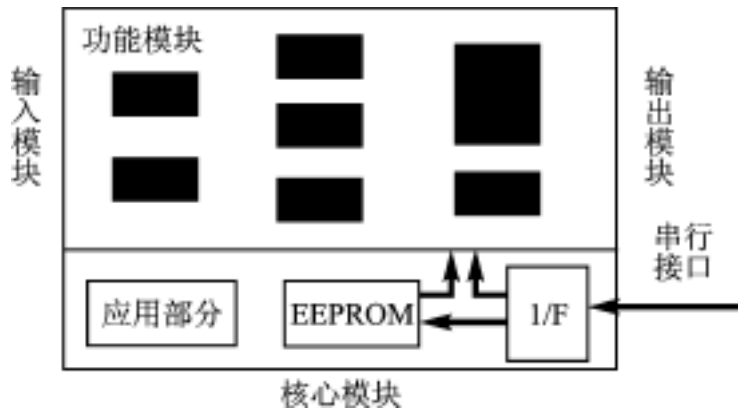


图 1.6-1 IMP50E10 简化框图

2.设计方法

如前所述,EPAC 的设计、调试由 Analog Magic 软件来完成。所见即所得的设计方法允许用户在数小时内完成 EPAC 器件的设计与调试工作。传统的模拟电路设计方法主要依靠硬件功能调试、功能块连接,最后重做电路板定型来完成。而 EPAC 主要依靠软件模拟、调试,最后形成下载代码、确定芯片内部连接来完成。

EPAC 具有缩短市场开发时间等种种技术上的明显优点。EPAC 可用于在数分钟内形成样机模型的场合,对小批量或大批量生产均能提供简单的低价格实现。如同 FPGA 一样,它是现场可编程的。

三、EPAC 应用实例:IMP50E10 在压力检测系统中的应用

在压力检测系统中,采用应变片作为传感器是常用的做法。传统的解决方案如图 1.6-2 所示。这种方案的缺点是:多个放大器的温度漂移、放大倍数一致性不好;多路选择器(内部的电子开关)的一致性难以保证;多路选择器内部的分布电容、控制信号的大小和各个部件间的连线影响检测的精度;各路模拟信号的滤波要求可能不一致,从而给设计带来一定的困难。

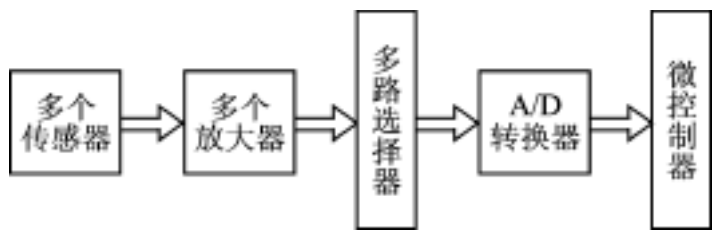


图 1.6-2 传统的信号采集系统框图

若采用 EPAC 器件,则系统将大大简化。各路传感器先经过多路选择器,再经过一组可编程放大器,输出至 A/D 转换器。其中多路选择器和可编程放大器由 EPAC 提供,并且可以克服传统做法的诸多缺点。我们将应变片放入温斯顿桥,并采用单片 44 引脚封装的 IMP50E10(最多可接 8 个桥电路)与微处理器相连接。系统框图如图 1.6-3 所示。如果使用

IMP50E30,则 ADC 也被包含在 EPAC 器件内。可以看出,两种方案在设计思想上的优劣是十分明显的。

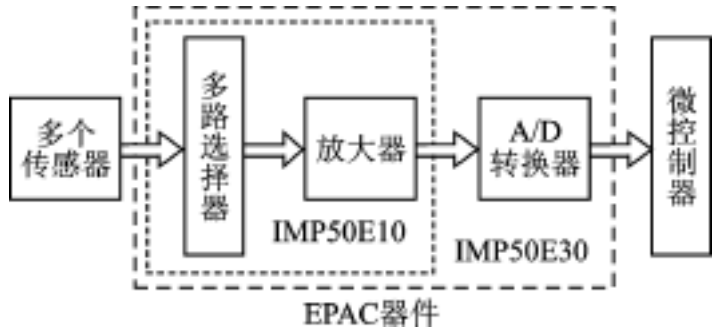


图 1.6-3 采用 EPAC 器件的信号检测系统框图

该系统实现的功能有:

- 提供桥电路的驱动电源,向桥电路提供的激励可以改变,以增加桥电路的灵敏度;
- 对各个传感器提供的信号提供相同或不同的滤波和增益;
- 提供跟踪/保持电路;
- 提供看门狗电路;
- 对增益值和自动归零灵敏度进行再编程,以补偿老化效应的影响;
- 可添加校正与诊断功能。

由于多路选择器与放大器是集成在 EPAC 中,多路选择器的控制信号由片内提供,且其分布电容相对减小(集成度比较高),同时可以对不同的回路提供不同的滤波电路,这使得系统的精度大为提高;同时由于采用公用的放大器,放大器的一致性自然得到保证,系统误差也相应减少。此外,系统提供传感器的电压驱动,使得传感器无须再配电源。整个系统的功耗也大为降低。

图 1.6-4 为系统的详细连接图。

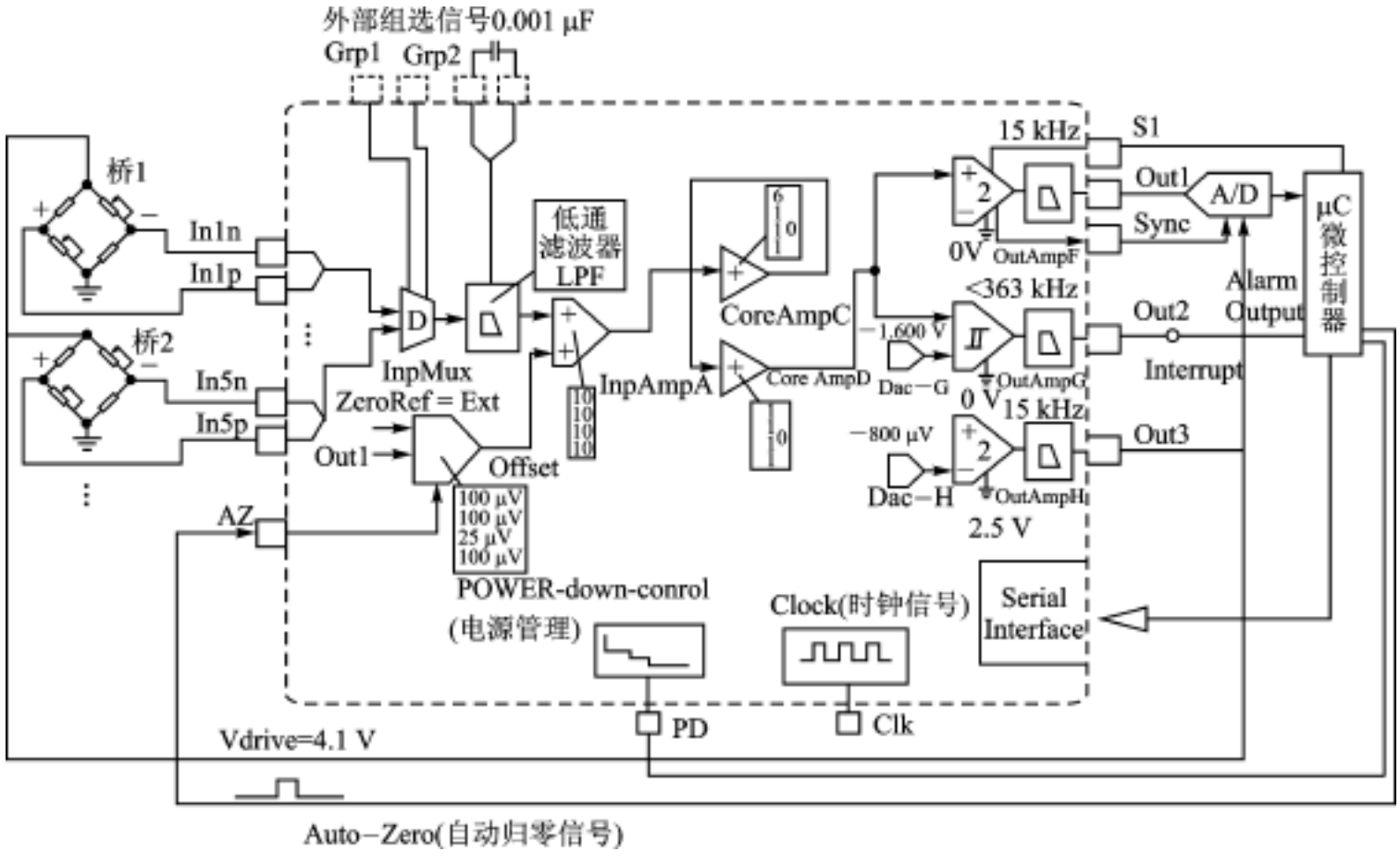


图 1.6-4 系统连接图

来自桥电路的信号进入“输入多路选择器模块”(InpMux)。为了在各个输入通道之间作出选择,可以通过改变组选引脚“Grp 1”和“Grp 2”上的 TTL 电平进行切换,也可以通过串口发出通道选择命令进行输入通道的切换。

低通滤波模块(LPF)含有内部电容(可以选择接通或断开),可以提供 15 kHz 的滤波。如果在外部电容引脚“Ca”、“Cb”上接入外部电容,则可以获得从 845 kHz 至接近零的滤波带宽。

通过在引脚“Ca”和“Cb”间使用一个 0.001 μF 的外接电容 Cext,低通滤波器模块(LPF)可将输入信号的带宽(fc)限制在 1 kHz 内,Cext 的计算用如下公式:

$$Cext = 1/89\,233 \times fc$$

输入放大器 A(InpAmpA)将来自桥电路的信号与设置成具有自动归零功能的偏移模块引入的偏移校正相加,然后再将相加结果乘以增益 10。

输出模块 F(OutAmpF)具有 2 倍增益并能对进入 A/D 变换器的信号进行缓冲。跟踪/保持特性的控制是通过微控制器或微处理器(μP)向引脚“S1”发出一保持指令信号而实现的。一旦输出信号被保持,引脚“Sync”发出信号启动 A/D 开始转换。

桥电路的 4.1 V 电压驱动是通过将 OutAmpH 设置为 2.5 V 的虚拟地而获得的。此外,Dac-H 被设置到 - 800 μV,它可提供 + 4.1 V 的输出。

输出模块 G(OutAmpG)用于提供看门狗功能,其作用是当它的输入上升到超过 1.6 V 时中断 μP。通过使用外部归零功能选项,桥电路和 IMP50E10 二者的初始失调及其漂移引起的误差均可置零。通过在每一次测量前进行的自动归零处理,由失调和共模移位引入的所有误差均可被消除到输入折合电平为 100 μV 范围之内。

从微控制器到“PD”引脚的一个控制信号用于使 IMP50E10 进入静止模式。功率消耗降至小于 70 μA。提供给桥电路的功率降至零。这样可以在不需测量桥电路输出信号的时候限制功率,以降低整体功耗。

当掉电模式仅用于 IMP50E10 中的选定模块时,只有必须使用看门狗功能的电路维持加电状态。当看门狗电路检测到某一情况后,其输出用于唤醒 IMP50E10 中的其余部分,如模数转换器和微控制器等部分。

四、小 结

通过 EPAC 器件 IMP50E10 在压力检测系统的成功应用,我们可以看到:EPAC 器件对于混合式电路的设计来说,是一套强有力的设计工具和开发系统,在线可重复配置与可编程性使 EPAC 的设计思想得以在更高层次上发挥作用。

结合我们的实际使用经验,我们认为 EPAC 器件在以下方面会使我们使用者受益。

(1) 设计容易

基于 Windows 界面的开发系统使没有集成电路或元件级模拟电路设计经验的用户可以很快投入设计工作。这种基于 EPAC 的试验校正法既无冒险也无代价。

(2) 调试容易

EPAC 器件带有片内全可编程探测模块(Magic Probe),允许开发者监测内部的模拟数字信号。当该集成电路芯片在系统中运行时,其探测模块由 Analog Magic 控制。拥有微处理器的系统可以使用这一模块进行在线系统测试和监测。

(3) 校准容易

在许多应用中需要进行增益放大或偏移校准。如果使用一次性编程技术,或由分离元件与标准集成电路芯片构成的电路,这一过程相当繁琐,且经常校准也是不大容易现实的。相反,使用可重复编程技术使其变得非常容易。

(4) 保护设计方案

设计方案保护常常是保持市场领先的关键。基于标准元件的电路容易复制,而 EPAC 器件则极难译码。保密位的设置可有效防止回读配置数据,而打开封装将破坏所有的存储器内容。

正由于 EPAC 器件有如上之优点,并且有精度高,速度快,简单易实现的特点。相信其必将在电子测量领域中有着广泛的应用前景并发挥更大的作用!

参 考 文 献

1 lectrically Programmable Analog Circuit Design Handbook . Imp, 1994

2 武汉力源电子股份有限公司 .EPAC 可编程模拟器件使用手册 .1996

3 薛宏熙,等 .MACH 可编程逻辑器件极其开发工具(第二版) 北京:清华大学出版社,1998

选自《电子测量技术》双月刊,2001 年第 2 期

1.7 VHDL 设计中电路简化问题的探讨

深圳大学 EDA 技术中心(518060) 朱明程 林欣荣

近年来,随着集成电路技术的发展,用传统的方法进行芯片或系统设计已不能满足要求,迫切需要提高设计效率。在这样的技术背景下,能大大降低设计难度的 VHDL 设计方法正越来越广泛地被采用;但是 VHDL 设计是行为级的设计,所带来的问题是设计者的设计思考与电路结构相脱节。设计者主要是根据 VHDL 的语法规则,对系统目标的逻辑行为进行描述,然后通过综合工具进行电路结构的综合、编译、优化,通过仿真工具进行逻辑功能仿真和系统时延的仿真。实际设计过程中,由于每个工程师对语言规则、电路行为的理解程度不同,每个人的编程风格不同,往往同样的系统功能,描述的方式是不一样的,综合出来的电路结构更是大相径庭。因此,即使最后综合出的电路都能实现相同的逻辑功能,其电路的复杂程度和时延特性都会有很大的差别,甚至某些臃肿的电路还会产生难以预料的问题。从这个问题出发,我们就很有必要深入讨论在 VHDL 设计中如何简化电路结构,优化电路设计的问题。

一、描述方法对电路结构的影响

用 VHDL 进行设计,其最终综合出的电路的复杂程度除取决于设计要求实现的功能的难度外,还受设计工程师对电路的描述方法和对设计的规划水平的影响。最常见的使电路复杂化的原因之一是设计中存在许多本不必要的类似 LATCH 的结构。而且由于这些结构通常都由大量的触发器组成,不仅使电路更复杂,工作速度降低,而且由于时序配合的原因而导致不可预料的结果。例如对于同一译码电路有不同 VHDL 描述:

```
1:  F INDEX = "0000" THEN
    STEPSIZE < = "0000111";
    ELSIF INDEX = "0001" THEN
    STEPSIZE < = "0001000";
    LSIF INDEX = "0010" THEN
    STEPSIZE < = "0001001";
    ……
    ELSE
        STEPSIZE < = "0000000";
    END IF;
2:  TEPSIZE < = 0000111" WHEN INDEX = "0000" ELSE
                "0001000" WHEN INDEX = "0001" ELSE
                "0001001" WHEN INDEX = "0010" ELSE
    ……
                    "0000000";
```

以上两段程序描述了同一个译码电路。第二段程序由于 WHEN…ELSE 的语句不能生

成锁存器的结构且 ELSE 后一定要有结果,所以不会有问题;而第一个程序如果不加 ELSE STEPSIZE < =“0000000”这句,则会生成一个含有 7 位寄存器的结构。虽然都能实现相同的译码功能,但是电路复杂度会大增。由于每位工程师的写作习惯不同,有的喜欢用 IF…ELSE 的语句,有的喜欢用 WHEN…ELSE 的方式,而用 IF…ELSE 时,如稍不注意,在描述不需要寄存器的电路时没加 ELSE,则会引起电路不必要的开销。所以在 VHDL 设计中要慎用 IF…ELSE 这类能描述自身值代入的语句。

二、设计规划的优劣直接影响电路结构

另一引起电路复杂化的主要原因是对设计规划得不合理。虽然 VHDL 语言能从行为描述生成电路,但一个完整的设计一般来说都不可能由直接描述设计的目标功能来实现,总要把设计分成若干部分,每一部分再分别描述其行为。这就涉及如何划分功能模块的问题,要求对设计了解得较深入,才能使划分更有效,才能降低电路的复杂程度。例如对于设计一个时钟源为 1 kHz,每 32 s 发出一组信号(共八组)的简单的控制器来说,有下面两种实现方法。

(1) 用 15 位的计数器实现把输入 1 kHz 的时钟分频为 1/32 Hz,然后用这个作为时钟驱动一个 3 位的计数器,这个计数器的八个状态分别通过一个 3-8 译码器发出所要求的信号。

(2) 直接用 18 位的计数器把输入的 1 kHz 时钟进行分频,再利用计数器的八个相距 32 s 的状态来推动一个 12-8 译码器来实现。

对于如此的设计要求,VHDL 程序分别如下所示:

1.第一种设计方法的 VHDL 源程序

```
process(clk,cclk,count2)
begin
if(clk = '1' and clk'event)then
   count2 < = count2 + 1;
if(count2 = "000000000000000")then
      cclk < = '1';
   else
      cclk < = '0';
   end if;
end if;
end process;

process(cclk,count3,ctemp)
begin
if( clk = '1' and cclk'event)then
   count3 < = count3 + 1;
   if( ount3 = "000")then
      ctemp < = "00000001";
   elsif(count3 = "001")then
      ctemp < = "00000010";
   elsif(count3 = "010")then
```

```
        ctemp <= "00000100";
    elsif(count3 = "011") then
        ctemp <= "00001000";
    elsif(count3 = "100") then
        ctemp <= "00010000";
    elsif(count3 = "101") then
        ctemp <= "00100000";
    elsif(count3 = "110") then
        ctemp <= "01000000";
    elsif(count3 = "111") then
        ctemp <= "10000000";
    else
        ctemp <= "00000000";
    end if;
end if;
end process;
```

2 . **第二种设计方法的** VHDL **源程序**

```
procss(clk,ctemp,count)
begin
if( lk = '1' and clk'event) then
    count <= count + 1;
    if( ount = "0000000000000000") then
        ctemp <= "00000001";
    elsif(count = "0100000000000000") then
    ctemp <= "00000010";
    elsif(count = "1000000000000000") then
        ctemp <= "00000100";
    elsif(count = "1100000000000000") then
        ctemp <= "00001000";
    elsif(count = "0000000000000000") then
        ctemp <= "00010000";
    elsif(count = "0100000000000000") then
        ctemp <= "00100000";
    elsif(count = "1000000000000000") then
        ctemp <= "01000000";
    elsif(count = "1100000000000000") then
        ctemp <= "10000000";
    end if;
end if;
end process;
```

对于第一种的程序可以综合出的电路如图 1 .7 - 1 **所示。**

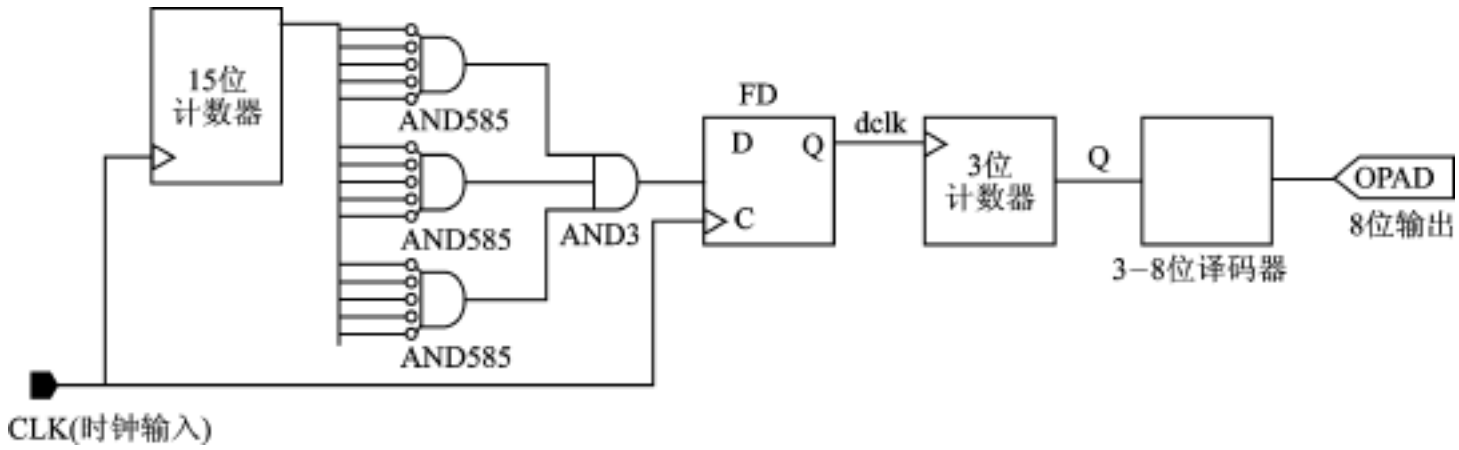


图 1.7-1　第一种程序综合出的电路

该电路用一个 15 位的加法器和寄存器组成一个 15 位的计数器。在计数器记完一周回到“000000000000000”时,通过后面的 15 输入的与非门和 1 位的触发器就可以实现同步进行 215 次分频,同步输出 32 Hz 的时钟 CCLK。CCLK 再驱动一 8 位的移位寄存器,便可实现每 32 s 输出一信号。

而用第二种的程序设计综合出的电路如图 1.7-2 所示。

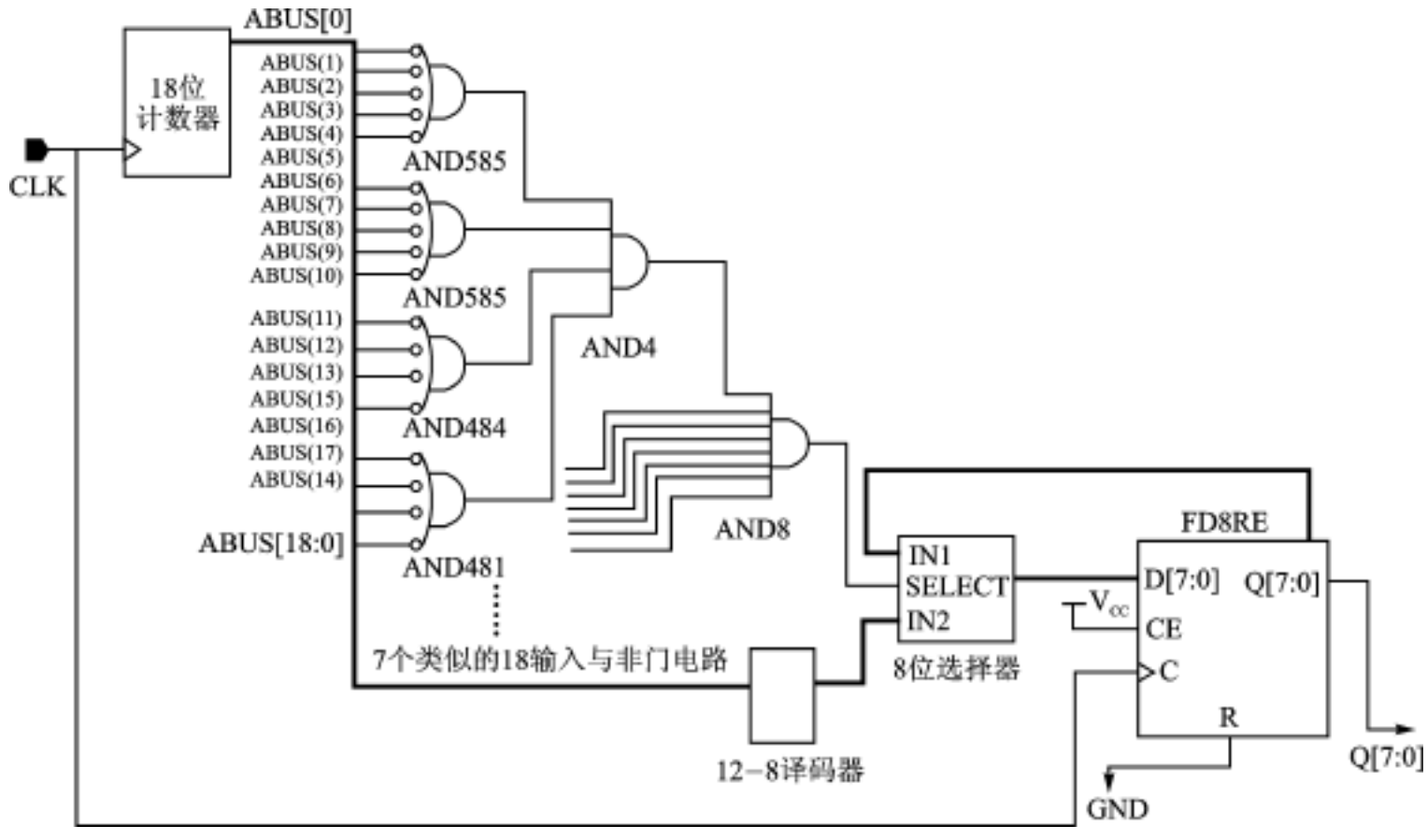


图 1.7-2　由第二种程序综合出的电路

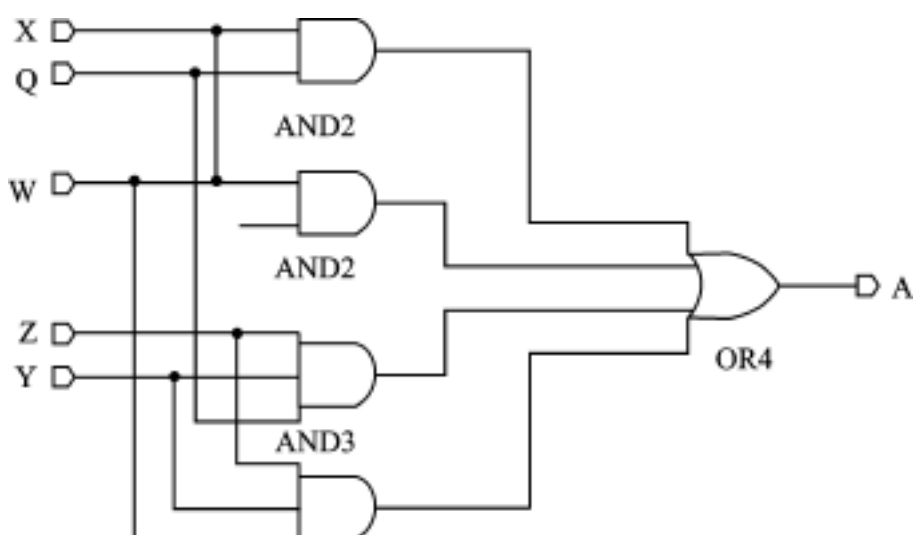
图 1.7-2 所示的电路用一个 18 位的加法器和寄存器组成一个 18 位的计数器。后接了 8 个 18 输入的逻辑门和 8 输入的或门。输入的 1 kHz 时钟经过计数器被分频,其中有 8 个相隔 32 Hz 的计数状态,逻辑门就负责把这 8 个状态译码成所需的 8 组信号。译码后的数据通过选择器输出到 8 位的触发器,以实现同步输出。还有个锁存器,用来保持输出信号不变,在 8 个状态中的从一个状态变到下一个之前,保持前一个的数值。选择器当逻辑门输出新的数据时让其输出数据通过,在新数据到来之前输出锁存器的数据。

以上两种方法都能实现相同的逻辑功能,但图 1.7-1 所示的方法由于运用了较少位数的

计数器,所用的逻辑门也较简单,而且还少用了多路选择器和锁存器资源,所以综合出来的电路较简单。以 XILINX Spartan S05-3 芯片为例:第一种方法占用芯片 CLB 的 12%,其中 FMAPS 为 9%,最高工作速度为 82 MHz;而第二种方法占用了 15%的 CLB,FMAPS 占用 15%,最高工作速度只有 69 .9 MHz。在这一个简单的设计之中就能省 20%的电路,提高 12 .1 MHz的工作速度。由此可见科学的划分设计对降低电路复杂程度的重要意义。

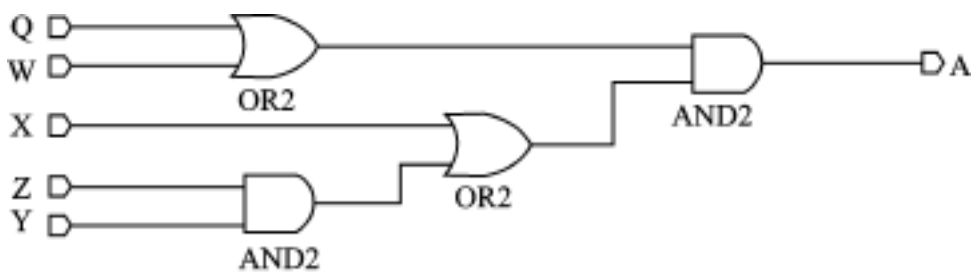
三、逻辑设计对电路结构的影响

还有一个使电路复杂化的原因是逻辑电路的输入项太多,以致需占用过多的面积。我们从图 1 .7 - 3 和图 1 .7 - 4 两个相同功能的逻辑电路和它们对应的 VHDL 描述来分析。




```
A < = (X AND Q)
OR(X AND W)
OR(Y AND Z AND Q)
OR(W AND Y AND Z);
```

图 1 .7 - 3 采用 2 级逻辑门及其逻辑描述




```
A<=B AND C;
B<=X OR (Y AND Z);
C<=Q OR W;
```

图 1 .7 - 4 采用 3 级逻辑门及其逻辑描述

比较两图可知,图 1 .7 - 3 是 2 级逻辑门,每个输入信号与不只一个逻辑门相连;图 1 .7 - 4 是 3 级的逻辑门,每个输入信号只与一逻辑门相连。由于级数少,延时也较小,因此图 1 .7 - 3 的速度要比图 1 .7 - 4 快。然而,由于图 1 .7 - 3 的输入项要比图 1 .7 - 4 多得多(10:5),因此,占用的面积必然也比图 1 .7 - 4 大。图 1 .7 - 4 是图 1 .7 - 3 通过提取公因数(例中是 B 和 C)得来的。这是把附加的中间项加到结构描述中去的一种过程,它使输入到输出中的逻辑级数增加,牺牲速度换来电路占用面积的减小。对于对延时要求不高的情况,采用这种方法分解逻辑

电路以达到降低电路复杂度的目的。

通过以上简单、初步的探讨,我们可以知道,用 VHDL 进行集成电路的设计,牵涉到对 VHDL 语言的使用方法和对设计的理解程度。本文讨论了以下简化和优化电路设计的 3 个值得注意的方面。

(1) 在用 VHDL 进行设计中要注意避免不必要的寄存器描述。

(2) 在编写程序前要先对整个设计进行较深入的了解,科学地划分设计,多设想几种方案,再进行比较,用多个较少位数的单元取代较多位数的单元。

(3) 在时延要求不高的情况下,可提取逻辑电路公因子,把它分解成含有中间变量的多级电路。

参 考 文 献

1 朱明程,孙普译.可编程逻辑系统的 VHDL 设计技术.南京:东南大学出版社,1998

2 侯伯亨,顾新编著.VHDL 硬件描述语言与数字逻辑电路设计.西安:西安电子科技大学出版社,1997

3 沈绪榜,杜敏.VLSI 设计导论.北京:高等教育出版社,1994

选自《电子技术应用》月刊,2000 年第 8 期

1.8 8031 芯片主要模块的 VHDL 描述与仿真

南京航空航天大学电子系(210016) 王成华 吕 勇

一、前 言

VHDL 语言是美国国防部开发的一种硬件描述语言,自问世以来,因其突出的优点而成为 IEEE 的一种工业标准,是惟一已被公认的 HDL 语言。

本文根据 8031 芯片的功能特点,将 8031 芯片划分为便于使用 VHDL 语言描述的几个功能模块,并分析各模块的结构以及使用 VHDL 语言进行描述,最后进行模块的仿真及分析。8031 芯片应用广泛,进行 8031 芯片各模块的描述与仿真是进行更复杂数字系统分析与设计的基础。

二、总体结构分析

本文采用自上至下的设计方法,首先分析 8031 芯片的整体结构。根据 8031 芯片的结构及功能特点,将其分成以下几个模块:

(1) 并行数据接口,包括 P0、P1、P2、P3 四个端口;
(2) 运算器;
(3) 程序计数器 PC;
(4) 堆栈指针 SP 及片内数据存储器 RAM;
(5) 特殊寄存器 SFR 及其控制电路;
(6) 控制器;
(7) 定时/ 计数器;
(8) 中断控制器;
(9) 片内时钟发生电路;
(10) 串行数据接口。

整个系统的结构框图如图 1.8-1 所示。

三、运算器的设计

1. 运算器结构设计

8031 芯片的运算器由两个暂存器、ALU、一个布尔处理机及一些控制逻辑构成。布尔处理机结构与 ALU 类似,主要完成位操作。其核心部分为 ALU,ALU 采用微指令的方法设计,即每种运算均有一个过程与之对应,根据不同的指令调用不同的过程来完成不同的运算。运算器的结构框图如图 1.8-2 所示。图中 ALU 的运算结果为 16 位,经选择器选择输出高 8 位或低 8 位。运算指令来自控制器,ALU 根据不同指令进行不同的运算操作,其操作数为两个暂存器的内容、ldtmp 1 和 ldtmp 2 为控制两个暂存器锁存的信号,暂存器的输入来自于数

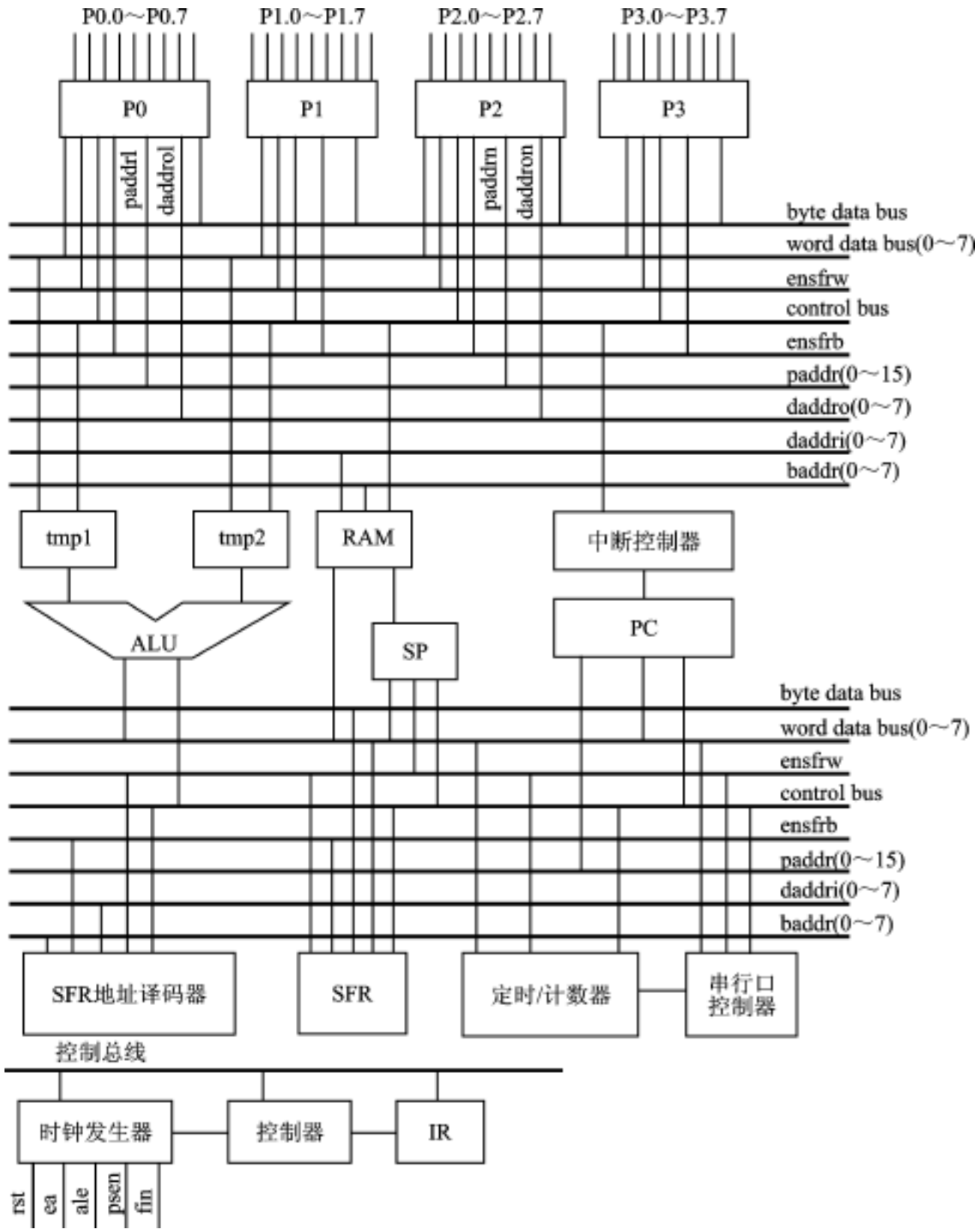


图 1 .8－1　8031 芯片整体结构框图

据总线上的数据。布尔处理机的输入分别来自进位标志位 C 和位数据总线。它根据来自控制器的位指令进行不同的位运算,结果输出到位数据总线。

2 . 运算器的 VHDL 语言描述

现根据以上结构分析,用 VHDL 语言描述运算器。在其结构图中,寄存器、数据选择器和三态门的描述较简单。本节仅给出核心部件 ALU 的 VHDL 描述。

先编制一个程序包,其中包括 ALU 中运算对应的各个进程,该程序包的部分代码如下 :

```
package alu_package is
        procedure add(------);
        ------
```

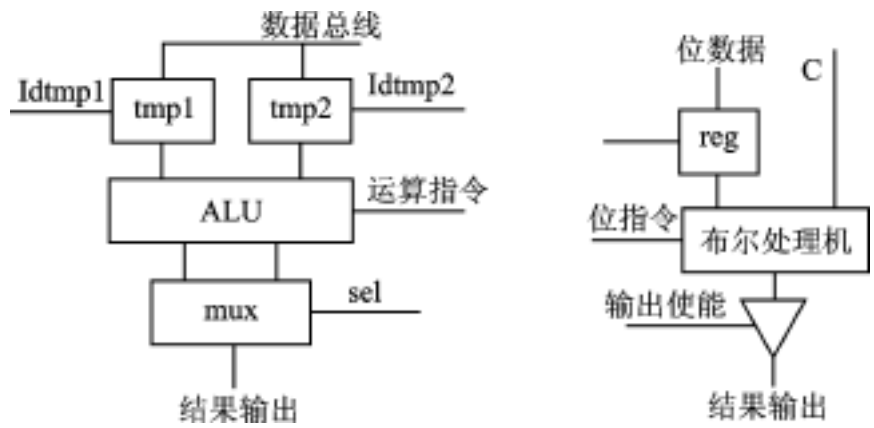


图 1.8-2 运算器结构图

```
        procedure dec(------);
end alu_package;
package body alu_package is
    procedure addc(a,b:in std_logic_vector(7 downto 0);
                   result:out std_logic_vector(15 downto 0);
                   c:in std_logic;
                   ov,cy,ac:out std_logic)is
        variable c7,cy_v:std_logic;
        begin
            cy_v:=(a(7) and b(7)) or ((a(7) or b(7)) and (a(6) and b(6))) or ((a(7) or
b(7)) and (a(6) or b(6)) and (a(5) and b(5))) or ((a(7) or b(7)) and (a(6) or
b(6)) and (a(5) or b(5)) and (a(4) and b(4))) or ((a(7) or b(7)) and (a(6) or
b(6)) and (a(5) or b(5)) and (a(4) or b(4)) and (a(3) and b(3))) or ((a(7) or
b(7)) and (a(6) or b(6)) and (a(5) or b(5)) and (a(4) or b(4)) and (a(3) or b(3))
and (a(2) and b(2))) or ((a(7) or b(7)) and (a(6) or b(6)) and (a(5) or b(5)) and
(a(4) or b(4)) and (a(3) or b(3)) and (a(2) or b(2)) and (a(1) and b(1))) or
((a(7) or b(7)) and (a(6) or b(6)) and (a(5) or b(5)) and (a(4) or b(4)) and
(a(3) or b(3)) and (a(2) or b(2)) and (a(1) or b(1)) and (a(0) and b(0))) or
((a(7) or b(7)) and (a(6) or b(6)) and (a(5) or b(5)) and (a(4) or b(4)) and
(a(3) or b(3)) and (a(2) or b(2)) and (a(1) or b(1)) and (a(0) or b(0)) and c);
            ac:=(a(3) and b(3)) or ((a(3) or b(3)) and (a(2) and b(2))) or ((a(3) or b(3))
and (a(2) or b(2)) and (a(1) and b(1))) or ((a(3) or b(3)) and (a(2) or b(2)) and
(a(1) or b(1)) and (a(0) and b(0))) or ((a(3) or b(3)) and (a(2) or b(2)) and (a
(1) or b(1)) and (a(0) or b(0)) and c);
            c7:=(a(6) and b(6)) or ((a(6) or b(6)) and (a(5) and b(5))) or ((a(6) or b(6))
and (a(5) or b(5)) and (a(4) and b(4))) or ((a(6) or b(6)) and (a(5) or b(5)) and
(a(4) or b(4)) and (a(3) and b(3))) or ((a(6) or b(6)) and (a(5) or b(5)) and (a
(4) or b(4)) and (a(3) or b(3)) and (a(2) and b(2))) or ((a(6) or b(6)) and (a(5)
or b(5)) and (a(4) or b(4)) and (a(3) or b(3)) and (a(2) or b(2)) and (a(1) and b
(1))) or ((a(6) or b(6)) and (a(5) or b(5)) and (a(4) or b(4)) and (a(3) or b(3))
and (a(2) or b(2)) and (a(1) or b(1)) and (a(0) and b(0))) or ((a(6) or b(6)) and
(a(5) or b(5)) and (a(4) or b(4)) and (a(3) or b(3)) and (a(2) or b(2)) and (a(1)
or b(1)) and (a(0) or b(0)) and c);
```

```
        ov: = cy_v xor c7;
        cy: = cy_v;
        result: = a + b + c;
end addc;
procedure inc ------;
procedure dec ------;
procedure subc ------;
procedure mul ------;
procedure div( ,b:in std_logic_vector(7 downto 0);
                result:out std_logic_vector(15 downto 0);
                cy,ov,ac:out std_logic)is
variable a1,b1:integer;
begin
        a1: = conv_integer(a);
        b1: = conv_integer(b & "0000000");
        for i in 0 to 7 loop
                if(a1-b1 < 0)then
                        a1: = a1 * 2;
                else
                        a1: = (a1 - b1) * 2 + 1;
                        end if;
                end loop;
                result: = conv_std_logic_vector(a1,16);
                cy: = '0';
                if(b = "00000000")then
                        ov: = '1';
                        else
                        ov: = '0';
                end if;
                ac: = '0';
        end div;
        procedure daa ------;
        procedure rla ------;
        procedure rra ------;
        procedure rlc ------;
        procedure rrc ------;
        procedure swap ------;
        procedure cpla ------;
        procedure anl ------;
        procedure orl ------;
        procedure xrl ------;
end alu_package;
```

有了这个程序包,即可编制 ALU 的 VHDL 代码如下:

```
library ieee;
use ieee .std_logic_1164 .all;
library work;
use work .alu_package .all;
entity alu is
        port(a,b:in std_logic_vector(7 downto 0);
                alui:in std_logic_vector(4 downto 0);   --alu instruction
                c,inac:in std_logic;
                rst:in std_logic;
                cy,ac,ov:out std_logic;
                result:out std_logic_vector(15 downto 0));
end alu;
architecture rtl of alu is
        begin
                process(a,b,alui,rst)
                variable result_v:std_logic_vector(15 downto 0);
                variable cy_v,ac_v,ov_v:std_logic;
                begin
                        if(rst = '1')then
                        result < = "0000000000000000";
                        cy < = '0';
                        ov < = '0';
                        ac < = '0';
                        elsif(alui = "00000")then
                        add ------
                        ------
                        elsif(alui = "10000")then
                        dec ------
                        else
                        ------
                        end if;
                end process;
end rtl;
```

在程序中,ALU 根据不同的指令,调用不同的过程,通过参数传递,完成各运算功能。

3 .运算器模块的仿真及波形分析

为验证以上描述的正确性,需进行波形仿真。ALU 模块的仿真波形如图 1 .8 - 3 所示。该仿真波形是用 MAX + PLUSII 的仿真器进行仿真的结果,复位信号 rst 有效时,结果 result 为 0;运算指令为“01”时,做带进位加运算;运算指令为“04”时,做除法运算;运算指令为“03”时,做乘法运算;运算指令为“05”时,做十进制调整;运算指令为“09”时,做带进位循环右移;运算指令为“08”时,做带进位循环左移;运算指令为“0C”时,做“与”运算;运算指令为“0B”时,做

加 1 运算。从仿真波形中可看出,对 ALU 模块的描述是正确的。

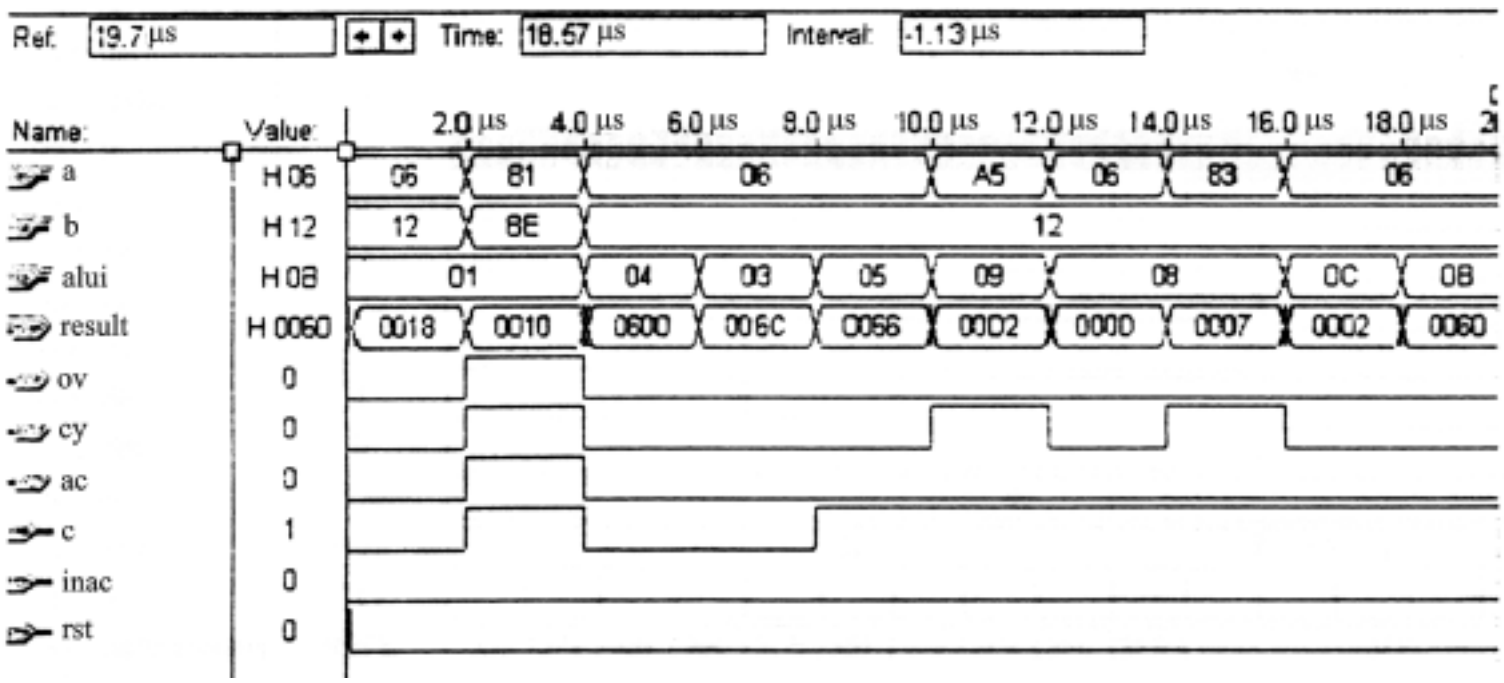


图 1 8－3　ALU 模块仿真波形图

四、串行数据接口设计

8031 芯片的串行口为可编程全双工的串行通信接口。它有四种工作方式,这里仅以方式 1为例说明这种数据接口的描述方法。

1 .串行口结构分析

串行口工作在方式 1 时,其发送和接收缓冲器均为 10 位,其移位时钟来自定时器 T1 的溢出率。为便于设计,将其分成接收控制、发送控制、时钟控制和移位寄存器等几个模块分别描述。串行口的结构框图如图 1 8－4 所示。图中,采样器判断 rxd 的值,fin 为输入时钟,txd 为串行数据输出,输入移位寄存器和输出移位寄存器地址相同,tb8、ren 均来自控制串行口的特殊功能寄存器。接收控制和发送控制分别控制输入移位寄存器和输出移位寄存器的工作。

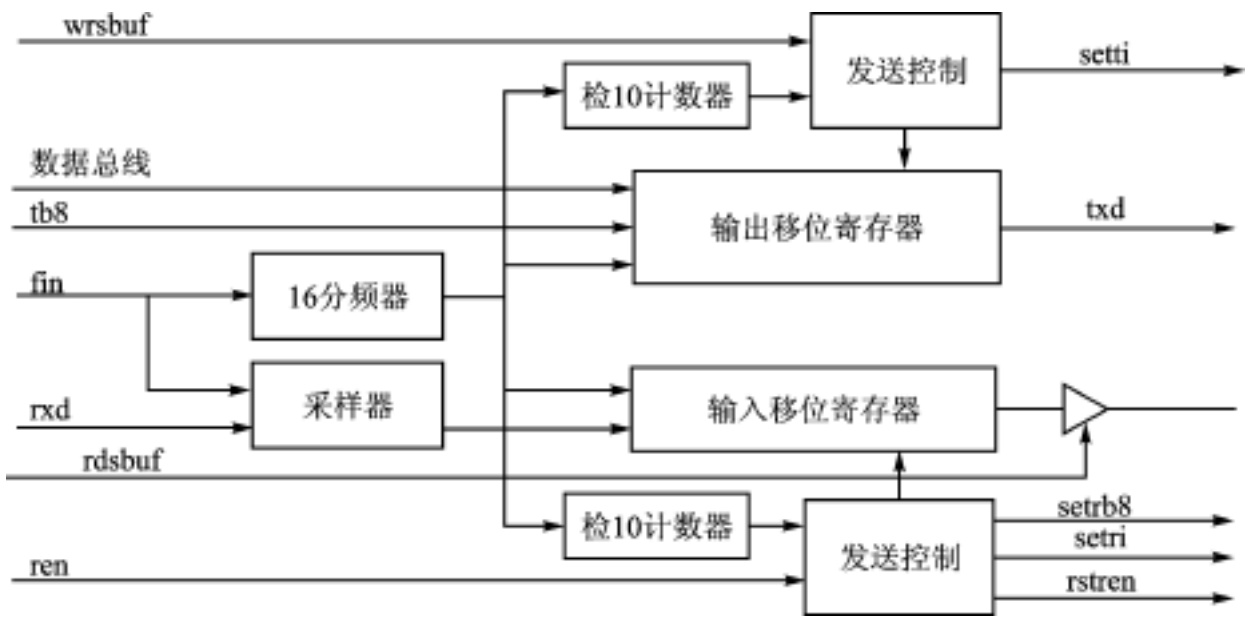


图 1 8－4　串行口结构框图

2．串行口仿真波形

串行口的仿真波形如图 1.8－5 所示。图中，wrsbuf 的上跳沿将 din 上的数据置入发送移位寄存器的 1～8 位，将 tb8 置入其第 9 位，‘0’置入其第 0 位，同时触发一次发送过程。由图中可看出，txd 上的数据为“0100111001”，第 1 位为起始位‘0’，最后一位为 tb8 上的数据，中间 8 位为 din 上的数据，发送过程正确。ren 等于 1 且 rxd 上出现下跳沿时启动一次接收过程。由图中可看出 rxd 上的数据为“0100111111”，第 1 位为起始位，最后一位为停止位，中间位为数据，其值为 F9H。经过一次接收过程后，由 rdsbuf 信号读到 dout 上的数据正好也为 F9H，说明接收过程也正确。

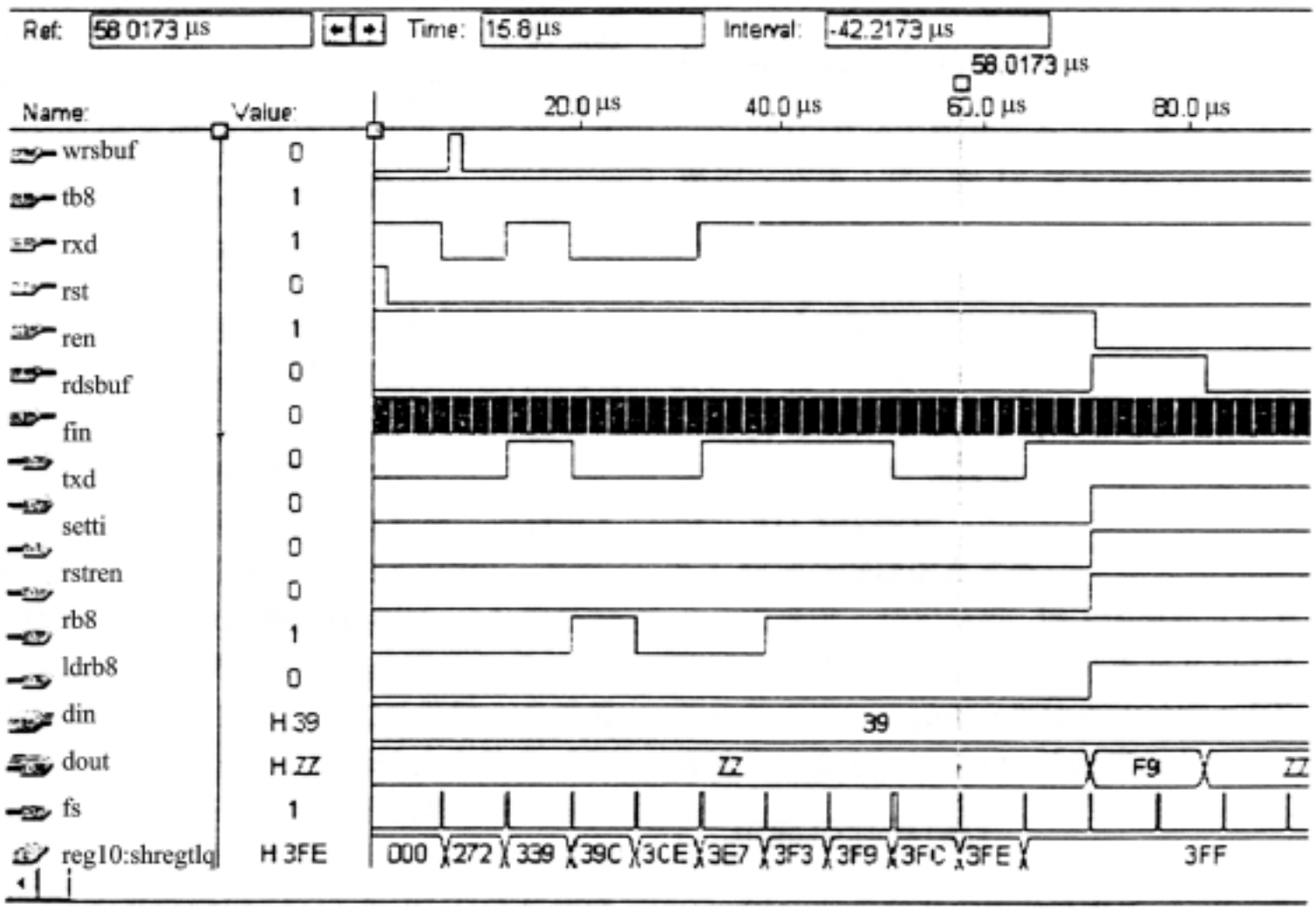


图 1.8－5 串行口仿真波形图

五、并行数据接口设计

8031 芯片有四个 8 位并行数据接口，它们的结构基本类似，此处以 P3 口为例说明接口的设计方法。

1．并行数据接口结构分析

P3 口可用作通用 I/O 口和第二功能输入/输出使用，其结构框图如图 1.8－6 所示。图中，寄存器为特殊功能寄存器 P3，set 信号为 P3 口用作第二功能使用时的置位信号。P3 口用作通用 I/O 口时，第二功能输入自动置 1。t 为方向选择信号，选择输出或输入；p3.i 信号为双向信号；w/b 信号选择位/字操作；di、bin 为字数据和位数据；wr 为写寄存器信号；rst 为复位信号；enp3 为 P3 口读写使能信号；enp3.i 为 P3 口第 i 位读写使能信号。

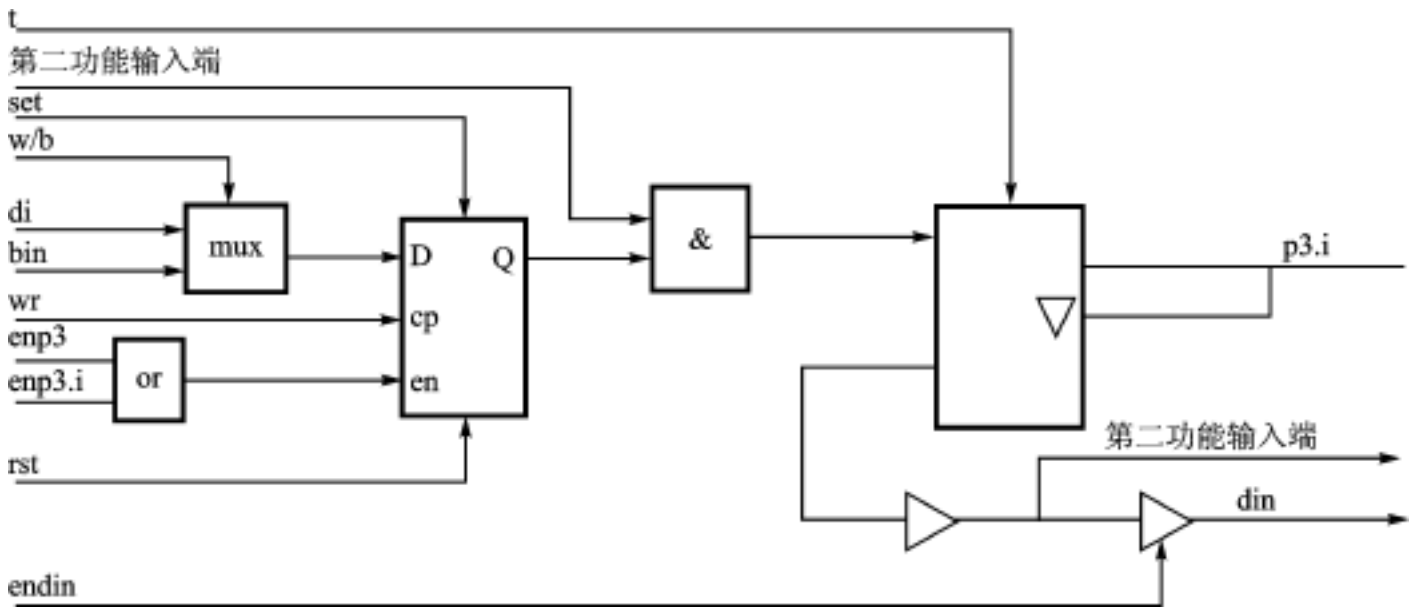


图1.8-6 P3口某位结构框图

2.并行数据接口仿真波形

根据图1.8-6结构框图不难采用VHDL语言来描述P3口。该端口的仿真波形如图1.8-7所示。其中p3reg为特殊功能寄存器P3的输出。由波形可以看出,该端口的描述是正确的。

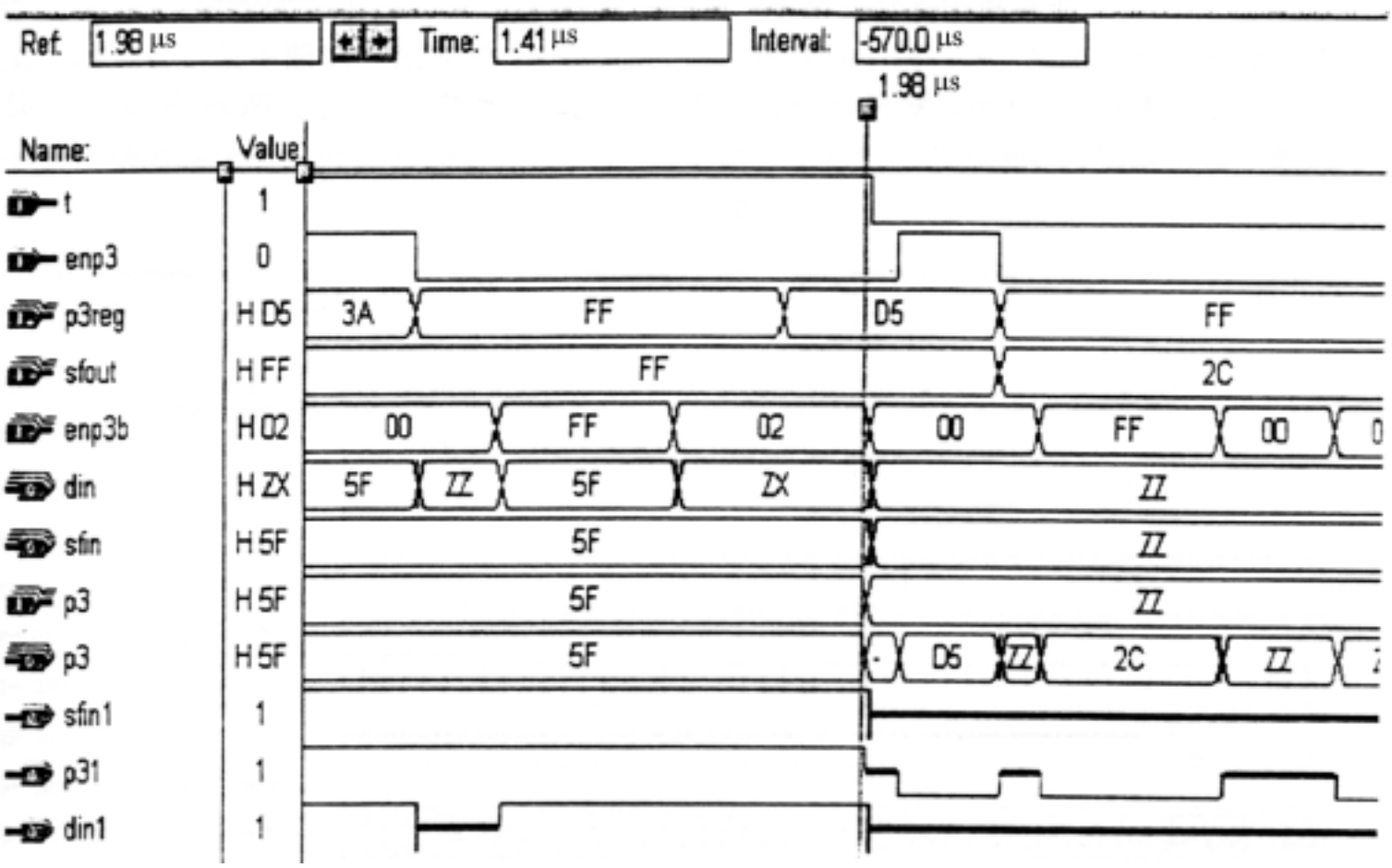


图1.8-7 P3口仿真波形图

六、片内数据存储器的设计

8031芯片内部有一个128字节的片内数据存储器,其中有16字节的区域可位寻址,也可字寻址。

1.片内数据存储器结构分析

该模块的结构框图如图1.8-8所示。图中A0～A7为字地址,B0～B7为位地址,ENB为位操作使能,D为8位的字数据,B为位数据,WR为写信号,RST为复位信号,RD为读信号。

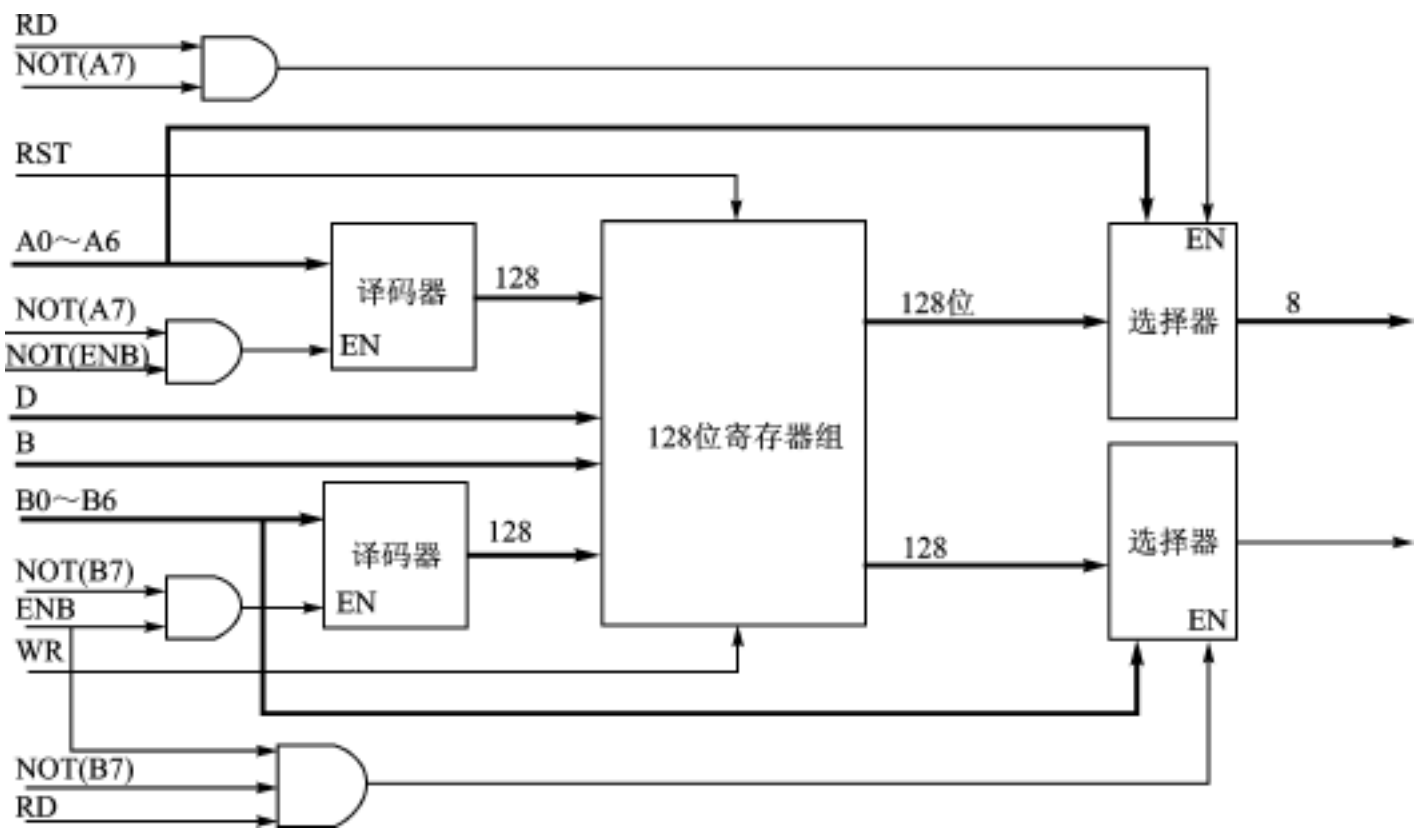


图1.8-8 数据存储器结构框图

2.数据存储器仿真波形

根据以上结构分析,描述该模块并仿真的波形如图1.8-9所示。该波形为对RAM中某些单元进行写操作后再读出的过程。由仿真波形可看出,该RAM功能正确。

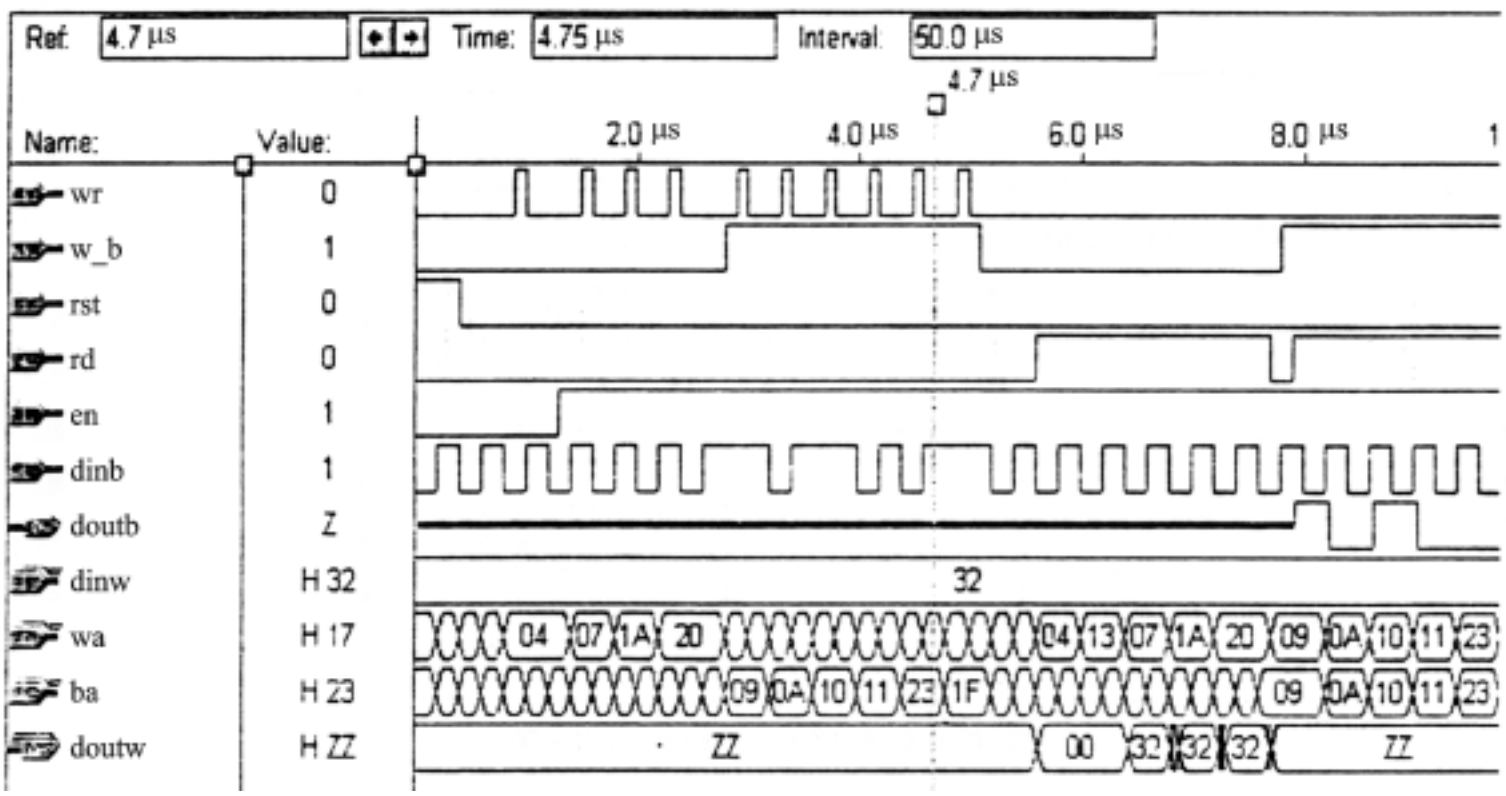


图1.8-9 数据存储器仿真波形

七、中断控制器设计

8031 芯片有 5 个中断请求源,每个中断请求源可以编程为高优先级中断或低优先级中断,可以实现二级中断服务程序的嵌套。

1 .中断控制器结构分析

根据中断控制器的功能将其分成中断允许控制 IE、中断优先级控制 IP、中断矢量发生器 IVA 及中断响应控制 INTA 几个模块,其结构框图如图 1 .8 - 10 所示。其中 PUSHPC 信号使 PC 入栈,CLRI 清除中断标志。

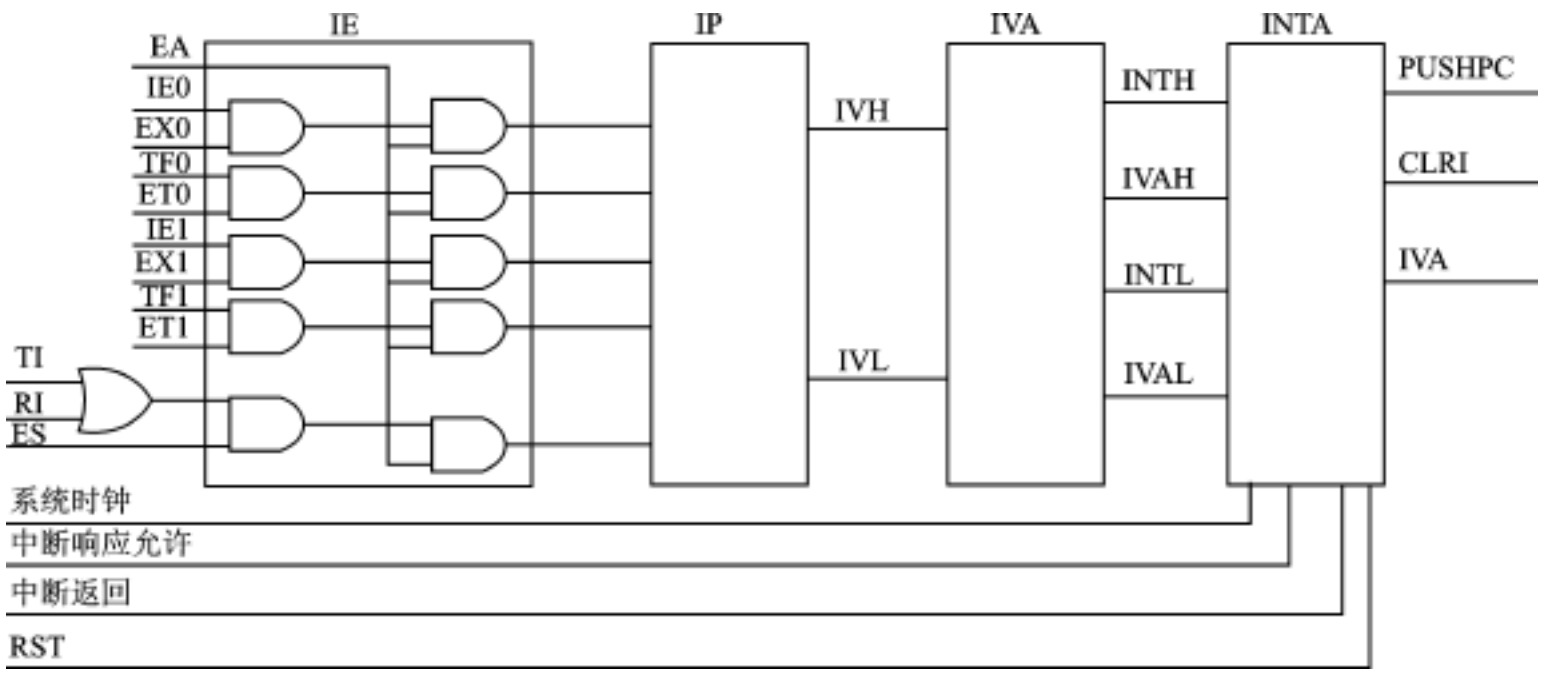


图 1 .8 - 10　中断控制器结构框图

2 .中断控制器仿真波形

根据图 1 .8 - 10 结构图,描述中断控制器并仿真的波形如图 1 .8 - 11 所示。由该图可看出,此中断控制器的功能正确,可实现二级中断的嵌套。

八、总　　结

以上对各个模块的描述是实现 8031 芯片整体 VHDL 描述的基础,在此基础上只需编写 8031 芯片的顶层代码,在顶层代码中调用已仿真通过的各个模块,即可实现对 8031 芯片的整体描述。

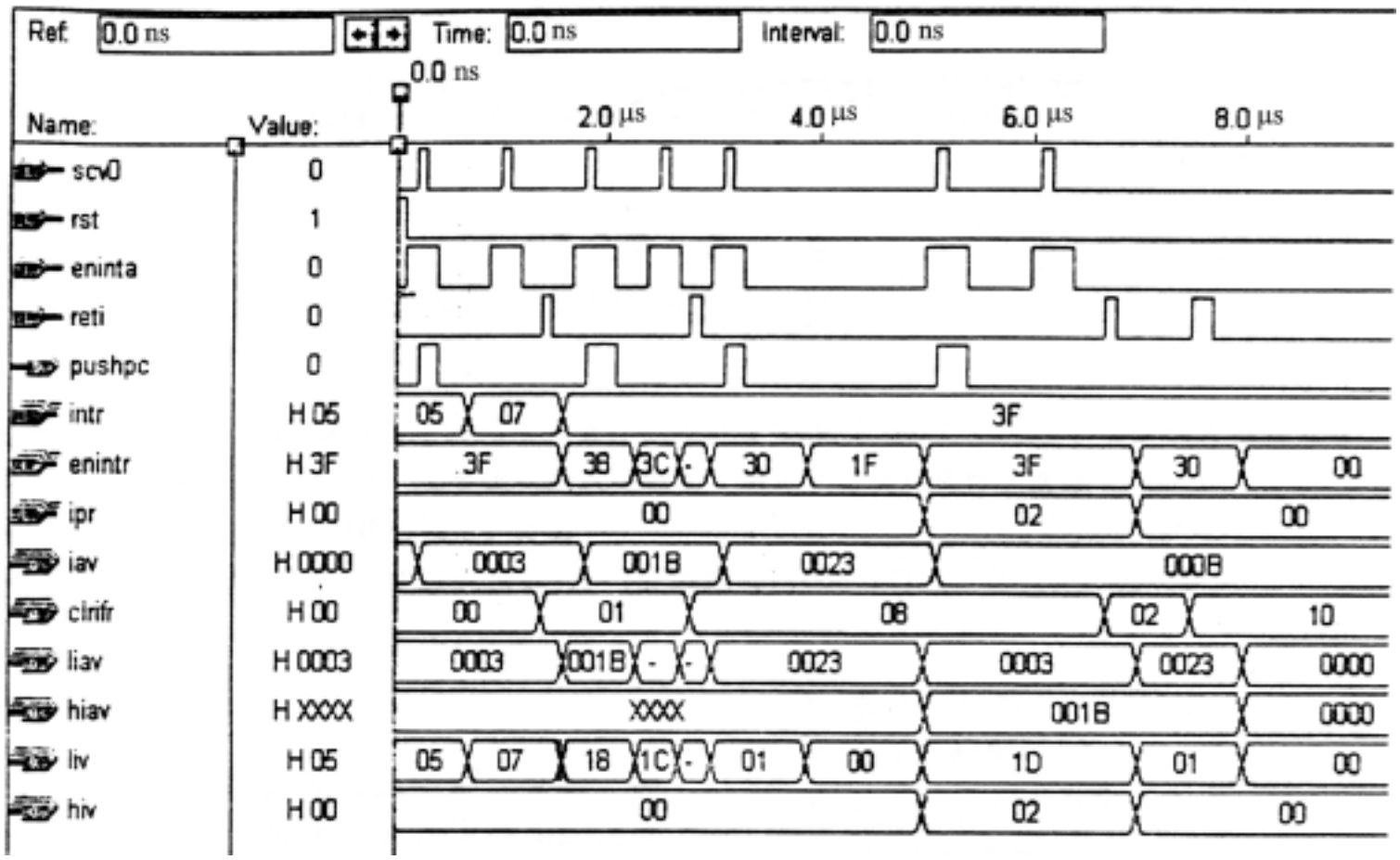


图 1 .8 - 11 中断控制器仿真波形图

参 考 文 献

1 illiam D Richard . A Capstone Computer Engineering Design Course . IEEE Transactions on Education, 1999 ,42 (4)

2 侯伯亨,顾新主编 .VHDL 硬件描述语言与数字逻辑电路设计 .西安:西安电子科技大学出版社,1999

3 李朝青主编 芯片原理及接口技术 .北京:北京航空航天大学出版社,1998

选自《航空电子技术》季刊,2000 年第 4 期

1 .9　ISP 技术在数字系统设计中的应用

杭州中国计量学院机电学院自动化系(310034)　吴　霞
天 津 大 学 自 动 化 学 院 (3 0 0 0 7 2)　吴　雪

一、ISP 技术介绍

采用 E^2 PROM 编程技术的 PLD 器件具有可反复编程且数据可长期保存的优点,编程时需用专用的编程器;而采用静态 RAM 技术的 PLD 器件具有灵活的现场编程的优点,但其编程数据在掉电的情形下却无法保留。Lattice 公司发明并批量生产的在系统可编程逻辑 isp 器件则将上述两种器件的优点完美地结合起来。

在系统可编程是指在用户自己设计的目标系统中或印刷电路板上为重新配置逻辑而对逻辑器件进行编程或反复编程的能力。具有这一功能的器件在编程时无需专用的编程器,通过器件内部所具有的在系统编程的电路直接接收编程的命令与数据,然后在编程状态机的控制下对器件进行编程。在系统可编程逻辑器件的出现使得数字电路系统设计的方法得以改进。采取这种方法时,硬件电路的逻辑功能的修改可以通过可编程逻辑器件内部逻辑功能的修改来实现,这使硬件的修改有可能像软件那样灵活和方便,为逻辑电路的设计开创了一个全新的局面。

Lattice 公司研制并生产的在系统可编程逻辑器件及其产品是在该公司生产的可编程器件的基础上增加了在系统的编程电路,具有在系统编程和逻辑重新配置的能力。编程时只需标准的 5 V 电源,可在印刷电路板上编程,不必将器件从试验电路板拔下,因而减少了试制过程中因反复修改逻辑而造成的人力和物力的耗费,缩短了产品的开发周期。在产品的研制过程中,在系统可编程逻辑器件可构成多功能硬件系统,减少印刷电路板的种类。利用在系统可编程逻辑器件的另一个好处是有利于产品的升级。

进入 20 世纪 90 年代以后,在系统可编程逻辑器件的出现,给数字系统设计带来了革命性的变化,传统的‘固定功能集成块 + 连线’的设计方法正逐步退出历史舞台,而基于芯片的设计方法正在成为数字系统设计方法的主流。

下面介绍 Lattice 公司高密度器件 ispLsI1016 结构(见图 1 .9 - 1)及特点。

ispLsI1016 器件共有 44 只引脚,采用 PLCC 扁平封装,绝大多数引脚都可以编程复用。ispLsI1016 主要由 2 个宏块(Magablock)、1 个全局布线区和一个时钟网络构成。每个宏块内部又可细分为 8 个通用逻辑块、1 个输出布线区、1 个输入总线、16 个 I/O 端口和 2 个输入引脚。

ispLsI1016 内部的信号流向为:(1) 由 I/O 引脚来的输入信号经过输入总线后进入全局布线区,然后由全局布线区统一调度,将信号分配给具体的通用逻辑块(GLB);由输入引脚来的输入信号不经过全局布线区(GRP),而是直接进入自己所属宏块的 GLB 中。(2) 系统的逻辑功能主要在各个 GLB 内实现。(3) 各 GLB 的输出可以反馈回全局布线区,也可经过输出布

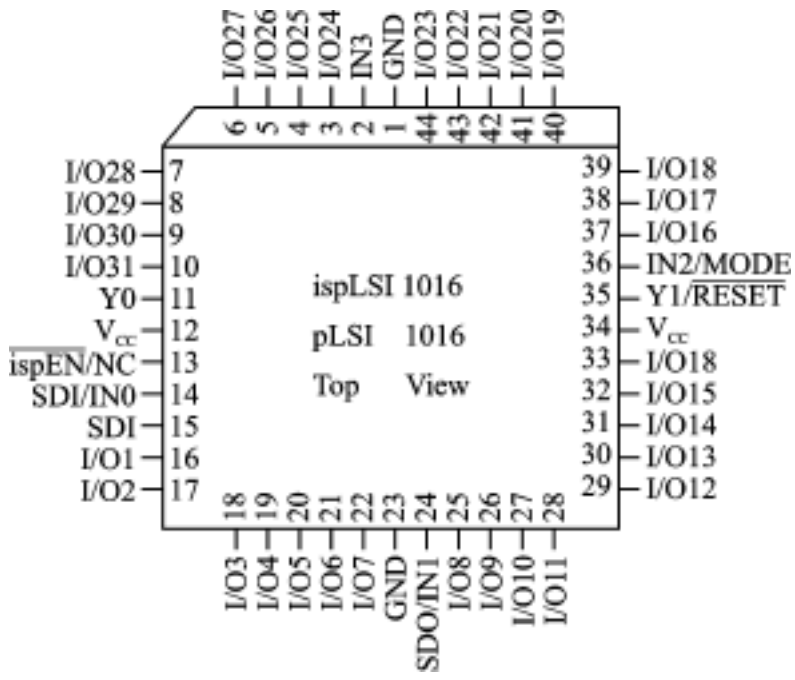


图 1.9-1 ispLsI1016 顶视图

线区后分配给具体的 I/O 引脚。(4) 系统所需的时钟信号由时钟网络产生。Y0、Y1、Y2 为器件的专用时钟输入脚,其中 Y1 引脚缺省时为系统复位端。(5) 编程引脚共 5 个:$\overline{\text{ispEN}}$、SDI、SDO、MODE 和 SCLK,其中后 4 个可以功能复用。当$\overline{\text{ispEN}}$为低电平时,进入编程状态,其余 4 个作编程引脚,否则作为功能引脚。

二、ISP Synario System 器件的开发系统

ISP Synario System 开发系统是一套完整的逻辑设计系统。它能够支持 Lattice 公司的 ispLSI、pLSI、ispGAL、ispGDS 器件及全新系列 GAL 器件的设计、编译和逻辑模拟。

1. 顶层原理图设计

ISP Synario System 为用户提供了一个自顶而下的设计方法,非常适合大型数字系统的设计。系统分析主要负责各模块分割,确定各模块的输入、输出。各模块还可多次嵌套,各子模块可分配给多人同时设计。顶层设计常采用原理图输入,也可采用 ABEL 语言编写源文件输入,其中以原理图法最为直观。对于小规模数字电路的设计,可以省略顶层原理图设计,单独由 ABEL 源文件来描述整个电路的逻辑功能。小规模数字电路的设计,也可以利用开发软件中提供的常用元件库,如门电路库、触发器库中的元件直接用原理图来描述完整的逻辑功能。

2. 编写 ABEL 源文件

ABEL 源文件常用来描述顶层原理图设计中的各模块的具体逻辑功能。对于小规模数字电路的设计,可以单独用 ABEL 源文件描述整个电路逻辑功能,并且可以通过设置中间变量(NODE)的方法实现简单的系统分割。如果电路逻辑功能或模块功能全部用原理图来描述,则这部分可以省略。

3. 编写顶层测试矢量

顶层测试矢量是用户期望的整个数字系统的输入和输出之间的逻辑关系,用于检验用户描述的逻辑功能的仿真结果的正确性。如数字系统采用原理图设计,则测试矢量文件必须单独编写;若数字系统设计全部由 ABEL 源文件来描述,测试矢量部分可以包含在 ABEL 源文件中,此时测试矢量文件不必单独编写。

4 . 各种资源的编译和简化

当编写好数字系统所需的各种资源(如原理图文件、ABEL 源文件、测试矢量文件等)后,下一步就是要利用 ISP Synario System 开发软件提供的各种处理程序对资源文件作相应的处理。这些工作都可以在项目导航器的指导下完成。主要检查用户输入逻辑功能的语法或符号错误并产生中间结果、各种文档等等。编译、链接过程中运行参数大多可用缺省值,但也可改变这些特性参数。另外,ISP Syanrio System 提供的波形仿真功能可直观地观察输入、输出信号之间的时序关系。

5 . 资源链接、器件映射 JEDEC 文件的形成

这部分主要是调用 ISP 的 Link Design 处理过程和 Fit Dhign 处理过程,对编译后的源文件自动进行链接,最后再进行分割、布局和布线。同时,产生下载所需的 JEDEC 文件和整个项目报告(逻辑功能的描述、仿真情况、器件管脚分配、器件资源的利用情况等等),这两项为 ISP Synario System 设计的主要结果。

6 . 下载 JEDEC 文件

下载过程主要由 Lattice 公司提供的菊花链编程软件 IDCD(ISP Daisy Chain Download) 来完成。它是个独立的软件,不包含在 ISP 系统软件中。若计算机和开发板之间的连接正确,启动 IDCD 后,软件会自动扫描开发板上的器件,并在 index 和 device 两栏中显示出来。经过简单的设置和操作就能将 JEDEC 文件下载到实际器件中去。

7 . 编程器件的功能测试

isp 器件被编程之后,除了软件检验下载是否正确外,还需给被编程器件加上实际的输入信号,观察其输出是否满足设计需要,经过检验正确后完成最终设计。

三、ABEL 语言输入方法与测试向量

1 . ABEL 描述逻辑

ABEL 描述逻辑功能的表达方法有三种:布尔方程、真值表、状态机。对应的语句和使用说明如下:

(1) Equations 方程语句

方程语句表示与某一器件有关的一组方程的开始,方程用布尔代数的形式来描述器件的逻辑功能。方程后可跟任何合法的布尔方程表达式。

(2) Truth-table 真值表

真值表是 ABEL 描述逻辑功能的又一方法,它用表格的形式直观地说明不同输入下的逻辑输出。真值表可用于组合和时序电路。

(3) State-diagram 状态@语句

利用状态图设计可以很容易实现复杂的时序逻辑。

2 . 测试向量的规则

语法:test-vectros[in 器件名]

(输入向量 - > 输出向量)

[输入信号值 - > 输出信号值:]

测试向量表用具体的输入向量表定义了一个逻辑器件应具备的逻辑功能。测试向量用于对器件的内部模式进行仿真,并对编程后的器件进行功能测试。各个测试向量都包含一个表

头格式及向量本身,表头格式用来定义测试向量的开始,并说明向量的排列,向量表示了器件应具有的逻辑功能。测试向量表列出了各种输入信号的组合及相应的输出信号,表中可包含所有的组合形式,也可以只写出其中一部分。要求表中所有信号值必须用定义过的常量或数值常量及特殊常量 .X ., .Z .等,表中每一行必须以分号结尾。

四、编程实例

下面用 ispLsI 器件 1016 实现的十进制加法计数器为例说明 ABEL 语言设计的方法。在图 1 .9 - 2 中,模块 CNT10 为计数器模块。本文仅介绍实现这个模块的 ABEL 语言、测试向量源程序。

```
MODULE CNT10                          实现十进制加法计数器模块 ABEL 语言源程序
EN,CLK,CLR PIN;                       '定义计数器使能端、清零端、时钟端
CAO PIN ISTYPE 'COM';                 '定义计数器输出进位端
Q3 ..Q0 PIN ISTYPE 'REG';             '定义计数器输出端
COUNT = [Q3 ..Q0];
EQUATIONS
COUNT .CLK = CLK;
COUNT .AR = CLR;
COUNT .CE = EN;
CAO = Q0 & ! Q1 & ! Q2 & ! Q3;
WHEN(COUNT > =9) THEN COUNT: =0 ELSE COUNT: = COUNT .FB+ 1;
END
MODULE CNT10                          '实现十进制加法计数器模块测试向量源程序
ENN,RESET,CK PIN;
COUT PIN ISTYPE 'COM';
QQ3 ..QQ0 PIN ISTYPE 'REG';
TEST_VECTORS ([ENN,RESET,CK] - > [QQ3 ..QQ0])
@REPEAT 11{[1,0, .C .] - > [ .X ., .X ., .X ., .X .];}
END
```

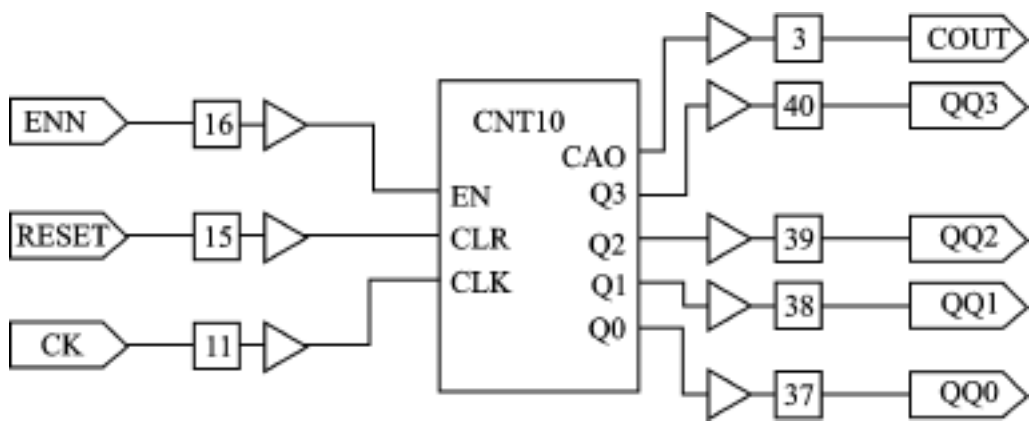


图 1 .9 - 2 isp 器件实现十进制加法计数器模块

五、结束语

采用 ISP 技术后,单片 ISP 器件即可实现以往需要几十片中规模集成电路(MSI)才能完成的电路功能。应用 ISP 器件简化了系统结构,减小了设计单元的体积,提高了系统质量、可靠性,方便了系统设计和调试。即使将在系统可编程器件焊到电路板上以后,用户也可以在不改变电路系统设计或线路板硬件设置的情况下,为重构逻辑而进行反复编程和随意修改,甚至可以在同一块板子上实现不同的硬件结构。使得硬件设备的修改像软件一样容易。

参 考 文 献

1　黄正瑾 .可编程逻辑器件设计[M] .上海:复旦大学出版社,1997

2　扬晖,张凤言 .大规模可编程逻辑器件与数字系统设计[M] .北京:北京航空航天大学出版社,1998

选自《仪表技术》双月刊,2001 年第 1 期

1 .10 单片机单总线技术

信息产业部电子五所 黄宇飞 吴 江
桂林电子工业学院 秦 旭 陈尚松

计算机用于测控系统时,测控对象与计算机之间的信息交换是通过总线进行的。不管是通用的并行、串行总线,还是专用总线,其传送数据的信号线总是多根的。最简单的 RS-232 串行总线也有两根信号线,且只能通向一路被测或被控对象,否则要用下位机扩展。这样,当遇到多路多个测控对象时,系统中相互连接则非常复杂。若只用一根信号线(1-wire),在其上可以挂上许多测控对象,且电源也经这根信号线馈给,这样组建一个系统就方便多了。本文介绍使用这种一根信号线的技术。为了表述简明易懂,并区别于计算机的其他总线,将其称为单总线(1-wire bus)技术。

单总线及相应芯片是美国 Dallas 半导体公司近年推出的新技术。在该公司支持下,信息产业部电子五所和桂林电子工业学院联合对单总线技术进行了开发应用。我们正在对环境状态监测系统、实时气象监测系统(自动气象站)、军用仓库测控系统、农业塑料大棚测控系统、宾馆楼宇监管系统、停车收费系统、考勤管理系统等领域进行应用开发。应用中主控计算机视要求,既可以采用 PC 机也可以采用单片机。本文主要介绍在单片机控制下单总线技术的应用。

一、单总线技术

单总线技术是只有一个总线命令者和一个或多个从者组成的计算机应用系统。单总线系统由硬件配置、处理次序和单总线信号三部分组成。

系统按单总线协议规定的时序和信号波形进行初始化、识别器件和进行数据交换。

1 .硬件配置

单总线系统只定义了一根信号线。总线上的每个器件都能够在合适的时间驱动它,相当于把计算机的地址线、数据线、控制线合为一根信号线对外进行数据交换。为了区分这些芯片,厂家在生产每个芯片时,都编制了惟一的序列号,通过寻址就能把芯片识别出来。这样做能使这些器件挂在一根信号线上进行码分多址、串行分时数据交换,组成一个自动测控系统或一个自动收费系统,甚至还可以用单总线组成一个微型局域网。

厂家对每个芯片用激光刻录的一个 64 位二进制 ROM 代码。从最低位开始,前 8 位是族码,表示产品的分类编号;接着的 48 位是一个惟一的序列号;最后 8 位是前 56 位的 CRC 校验码。CRC(Cyclic Redundancy Check)称为循环冗余码检测,是数据通信中校验数据传输是否正确的一种方法。在使用时,总线命令者读入 ROM 中 64 位二进制码后,将前 56 位按 CRC 多项式(这里是 $X^8 + X^5 + X^4 + 1$)计算出 CRC 值,然后与 ROM 中高 8 位的 CRC 值比较,若相同则表明数据传送正确,否则要求重传。48 位序列号是一个 15 位的十进制编码,这么长的编码完全可为每个芯片编制一个全世界惟一的号码,也称之为身份证号,可以被寻址识别出来。此外,芯片内还含有收、发控制和电源存储电路,其示意图如图 1 .10 - 1 所示。

这些芯片采用 CMOS 技术,耗电量都很小,从单总线上“偷”一点电(空闲时几 μW,工作时几 mW)存在芯片内电容中就可正常工作了,故一般不用另附电源。单总线上通常处于高电位(5 V 左右),每个器件都能在需要时驱动它。因此,挂在总线上的每个器件必须是漏极开路或者是三态输出的,这样,不工作时不会给总线增加功耗。

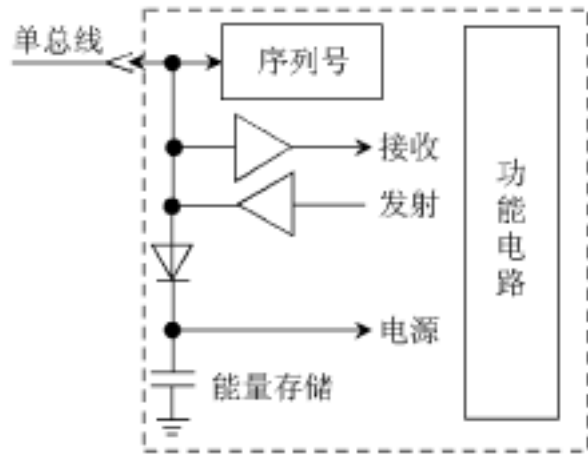


图 1 .10 - 1　单总线芯片入口的示意图

单总线的数据传输有两种模式,通常以 16 .3 Kbps的速率通信,超速模式可达142 Kbps。因此,只能用于对速度要求不高的场合,一般用于 100 Kbps 以下速率的测控或数据交换系统中。

以上内容是单总线技术协议所要求的,各种芯片都具备这些基本内容,然后才进入某种具体的芯片功能,如 A/ D 转换器、温度计等。

应当指出,单总线技术作用距离在单片机 I/ O 直接驱动下可达 200 m,经扩展可达 1 000 m以上,允许挂上百个器件,能满足一般测控系统的要求。

2 .处理次序

处理次序是软件设计的任务。在单总线系统中,软件设计是技术的关键。简洁的硬件配置是靠复杂的软件来支撑的。在 PC 机作主控机时,单总线软件设计基于 Dallas 公司授权的软件开发商提供的成套开发工具,为软件开发应用带来很大的便利。而用单片机作主控机时,得由自己依据单总线协议,用汇编语言和 C 语言来编写全部软件,给开发应用增加了一定的难度。

处理次序保证数据可靠地传送,任一时刻单总线上只能有一个控制信号或数据。处理次序操作时,一般有以下四个过程:① 初始化;② 传送 ROM 命令;③ 传送 RAM 命令;④ 数据交换。

单总线上所有处理都从初始化开始。初始化时序由总线命令者发出的复位脉冲和一个或多个从者发出的应答脉冲组成。“应答脉冲”是从者让总线命令者知道某器件是在总线上,并准备工作。其信号波形如图 1 .10 - 2 所示。

当总线命令者检测到某器件的存在,就会发出传送 ROM 功能命令。单总线协议规定其层次结构如图 1 .10 - 3 所示。

单总线命令者首先必须发送 7 个 ROM 功能命令中的一个命令:① 读 ROM(总线上只有一个器件时,如读 DS2401 的序列号);② 匹配 ROM(总线上有多个器件时,寻址某个器件);③ 查找 ROM(系统首次启动后,须识别总线上各器件);④ 跳过 ROM(总线上只有一个器件时,可跳过读 ROM 命令直接向器件发送命令,以节省时间);⑤ 超速匹配 ROM(超速模式下寻址某个器件);⑥ 超速跳过 ROM(超速模式下跳过读 ROM 命令);⑦ 条件查找 ROM(只查找输入电压超过设置的报警门限值的某个器件)。这些操作在手册中都有具体的命令码供编程使用。当成功执行上述命令之一后,总线命令者可发送任何一个可使用的命令来访问存储和控制功能,进行数据交换。所有数据的读写都是从最低位开始的。

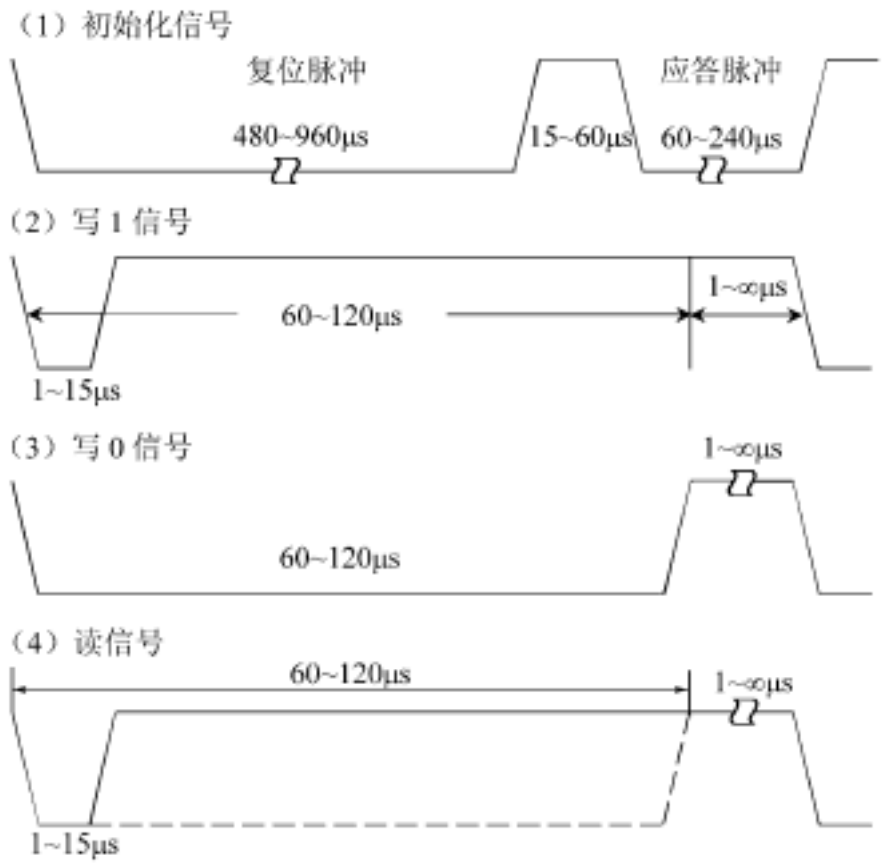


图 1 .10－2 单总线的时序信号波形

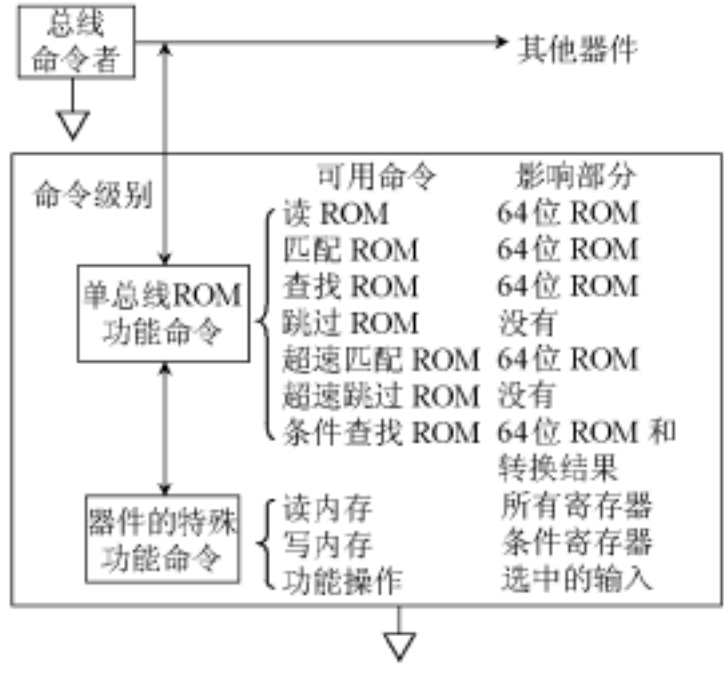


图 1 .10－3 ROM 的功能层次结构

3 . 单总线信号

单总线传送数据或命令是由一系列的时序信号组成的,单总线上共有 4 种时序信号:① 初始化信号;② 写 0 信号;③ 写 1 信号;④ 读信号。图 1 .10－2 给出了常规模式下这 4 种波形的示意图。各器件的应用手册中对这 4 种波形参数(如脉冲上升时间、宽度和间隙等)都作了具体的要求,设计中应保证指令执行时间小于或等于时序信号中的最小时间。这部分软件必须用单片机的汇编语言进行编程,以确保严格的时间关系,且注意当单片机工作频率不同时,单总线的时延值是不同的。

根据单总线的时序信号波形,用汇编语言写出接口程序,C51 将其作为外部函数调用即

可。C 语言的外部调用函数声明如下:

```
extern WDS(unsigned char x);                              * 写单总线命令外部函数声明 */
extern RDSCRC8(unsigned char pt,unsigned char num);    / * 读单总线数据外部函数声明 */
extern bit RTDS(void);                                    / * 复位单总线外部函数声明 */
extern Delay15(unsigned char n);                          / * 延时 15 μs 外部函数声明 */
```

例如,对于单总线上的器件要读其序列号,用 0x33H 命令,程序如下:

```
if (RTDS() ! = 1) {;} Delay15(0x1f);
WDS(0x33); RDSCRC8(temp,8);while(1){};      / * Read serial number */
```

这段程序首先对单总线发复位,然后再对单总线写 0x33H 命令,最后从单总线上读取 64 位的序列号,存入 Temp 变量中。

二、单总线器件

单总线系统中配置的各种器件是由 Dallas 等公司提供的专用芯片来实现的。专用芯片的种类和型号很多,可以参阅 Dallas 公司的数据手册和光盘,也可从互联网上访问。这里简单介绍常用的典型芯片。

1 .数字温度计 DS18S20

Dallas 公司提供了多种数字温度计,如 DS1820、DS18B20。下面是其新推出的 DS18S20 的主要特性。

- 温度测量范围: - 55～ + 125℃;
- 分辨率:±0 .5℃(- 10～ + 85℃时);
- 温度值输出:9 位二进制数字量;
- 转换时间: 750 ms(最大值);
- 用户可设置报警温度的上下限;
- 不需外围电路,电源可由单总线提供;
- 两种封装形式: 3 端 PR-35 塑封或 8 脚 SOIC 封装。

该温度计采用了与众不同的原理,是利用温敏振荡器的频率随温度变化的关系,通过对振荡周期的计数来实现温度测量的。为了扩大测温范围和提高分辨率,使用了一个低温系数振荡器和一个高温系数振荡器分别进行计数,并采用了非线性累加器等电路来改善线性,故此 DS18S20 具有上述良好的特性,而且售价低廉。

2 .A/ D 转换器

在单总线上直接挂上 A/ D 转换器,会使系统的检测功能大大增强。各种物理量只要通过传感器变为电压量,就可由 A/ D 采集后经单总线送到计算机进行处理。Dallas 公司 1999 年推出的 DS2450 就是这样的 A/ D 转换器,其主要特性为:

- 4 路模拟输入通道,两种模拟输入量程为 0～2 .56 V 和 0～5 .12 V;
- 未用做输入的通道可作为输出通道使用;
- 一个数据口,以 16 .3 Kbps 的速率通信,超速模式可达 142 Kbps;
- 逐次逼近的变换原理,可选择的 8 位转换精度;
- 响应模拟电压超门限报警设置;

· 不用另接电源和外围电路;

· 8 脚 SOIC 封装。

这样,温、湿度的检测也可改用 A/ D 转换器 DS2450 和模拟式温、湿度传感器串接来实现。同理,其他类型的传感器,如防火、防盗等传感器,其输出电压也可由 A/ D 判定来报警。

3 . 可寻址控制开关

在测控系统中,开关量控制是应用最多的。对计算机来讲,只要送出一位 0 或 1 控制码信号,就可用它去触发被控电路。通常是先触发光电耦合器,然后启动继电器、晶闸管或固体继电器,视被控设备功率大小选用合适的开关器件。

Dallas 公司提供了一些可寻址的控制开关,如 DS2405、DS2406、DS2409 等。DS2405 的主要特性为:

· 由单总线上数据决定漏极开路输出的逻辑电平作为开关控制信号,如图 1 .10 - 4 所示;

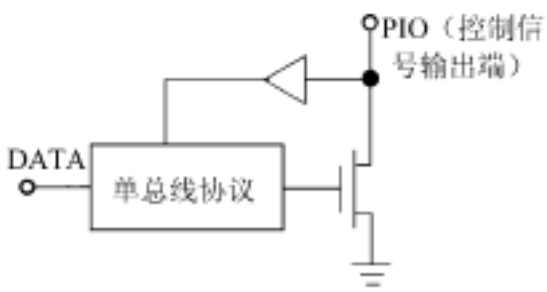


图 1 .10 - 4 可寻址开关 DS2405 原理示意图

· PIO 引脚吸收能力大于 4 mA,0 .4 V;

· 不用外接电源;

· 三种封装形式: TO-92 三脚塑封、SOT-223 四脚平面封装和 C-Lead 六脚表面安装封装。

4 . 硅序列号 DS2401

该芯片实际上是块符合单总线协议的 ROM 硅片,厂家在其中写入了惟一的序列号,用做寻址定位的标识。图 1 .10 - 5 中的电子门锁和防非法侵入,都使用了 DS2401。DS2401 广泛用于位置判定、身份识别和巡检监督等场合。

5 . 防静电保护二极管

在单总线线路的末端,为防止处在开路状态易受静电等干扰侵入,通常都接上 DS9502 之类的防静电(可高达 27 kV)保护二极管。

三、单总线应用

用单片机实现单总线应用,硬件连接简单,单片机并口 P1、P2、P3 中的任一位 I/ O 端口都可以与单总线进行双向数据传输。用单片机对单总线系统进行现场长期监控是最经济实惠的方案,而且还可通过 RS-232(或 485)串行口与上位机 PC 连接,这样还能在 Windows 平台上进行更高一级的软件管理。单总线技术可广泛应用到社会各领域,这里只列举了环境状态监控的应用情况,其方法也完全可以应用于其他领域。

环境状态监控系统通常用于程控通信机房、精密仪器室、档案馆、库房、宾馆客房、无人值守站、变电站等场所,实时监测现场环境中的温度、湿度、烟雾、浸水及非法侵入等情况,根据设定值自动报警并驱动相关执行器件。这是计算机在测控领域中典型应用的实例。通常的设计方案是选用一块性能符合要求的多路数据采集(A/ D)卡,与单片机的并行口相连,再把各种传感器送来的模拟信号经多路转换开关连到 A/ D 上,变成数字信号后交给计算机进行处理。这样设计至少有两大缺点:其一,要拉一大把线才能把现场传感器的信号送到采集卡上,布线施工麻烦,成本也高;其二,线路上传送的是模拟信号,易受干扰和损耗。因此,这种方案的性能

价格比较低。

采用单总线技术设计环境状态监控系统,只要一条双绞线(一根为信号线,一根为地线)从单片机拉向监控现场,然后将各种监控对象挂在其上就可以了,其示意图如图 1 .10 - 5 所示。图中只画出了一个监控现场的配置,其布线接头与通常电话线路用的一样,插入和拔出都很方便。

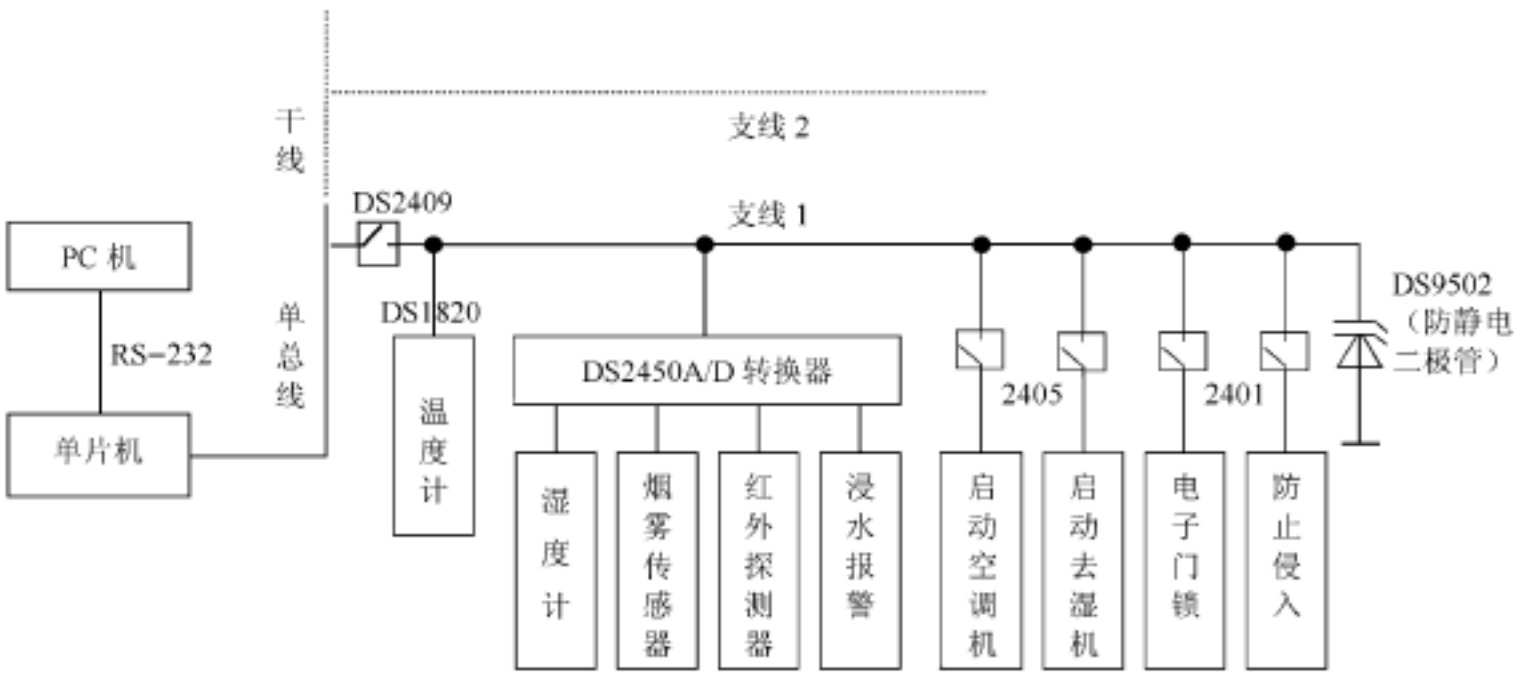


图 1 .10 - 5　单总线环境状态监控系统布局示意图

图 1 .10 - 5 中的系统可监控室内温、湿度。因为温度计为直接数字输出,不需 A/ D 转换器。湿度计、烟雾传感器、红外传感器和浸水报警需要 A/ D 转换器进行判别。当温、湿度超过设定值就会通过开关 DS2405 来开启空调机、去湿机。烟雾传感器用做防火报警,当其输出电压超过 A/ D 设定门限时就发出报警。对于红外式传感器,当有人侵入室内时,其输出电压可由 A/ D 判知并报警。对于浸水报警,当置于地面上的探测器被水淹而短路,接通了监视电路,被计算机查询到后就会发出浸水报警。

电子门锁和防非法侵入都是用 DS2401 序列号设计的。正常情况下,只有用对应的序列号钥匙才能打开房门。若非法侵入,门窗位移使磁控开关接通设置的序列号,则会发出报警。

综上所述,单片机单总线技术比采用传统的方案具有较高的性能价格比。而且,可以看出该技术具有以下特点:适用于低速测控场合;测控对象越多越显示其优越性;性价比高;硬件施工、维修方便;抗干扰性能好;具有 CRC 校验功能,可靠性高;软件设计规范;系统简明直观,易于掌握。因此,积极推广单总线技术的应用会有较好的经济效益和社会效益。

参　考　文　献

1　allas . Products data Book,1999(光盘版)

2　Dallas . Application Net Book,1999(网络版)

3　吴江,陈尚松 .单总线技术在测控系统中的应用 . 电测与仪表,1999(9)

4　吴江,陈尚松 .用单总线技术设计环境状态监控系统 . 电子技术应用,2000(6)

选自《单片机与嵌入式系统应用》月刊,2001 年第 1 期

1.11 智能信息载体 iButton 及其应用

华东理工大学 王永红 凌志浩

一、iButton 简介

计算机技术的蓬勃发展,使基于条形码、磁卡、IC 卡等技术的数字识别系统逐渐取代了人工识别,并被广泛应用于金融、电信、商业等领域,深刻影响着人们的日常生活。但是,条形码、磁卡、IC 卡等构成的数字识别系统存在着携带不便、易受损坏、不能应用于恶劣环境等不足。美国达拉斯半导体公司(Dallas Semiconductor Corporation)推出的一种智能化信息载体 iButton,较好地解决了数字识别系统存在的这些问题,为开发更加完善的新颖数字识别系统提供了技术支持和实现手段。

iButton(information Button)意为"信息钮扣"。它采用直径 17 mm、厚 3～6 mm 的钮扣状不锈钢外壳封装。内部由 I/O 处理器和存储器两个基本部分组成,某些应用于特殊场合的 iButton 还内置有感温、时钟等元器件。iButton 以 1-Wire 规范作为通信协议,仅用 1 根数据线实现与外界的信息交换。图 1.11-1 所示是 iButton 的外型结构示意图。

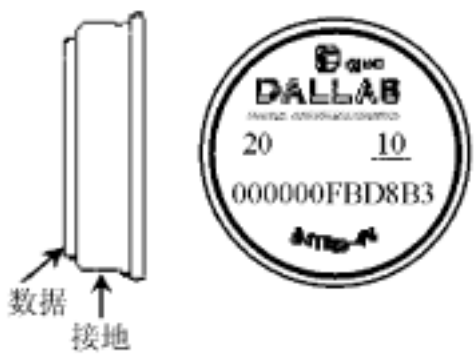


图 1.11-1 iButton 外型结构

iButton 作为一种新颖的智能化信息载体,相比传统的信息载体具有如下特点:

① 由于 iButton 采用不锈钢外壳封装,无暴露的易损部件或易腐部件,具有抗撞击、防水渍、耐腐蚀、抗磁扰、防折叠等显著特点,工作温度范围也较宽,可以在 -40～80℃的温度范围内正常工作,适用于恶劣的环境;

② 每片 iButton 内部均固化有惟一的 64 位标识号(ID),排列顺序依次为分类号 8 位、系列号 48 位以及 CRC 码 8 位,并且具有自毁功能,保密性能好,可适用于需要作硬件加密的场合;

③ 由于 iButton 采用独特的机械外形设计,具有 IC 卡、磁卡等信息载体无法比拟的优势,存储于其中的数据信息具有相当高的安全可靠性;

④ iButton 可以嵌在戒指、钥匙串、钱包或手表上,随身携带方便;

⑤ 存取数据时采用接触方式,与触头轻轻一碰,瞬间即可完成数据信息的读写操作。

二、iButton 的工作机理

1 根数据线和 1 根地线构成了 iButton 的电气接口。正面是数据线,底座为地线,结构极其简单。传输数据时,无需电源和时钟信号,信号同步靠内部定时逻辑解决,而数据通信的能量则由数据线提供。iButton 内部为一大规模集成电路,由 I/O 处理器和存储器两个基本部分构成,其功能模块如图 1.11-2 所示。

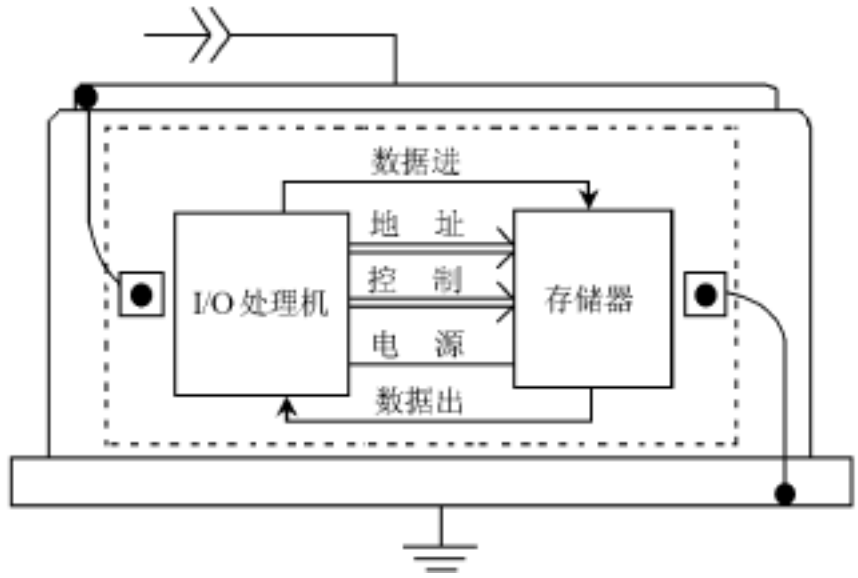


图 1.11-2 iButton 内部功能模块示意图

iButton 采用独特的主从式、位同步、半双工串行方式与外部进行通信。微机及有关读写设备处于主动(Master)地位,iButton 处于从属(Slave)地位。Master 与 Slave 之间以 1-Wire 协议为基础,按照特定的时序要求由数据线逐位交换数据。

iButton 作为从设备,其工作过程可描述为:首先,由微机主动向 iButton 发测试脉冲,以识别 iButton 是否已与其触头接触,若已正确连接,可接收到 iButton 发来的应答脉冲,表示可以进入数据通信过程。这时,微机先发操作 iButton 的 ROM 区的指令,如读 ROM 区数据指令、匹配操作指令、搜寻操作指令等,这些指令被 iButton 接收并执行。然后,发操作 iButton 的 NV RAM 区数据的指令,如读写 NV RAM 区数据指令、读写或复制读写缓冲区(Scratch-pad)数据的指令等。之后,微机与 iButton 间进行数据传输操作。最后,微机再发测试脉冲,当收到 iButton 的应答脉冲后,整个数据通信过程即告结束。图 1.11-3 示意了有关时序。它们可分为测试连接与应答、从 iButton 读取数据和向 iButton 写入数据 3 种类型。

在应用软件开发过程中,对 iButton 进行数据读写的过程需要遵循其工作机理和时序要求,具体包括:

(1) 测试连接及应答

微机发测试负脉冲给 iButton,查询 iButton 是否已与触头正确连接。若与触头连接良好,iButton 则将数据线拉低,产生应答负脉冲。如果微机检测到这个应答脉冲,就可以进行数据读写操作了。

(2) 从 iButton 读取数据

微机先向 iButton 发 1 个读负脉冲,iButton 接收该脉冲后立即将被读取位的内容送至数据线上,微机从数据线上获得数据。若数据线在 iButton 的采样时区内维持高电平,则读取值为“1”;否则,为“0”。最后,iButton 释放数据线,数据线恢复为高电平,为微机继续从 iButton 读取数据位作好准备。

(3) 将数据写入 iButton

与读取数据类似,微机向 iButton 发 1 个写负脉冲,然后开始写数据。微机维持数据线低电平特定时间,再恢复为高电平,则表明写入“0”;微机发出写负脉冲后立即将数据线拉高并维持特定时间,则表明写入“1”。完成数据写入后,数据线恢复为高电平,为微机继续向 iButton 写入数据位作好准备。

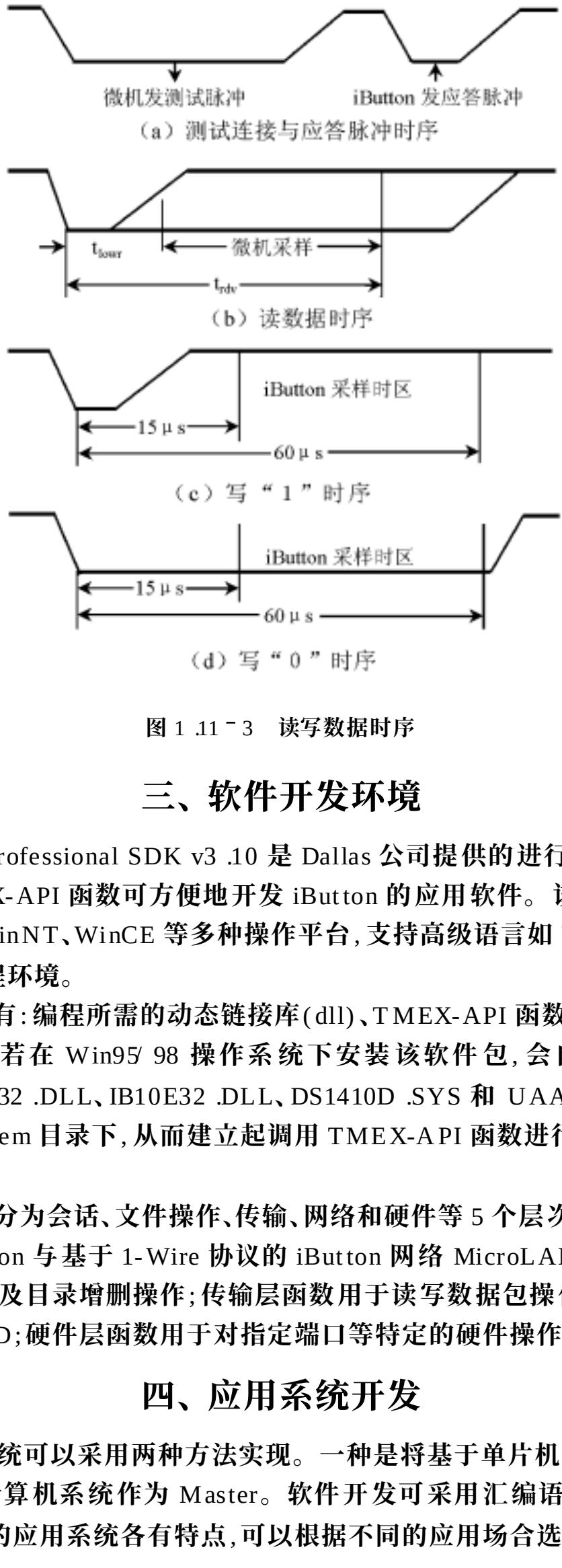


图 1 .11 - 3 读写数据时序

三、软件开发环境

iButton-TMEX professional SDK v3 .10 是 Dallas 公司提供的进行应用程序开发的软件包,通过调用其 TMEX-API 函数可方便地开发 iButton 的应用软件。该软件包适用于 DOS、Win3 .1、Win95/ 98、WinNT、WinCE 等多种操作平台,支持高级语言如 VB、VC 和汇编语言如 8051、808x 等多种编程环境。

SDK 的主要内容有:编程所需的动态链接库(dll)、TMEX-API 函数的说明文档、Demo 程序及其说明文档等。若在 Win95/ 98 操作系统下安装该软件包,会自动将 IBFS32 .DLL、IB97E32 .DLL、IB97U32 .DLL、IB10E32 .DLL、DS1410D .SYS 和 UAAUTHD .UXD 等文件写入 Windows 的 System 目录下,从而建立起调用 TMEX-API 函数进行应用程序开发的软件环境。

TMEX-API 函数分为会话、文件操作、传输、网络和硬件等 5 个层次。会话层函数主要用于多任务环境下 iButton 与基于 1-Wire 协议的 iButton 网络 MicroLAN 的连接;文件操作层函数用于文件读/ 写以及目录增删操作;传输层函数用于读写数据包操作;网络层函数用于读取及验证 iButton 的 ID;硬件层函数用于对指定端口等特定的硬件操作。

四、应用系统开发

iButton 的应用系统可以采用两种方法实现。一种是将基于单片机的应用系统作为 Master,另一种是将微型计算机系统作为 Master。软件开发可采用汇编语言或高级语言进行编程。这两种方法实现的应用系统各有特点,可以根据不同的应用场合选用。

1 . 基于单片机的应用系统设计方法

将基于单片机的应用系统作为 Master, 自行设计单片机与 iButton 的接口电路。图 1 .11 - 4为单片机与 iButton 的接口电路示意图。使用 P0 .0(指定为 DATA_BIT)作为 1-Wire 的 I/ O 口; 晶振频率选用市场上常有的 11 .059 2 MHz; 齐纳二极管将数据线的电压限制在 5 .6 V 以下, 防止过高的电压损坏器件。

按此思路设计的系统小巧, 可以嵌入在有关的仪器仪表中, 从而开发出各种具有智能化数字识别功能的便携式产品或嵌入式产品, 如预付费水电表、电子公交车票、电子防盗门锁等。但是, 必须按照严格的时序编程。

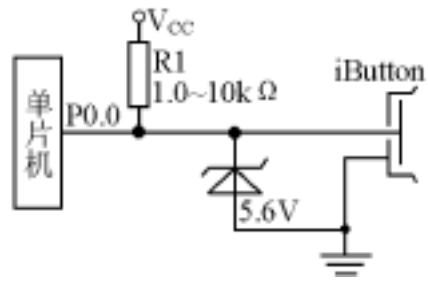


图 1 .11 - 4　单片机与 iButton 的接口示意图

以下汇编程序能够测试 iButton 是否已与触头良好接触, 具体过程为: 先发 1 个测试脉冲给 iButton, 检测 iButton 的应答脉冲, 若检测到该脉冲, 则将进位位置‘ 1 ”, 表明可继续对 iButton 进行数据读写操作; 否则, 清‘ 0 ”。

```
Procedure TouchReset
DATA_BIT BIT P0 .0
TRESET:     USH
            PUSH   ACC
            MOV    A, # 4
            CLR    DATA_BIT          ;开始发复位脉冲
            MOV    B, # 221
            DJNZ   B, $              ;至少保持 480 μs 低电平
            SETB   DATA_BIT          ;释放数据线
            MOV    B, # 6
            CLR    C                 ;清应答脉冲标志
WAITLOW:    JB     DATA_BIT, WH      ;若数据线拉高,则退出循环
            DJNZ   B,WAITLO
            DJNZ   ACC,WAITLO        ;延迟 3 360 μs
            SJMP   SHORT             ;退出子程序
WH:         MOV    B, # 111
HL:         ORL    C,/ DATA_BIT      ;检测应答脉冲
            DJNZ   B,HL
SHORT:      POP    ACC
            POP    B
            RET
```

2 . 基于微机系统的设计方法

采用微机系统作为 Master, 利用 Dallas 公司提供的触头、串(并)口适配器等现成的外围产品, 加之 iButton 构成应用系统。触头用于与 iButton 接触, 实现数据的读写操作; 串(并)口适配器, 用于实现 1-Wire 信号与计算机串(并)口数据信号的转换。触头可通过 DS9097、DS9097E 或 DS9097U-09 串口适配器连到计算机的 COM 口上, 也可通过 DS1410E 并口适配

器实现与计算机的 LPT 口连接。如图 1 .11－5 所示,DS9092GT 触头与 DS9097E 串口适配器相连,DS9097E 直接插到微机的 COM 口上,从而构成 1 个 iButton 的应用系统。

图 1 .11－5 触头与串(并)口适配器连接

用该方法实现的系统,其应用软件的开发相当方便,允许在高级语言的环境下,直接调用 TMEX-API 函数编程,使 iButton 的读写操作对用户完全透明,不必考虑复杂的时序等细节,开发效率高,人机交互界面友善。适合于开发大型企业考勤系统、单位就餐结算系统、智能化小区人员出入管理系统等。

以下 VB 程序调用 TMEX-API 函数,读取 iButton 的 ID。若与数据库技术结合,将读到的信息加上时间标签写进数据库,就能实现显示、查询、打印等功能,构成企业考勤、智能小区人员出入管理等应用系统。

```
……
Dim ROM(8) As Integer
Dim dmmy, search, flag As Integer
Dim MyHandle As Long
Dim RomString As String
Dim state_buffer(15360) As Byte
Private Declare Function TMExtendedStartSession Lib "IBFS32 .DLL"(ByVal PortNum As Integer,
ByVal PortType As Integer, ByVal Reserved As Any) As Long
Private Declare Function TMSetup Lib "IBFS32 .DLL"
(ByVal session_handle As Long) As Integer
Private Declare Function TMFirst Lib "IBFS32 .DLL"
(ByVal session_handle As Long, state_buffer As Byte) As Integer
Private Declare Function TMRom Lib "IBFS32 .DLL"
(ByVal session_handle As Long, state_buffer As Byte, ROM As Integer) As Integer
Private Declare Function TMEndSession Lib "IBFS32 .DLL"(ByVal session_handle As Long) As Integer
……
MyHandle = TMExtendedStartSession(2, 1, vbNullString)
    '建立会话
If (MyHandle > 0) Then          '成功建立会话
  dmmy = TMSetup(MyHandle) '初始化
  search = TMFirst(MyHandle, state_buffer(0))
    '找设备,找到设备返回"1"
  If (search = 1) Then
    ROM(0) = 0                  '表示读数据
    flag = TMRom(MyHandle, state_buffer(0), ROM(0))
    '读得值放入 ROM(0 to 7)
    RomString = "  "
```

```
    For I = 7 To 0 Step -
      If(ROM (I)  < = &HF) Then RomString = RomString + "0"
        RomString = RomString + Hex$ (ROM(I))
    Next I                                    生成 ID 串 RomString
    Text1 .Text = RomString
  End If
  dmmy = TMEndSession(MyHandle)               结束会话
End If
……
```

五、应用前景展望

作为一种新型智能化数据信息载体,iButton 正越来越广泛应用于日常生活,其应用系统大体可分为如下几个方面:

① 应用于身份识别,如电子防盗门锁、单位考勤系统、智能小区人员出入管理系统、计算机软件的硬件加密、电子防伪、暂住人口管理系统等。

② 应用于预收(付)费系统,如预付费水电表、停车场收费系统、电子钱包、电子车票、高速公路收费、公共收费电话等。

③ 应用于数据采集系统,如机动车运行状态参数采集器、温度遥测系统、小型气象站等。

④ 应用于军事领域,如军事保密、情报系统的人员联络等。

综上所述,Dallas 公司推出的智能信息载体 iButton,具有广阔的应用前景,必将给人们的生活带来深刻的影响。相信基于 iButton 的新型数字识别系统不久将会得到广泛的应用。

参 考 文 献

1 allas Semiconductor Corporation . Book of DS19xx iButton Standards[M] . http:// www .ibutton .com

2 Dallas Semiconductor Corporation . DS0621-SUL iButton-TMEX Revision 3 .10 Reference Manual[M] .http:// www .ibutton .com

3 Dallas Semiconductor Corporation .DS0621-SDK iButton TMEX Professional Software Developer s Kit [M] . http:// www .ibutton .com

4 潘日芳,凌志浩 . 微型计算机原理及应用基础 . 上海:华东化工学院出版社, 1993

5 何立民 . 单片机应用系统设计 . 北京: 北京航空航天大学出版社, 1990

选自《单片机与嵌入式系统应用》月刊,2001 年第 4 期

1 .12 基于单片机的高新技术产品加密方法探讨

长沙铁道学院防灾科学与安全技术研究所(410075)
赵望达 鲁五一 徐志胜 裘志浩 蒋晴霞

随着单片机和大规模集成电路技术的飞速发展,人们应用单片机及相关的外围电子器件,研制开发了各种各样的智能化仪器仪表和小型控制系统等自动化高新技术产品。在此类产品开发及推广应用中,碰到的一个令人头疼的问题就是新产品刚一推出就被仿制和剽窃。这样,加密问题就成为新产品开发中必须考虑的重要问题之一。特别是知识产权制度仍处于不断完善之中的我国,其重要意义不言而喻。

早期的单片机开发产品一般采用 MCU + EPROM 的模式,其内部电路和软件均是透明的,即硬件电路可以测绘出来,存放在 EPRO 中的软件亦可方便地读出。因此,这样的产品几乎没有什么防仿制的功能,研制开发人员花费大量心血和资金经长期研制而成的产品就被人轻而易举地窃取了。所以,基于单片机的高新技术产品的研制人员在不断开发新产品的同时亦一直在寻求提高新产品防仿制能力的加密技术。

一、加密技术的基本思路

为了防止新产品的核心技术被窃,需对其硬件电路和软件进行加密。加密方法的基本思路是对硬件电路和软件程序均采取一切可用的方法增加其读出难度,防止硬件电路原理被别人测试和软件被破译。其具体方法有软件的,有硬件的,亦有软硬件结合的。总之,要结合实际产品特点,从设计阶段即对产品的加密方法作全盘考虑,如一般产品或高科技含量的尖端技术产品,视其综合仿制能力的难易程度等,根据不同产品,选择力所能及的加密技术,以使自己开发出的新产品不被仿制或尽可能降低其被仿制的可能性。笔者在主持或参加的众多单片机产品的研制开发工作中,对防仿制和软硬件加密有着浓厚的兴趣,并一直在摸索着这方面的新方法和技术。在此,笔者基于十余年来这方面的应用经验,总结出以下一些加密方法,以供有此爱好的人们参考。

二、基于硬件的加密方法

单片机产品的硬件加密方法有很多,只要能使其硬件电路核心部分不能或很难被破译,就是有效的硬件加密方法。下面介绍几种常见的硬件加密方法。

1 . 采用内置程序存储器的单片机的加密

早期一般采用 8031 + EPROM 的组合结构,而 EPROM 内程序是很容易被拷贝的。所以,近年来随着新型单片机的不断推出,人们纷纷选用与 8031 兼容的内置程序存储器的单片机,如内置 EPROM 的 INTEL 公司的 8751 和内置 FlashRAM 的 ATMEL 公司的 AT89C××系列,还有和 8031 不兼容的 MOTOROLA 的 MC68××系列, MICROCHIP 公司的 PIC16C××系列等。将调试好的程序写到上述单片机的程序存储器中,并选择相应的加密方

式,这样,程序就很难被破译了。虽然,不少单片机开发仿真系统声称能对加密了的存于单片机内置程序存储器 EPROM、FlashRAM、E^2PROM 等之中的程序进行解密,但是,新的加密算法亦不断推出。随着加密和解密之间不断地战斗,单片机及其外围器件提供的加密功能也越来越完善。

2 .采用多单片机结构的加密

现在,市面上 INTEL、MOTOROLA、MICROCHIP 等公司的各式单片机型号品种众多,由于应用面广,其价格同一般集成电路差不多。这样,给我们提供了一种思想:设计一个系统或产品可采用两个或多个单片机,其中一个为主处理器,其余完成其他集成电路的功能。

3 .采用新型硬件电路的加密

采用 PLD 和最新的场可编程器件 FPGA 取代单片机系统中的逻辑电路和计数器等元件,并在编程时对其进行加密,既简化了电路,又达到了加密效果。我们在设计一超声波流量计时,用 Lattice 1032 芯片处理超声波信号和高速计数,而用 AT98C51 单片机及一些外围电路完成仪表常规功能。Lattice 芯片一并完成硬件电路的加密功能。

4 .采用排乱硬件线路的加密

对于采用 8031 + EPROM 结构的产品,设计时,将 8031 和 EPROM 之间的数据线和地址线的顺序排乱,然后在将仿真器上调试好的程序写入 EPROM 前,必须按排乱的数据线和地址线将原程序(明文)一一转换成 EPROM 中的目标程序(密文)。这样,剽窃者如果要破译 EPROM 中的程序,必须同时破译硬件电路,并将其排乱的数据线和地址线恢复正常;然后根据排乱和正常状态的对应关系,将 EPROM 芯片中读出的数据(密文)反向转换成原始程序(明文);再考虑程序的反汇编和分析等。其工作量之大可想而知。如果将二、3 节中的加密方法纳入进来,就能达到更好的保密效果。

三、基于软件的加密方法

本文以 8051 及其兼容单片机为例介绍以下几种软件加密方法:

(1) 插入多条跳转指令,降低程序可读性。在程序调试阶段,就考虑留有增加加密方法所需的足够空间,如程序需占用 4 KB,就必须选具有 8 KB 的程序存储器。在产品应用程序调试通过后,增加一些跳转指令(AJMP、SJMP、LJMP、JMP @ A + DPTR 等)和子程序调用指令(LCALL、ACALL 等),这样,可使别人不容易读懂该程序。

(2) 采用模块化设计方法。将整个程序分为 n 个模块,如图 1 .12 - 1 所示,每个模块后插入一组 3 个字节的 NOP 指令,调试完毕后,将前 2 条 NOP 指令修改为 SJMP Mi(Mi 为第 i 个模块地址),第 4 条指令改为任意指令码,最好是 02 H、12 H、50 H、60 H 等与调用、跳转有关的指令。这样,用开发系统反汇编时,导致插入的 LJMP 指令以后一段程序与原程序完全不一样。所以,若对每个模块都进行此类操作,则整个程序反汇编后读来相当混乱,毫无逻辑。当然,假如剽窃者硬是‘青睐”你的软件,他亦可采用手工汇编的方式,将原程序原原本本汇编出,而你的程序本来就是很难读懂的,则剽窃者处心积虑的剽窃行为亦是白费工夫。

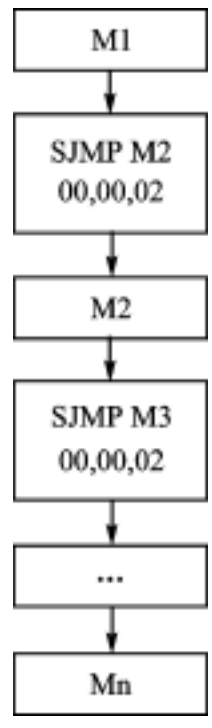


图 1 .12 - 1

(3) 将程序中的固定参数、常数(如显示键盘代码和数值表格等)存

于带掉电保护的数据区域,这些区域可以是 DS12887、MC14618 或 28C64 等芯片的掉电保护存储器。当然,这些芯片是所设计的新产品其他功能必须用的,在此,仅作为一个附加功能而完成加密数据、保护程序的任务。这样,剽窃者即使破译出了你的原程序,要对此程序作分析、解剖,他还必须找出程序中的表格地址;然后将 DS12887 等芯片拿到某一个能读出其内部存储器内容的开发系统上,并对照原程序地址,得出表格数据;最后对表格和原程序作统一分析、解剖。如此过程需要剽窃者具有较深的单片机软硬件知识和应用经验。这样一来,他与其苦心去猜测破译别人的程序,不如自己重新设计一个来得容易。

(4) 其他防止别人分析、解剖软件的加密方法。软件加密的一个重要目的就是防止别人分析、解剖自己的程序,因而广义地理解任何能降低程序可读性的方法都可认为是一种“加密”。除上面所述的转移指令,还可利用大量不用的程序空间做文章。譬如说可以将正常程序中夹上几段无用的程序,这些无用的程序段也会执行到,但是它们的执行不会对系统的工作产生不良影响,仅是用来混淆破译者的视听,打乱其分析程序的思路而已。

软件加密方法的基本思想是通过某些方法对程序和数据进行处理,迫使剽窃者在解密时花费较高的代价和较长的时间,从而望而生畏,放弃解密,因此程序并非是不可破译的。此外,软件加密方法不能防止别人复制,只能防止别人解剖、分析和改进自己的产品,以延长产品占据市场的时间,保持产品优势。所以,软件加密只是硬件加密的补充和延伸。诸如此类的软件加密方法,只要不影响到产品运行功能,是多多益善的,亦是新产品开发者保护自己产品的有力武器。

四、单片机加密方法实例

笔者在设计一智能仪器时,采用了双单片机结构,如图 1 .12 - 2 所示。其主要器件有 AT89C52(主单片机)、AT28C64、ATF16V8B 和 PIC16C71(辅单片机)等。它们在仪器中的作用是:AT89C52 作为微控制器 MCU 并存储 8 KB 的程序;AT28C64 作为外部数据存储器;2 片 ATF16V8B 对 8 位数据总线及控制线进行变换;PIC16C71 运行自己的程序并完成 1 路 8 位的 A/ D 采集。该产品采取的加密措施为:

(1) 对 AT89C52 中程序进行 3 级加密,加密后原存放在内部程序存储器的数据表格必须存放在外部数据存储区,即 28C64 中;

(2) 对写入 ATF16V8B 的“熔丝图”进行加密;

(3) 对 PIC16C71 的程序进行加密,烧断其保密熔丝。

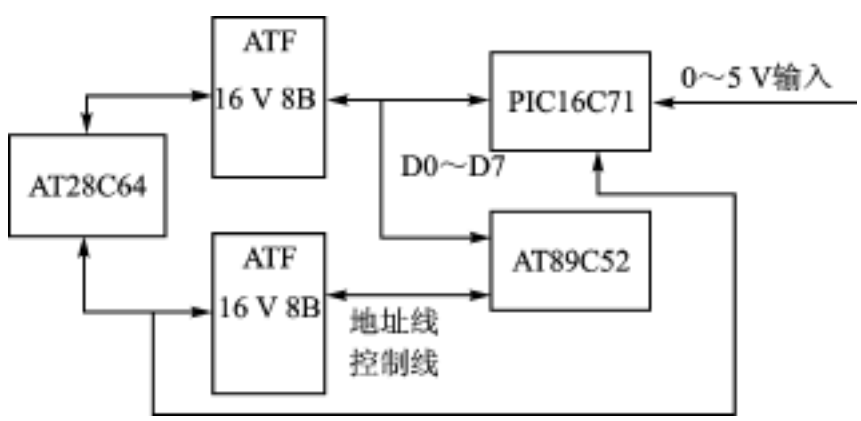


图 1 .12 - 2

这样,如果要对该产品进行仿制,必须对上述 3 种芯片进行解密。如果还要对其分析、解剖,则还须对 28C64 内数据及上述程序进行综合分析。此过程投入的时间和资金将不亚于一次自主的开发。

五、结束语

本文介绍了软硬件加密方法及其在实际产品开发中的应用经验,其中有些软件方法已在其他单片机应用文献中有过类似的介绍,但还未见有系统的针对单片机产品加密方法的论著发表。所以,笔者基于自己的单片机产品开发中的应用体会,对上述加密方法作一简要探讨,旨在对单片机高新技术产品开发研制人员起一抛砖引玉的作用,为大家在具体产品开发中设计合适有效的加密方法提供参考。

加密和解密长期以来就是一对矛盾,世界上只要存在偷窃者,人们就会不断寻求保护自己的防盗方法。在新产品开发推广中,研制开发者可根据具体情况和自己的设想大胆地去研究,并吸取别人的经验,开发研究出更多更新的‘防盗武器”。

参 考 文 献

1 张卫东,等.MCS-51 程序的软件加密方法[J].电子技术,1994

2 赵望达.一种通用型智能单回路测控仪[J].基础自动化,1995

3 李济生.一种新型产品防护措施——程序下载技术[J].电子技术应用,1995(5)

选自《制造业自动化》月刊,2000 年第 2 期

1.13 新一代私钥加密标准 AES 进展与评述

成都西南交通大学(610031) 范平志
成都四川工业学院(610039) 何明星

一、引 言

密码体制大致可分为两类:私钥加密与公钥加密。私钥加密体制由来已久,尤其是在军事领域,它是利用同一共享密钥进行保密通信。与之不同,公钥加密体制始于 20 世纪 70 年代中期,是利用密钥对——公开的公钥与保密的私钥来进行通信。公钥加密体制可以避免私钥加密体制涉及的密钥分配安全性与复杂性问题,但明显的缺点是计算复杂性高,因而私钥加解密从速度上讲具有优势。实际上,通常把这两种方法混合使用:公钥体制用来建立对话密钥,而对话的数据则用私钥体制来进行加密。如电子商务安全交易协议 SET 中通过 DES 算法和 RSA 算法的结合使用,在一定程度上保证了数据的一致性和完整性[1,2]。其中 DES 算法就是 1977 年公布的(IBM 专利)第一代私钥数据加密标准(Data Encryption Standard)。但 DES 经实践检验证明已经不是可靠的密码算法了,从各方面来看,已走到了它生命的尽头。事实上,1997 年由 RSA 公司发起的 DES 挑战赛已证明 DES 的 56bit 密钥太短,虽然三重 DES 可以解决密钥长度的问题,但是它并不十分有效,而且 DES 的设计主要针对硬件实现[1~3]。而今,在许多领域,需要针对软件实现相对有效的算法。鉴于此,1997 年 4 月 15 日,美国国家标准和技术研究所(NIST)发起征集新一代数据加密标准即高级数据加密标准 AES(Advanced Encryption Standard)算法的活动,并成立了 AES 工作组。其目的是确定一个非保密的、全球免费使用的加密算法,用于保护 21 世纪政府的敏感信息,也希望能够成为秘密和公开部门的数据加密标准[5]。1997 年 9 月 12 日,NIST 在联邦登记处(FR)公布了征集 AES 候选算法的通告。对 AES 的基本要求是:它必须是私钥分组加密算法,比三重 DES 快,且至少和三重 DES 一样安全,分组长度是 128 位,密钥长度为 128(3.4×10^{38} 个密钥)、192(6.2×10^{57} 个密钥)和 256(1.1×10^{77} 个密钥)位。1998 年 8 月 20 日,NIST 召开了第 1 次 AES 候选会议,并从 21 个算法中公布了 15 个满足基本要求的 AES 候选算法(其中 5 个出自美国)[1]。3 年来,NIST 对这些候选算法做了大量的分析和测试,也收集了来自 40 多个国家对此感兴趣的部门和个人提出的大量看法和意见[6,8,9,11]。1999 年 6 月,NIST 举行了第 2 次 AES 候选会议,公开了 15 个候选算法的分析讨论结果,并于 1999 年 8 月从中选出了 5 个进行第二轮评估。2000 年 10 月 2 日 NIST 宣布选择来自比利时的 Rijndael(发音为“Rain Doll”)[4,5]作为 AES 算法。目前的工作正处在对 FIPS 草案进行公众评论阶段,2001 年 6 月正式出台并使用。为了让更多的读者了解 AES 算法,本文简要介绍 AES 算法的遴选过程,并对 AES 的基本设计思想进行分析讨论,同时对 AES 算法的最新进展进行评述。

二、AES 评选进程

最早考虑用 AES 代替 DES 是在 20 世纪 90 年代初期,表 1 .13 - 1 给出 AES 产生的详细时间表。

表 1 .13 - 1　AES 评选时间表

时　间	活　动	评　注
90 年代早期	考虑 AES 问题	
1993	指出 AES 代替 DES 问题	在 1993 年 DES 评估会上
1997 .1 .2	发布 AES-FIPS 研发计划	1996 年已开始准备
1997 .9 .12	征集候选算法	密码界的广泛响应(共 21 个)
1998 .8 .20	第一次 AES 会议召开(AES1, Califomia)	公布 15 个候选算法,第一轮开始
1999 .6	第二次 AES 会议召开(AES2, Rome)	分析 15 个候选算法
1999 .8	选出 5 个进行第二轮评估	第二轮开始
2000 .4	第三次 AES 会议召开(AES3, New York)	对 5 个候选算法进行分析
2000 .10 2	宣布 NIST 的评选结果	Rijndael 获胜
2000 .11	AES 发布 FIPS 草案	对 FIPS 草案进行公众评论

注:(FIPS) Federal Information Processing Standards(联邦信息处理标准)。

现在 AES 暂时还不能正式成为美国政府的加密标准。NIST 仅仅宣布此算法被列为 FIPS 的一个新草案,用以征集公众的评论,之后还要进行标准的修订。如果合适,才会被采纳为正式的政府标准来维护商业的安全。AES 以后的发展有以下两个阶段:2001 年 2 月,评述期结束;2001 年 4～6 月,AES FIPS 正式使用,并通过规范测试。这个时间阶段可能有所变化,估计在 2001 年春季的某一时间。以后 NIST 继续关注 AES,并每 5 年对其进行一次评估。

三、5 个 AES 候选算法

第三次 AES 会议于 2000 年 4 月在纽约召开,接受了提交的 37 篇论文及报告以及 136 组公众评论意见[2,6,7,11]。这次会议按照 1997 年制定的基本评估标准,从安全性、成本、算法实现特性等方面对 5 个最后候选算法[1] (Mars, RC6, Serpent, Twofish, Rijndael)进行了进一步的分析与评估。

Mars 算法是 IBM 公司提供的一个候选算法,特点是充分使用非平衡的 Feistel 网络。为了保证加密和解密的强度相当,Mars 由结构类似的 2 部分组成。Mars 的加密算法由 6 部分组成:① 密钥加;② 不受密钥控制的 8 轮前期混合运算;③ 密钥控制下的 8 轮前期加密变换;④ 密钥控制下的 8 轮后期加密变换;⑤ 不受密钥控制的 8 轮后期混合运算;⑥ 密钥减。从现有的分析结果来看,Mars 对现有的密码分析方法是免疫的,其缺陷是有弱密钥,且速度相对较慢。

RC6 算法是在 RC5 算法的基础上设计的。众所周知,RC5 是一个非常简洁的算法,它的特点是大量使用数据依赖循环。RC6 继承了这些优点。为了满足 NIST 的要求,即分组长度为 128 位,RC6 使用了 4 个寄存器,并加进 32 位的整数乘法,即二次函数 $B\times(2B+1)$(相应为 $D\times(2D+1)$),用于加强扩散特性。

关于 RC6 的分析结果可归纳如下:

(1) 对 RC6 的最好攻击似乎是穷举搜索用户的密钥。

(2) 对 RC6 进行差分和线性密码分析,所需的数据超过现有的数据。

值得注意的是,由于使用了 32 位的整数乘法,RC6 的速度可能受到影响。

Serpent 算法是 Anderson、Biham 和 Kundsen 提交的一个候选算法。它采用的是代替/ 置换网络。在 Serpent 的最初版本中,使用了 DES 的 S-盒,目的是使公众相信设计者没有设置任何陷门。对于 Serpent 有类似的保证,这是因为 S-盒以简单的、确定的方式生成。Serpent 的加密算法由 3 部分组成: ① 初始置换; ② 32 轮的加密操作,每一轮包含密钥混合运算、S-盒及线性变换; ③ 末尾置换。

Twofish 算法是美国的 Bruce Schneier 等人提交的一个候选算法。它的总体结构是一个 16 轮的 Feistel 结构,主要特点是 S-盒由密钥控制。Twofish 的加密分 3 部分:第 1 部分是初始变换;第 2 部分是 16 轮的加密;第 3 部分是末尾变换。关于 Twofish 的分析结果,公开的最佳攻击是用 $2^{22.5}$ 个选择明文和 2^{51} 的计算量攻破 5 轮 Twofish。

在第三次 AES 会议上,尽管并未完全达成一致,但与会者认为, Rijndael 按时间测试原理设计简单,在各种测试环境下总体性能良好。而且,其安全强度指标经过各种算法分析与密码攻击证实是相当高的。惟一不足的是加密轮数还需提高,以保证更好的安全性(注:算法在不同的密钥长度下采用不同的加密轮数)。在这次会议上,一般都认为 Rijndael 会最终胜出。表 1.13-2 给出与会者对最后 5 种候选算法的一般看法。

表 1.13-2 对最后 5 种候选算法的一般看法

加密算法	提交单位(个人)	主要特性	实现复杂性
Mars	IBM	安全强度高	复杂 密钥生成快
RC6	RSA 实验室	安全强度低	非常简单 密钥生成一般
Rijndael	Joan Daemen, Vincent Rijmen	设计简单,总体良好,合理的强度	轮数不够 密钥生成一般
Serpent	R. Anderson, E. Biham, L. Knudsen	强度高,设计与分析复杂	软件性能差 密钥生成一般
Twofish	Bruce Schneier 等	强度高与复杂性	比较复杂 密钥生成慢

四、AES——Rijndael 算法思想与特点评述

不出所料,第三次 AES 会议后, NIST 于 2000 年 10 月 2 日宣布选择 Rijndael 作为 AES 的惟一算法。Rijndael 算法是比利时 Proton World International 的 Joan Daemen 博士和天主教大学电子工程系的 Vincent Rijmen 博士后提交的一种候选算法[5,7]。Rijndael 汇聚了高安全、高性能、高效率、易用和灵活等优点。它是一个迭代分组密码算法,其分组长度和密钥长度都是可变的,但为了满足 AES 的要求,分组长度为 128 位,密钥长度为 128/ 192/ 256 位,相应

的轮数为 10/ 12/ 14。比较而言, AES 的 128 位密钥比 DES 的 56 位密钥多 10^{21} 倍。

该算法的原形是 Square 算法,它的设计策略是宽轨迹策略(wide trail strategy)[1]。宽轨迹策略是针对差分分析和线性分析提出的,最大优点是可以给出算法的最佳差分特征的概率以及最佳线性逼近的偏差的界。由此,可以分析算法抗击差分密码分析及线性密码分析的能力。

Rijndael 采用的是代替/ 置换网络。每一轮由 3 层组成: ① 线性混合层,确保多轮之上的高度扩散; ② 非线性层,由 16 个 S-盒并置而成,起到混淆的作用; ③ 密钥加层,子密钥简单地异或到中间状态上。S-盒选取的是有限域 GF(28)中的乘法逆运算,它的差分均匀性和线性偏差都达到了最佳。

Rijndael 的软件实现测试速度[6] 如表 1 .13-3、表 1 .13-4 所列。

表 1 .13-3　软件加密速度(32 位处理器,C)

A		B		C		D		E	
Clocks	Norm	Clocks	Norm	Clocks	Norm	Clocks	Norm	Clocks	Norm
237	0 .94	1276	0 98	805	0 .40	52 .6	0 .42	362	0 .74
				981	0 .32	44 .3	0 .35	428	0 .63
				1155	0 .28	38 .2	0 .30	503	0 .53

表 1 .13-4　软件解密速度(32 位处理器,C)

B		C		D		E	
Clocks	Norm	Clocks	Norm	Clocks	Norm	Clocks	Norm
1276	0 91	784	0 .39	57 .1	0 .46	358	0 .65
		955	0 .32	47 .9	0 .39	421	0 .55
		1121	0 .23	41 .3	0 .33	492	0 .47

A: Intel Pentium Ⅱ; B: Linux/ GCC-2 .7 .2 .2/ Pentium 133MHz MMX;
C: Intel Pentium Ⅲ 600MHz; D: Apple G4 PowerPC; E: Intel Pentium Ⅱ/ Ⅲ
Clocks 指实际的加密性能指标。通常用时钟周期数(clock cycles)来表示。
Norm 指数据的相对比较指标。最大取值为 1 .00,表示加密速度最快。亦即,以最大的时钟周期数(加密算法最快的)指标作为被除数,每种算法的时钟周期数作为除数,得值就是 Norm 值。

Rijndael 的安全性[7,11] 是:4 轮 Rijndael 的最佳差分特征的概率及最佳线性逼近的偏差分别为 2 - 150 和 2 - 76;8 轮 Rijndael 的最佳差分特征的概率及最佳线性逼近的偏差分别为 2 - 300 和 2 - 151。“ Square ”攻击是针对 Square 算法提出的一种攻击方法,它对 Rijndael 也是适用的。分析结果显示,7 轮以上的 Rijndael 对“ Square ”攻击是免疫的。Rijndael 可以抵御功率(power)与计时(timing)攻击。但有一些评论批评 Rijndael 的数学结构[8]。另外,当同时进行加解密运算时,ROM 需求也随之增加。

全方位考虑,正如 NIST 所称,Rijndael 汇聚了所有的优点,使它成为 AES 最合适的选择,尤其是它在无论有无反馈模式的计算环境下的硬、软件中都能显示出其非常好的性能。它的密钥安装时间很好,也具有好的灵敏度。它非常低的内存需求也使其很适合用于受限环境

中。Rijndael 的操作简单,并可抵御强大和实时的攻击。此外,它还有许多未被特别强调的防御性能。

Rijndael 在数据块和密钥长度的设计上也很灵活,算法可提供不同的迭代次数,尽管这些特征还需更深入地研究,短期内不可能被利用。在将来,Rijndael 内在的迭代结构很可能会显示其潜能,以防御入侵行为。

NIST 在其报告中称:“所有这 5 种算法对 AES 都很安全”。NIST 并未说其他 4 种算法有何不妥,然而,当经过大量的分析及评估后,NIST 最终选择了 Rijndael。基于公众的评述,NIST 决定不从其他 4 种算法中选出一个“备份”算法。NIST 认为在公众评述期如果对多个算法进行讨论会增加无谓的资源浪费,而公众对某一算法的详细评述要比评估这些算法哪个好更重要。

NIST 只选一种算法的原因有以下几点:

(1) 其他 FIPS 算法(如 Tripple DES)提供了一定程度的系统弹性,这给 AES 提出了新问题;

(2) AES 的多种密钥长度加强了安全程度;

(3) 单 AES 算法会促进研发队伍相互间的协作而减少操作复杂性,并能降低费用;

(4) 单 AES 算法使供应商减少了智力资源的浪费。

一些供应商也表示如果过于关注研发一个备份算法,并使之立即成为产品出售,会导致高额的费用,也会潜在地降低 AES 产品开发的协作性。然而,NIST 预测其他算法将用于商业产品而被继续开发。没人知道 AES 或其他的加密算法会持续多久。从历史来看,NIST 的第一代数据加密标准(DES)作为一个美国政府标准,直到被庞大的电脑并行网络攻击和特定的“DES Cracking”硬件攻克时,它已连续使用了 20 年。AES 比 DES 支持更长的密钥。假如能建造一个 DES-Cracking 机,能以每秒 2^{55} 个密钥进行 DES 密钥搜索,则需 149 万亿年才能搜索完 128 位 AES 密钥。除非一些对 AES 的攻击速度比密钥穷尽搜索(Key exhaustion)要快得多,否则 AES 应能保持超过 20 年之久的安全。

五、结束语

基于对 AES 算法的产生过程和设计思想的认识,我们相信,随着 AES 的正式使用,将不断产生对 AES 算法的各种分析与攻击结果,AES 的安全性、有效性和生命力也有待于在实践中进一步得到检验。

参 考 文 献

1 吴文玲,冯登国,卿斯汉 .简评美国公布的 15 个 AES 候选算法[J] .软件学报,1999,19(3):23～25

2 Raghavan N Srinivas . AES: Who Won ?Discover the Results of the Advanced Encryption Standard Contest [M] .2000

3 Raghavan N Srinivas . AES: Cryptography Advances into the Future[J] . Java World,2000,(4)

4 AES Home Page [EB/ OL] . http:// www .nist .gov/ aes

5 The Rijndael Homepage [EB/ OL] . http:// www .esat .kuleuven .ac .be/ ～rijmen/ rijndael

6 Susan Landau . Communications Security for the Twenty-first Century: Teh Advanced Encryption Standard [EB/ OL] . http:// www .ams .org/ notices/ 200004/ fea-landau .pdf

7 J Nechvatal, E Barker, et al . Report on the Development of the Advanced Encryption Standard (AES) [EB/ OL] . http:// csrc .nist .gov/ encryption/ aes/ index .html

8 The Original Description of Rijndael [EB/ OL] . http:// www .esat .kuleuven .ac .be/ ～rijmen/ rijndael/

9 Murphy S, Robshaw M . Further Comments on the Structure of Rijndael, AES Forum Comment [EB/ OL] . http:// www .cs .rbbnc .ac .uk/ ～sean/ .2000-08-17

10 urphy S, Robshaw M . New Observation on Rijndael, AES Forum Comment [EB/ OL] . http:// www . cs .rbbnc .ac .uk/ sean/ .2000-08-07

11 AES Discussion Forum [EB/ OL] . http:// aes .nist .gov/ aes/

1 .14 基于单片机的实时 3DES 加密算法的实现

长沙国防科技大学电子工程学院(410073) 郑 磊 易 波

一、引 言

现代社会已进入信息时代,社会的发展越来越需要信息安全技术的广泛应用。而单片机技术是迄今为止发展较为成熟且应用非常广泛的一门技术,如果能够将单片机技术和信息安全技术紧密结合起来,将对信息安全技术的广泛应用起到强有力的推动作用。

数据加密技术是信息安全技术的核心,许多优秀、成熟的加密算法(如 3DES[1]、RSA[2])的运算复杂度都非常大。由于单片机的工作频率一般不高,机器字长较短,指令的位运算功能不强,因此使用它来实现这些算法时运算速度通常无法满足实时应用的要求。如何提高用单片机实现这些算法时的实时运算速度已成为实现这一结合急待解决的问题。本文针对这一问题给出了用单片机来实现 3DES 加密算法时提高实时运算速度的两种方法。

二、3DES 加密算法简介

3DES(又名 3 重 DES)加密算法是基于 DES[2] (Data Encryption Standard)且强度更高的一种加密算法,其算法可以用式(1)和式(2)表示。令 DES_K 表示密钥 K 作用下的加密变换,且 DES_K^{-1} 为相应的解密变换。128 位密钥 K 被分成密钥 K_L (K 的左边 64 位)和 K_R (K 的右边 64 位)。明文 M 被加密成

$$C = DES_{KL}[DES_{KR}^{-1}[DES_{KL}(M)]] \tag{1}$$

密文 C 的解密过程是加密的逆过程

$$M = DES_{KL}^{-1}[DES_{KR}[DES_{KL}^{-1}(C)]] \tag{2}$$

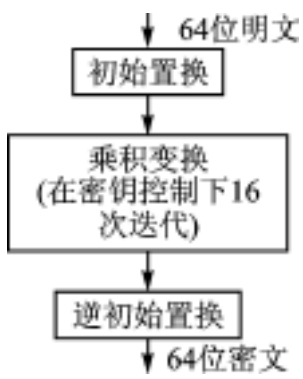


图 1 .14－1 DES 算法框图

由式(1)和式(2)不难看出,提高 3DES 运算速度的关键是提高 DES 算法的运算速度。DES 算法的运算步骤如图 1 .14－1 所示。其中耗时最多的是乘积变换部分(其原理见图1 .14－2,图中 L_{I-1} 和 R_{I-1} 分别是进行一次迭代时的左边 32 位和右边 32 位)。如果能减少这部分的用时,则整个算法的运算速度将得以提高。下面给出了用单片机来实现该算法时提高运算速度的两种方法。

三、通过改进选择压缩运算 S 的查找策略来提高 3DES 的运算速度

传统的选择压缩运算 S 将前面送来的 48 位数据自左至右分成 8 组,每组 6 位,而后并行送入 8 个 S 盒,每个 S 盒输出 4 位。以 S1(见表 1 .14－1)为例,当输入的数据为 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ 时,先找到 $x_6\ x_1$ 对应的 S1 中的一行,再找 $x_5\ x_4\ x_3\ x_2$ 对应的 S1 中的一列,行列交叉处的数据即为 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ 经过 S1 的输出,这种查找方法是二维查找。用单片机实现这一过程时,首先要从 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ 这 6 个比特位中提取 $x_6\ x_1$ 这两位,分别计算出 $x_6\ x_1$ 和 $x_5\ x_4\ x_3\ x_2$ 的值,然后在表 S1 中进行二维查找。由于用单片机实现位提取时要用到多句的位运算指令,这就使得在完成乘积变换的 16 次迭代时要在这部分耗去大量的时间,而且二维查找也使算法执行速度降低。

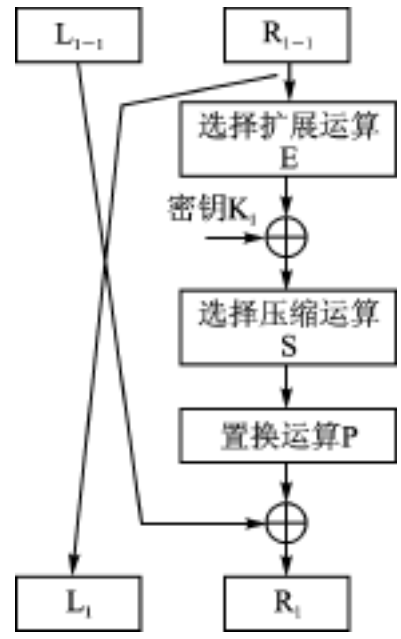


图 1 .14－2　乘积变换框图(I = 1～16)

表 1 .14－1　S1 函数

行＼列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	2	14	10	0	6	13

通过改进选择压缩运算 S 中的查找策略,可以大量减少位运算且将原来的二维查找变为一维查找,而运算结果与原来的选择压缩运算 S 的运算结果完全一致。在选择压缩运算 S 中改进后的查找策略是将原有的各 S 盒重新排列,得到新的 S′盒,其输出是输入的线性函数且与 S 盒等价。具体的排列方法是这样的:以 S1 为例,将 S1 表中的每个值按其对应的二进制数 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ (x_6 是最高位,x_1 是最低位)按由小到大的顺序排列成一个一维数组 S_1'(见表 1 .14－2),当输入为 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ 时,输出为 $S1'[x_6\ x_5\ x_4\ x_3\ x_2\ x_1]$。例如,输入 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ = 101100 时,按以前的查找方法将在 S1 的第二行($x_6\ x_1$ = 10)第六列($x_5\ x_4\ x_3\ x_2$ = 0110)交叉处找到输出值 2;而采用新的查找方法只需由 $x_6\ x_5\ x_4\ x_3\ x_2\ x_1$ = 101100 = 38 在 S_1'中找到 $S_1'[38]$ = 2。其他 7 个 S 盒也采用类似方法进行重排。

新的查找方法不需要进行位提取,这就省去了大量的位操作,而且新的查找方法进行的是一维查找,因此使得整个运算速度得以提高。在用单片机实现加密算法时,可以将重新排列得到的 S_1'盒按一维数组的方式存放在内部 ROM 中,每个 S_1'盒占用 ROM 中连续的 64B。

表 1.14-2 S_1'函数

输 入 $x_6x_5x_4x_3x_2x_1$	输 出	输 入 $x_6x_5x_4x_3x_2x_1$	输 出	输 入 $x_6x_5x_4x_3x_2x_1$	输 出	输 入 $x_6x_5x_4x_3x_2x_1$	输 出
000000	14	010000	3	100000	4	110000	15
000001	0	010001	10	100001	15	110001	5
000010	4	010010	10	100010	1	110010	12
000011	15	010011	6	100011	12	110011	11
000100	13	010100	6	100100	14	110100	9
000101	7	010101	12	100101	8	110101	3
000110	1	010110	12	100110	8	110110	7
000111	4	010111	11	100111	2	110111	14
001000	2	011000	5	101000	13	111000	3
001001	14	011001	9	101001	4	111001	10
001010	15	011010	9	101010	6	111010	10
001011	2	011011	5	101011	9	111011	0
001100	11	011100	0	101100	2	111100	5
001101	13	011101	3	101101	1	111101	6
001110	8	011110	7	101110	11	111110	0
001111	1	011111	8	101111	7	111111	13

四、采用数据并行存放的方式来提高运算速度

通常人们使用硬件来实现 3DES 加密算法时,明文、密文以及中间数据都是按位连续存放的。如 64 位数据 $d_1\ d_2\ d_3 \cdots d_{63}\ d_{64}$ 在 RAM 中的存放方式见表 1.14-3。

表 1.14-3 一般的数据存放方式

$d_1\ d_2\ d_3\ d_4\ d_5\ d_6\ d_7\ d_8$	$d_9\ d_{10}\ d_{11}\ d_{12}\ d_{13}\ d_{14}\ d_{15}\ d_{16}$	…	$d_{57}\ d_{58}\ d_{59}\ d_{60}\ d_{61}\ d_{62}\ d_{63}\ d_{64}$
第一字节	第二字节	…	第八字节

这种存放方式使得在乘积变换中对数据进行选择扩展运算,E 和置换运算 P 时要用到大量的位运算,从而严重影响了 3DES 的运算速度,限制了它的实时应用。

我们将采用一种新的数据存放方式,即将每组数据的 64 位分开分别存放到 64B 中,属于同一组数据的各个位占用每个字节中相同位置的位,这样 64B 可以并行存放 8 组数据。例如有 8 组数据 a～h,每个字母代表一组 64 位的数据,属于同一组数据的不同位用字母加下标表示,这样这 8 组数据的存放方式如表 1.14-4 所列。

表 1.14-4 改进后的数据存放方式

字 节	高 位 低 位
第 1 字节	$a_1\ b_1\ c_1\ d_1\ e_1\ f_1\ g_1\ h_1$
第 2 字节	$a_2\ b_2\ c_2\ d_2\ e_2\ f_2\ g_2\ h_2$
第 3 字节	$a_3\ b_3\ c_3\ d_3\ e_3\ f_3\ g_3\ h_3$
第 4 字节	$a_4\ b_4\ c_4\ d_4\ e_4\ f_4\ g_4\ h_4$
…	…
第 63 字节	$a_{63}\ b_{63}\ c_{63}\ d_{63}\ e_{63}\ f_{63}\ g_{63}\ h_{63}$
第 64 字节	$a_{64}\ b_{64}\ c_{64}\ d_{64}\ e_{64}\ f_{64}\ g_{64}\ h_{64}$

数据采用这种方式存放后,在进行选择扩展运算 E 和置换运算 P 时将不必像以往那样频繁地使用位操作指令,替代它们的是简捷的字节交换;而且由于数据是按位并行存储的,所以每做一次换位运算时 8 组数据同时得到了处理。这种并行处理的方式大大提高了整个算法的运算速度。数据采用这种方式存放后在进行选择压缩运算 S 时,先要从不同的字节中提出属于同一组数据的连续 6 个比特位,然后计算出它们表示的二进制值,因此使得采用这种方式存放后的选择压缩运算 S 耗时比以前略长,但是由于有并行处理的优势(对于机器字长越长的单片机这种优势越明显,当机器字长为 16 位时可以并行处理 16 组数据),整个 3DES 算法的速度还是有了很大的提高。

五、实际测试结果

笔者用 AT89C51[3] 实现 3DES 加密算法时(使用 8M 的外部时钟),采用以上介绍的两种方法后,对 64B 明文进行加密所用的时间由以前的 150 ms 提高到 12 ms(也即 42 .7k bit/ s),完全满足了通过调制解调器进行串行通信的需要。

由以上的分析和实际测试的结果我们可看出,本文给出的两种方法切实可行,而且可以作为用单片机实现别的加密算法时提高运算速度的借鉴。

参　考　文　献

1　刘尊全著 .刘氏高强度公开加密算法设计原理与装置 .北京:清华大学出版社,1998

2　王育民,何大可著 .保密学——基础与应用 .西安:西安电子科技大学出版社,1990

3　余永权著 .ATMEL89 系列(MCS－51 兼容)Flash 单片机原理及应用 .北京:电子工业出版社,1997

选自《微处理机》季刊,2000 年第 3 期

1 .15 ATA 接口技术

南京工程兵工程学院计算机教研室(210007) 伍红兵 沈鑫剡

一、ATA 简介

硬盘接口 IDE(Integrated Drive Electronies)也称 AT 总线接口,是当前新型硬盘驱动器普遍采用的一种接口。它最早由 Texan 和 Compaq 公司提出,目的是把硬盘控制器嵌入到驱动器中。1988 年 10 月,ANSI 中的 X3T9 .2 工作组的一个委员会开始讨论 IDE 的有关问题。1993 年 2 月发表了该标准的 3 .1 版本,使其成为正式的 ANSI 标准,并赋予了一个新的名称——ATA(AT Attachment)。从概念上说,ATA 与 IDE 具有基本相同的含义。

随着软件要求的提高及硬盘驱动器技术的进步,ATA 标准也在不断地改进,近几年出现了以下几个新版本。

(1) ATA—2

这一标准兼容 ATA,并扩展了一些功能,它不仅增加了 faster PIO 模式和 DMA 模式,而且提高了即插即用性及与未来版本的兼容性。同时它还增加了一种新的寻址方式——LBA。

(2) ATA—3

这是最新的版本,其主要特征是:提高了稳定性,尤其是 PIO mode 4 的稳定性;提供了一种简单的基于口令的安全方案;更复杂的电源管理;自我监视、分析、报告技术(S .M .A .R .T .),使得驱动器能警告用户一些紧急错误。ATA—3 并没有定义一种更快的模式。虽然一些设备厂商推出"mode 5"的设备,但并不是指一种比 ATA—2 更新的 PIO 模式。

(3) Ultra—ATA

Ultra—ATA 在概念上类似于 Ultra—SCSI,它在目前的 ATA—3 标准和未来的 ATA—4 标准(目前还未完成)之间架起了一座桥梁。它增加了一种新的、高性能的模式——DMA/33,它的带宽位为 33 MB/ s,2 倍于 DMA mode 2。Intel 的 Pentium Ⅱ芯片组和 TX Pentium 芯片组支持这种模式。

值得一提的是以下 2 个容易混淆的名词——EIDE、Fast—ATA。EIDE(Enhanced IDE)是由 Western Digital 公司提出的一个用于营销的商业名称,它建立在 ATA—2 和 ATAPI 这二个事实标准上。Fast—ATA 则是为了与 WD 公司竞争,由 Seagate 和 Quantum 发布的名称,但它只建立在 ATA—2 标准之上。

二、IDE 接口定义

IDE 接口的 ATA 标准包含信号电缆和电源线的电器特性。该标准考虑到新技术的发展,不仅适用于通用的 5 V 逻辑,也适用于 3 .3 V 的电路逻辑。IDE 电缆长度短于 46 cm,除 DASP、PDIAG、IOCS16 和 SPSYNC: PSEL 信号外,几乎所有信号都使用 TTL 电平线路收发器。IDE 接口引脚定义如表 1 .15 - 1 所列。

表 1.15-1 IDE 接口引脚定义

引 脚	信 号	信号描述	信号方向	引 脚	信 号	信号描述	信号方向
1	RSET	复位	I	2	GND	地	I/O
3	DD7	数据位 7	I/O	4	DD8	数据位 8	I/O
5	DD6	数据位 6	I/O	6	DD9	数据位 9	I/O
7	DD5	数据位 5	I/O	8	DD10	数据位 10	I/O
9	DD4	数据位 4	I/O	10	DD11	数据位 11	I/O
11	DD3	数据位 3	I/O	12	DD12	数据位 12	I/O
13	DD2	数据位 2	I/O	14	DD13	数据位 13	I/O
15	DD1	数据位 1	I/O	16	DD14	数据位 14	I/O
17	DD0	数据位 0	I/O	18	DD15	数据位 15	I/O
19	GND	地		20	N.C	未 用	
21	DMARQ	DMA 请求	O	22	GND	地	
23	DIOW/	写选通	I	24	GND	地	
25	DIOR/	读选通	I	26	GND	地	
27	IORDY	通道就绪	O	28	DPSYNC:CXEL	同步电缆选择	
29	DMACK/	DMA 应答	I	30	GND	地	
31	INTRQ/	中断请求	O	32	IOCS16/	16 为 IO	O
33	DA1	地址 1	I	34	PDIAG/	诊断完成	O
35	DA0	地址 0	I	36	DA2	地址 2	I
37	CS1FX/	片选 0	I	38	CS3FX/	片选 1	I
39	DASP/	驱动器激活	O	40	GND	地	

三、IDE 接口时序

IDE 接口有二种数据传送方式:一种是可编程 I/O(PIO),另一种是 DMA 方式。ATA 标准为 PIO 和 DMA 方式定义了三种工作模式,模式 0 是常用的,也是最慢的一种方式。可以从“确认驱动器”命令的返回参数中了解驱动器当前的工作模式。

(1) 通过 PIO 传送数据:在该方式下,CPU 对控制器的访问都是通过 PIO 进行的,包括从控制器读取状态信息和错误信息,以及向控制器发送命令和参数,时序如图 1.15-1 所示。

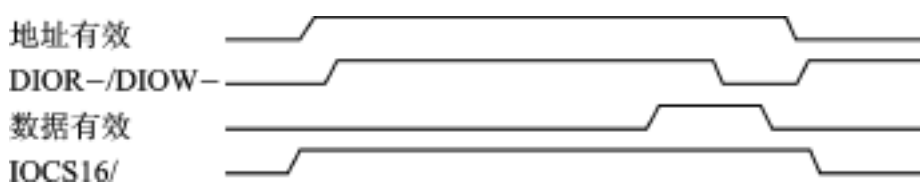


图 1.15-1 IDE 接口时序(PIO 模式)

对于 1 个 PIO 数据传输,CPU 置地址信号 DA0、DA1、DA2、CS1FX/ 、CS3FX/ 有效后(大约 70 ns),DIOW/ 或 DIOR/ 有效,同时,CPU 或控制器把传送数据放置到数据总线上,根据数据传送方向,控制器或 CPU 读取数据线上的数据。操作完成后,释放数据线、地址线和

IOCS16/ 线,这样,一个操作周期结束。

(2) 单字 DMA 方式:这种方式主要用于多用户系统中,在 I/O 操作的准备阶段,CPU 可以处理其他事务,单字 DMA 时序如图 1.15-2 所示。

图 1.15-2 IDE 接口时序(单字 DMA 模式)

当控制器数据准备好后,置 DMARQ 信号,启动 1 次 DMA 操作,DMA 控制器以 DMACK/ 信号作为 DMA 请求的应答信号,同时使 DIOR/ 为低,完成 1 个数据的传输;然后控制器撤销 DMARQ 信号,释放数据线,准备启动下一次操作。

(3) 多字 DMA 方式:这种方式与单字 DMA 方式的操作十分相似,只是在 1 个字传输结束后,控制器仍然保持 DMARQ 信号有效,直到最后 1 个字传输完成。多字 DMA 时序如图 1.15-3 所示。

图 1.15-3 IDE 接口时序(多字 DMA 模式)

四、IDE 寄存器及 IDE 命令

IDE 控制器中有 2 组寄存器:命令寄存器和控制寄存器。命令寄存器被用来接受命令和传送数据;控制寄存器组用作磁盘控制。这 2 个寄存器组用 CS1FX/ 和 CS3FX/ 信号区分。CS1FX/ 的地址范围是 1F0H～1F7H, CS3FX/ 的地址范围是 3F0H～3F7H。具体如表 1.15-2所列。

表 1.15-2 地址译码

信号					地址
CS1FX	CS3FX	DA2	DA1	DA0	
1	0	0	0	0	1F0
1	0	0	0	1	1F1
1	0	0	1	0	1F2
1	0	0	1	1	1F3
1	0	1	0	0	1F4
1	0	1	0	1	1F5
1	0	1	1	0	1F6
1	0	1	1	1	1F7
0	1	1	1	0	3F6
0	1	1	1	1	3F7
0	1	*	*	*	未用

目前,许多计算机配置了 2 个 IDE 接口,对于第 2 个 IDE 接口,这 2 个信号的地址范围分别是 170H～177H 和 370～377H。

在 ATA 标准中以寄存器方式传送数据、命令和地址。这些寄存器除数据寄存器为 16 位以外,其他寄存器均为 8 位。PC 机分配给寄存器的地址有二组,一组为 1F0H～1F8H,另一组为 170H～178H。通常 ATA 适配器采用 IOQ14 中断。

(1) 数据寄存器(1F0H R/ W):这是 1 个 16 位 PIO 数据寄存器,用于对扇区的读、写和格式化操作。CPU 通过该寄存器向硬盘控制寄存器写入或从硬盘控制寄存器读出扇区缓冲区的数据,如使用‘REP OUTSW’或‘REP INSW’指令,通过数据寄存器也可以进行 DMA 方式的数据传输。

(2) 错误寄存器(1F1H R):该寄存器是 1 个 8 位的寄存器,它反映控制寄存器在诊断方式或操作方式下的错误原因。在不同方式下有不同的意义。

① 诊断方式:硬盘控制器在加电、复位或执行驱动器诊断命令以后的工作方式。此时驱动器包含诊断码,该代码反映了诊断后的结果,如表 1.15-3 所列。

表 1.15-3 诊断代码

代 码	意 义	代 码	意 义
10H	无错误	04H	ECC 电路错
02H	控制器错	05H	控制器处理机错
03H	数据缓冲区错	8XH	从驱动器诊断代码

② 操作方式:硬盘控制器执行除诊断命令以外的所有命令后进入该方式。如果状态寄存器的 ERR = 1,则该寄存器包含命令执行后的错误代码。错误寄存器的内容如表 1.15-4 所列。

表 1.15-4 错误寄存器

D7	D6	D5	D4	D3	D2	D1	D0
BBK	UNC	MC	IDNF	MCR	ABRT	TKONF	AMNF

BBK(Bad block detected):坏扇区;

UNC(uncorrectable data error):ECC 错误;

MC(media change):介质改变;

IDNF(ID not found):ID 没找到;

MCR(media change requeted):介质改变请求;

ABRT(abouted command):命令放弃;

TKONF(track 0 not found):0 磁道错;

AMNF(address mark not found):地址标志没找到。

(3) 特性寄存器(1F1H W):一般情况下不使用该寄存器,根据 ATA 标准它被用来设置接口的某些特性。

(4) 扇区数寄存器(1F2 R/ W):它记录读、写命令的扇区数。当多扇区传输时,每完成 1 个扇区操作,该寄存器自动减 1,直至为 0。如果初值为 0,则表示 256。如果有错误发生,该寄存器包含已经操作成功的扇区数。

(5) 扇区号寄存器(1F3H R/ W):它记录读、写和校验命令指定的起始扇区号。如果驱动器使用逻辑块寻址(LBA,logical lock address)方式,则该寄存器记录逻辑扇区号的 0 字节。

(6) 柱面号寄存器:(1F4H 1F5H R/ W):它记录读、写、校验、寻址和格式化命令指定的柱面号,ATA 标准允许 65 536 个柱面,但早期的 IDE 控制器中只允许 1024 个柱面。低 8 位在 1F4H 寄存器中,高 8 位在 1F5H 寄存器中。如果是 LBA 寻址方式,这 2 个寄存器包含起始扇区的 1 和 2 字节。

(7) 驱动器/ 磁头寄存器(1F6H R/ W):它记录读、写、校验、寻道和格式化命令指定的驱动器号、磁头号和寻址方式。其定义如表 1 .15 - 5 所列。

表 1 .15 - 5 驱动器/ 磁头寄存器定义

D7	D6	D5	D4	D3	D2	D1	D0
1	L	1	DRV	HS3	HS2	HS1	HS0

HS0～HS3(磁头选择):在 LBA 方式中,是逻辑扇区号的高 4 位。

DRV(驱动器选择):0 选择主驱动器,1 选择从驱动器。

L(LBA 方式):L = 1,置驱动器为 LBA 模式;L = 0,置驱动器为 CHS 模式。

(8) 状态寄存器(1F7H R):它反映硬盘驱动器执行命令后的状态,读该寄存器要清除中断请求信号,如果要避免清除中断,可以读辅助状态寄存器 3F6H。这 2 个寄存器的内容完全相同,其定义如表 1 .15 - 6 所列。

表 1 .15 - 6 状态寄存器定义

D7	D6	D5	D4	D3	D2	D1	D0
BSY	DRDY	DWF	DSC	DRQ	CORR	IDX	ERR

BSY:驱动器忙;

DRDY:驱动器准备好;

DWF:驱动器失败;

DSC:寻道结束;

DRQ:请求服务,驱动器请求通过寄存器与处理器交换 1 个字节数据;

CORR:当可以纠正的读错误发生时,该位置 1,数据传输继续进行;

IDX:收到索引信号;

ERR:命令执行出错。

(9) 命令寄存器(1F7H W):该寄存器接收处理器输出的 HDC 命令,命令格式和含义如表1 .15 - 7所列。其中 12 种是强制性的(M),其他是选择性的(O)。有些命令有 2 个操作码,后 1 个是早期操作码,有的驱动器上仍在使用,如 CONNER 驱动器。

表 1.15-7 命令代码

命令名	代 码	O/M	命令名	代 码	O/M
承认介质改变	DBH	O	长读(长读-禁止重试)	22H,23H	M
启动	DCH,DDH	O	扇区读校验(禁止重试)	40H,41H	M
检查电源模式	98H,E5H	O	复位	E7H	O
门锁	DEH	O	恢复驱动器状态	EAH	O
开锁	DFH	O	寻道	7XH	M
驱动器诊断	90H	M	设置特性参数	EFH	O
格式化磁道	50H	M	设置多扇区模式	C6H	O
确认驱动器	ECH	O	设置休眠模式	99H,E6H	O
空闲	97H,E3H	O	设置准备模式	96H,E2H	O
立即进入空闲	95H,E1H	O	立即进入准备模式	94H,E0H	O
初始化驱动器参数	91H	M	写扇区缓冲区	E8H	O
预补偿	1XH	M	DMA 写(DMA 写-禁止重试)	CAH,CBH	O
读扇区缓冲区	E4H	O	多扇区写	C5H	O
DMA 读(DMA 读-禁止重试)	C8H,09H	O	部分写	E9H	O
读驱动器状态	E9H	O	写扇区(写扇区-禁止重试)	30H,31H	M
多扇区读	C4H	O	长写(长写-禁止重试)	32H,33H	M
读扇区(读扇区-禁止重试)	20H,21H	M	写校验	3CH	O

(10) 辅助状态寄存器(3F6H R):它包含与状态寄存器相同的内容,但读该寄存器时不会清除中断请求信号。

(11) 硬盘控制寄存器(3F6H W):该寄存器定义如表 1.15-8 所列。

表 1.15-8 硬盘控制寄存器

D7	D6	D5	D4	D3	D2	D1	D0
—	—	—	—	1	SRST	IEN/	0

SRST:软件复位,当该位为 1 时,复位驱动器。

IEN/ :中断允许位,该位为 0 时,允许中断。

(12) 驱动器地址寄存器(3F7H R):该寄存器包含命令执行后的某些信息,它与软盘驱动器共享,D7 位是软盘驱动器的更换磁盘位。寄存器的所有位都是负逻辑。其定义如表 1.15-9所列。

表 1.15-9 驱动器地址寄存器

D7	D6	D5	D4	D3	D2	D1	D0
—	WTG/	HS3/	HS2/	HS1/	HS0/	DS1/	DS0/

WTG/ :写操作门控信号;

HS3/ ～HS0/ :当前选择头号的反码;

DS1/：该位为 0,选从驱动器;

DS0/：该位为 0,选主驱动器。

五、硬盘的编址方式

ATA 标准允许 65 536 个柱面,每个扇区 512 B。扇区寻址有 2 种方式:物理寻址方式和逻辑寻址方式。

物理寻址方式(CHS 方式):用柱面、磁头和扇区号表示 1 个特定的扇区。起始扇区是 0 磁道、0 磁头、1 扇区,接下来是 2 扇区,直到 E0F 扇区,接下来是同一柱面 1 头、1 扇区……。

逻辑地址方式(LBA 方式):对于参数寄存器来说,其柱面值最大为 65 536,磁头数最大为 16,扇区最大为 255。因此参数寄存器可支持的最大磁盘容量为 65 536 × 16 × 255 = 13 .69 GB。而 BIOS 所能支持的最大值分别为 1 024、255、63,支持的最大容量为 1 024×255 ×63 = 8 .4 GB。

IDE 接口对磁盘的容量限制是由 BIOS 和参数寄存器二者结合产生的。因此柱面、磁头、扇区数被限制在 1 024、16、63,最大只能支持 1 024×16×63 = 528 MB。采用逻辑块方式寻址可以突破 528 MB 的容量限制。该方式以 28 位的宽度可寻址到 2^{28} = 268 435 455 块扇区,容量达 137 GB。如表 1 .15 - 10 所列。

表 1 .15 - 10 LBA 编址

寄存器名	D7	D6	D5	D4	D3	D2	D1	D0
起始扇区寄存器(1F3H)	LBA7	LBA6	LBA5	LBA4	LBA3	LBA2	LBA1	LBA0
柱面号低字节寄存器(1F4H)	LBA15	LBA14	LBA13	LBA12	LBA11	LBA10	LBA9	LBA8
柱面号高字节寄存器(1F5H)	LBA23	LBA22	LBA21	LBA20	LBA19	LBA18	LBA17	LBA16
驱动器/ 磁头号寄存器(1F6H)	1	LBA	1	DRV	LBA27	LBA26	LBA25	LBA24

逻辑块地址与物理地址的关系为:

LBA 地址 = (柱面号×磁头数 + 磁头号)×扇区数 + 扇区数 - 1

采用 LBA 方式寻址,没有磁头和磁道的转换操作,在访问连续扇区时,操作速度的物理地址方式要快。

LBA 寻址方式虽然需要 BIOS 做些修改,但它与 Microsoft 和 IBM 的 INT13 功能扩展规范是兼容的。为了能够用 LBA 方式存取大于 528MB 的硬盘,IDE 提供了二种方式供主机系统选择,这二种方式均在 CMOS 中设置。

第 1 种方式称为自动配置,这主要是向那些不用 BIOS 存取磁盘的操作系统(如 Netware 和 UNIX)提供的。它运用标准 DPT(磁盘参数表)从 IDE 驱动器上获得有关驱动器容量的信息传向操作系统。

第 2 种方式是为 DOS 和 Windows 设计的,称为自动转换。在 CMOS 设置中选择这种类型,主机加电初始化时,BIOS 会创建 1 张 EDPT(增强型磁盘参数表)。EDPT 表包含 2 组驱动器参数:一组来自 Identify 命令获取的驱动器信息;另一组则是 BIOS 给操作系统的。这些信息不是驱动器返回信息的简单拷贝,它还要做些转换。根据 Identify 命令返回的信息,BIOS 将实际的 CHS 转换成 IDE 的 CHS 或者转换成 LBA 的 CHS,能够支持的最大磁盘容量为

8 .4 GB,而且 BIOS 改动最小。

六、接口设计

由于 IDE 驱动器的控制器包含在驱动器的内部,使得 IDE 接口设计比较简单。接口卡除了提供 IDE 接口所必需的片选信号和控制信号,还要提供控制对硬盘和软盘共享的 I/O 端口 3F7H 进行访问所必需的逻辑。图 1 .15 - 4、图 1 .15 - 5 为 1 个具有缓冲的 IDE 硬盘接口的电路设计。

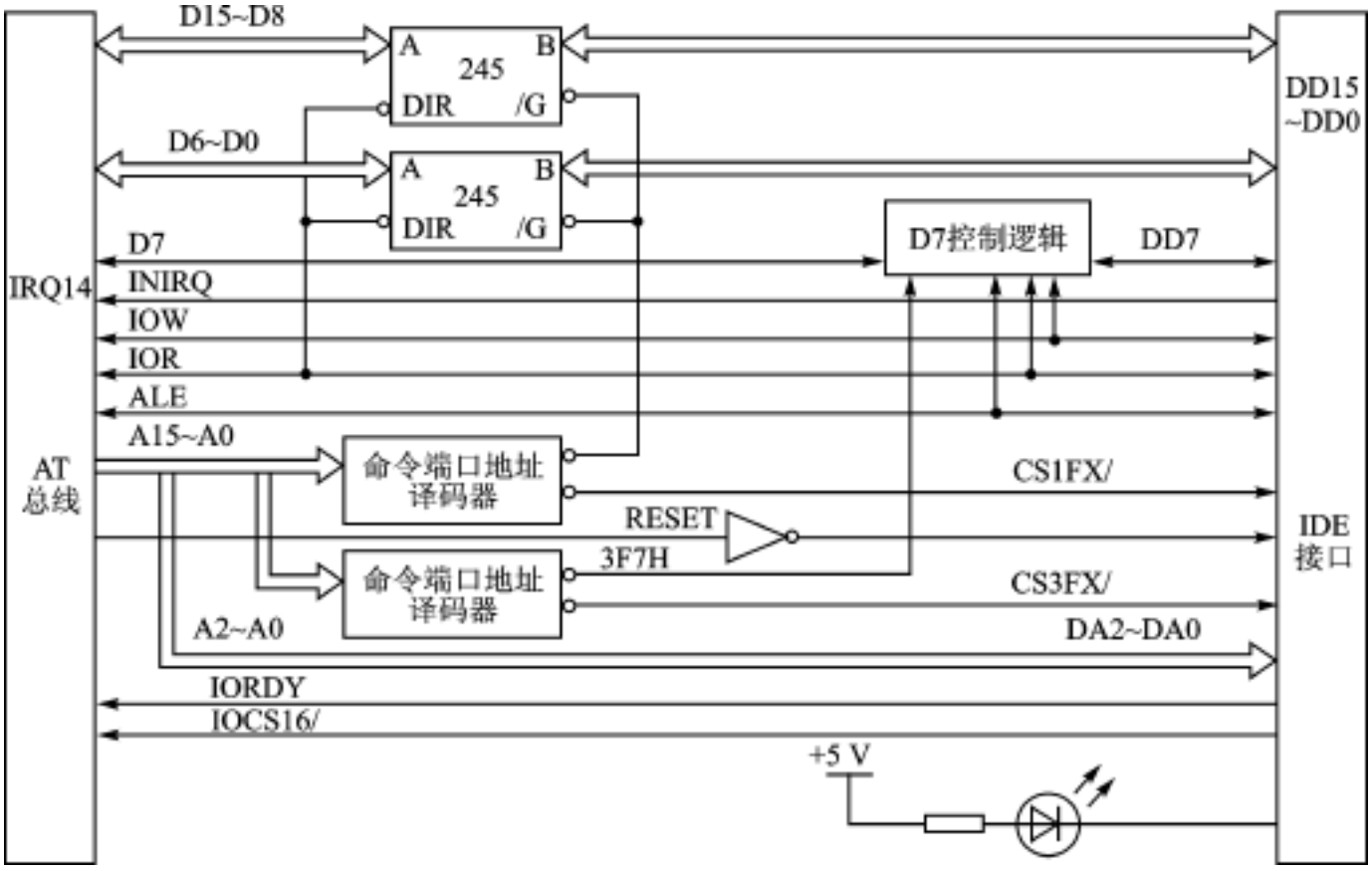


图 1 .15 - 4　具有缓冲的 IDE 硬盘接口的电路

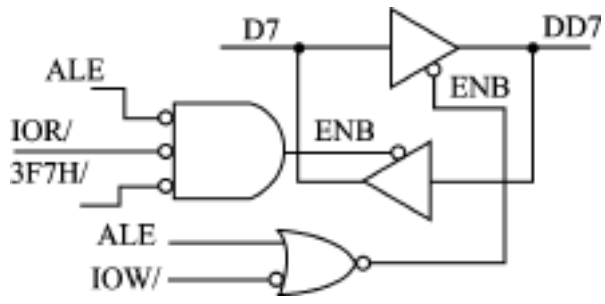


图 1 .15 - 5　D7 控制逻辑

参　考　文　献

1　chmidt F . The SCSI Bus and IDE Interface . AD—DISON—WESLEY,1993

2　张载鸿 .微机接口技术教程 .北京:清华大学出版社,1992

选自《微型机与应用》月刊,2000 年第 5 期

1 .16 基于 IDE 硬盘的高速数据存储器研究

哈尔滨工业大学电子与通信工程系(150001)
吴芝路 任广辉 王桂玲 赵雅琴

随着遥测技术的发展,被测参数迅速增加,数据传输速率越来越高,对系统的存储容量、体积、造价、稳定性等都提出了更高要求。为了实现较高的传输速率和较大的数据量,往往采用多处理机并行处理、传输和记录技术。但这类系统对工作环境要求较高,加之体积大、占用系统机时等缺点,很难适用于受空间限制的特殊环境。因此,研制性能可靠、体积小、造价低的数据存储系统是十分必要的。近年来作为数据存储媒介的硬盘,以其容量大、接口智能化程度高、控制方便越来越受到人们的重视。充分发挥硬盘的优势,脱离系统主机,可为用户开发速度高、存储量大、性能可靠的遥测数据存储系统。

基于上述原因,本文提出了采用单片机控制硬盘对遥测数据进行实时存储的方案,对数千秒遥测数据进行实时无丢失存储,其最大优点是不占用系统机时。本系统特点如下:

· 采用 PIO 模式进行块操作的写盘方法,从而大大提高了硬盘的写盘速度;

· 以单片机为核心,采用 I/O 口与缓冲存储器之间进行高速数据传输的方案,解决了高速遥测数据、CPU 和硬盘三者工作速度不一致的矛盾;

· 采用 EPLD 器件对部分电路进行集成,提高了系统的稳定性和可靠性,具有较好的通用性,可满足多种场合的需要而无需改动任何硬件。

一、系统的硬件介绍

系统分为遥测数据采集和存储两部分。采集部分包括采集控制和串/ 并转换电路;存储部分有:帧计数、读写缓冲地址产生电路、读写控制电路及主存储电路。结构框图如图 1 .16 - 1 所示。

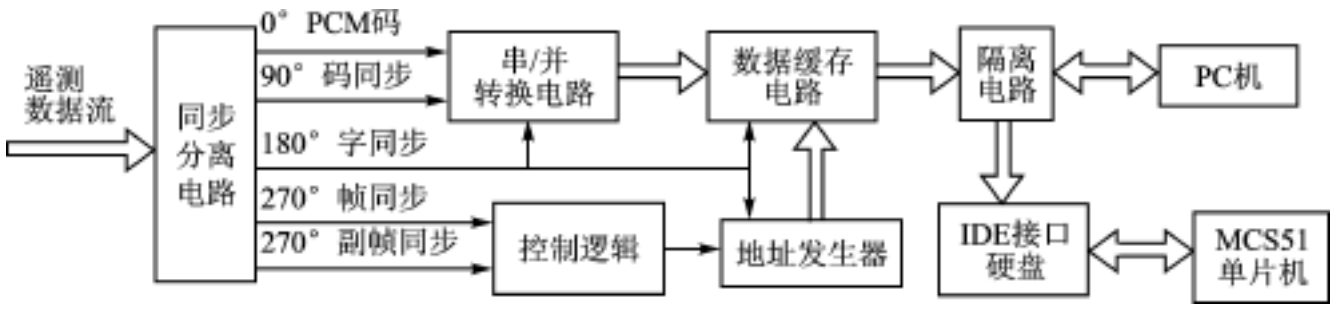


图 1 .16 - 1 系统硬件结构框图

由于硬盘的工作时序与遥测数据的速率不匹配,从数据采集部分得到的并行数据需采用 SRAM 作数据缓存,然后在单片机的控制下,将 SRAM 中的数据直接存入硬盘。而硬盘的寻道时间相对于遥测数据的速率来讲比较慢,因此采用适当的写盘方法,提高硬盘的读写速度是本系统的重点之一。其难点在于如何利用单片机控制硬盘进行读写、复位、检测等操作。

1 . 双片交替式缓冲存储器

由于时序不匹配,遥测数据无法直接存入硬盘,因此需要一个缓冲存储器进行匹配或缓冲,使遥测数据和硬盘存储可以分别按各自不同的时序和速度对缓冲存储器 SRAM 进行操作,解决了不同时序的匹配问题。因此采用缓冲存储方法,将遥测数据存满一定容量的 SRAM,再对其进行块操作存储,可极大地提高存储的速度。为了实现遥测数据无丢失存储,缓冲存储器采用双片交替式,即两片 SRAM 分别交替地被写入遥测数据。首先在地址产生电路控制下将遥测数据写入其中一片 SRAM,写满后发出溢出中断,并且封闭地址产生电路及遥测数据的通路而接通硬盘地址和数据通路,等待 CPU 响应中断后,读取数据存入硬盘。与此同时,另一片 SRAM 的地址和数据端马上被接通到地址产生电路和遥测数据通路上,接受遥测数据的写入。同样写满后发出中断,切换至被读取状态。两片 SRAM 如此交替地工作,连续不断地将遥测数据缓存、写盘,只要写盘所用的时间不长于遥测数据写满 SRAM 的时间,数据就会无丢失地全部存储。

2 . 采用 MCS—51 系列单片机驱动硬盘的方案

由于目前还没有为单片机设计的专用硬盘驱动器及接口电路,利用单片机系统控制现有的硬盘驱动器,可极大地提高系统的性能价格比。因此,本系统采用单片机控制硬盘进行高速数据的存储。

IDE 接口的硬盘驱动器提供了两种数据传输模式:DMA 模式和 PIO 模式。由于 PIO 模式控制相对容易,提供了一种编程控制输入输出的快速传输方法。该模式采用了高速的数据块 I/ O,以扇区为单位,用中断请求方式与处理机进行批量数据交换。在扇区读写操作时,一次按 16 位长度通过内部的高速 PIO 数据寄存器实现传输。通常情况下,数据传输以扇区为单位,每传输一扇区数据产生一个中断。在块模式下以块为单位,在读写一个块期间,硬盘驱动器不产生中断,这样就大大地节省了时间。由于本系统遥测数据的码速率提高到 8 Mb/ s,对硬盘的写盘速度要求非常高,因此本系统采用了块传输模式以提高硬盘的读写速度。硬盘在读写 16 位数据时,PC 机中使用 INSW 指令或 OUTSW 指令实现 16 位数据读/ 写操作,由 I/ O 端口直接到内存。而在本系统中采用 MCS—51 系列单片机控制硬盘,使数据传输在硬盘端口寄存器与缓冲存储器之间进行,不经过单片机,单片机只是对硬盘进行初始化,并发相应的写命令。这样,在单片机的控制下,两片 SRAM 交替工作,实现了高速遥测数据无丢失存储。

二、系统的软件介绍

1 . 硬盘速度测试

硬盘的存取速度是决定 8 Mb/ s 码速率硬盘存储方案可行性的关键。是否存在足够高速的硬盘与系统相配,是本系统方案能否实现的一个决定性问题。但是,利用现有的硬盘测试软件只能比较几种硬盘之间的优劣和差异,而不能比较同一硬盘采用不同写盘方法的速度差异。为了得到一种较快的写盘方法,在对硬盘读写原理进行深入剖析后,自己编程进行测试。以迈拓 4 .3 GB 硬盘为例,采用三种不同的写盘方法:调用 BIOS 的 INT13;利用非块模式和块模式写盘;对硬盘速度进行测试。结果证明:采用 PIO4 及块模式写盘速度最快,可达到 3 .496 MB/ s。完全可以满足本系统的需要。

2 .单片机控制硬盘读写操作

IDE 接口是一种任务寄存器结构的接口,所有输入输出操作均通过对相应寄存器的读/ 写来完成的。如果主机要对硬盘机进行写数据操作,首先进行(命令和参数)寄存器选择,通过数据总线将相应的命令码用 IOW 写入命令寄存器,以及有关参数写入硬盘参数寄存器。数据由数据总线(16 位宽)传递至数据寄存器,通过数据寄存器存入缓存器。IDE 接口会根据命令自动将数据写到由参数寄存器指的磁道号、头号、扇区号。硬盘的读操作与写操作相似,区别在于首先发出中断请求,然后进行数据传输。IDE 控制器端口寄存器地址分配见表 1 .16 - 1。

表 1 .16 - 1 IDE 硬盘控制器端口寄存器的地址分配

CS0	CS1	A2	A1	A0	读操作	写操作
0	1	0	0	0	数据寄存器:扇区读	数据寄存器:扇区写
0	1	0	0	1	错误寄存器:错误状态	写预补偿寄存器
0	1	0	1	0	扇区计数器:扇区数(读、写、检验、格式化)	
0	1	0	1	1	扇区号寄存器:起始扇区(读、写、检验)	
0	1	1	0	0	柱面号寄存器:低字节(读、写、检验、寻道)	
0	1	1	0	1	柱面号寄存器:高字节(读、写、检验、寻道)	
0	1	1	1	0	驱动器/ 磁头寄存器:驱动号/ 磁头号	
0	1	1	1	1	主状态寄存器	命令寄存器

主状态寄存器(CS0 置 0)反映硬盘控制器的操作状态,决定查询状态后的不同流向。定义如下:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
控制器忙碌	驱动器就绪	写故障	寻道结束	数据请求	ECC 纠错	盘片索引	错误

在向控制器发出命令之前,必须先检测控制器是否忙碌(D_7 = 1)。如果在规定时间内控制器一直忙碌,则置超时错,否则表示控制器空闲可接受命令。

设计过程中,采用 PIO 模式以块为单位进行读写操作,从硬盘读数据的过程描述如下:

(1) 在相关寄存器中写入所需的参数,如读取扇区的起始柱面号、磁头号、扇区号、读写扇区数等;

(2) 向命令寄存器中写入命令代码;

(3) 驱动器置 BSY 位,准备数据传输;

(4) 当驱动器准备好数据后,置 DRQ 位,清除 BSY 位,发出中断请求;

(5) 主机检测到中断,读出状态寄存器,测试 ERR 位,若为 1 则转入出错处理,否则循环使用 IN 指令通过数据寄存器读一个扇区或一个块的数据;

(6) 驱动器清除 DRQ 位,如果还有要传输的数据,从第(4)步重复执行。

数据的写入过程与读出过程大致相似,区别在于首先进行数据传输,然后发出中断请求,具体过程不再赘述。PIO 模式编程的简单流程图见图 1 .16 - 2。

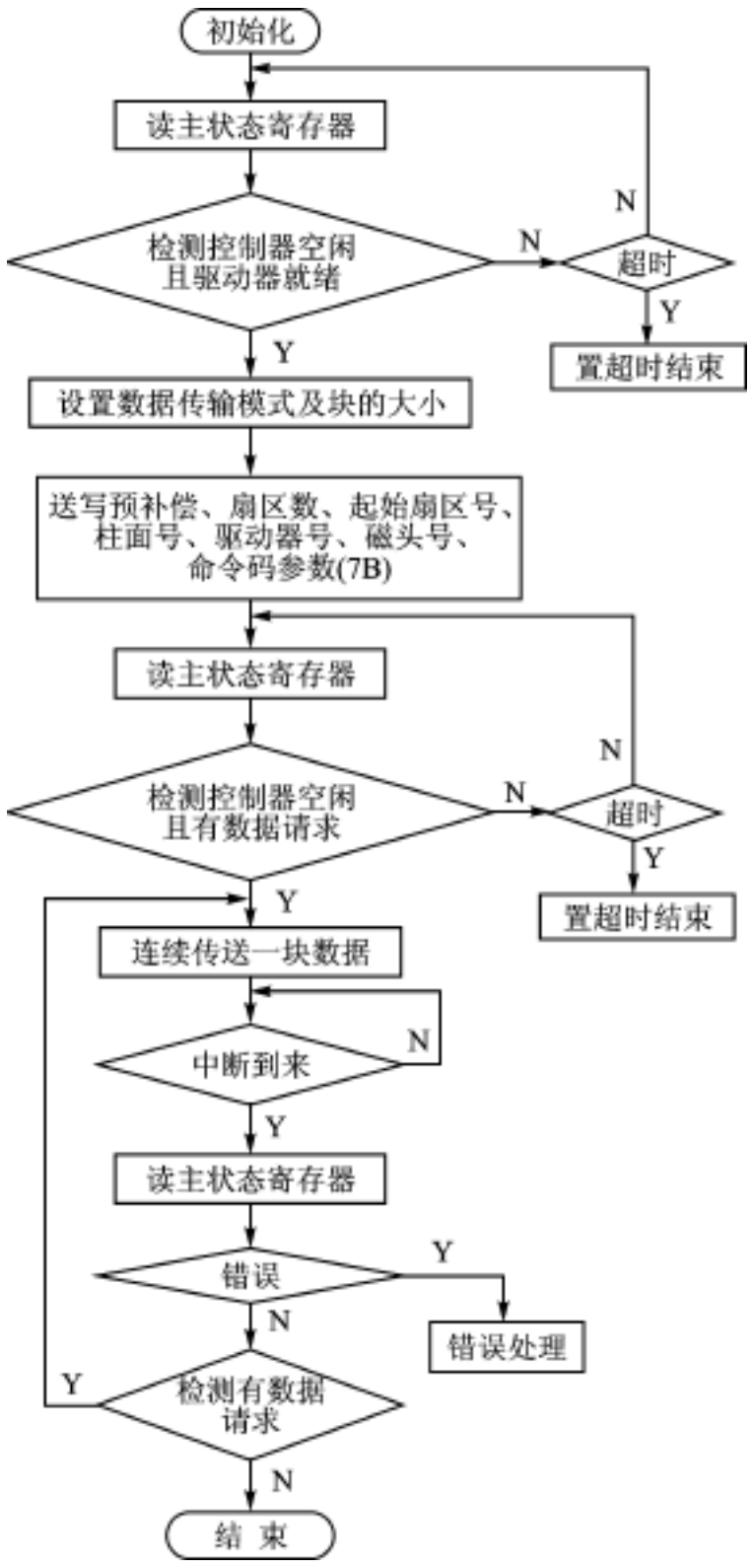


图 1 .16 - 2　采用 PIO 模式写盘软件流程图

实验证明,本存储方案可行,系统运行稳定,实现了遥测数据正确、无丢失存储,并应用到实际系统中。

参　考　文　献

1　aury Wright . Disk Drivers at 40 Lean . Mean Storage Machines, 1996, 41(23)

2　魏梓栋 .硬磁盘机的接口技术 .计算机与通信,1996(9):20～23

3　徐厚骏,等 .IDE 接口和 IDE 硬盘驱动器 .新浪潮,1996(7):17～183

选自《电子技术应用》月刊,2000 年第 5 期

1 .17 模拟比较器的应用

华东地质学院 周航慈 涂水林 张福柳 付 萍 李跃忠

模拟比较器通常用来监测模拟信号的变化情况。如果超过某个限度,就输出一个对应的逻辑信号;如果需要对模拟信号进行更精细的分辨,必须采用 A/ D 转换芯片或者内含 A/ D 部件的单片机来进行 A/ D 转换。当对模拟信号的 A/ D 转换精度要求不是很高(如精度要求在 1%左右),每秒采样次数不超过 20 次时,利用内含模拟比较器的单片机来完成 A/ D 转换将明显降低系统的硬件成本,这在很多家电产品中是非常有意义的。本文采用内含模拟比较器的 P87LPC76X 系列单片机作为讨论对象,在介绍模拟比较器的一般用法后,重点介绍利用模拟比较器完成 A/ D 转换的实现方法和注意事项。

一、模拟比较器用于超限监测

1 . 两路相关模拟信号的监测

将两路动态变化的模拟信号分别输入模拟比较器的正向输入端和反向输入端,从输出端就可以监测到它们之间的相互关系。有时可能需要在输入信号和单片机的模拟输入端之间加入分压电阻,以满足单片机输入端的安全需要和两路信号之间关系判断的需要。这一类问题的设计比较简单,示例如下。

已知条件:信号 A 在 10～20 V 之间变化,信号 B 在 6～15 V 之间变化。正常情况下,信号 B 的幅度小于信号 A 幅度的 80%。

设计要求:当信号 B 达到或超过信号 A 的 80%时,输出低电平(使一个 LED 发光),并在信号 B 达到信号 A 的 80%瞬间触发一个中断子程序,以便在中断子程序中作相关处理。

设计过程:硬件电路如图 1 .17 - 1 所示。由于两路信号均超过了 5 V,不允许直接加到单片机的输入端。为此,信号 A 通过分压电路输入到比较器的正向输入端 CIN1A(P0 .4),信号 B 通过分压电路输入到比较器的反向输入端 CMPREF(P0 .5)。设两路信号的瞬时值分别为 u_A 和 u_B,分压系数分别为 k_A 和 k_B,加到单片机输入端的信号电压最好不要超过 4 V,则:

$$u_A k_A < 4$$

$$u_B k_B < 4$$

将两路信号的最大值代入上式,可以求出两路分压系数取值限度:

$$k_A < 4\ V/ 20\ V = 0\ .200$$

$$k_B < 4\ V/ 15\ V = 0\ .267$$

根据中断触发条件,有:$u_B = 0\ .8\ u_A$,信号 B 达到信号 A 的 80%;$u_B k_B = u_A k_A$,这时加到模拟比较器输入端的信号幅度相同,引起比较器输出端反转。由此得到:$0\ .8\ u_A k_B = u_A k_A$,即 $k_B = 1\ .25\ k_A$。

信号 A 由电阻 R_1 和 R_2 分压,则分压系数 $k_A = R_2/ (R_1 + R_2)$。取 $R_1 = 5\ .1$ kΩ, $R_2 = 1$ kΩ,得到分压系数 $k_A = 1/ (5\ .1 + 1) = 0\ .164$,小于 0 .200,符合安全要求。

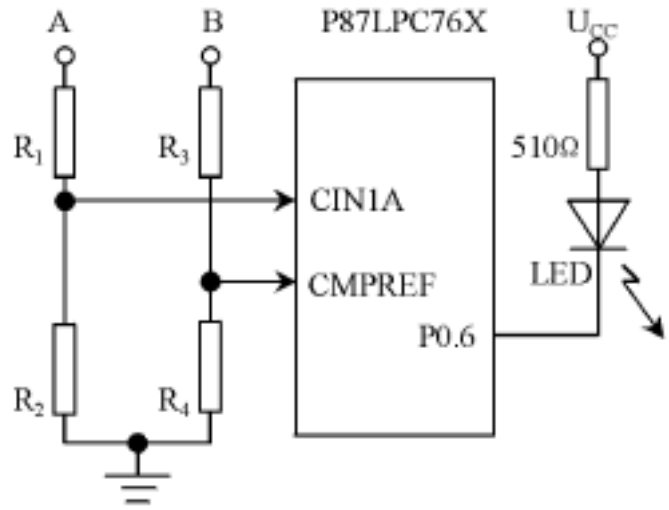


图 1 .17 - 1　两路相关模拟信号的监测电路

信号 B 由电阻 R_3 和 R_4 分压,分压系数 k_B = 1 .25 k_A = 0 .205,也符合安全要求。如果取 R_4 = 1 kΩ,就可以通过方程 $R_4/(R_3 + R_4) = k_B$ 求解出 R_3 = 3 .88 kΩ(取 R_3 = 3 .9 kΩ)。

根据要求,当信号 B 达到或超过信号 A 的 80%时,比较器应该输出低电平。如果采用比较器 1,则信号 A 应该输入比较器 1 的正向输入端 CIN1A(P0 .4),信号 B 应该输入比较器 1 的反向输入端 CMPREF(P0 .5),LED 应该接在它的输出端 CMP1O(P0 .6)。

在程序设计中,有三件事要做:首先,在主程序的初始化部分应该将比较器 1 的输入端设置为模拟输入状态,将比较器 1 的输出信号从 CMP1O 输出,以便控制 LED,设置好比较器 1 的中断功能。其次,在中断向量区填写一条转移指令,引导到比较器 1 的中断子程序。最后,编写比较器 1 的中断子程序。由于比较器输出端的上升沿和下降沿均能触发中断,故在中断子程序中要根据情况分别处理,相关程序如下:

```
CMP1      ATA    ACH          比较器 1 控制寄存器
P0M1      DATA   84H          ;P0 方式寄存器 1
P0M2      DATA   85H          ;P0 方式寄存器 2
PT0AD     DATA   0F6H         ;P0 数据输入禁能
IEN1      DATA   0E8H         ;中断使能寄存器 1
EC1       BIT    IEN1 .5      ;比较器 1 中断使能
CIN1A     BIT    P0 .4        ;比较器 1 输入 A(正向输出端)
CMPREF    BIT    P0 .5        ;比较器参考输入(反向输出端)
CMP1O     BIT    P0 .6        ;比较器 1 输出

          ORG    0000H
          LJMP   MAIN         ;至主程序

          ORG    0063H
          LJMP   CMP1         ;至比较器 1 中断子程序

          ORG    0080H
MAIN:     .                   ;主程序例行操作:自检、初始化等
          .
```

```
        .
        MOV     PT0AD, #30H      ;禁止 CIN1A 和 CMPREF 上的数字输入功能,使能比较器
        ANL     P0M2, #0CFH
        ORL     P0M1, #30H       ;开启比较器 1 并进行如下设置
        MOV     CMP1, #24H       ;正向输入为 CIN1A,反向输入为 CMPREF,结果输出到
                                 ;CMP1O
        MOV     R2, #5           ;延时
DELAY:  DJNZ    R2,DELAY
        ANL     CMP1, #0FEH      ;清除比较器 1 的中断标志
        SETB    EC1              ;使能比较器 1 中断,保持当前中断优先权
        SETB    EA               ;开中断
        .                        ;主程序其他内容
        .
        .

CMP1:   MOV     A,CMP1           ;只响应下降沿引起的中断
        JB      ACC.1,CMP1E
        .                        ;比较器 1 中断子程序,执行相关操作
        .
        .
CMP1E:  ANL     CMP1, #0FEH      ;清除比较器 1 的中断标志
        RETI                     ;中断结束,返回
```

2. 多路独立模拟信号的监测

P87LPC76X 系列单片机虽能有 2 个模拟比较器,可以输入 4 路模拟信号,但只有一个公共的反向输入端,这意味着各路模拟信号必须与同一个参考电压进行比较。如果各路模拟信号的监测电压互不相同,只能通过调节各路信号的分压系数的方法实现多路信号监测。如果选用 1.28 V 的内部参考电压,可以简化外部电路。下面以一个示例说明多路独立模拟信号的监测方法。

有 4 路模拟信号需要监测,条件为:当信号 A 超过 8 V 时,LED1 点亮;当信号 B 超过 2 V 时,LED2 点亮;当信号 C 不足 4 V 时,LED3 点亮;当信号 D 不足 5 V 时,LED4 点亮。

电路设计:硬件电路如图 1.17-2 所示。A 路模拟信号通过电阻 R_1 和 R_2 分压后输入到比较器 1 的正向输入端 CIN1A(P0.4);B 路模拟信号通过电阻 R_3 和 R_4 分压后输入到比较器 1 的正向输入端 CIN1B(P0.3);C 路模拟信号通过电阻 R_5 和 R_6 分压后输入到比较器 2 的正向输入端 CIN2A(P0.2);D 路模拟信号通过电阻 R_7 和 R_8 分压后输入到比较器 2 的正向输入端 CIN2B(P0.1)。P0 口的另外 4 个端口用来驱动 4 个 LED。

各路分压电阻计算:公共参考电压为 1.28 V,取 $R_2 = R_4 = R_6 = R_8 = 1\ k\Omega$。

$R_1 = [(8/1.28) - 1]R_2 = 5.25\ k\Omega$,选用 7.5 kΩ 可调电阻。

$R_3 = [(2/1.28) - 1]R_4 = 0.56\ k\Omega$,选用 750 Ω 可调电阻。

$R_5 = [(4/1.28) - 1]R_6 = 2.13\ k\Omega$,选用 3.3 kΩ 可调电阻。

$R_7 = [(5/1.28) - 1]R_8 = 2.90\ k\Omega$,选用 3.3 kΩ 可调电阻。

由分压系数计算出来的电阻值一般都不是标称值,即使正好是标称值也不能直接采用,因

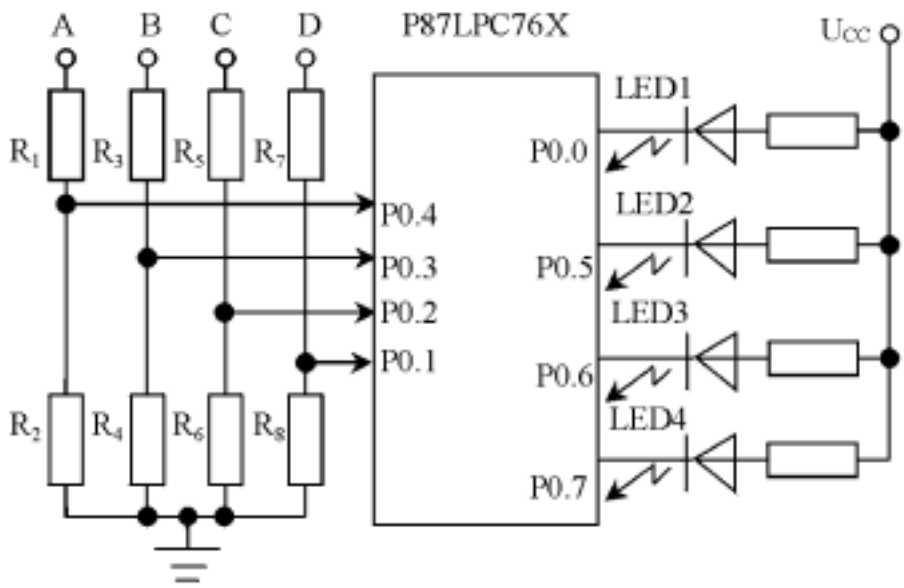


图 1.17-2 四路模拟信号的监测电路

为内部参考电压 1.28 V 有 10%的误差。因此,每路信号均接入一个可调电阻,通过实际调整达到预定的监测条件。

程序设计:主程序初始化过程中将 P0 口设置为 4 路模拟输入和 4 路开漏输出,以便检测 4 路模拟信号和控制 4 路 LED。由于检测过程很简单,耗时也不多,可以设计成一个检测子程序,既可以在主程序中调用,也可以在定时中断子程序中调用。

```
CMP1    DATA   ACH          比较器 1 控制寄存器
CMP2    DATA   0ADH         ;比较器 2 控制寄存器
P0M1    DATA   84H          ;P0 方式寄存器 1
P0M2    DATA   85H          ;P0 方式寄存器 2
PT0AD   DATA   0F6H         ;P0 数据输入禁能
LED1    BIT    P0.0         ;A 路信号指示(低电平有效)
LED2    BIT    P0.5         ;B 路信号指示(低电平有效)
LED3    BIT    P0.6         ;C 路信号指示(低电平有效)
LED4    BIT    P0.7         ;D 路信号指示(低电平有效)

        ORG    0000H
        LJMP   MAIN         ;至主程序

        ORG    0080H
MAIN:   .                   ;主程序例行操作:自检、初始化等
        .
        .
        MOV    PT0AD,#1EH   ;禁止 4 路模拟信号输入端上的数字输入功能
        MOV    P0M1,#0FFH   ;P0 口设置为 4 路模拟输入端,4 路开漏输出端
        MOV    P0M2,#0E1H
        .                   ;主程序其他内容
        .
        .
LOOP:   LCALL  TEST
```

```
        .                           ;主程序其他内容
        .
        LJMP    LOOP
TEST:   MOV     CMP2, #0            ;关闭比较器 2
        MOV     CMP1, #28H          ;使能比较器 1,选通 CIN1A,选用内部参考电压
        MOV     R2, #5
TEST1:  DJNZ    R2,TEST1            ;延时
        MOV     A,CMP1
        MOV     C, ACC .1           ;取信号 A 的检测结果
        CPL     C
        MOV     LED1,C              ;控制 LED1
        MOV     CMP1, #38H          ;使能比较器 1,选通 CIN1B,选用内部参考电压
        MOV     R2, #5
TEST2:  DJNZ    R2,TEST2            ;延时
        MOV     A,CMP1
        MOV     C, ACC .1           ;取信号 B 的检测结果
        CPL     C
        MOV     LED2,C              ;控制 LED2
        MOV     CMP1, #0            ;关闭比较器 1
        MOV     CMP2, #28H          ;使能比较器 2,选通 CIN2A,选用内部参考电压
        MOV     R2, #5
TEST3:  DJNZ    R2,TEST3            ;延时
        MOV     A,CMP2
        MOV     C, ACC .1           ;取信号 C 的检测结果
        MOV     LED3,C              ;控制 LED3
        MOV     CMP2, #38H          ;使能比较器 2,选通 CIN2B,选用内部参考电压
        MOV     R2, #5
TEST4:  DJNZ    R2,TEST4            ;延时
        MOV     A,CMP2
        MOV     C, ACC .1           ;取信号 D 的检测结果
        MOV     LED4,C              ;控制 LED4
        RET
```

3 .一路模拟信号的多条件监测

在很多情况下,我们检测的模拟信号只有一路,但检测的条件不只一个。最常见的情况是对一个模拟信号同时进行上下限检测,超过上限要报警,低于下限也要报警,并分别采取不同的措施。我们以一个简单的例子来说明设计方法。

有一个模拟信号,电压在 0～9 V 之间变化,正常值的范围是 4～6 V。有一个执行机构,它动作时可以使模拟信号慢慢升高;停止动作时,模拟信号会慢慢衰减。要求用模拟比较器设计一个简单的自动控制系统,使该模拟信号的幅度变化基本上保持在正常值范围之内。

该系统虽然只有一个模拟信号,但有两个检测条件,我们可以把这个信号分成两路,每路检测一个条件。利用前一节介绍的方法,很容易得到图 1 .17 －3 所示电路。R_1 和 R_2 对信号

进行分压,输入到比较器 1 的 CIN1A(P0.4)端口,用来进行上限检测;另一路 R_3 和 R_4 对信号进行分压,输入到比较器 1 的 CIN1B(P0.3)端口,用来进行下限检测。比较器采用内部 1.28 V的参考电压。LED1 用于上限超限报警(用 P0.6 控制);LED2 用于下限超限报警。用 P0.0 输出执行机构的控制信号 OUT,低电平动作,高电平停止。

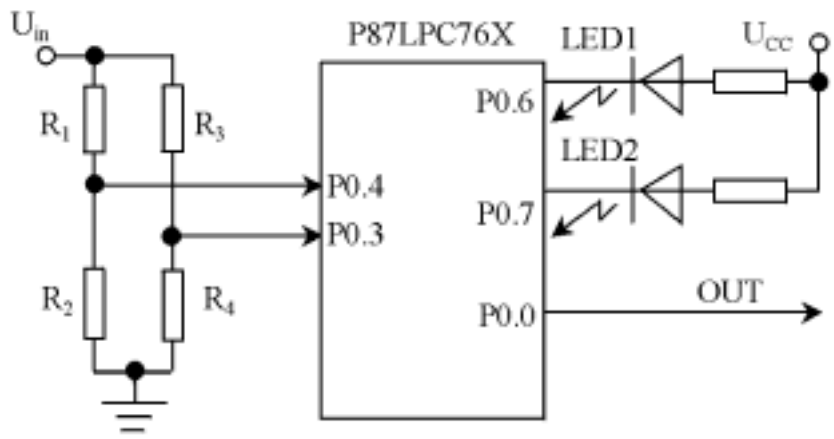


图 1.17-3 一路模拟信号的多条件监测电路

各路分压电阻计算:公共参考电压为 1.28 V,取 $R_2 = R_4 = 1\ k\Omega$。

$R_1 = [(6/1.28) - 1]R_2 = 3.69\ k\Omega$;选用 4.7 kΩ 可调电阻。

$R_3 = [(4/1.28) - 1]R_4 = 2.13\ k\Omega$;选用 3.3 kΩ 可调电阻。

系统运行时,如果信号低于下限,则点亮 LED2,进行下限超限报警,并输出低电平,启动执行机构;如果信号高于上限,则点亮 LED1,进行上限超限报警,并输出高电平,关闭执行机构;如果信号在正常范围之内,则熄灭两个 LED,维持执行机构的工作状态不变。在实际调试中,因为系统具有惯性,如果严格按照规定条件设置上限和下限,控制的效果反而不好,系统必然经常超限。如果上限检测条件适度低于 6 V(如 5.8 V),下限检测条件适度高于 4 V(如 4.2 V),执行机构就可以提前采取动作,使系统不至于超限。这时,LED1 和 LED2 点亮,说明系统将要超限。

程序设计:主程序初始化过程中将 P0 口设置为 2 路模拟输入和 6 路开漏输出,以便进行上限检测和下限检测,控制 2 路 LED 和 1 路执行机构。由于检测过程很简单,耗时也不多,可以设计成 1 个检测子程序,既可以在主程序中调用,也可以在定时中断子程序中调用。

```
CMP1    ATA   ACH          比较器 1 控制寄存器
P0M1    DATA  84H          ;P0 方式寄存器 1
P0M2    DATA  85H          ;P0 方式寄存器 2
PT0AD   DATA  0F6H         ;P0 数据输入禁能
LED1    BIT   P0.6         ;上限超限报警(低电平有效)
LED2    BIT   P0.7         ;下限超限报警(低电平有效)
OUT     BIT   P0.0         ;执行机构控制输出(低电平有效)
FLAG    DATA  20H          ;标志字节
UP      BIT   FLAG.0       ;上限检测结果
DOWN    BIT   FLAG.1       ;下限检测结果
        ORG   0000H
        LJMP  MAIN         ;至主程序
```

```
        ORG    0080H
MAIN:   .                          ;主程序例行操作:自检、初始化等
        .
        .
        MOV    PT0AD, #18H         ;禁止 2 路模拟信号输入端上的数字输入功能
        MOV    P0M1, #0FFH         ;P0 口设置为 2 路模拟输入端,6 路开漏输出端
        MOV    P0M2, #0E7H
        .                          ;主程序其他内容
        .
        .
LOOP:   LCALL  TEST
        .                          ;主程序其他内容
        .
        .
        LJMP   LOOP

TEST:   MOV    CMP1, #28H          ;使能比较器 1,选通 CIN1A,选用内部参考电压
        MOV    R2, #5
TEST1:  DJNZ   R2,TEST1            ;延时
        MOV    A,CMP1
        MOV    C, ACC .1           ;取上限的检测结果
        MOV    UP,C                ;保存上限的检测结果
        MOV    CMP1, #38H          ;使能比较器 1,选通 CIN1B,选用内部参考电压
        MOV    R2, #5
TEST2:  DJNZ   R2,TEST2            ;延时
        MOV    A,CMP1
        MOV    C, ACC .1           ;取下限的检测结果
        MOV    DOWN,C              ;保存下限的检测结果
        MOV    A,FLAG              ;取检测结果
        ANL    A, #3               ;将两个结果进行综合判断
        JNZ    TEST3
        CLR    LED2                ;低于下限,点亮 LED2,进行报警
        SETB   LED1                ;熄灭 LED1
        CLR    OUT                 ;启动执行机构
        RET
TEST3:  CJNE   A, #3,TEST4
        CLR    LED1                ;高于上限,点亮 LED1,进行报警
        SETB   LED2                ;熄灭 LED2
        SETB   OUT                 ;关闭执行机构
        RET
TEST4:  CJNE   A, #2,TEST5
        SETB   LED1                ;正常范围,熄灭 LED1
        SETB   LED2                ;熄灭 LED2,执行机构维持现状
```

```
        RET
TEST5:  SETB    LED1            ;故障状态,点亮 LED1
        SETB    LED2            ;点亮 LED2
        SETB    OUT             ;关闭执行机构
        RET
```

在程序中,我们将 2 次检测的结果进行综合考虑,就有 4 种情况。其中 3 种情况是合理的,分别进行对应的控制操作。有 1 种情况是不合理的:上限检测得到高电平,说明信号超过上限;下限检测得到低电平,说明信号低于下限。两个检测结果互相矛盾,这在现实情况里是不可能的。惟一能够解释这种检测结果的原因是电路出现故障。例如:当 R_2 发生开路故障,而信号电压下降到 4 V 以下时,就会同时检测到 2 个报警信号。同样,当 R_3 开路和信号电压上升到 6 V 以上时,也会同时检测到 2 个报警信号。综合判断可以发现故障状态,以便采取安全措施(关闭执行机构)。

二、模拟比较器用于直流信号的 A/D 转换(方法一)

1.原理电路

如图 1.17-4 所示,直流信号 U_{in} 输入比较器正向输入端,由 R 和 C 组成一个充电回路,电容器上的充电电压 U_C 输入比较器反向输入端,K 为电容器的放电开关。

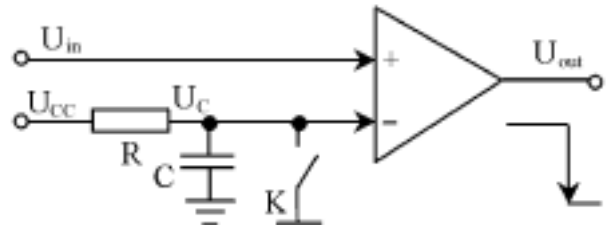


图 1.17-4　直流信号 A/D 转换原理电路

当电容器上的电压 U_C 比待检测的信号电压 U_{in} 低时,从比较器的输出端可以检测到高电平;当电容器上的电压 U_C 比待检测的信号电压 U_{in} 高时,从比较器的输出端可以检测到低电平。因此,在比较器输出端出现下降沿的瞬间,U_{in} 和 U_C 正好相等。U_C 的数值可以由充电公式计算:

$$U_C = U_{CC}(1 - e^{-t/RC})$$

式中,U_{CC}、R、C 三个数据为已知值,只要测量出 $U_C = U_{in}$ 时所需的充电时间 t,就可以计算出待测信号电压 U_{in}。

2.A/D 转换过程

充电时间 t 可用单片机中的定时器来检测。

(1) 开机上电时首先将 K 闭合,电容器 C 进行放电,为第 1 次检测作好准备。

(2) 当确保电容器放电结束后,将 K 断开,电容器 C 通过 R 开始充电,同时定时器开始计时。

(3) 当比较器输出端出现下降沿时,立即从定时器中读取充电时间 t,然后将 K 闭合,使电容器 C 开始放电,为下一次测量作好准备。

(4) 根据检测到的充电时间 t,通过数据处理计算出 U_C 值,便得到待测信号电压 U_{in} 的值。由于这种检测方法的精度不可能很高,通常采用查表和插值运算的方法求得结果。

(5) 由于第 3 步已将电容器放电,以后各次检测可以从第 2 步开始。

3 . 误差分析

引起检测误差的主要原因有:

(1) 系统时钟频率:如果使用外部晶体振荡器,其精度自然满足要求。采样单片机内部 RC 振荡器时,其振荡频率误差较大,离散性也较大。从计算公式可以看出,系统时钟误差可以通过 RC 时间常数"误差"进行补偿。实际精度可通过标定过程来检定。

(2) 电容器放电残留电压 U_{C0}:在实际的单片机电路中,放电开关 K 的功能是由单片机的一个具有开漏特性的端口来完成的。当该端口处于高电平时,相当于 K 断开,电容器可以进行充电;当该端口处于低电平时,相当于 K 闭合,电容器进行放电。为了保护单片机的端口,电容器是通过一个小电阻进行放电的,因此,放电结束时残留电压 U_{C0} 不能忽视。当待测信号电压 U_{in} 与 U_{C0} 接近时,使用基本公式计算将产生很大误差。在这种情况下,电容器实际充电过程的计算公式应该修正为:

$$U_C = (U_{CC} - U_{C0})(1 - e^{-t/RC}) + U_{C0}$$

(3) 电容器充电终止电压 U_{CC}:在三个已知数据 U_{CC}、R、C 中,U_{CC} 的稳定性最差。当 U_{CC} 的供电负载较重或变化不定时(如显示电路),U_{CC} 的波动直接影响检测结果。当 U_{in} 接近 U_{CC} 时,充电曲线非常平缓,U_{CC} 的变化对充电时间 t 的影响非常大。

(4) 充电时间的检测:在廉价单片机中,定时器一般不具备硬件捕获功能,通常是用软件指令来判断出现下降沿或出现超时,然后再冻结定时器,读取充电时间。冻结定时器的实际时刻比出现下降沿的时刻延误了 2～6 μs,导致从定时器中读取的数据比实际充电时间多出几个 μs。这多出的几个 μs 对于大信号影响不大,但对于小信号影响比较明显。

(5) 电阻 R 和电容器 C 的稳定性:在温度和湿度等环境参数变化时,R 和 C 的数值多少都会产生一些变化,从而对检测结果的精度产生一定的影响。为此,我们应该选用稳定性比较好的电阻和电容器,如聚苯乙烯薄膜电容器(注意:不要把涤纶薄膜电容器当成聚苯乙烯薄膜电容器)和金属膜电阻。

因此,这种检测方法在检测小信号(< 20% U_{CC})和大信号(> 90% U_{CC})时,精度会下降。当 U_{CC} = 5 V 时,U_{in} 的变换范围最好保持在 1 .0～4. 5 V,以便获得比较满意的检测结果。现实工程中,很多检测信号的数值变化范围并不大,通过信号调理(放大、衰减、窗口变换),一般可以控制在这个范围之内。

4 . 电路参数选择

如果信号调理电路已经使 U_{in} 的范围控制在(30%～80%)U_{CC} 之间,通过公式计算,所需充电时间 t 的范围为(0 .357～1 .61)RC。RC 为充电时间常数,即充电电阻 R 和电容器 C 的乘积。

为了保证一定的测量精度,充电时间不能太短(如定时器读数大于 100),但也不能太长,以免使检测速度降低。若选定定时器时间分辨力为 1 μs,最小读数为 100,即 0 .357RC = 100 μs,从而计算出充电时间常数 RC = 280 μs,可选取 RC = 300 μs。电容器 C 的数值不能太大,否则放电时很容易损伤单片机,但也不能太小,以免受电路分布参数的影响。若选 C = 0 .01 μF,则 R = 300 μs/ 0 .01 μF = 30 kΩ。

5 . 表格设计

系统工作时,每次检测可以读取到一个充电时间 t,用公式计算 U_{in} 需要浮点子程序库支

持。程序设计是费时费力的,程序运行速度也受限制,而查表插值算法在这里非常有效。表格的生成过程如下:预先选定一个充电时间常数(如 300 μs),按一定的时间间隔(如 16 μs),用公式计算出相应的 U_{in} 理论值,组成一个表格(该过程可以用 C 语言或 BASIC 语言编写一个简单的程序来完成)。表格中相邻两项之差称为增量,在进行插值运算时需要使用,为了加快运算速度,我们将增量表也事先计算出来。

6.程序设计

PHILIPS 公司的 P87LPC76X 片内含有两个模拟比较器,当只需要检测一路模拟信号时,相关电路如图 1.17-5 所示。正向输入端为 P0.4,反向输入端(基准电压输入端)为 P0.5,利用软件指令访问 CMP1 的 CO1 位就可以得到比较的结果。直流信号 U_{in} 从 CIN1A 输入,充电电路由 R_1 和 C 组成,通过放电保护电阻 R_2 输入到 CMPREF 端口。当电容器上的电压 U_C 和待检测的信号电压 U_{in} 相同时,从 CMP1 的 CO1 上可以检测到下降沿。程序设计中须注意以下几点:

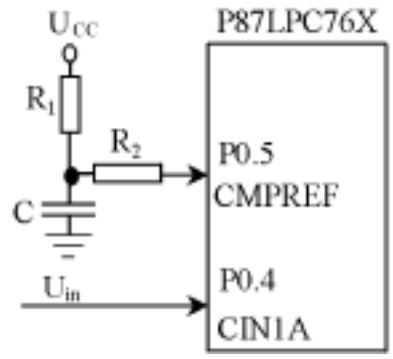


图 1.17-5　直流电压检测电路

(1) 信号输入端口初始化(CIN1A):必须关闭模拟信号输入端口的数字 I/O 功能,使其处于高阻状态。

(2) 电容器充放电端口(CMPREF)的处理:在放电阶段(准备阶段)设置成为开漏输出端口,并输出低电平,使电容器放电;在充电阶段(检测阶段),设置成为模拟输入端口,使其处于高阻状态。

(3) 比较器输出端下降沿的检测:用软件查询,从 CMP1 的 CO1 上可以检测到下降沿。

(4) 系统时钟:P87LPC76X 单片机的系统时钟设置方案较多,当定时器时间分辨力为 1 μs时,可以选用 12 MHz 外部晶振(二分频),也可以选用 6 MHz 外部晶振(不分频),还可以选用片内约 6 MHz 的 RC 振荡器(不分频)。对于批量较大、对系统时钟精度有一定要求的产品,为了降低调试成本,应该选用外部晶振。为了降低成本或者节省引脚,也可以采用片内 RC 振荡器,这时应该采用下一节介绍的方案,才能确保检测精度。

(5) 如果系统对速度要求不高,为了减少干扰的影响,可以连续检测多次,求充电时间的平均值。

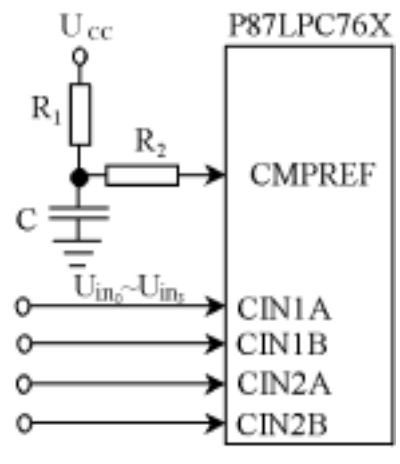


图 1.17-6　多路 A/D 转换电路

(6) 当信号电压 U_{in} 接近或超过 U_{CC} 时,充电时间趋于无穷大,比较器的输出端有可能检测不到下降沿,系统将死机。为此,将定时器预置一个时间上限,超过此时间后即不再检测下降沿,从而避免死机。

P87LPC76X 片内包含两个模拟比较器,最多可以检测 4 路模拟信号,如图 1.17-6 所示。程序设计和检测电路信号基本相同,需要考虑的新问题有:

(1) 4 个信号输入端口都必须关闭数字 I/O 功能,使其处于高阻状态。

(2) 如果 4 路信号都是慢信号,可以共用一套临时累加器和结果寄存器,采用分时工作方式。在一段时间之内,只检测和显示固定的 1 路信号。

(3) 如果 4 路信号变化较快,需要同时采集,则必须为每 1 路信号单独设置一套临时累加

器和结果寄存器,采用“并行”工作方式,即在每 1 次定时中断里对 4 路信号分别检测 1 次。

(4) 4 路信号的选通方法有一些不同,需要通过设置对应的特殊功能寄存器来完成。

(5) 比较器输出端下降沿的检测有一些不同:对于第 1 路和第 2 路,通过检测 CMP1 寄存器的 CO1 位来完成;对于第 3 路和第 4 路,通过检测 CMP2 寄存器的 CO2 位来完成。

(6) 不管检测哪一路,检测到下降沿后进行放电的方法都相同。

7 . 系统标定

在实际电路中,由于使用普通元件,参数的离散性比较明显,R_1 和 C 的乘积不可能正好等于选定的时间常数,这时 R_1 可以用 1 个微调电阻和 1 个固定电阻串联组成。标定时令 U_{in} = 2 .50 V,通过调节微调电阻,使数码管显示的电压值正好为 2 .50 V,则电路的充电时间常数和预先选定的时间常数就吻合了。再将 U_{in} 从 0 .25～5 .00 V 进行改变,检查各点的测量误差,做到心中有数。

如果 U_{in} 是原始检测量经过传感器变换而来的,而且是线性变换,则可以用 U_{in} 进行线性反变换,直接得到原始检测量,并直接显示原始检测量。例如,有一个压力传感器,灵敏度为每 500 kg 压力输出 1 mV 信号,经过放大器放大 1 000 倍后进行检测,变换系数成为 500 kg/ V,当检测到 U_{in} 为 3 .57 V 时,可变换为 $500\times3.57\times9.8 = 17.5\times10^3$ N,并直接显示出三位压力有效数据。

如果 U_{in} 是原始检测量经过变换而来的,但不是线性变换,就不能通过简单的系数换算来得到原始检测量。这时必须首先通过一系列的标定测试,得到原始检测量和 U_{in} 的关系曲线,再用这个关系曲线来修正数字电压表的表格数据,这样就可以直接显示出原始检测量。

三、模拟比较器用于直流信号的 A/ D 转换(方法二)

1 . 电路原理

前一节介绍的 A/ D 转换方法适用于需要外接晶体的精确定时系统,对于不需要精确定时的系统,为了简化系统和降低成本,一般采用片内 RC 振荡器。前一节误差分析中已经指出,采用片内 RC 振荡器将严重影响测量精度,虽然可以通过可调电阻进行单机标定来校准,但必然要增加成本,且不利于批量生产。本节介绍另一种利用模拟比较器完成 A/ D 转换的方法,可以在采用片内 RC 振荡器的系统中正常运行,且不需要进行标定就可以达到预定的精度,其原理电路如图 1 .17 - 7 所示。比较器正向输入端有两个信号:一个是待检测的直流信号 U_{in};另一个是电压值为 U_{CC} 的一半的已知信号(U_{CC}/ 2)。R 和 C 组成一个充电回路,电容器上的充电电压 U_C 输入比较器反向输入端,K 为电容器的放电开关。

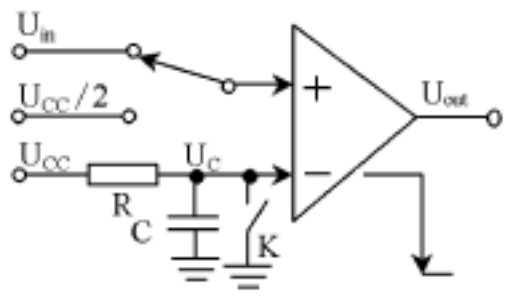


图 1 .17 - 7 直流信号 A/ D 转换原理电路

系统首先检测已知信号(U_{CC}/ 2)的充电时间 t′:

$$\frac{1}{2}U_{CC} = U_{CC}(1 - e^{-t'/RC})$$

从而得到: $t' = RC\ln 2 = 0.693RC$ 或者 $RC = 1.443t'$。

这里的 t′是以片内 RC 振荡器为时钟源检测出来的,不同的芯片检测出来的数据肯定不一样;但同一块芯片内 RC 振荡器有一个固定的频率,我们用同一个频率再来检测输入待测信号 U_{in} 的充电时间:

$$U_{in} = U_{CC}(1 - e^{-t/RC})$$

将 $RC = 1.443t$ 代入上式,得到 $U_{in} = U_{CC}(1 - e^{-t/RC}) = U_{CC}(1 - e^{-t/1.443t'})$

令 $K = t/t'$,最后得到待测信号的幅度为:

$$U_{in} = U_{CC}(1 - e^{-0.693K})$$

从式中可以看出,检测结果只与 U_{CC} 和 K 有关,而 K 是两个检测对象充电时间的比值。当硬件电路固定后,系统时钟的变化对 K 值没有影响(如果系统频率升高,t 和 t′的计数值将按相同的比例升高,但其比值保持不变),这就是相对测量方法的优点。当然,为此付出的代价是增加一个 $U_{CC}/2$ 的固定输入通道,从而减少可检测的模拟信号数量。

2.A/D 转换过程

充电时间的检测过程和前一节相同。首先检测 $U_{CC}/2$ 的固定信号的充电时间 t′,再检测 U_{in} 的充电时间 t,然后计算出对应的 K 值,就可以通过公式计算或查表插值运算得到 U_{in} 的检测结果。

3.误差分析

误差分析可以参考前一节。U_{CC} 和 $U_{CC}/2$ 的误差将直接影响检测结果,因此,U_{CC} 应该尽可能稳定在一个预定的数值上(例如 4.95 V),$U_{CC}/2$ 应该尽可能精确地等于 U_{CC} 的一半。系统时钟的影响可以不再考虑,从而可以放心使用片内 RC 振荡器。另外,我们可以通过定时检测系统时间常数的方法来消除元件参数的漂移,故电容器和电阻稳定性的影响不必考虑,可以放心使用廉价的元器件。

4.电路参数选择

实际电路如图 1.17-8 所示,充电电路可参考前一节。选时间常数 1 ms 左右,取 C = 0.033 μF,R_1 = 30 kΩ,R_2 = 100 Ω。信号($U_{CC}/2$)可由 R_3 和 R_4 分压产生,可选 $R_3 = R_4$ = 2～10 kΩ。为了提高系统精度,R_3 和 R_4 要用数字万用表仔细挑选,尽可能使电阻值相同。

5.表格设计

表格的生成过程如下:设 U_{CC} = 4.95 V,将 K 值按一定的间隔(例如 1/16)变化,代入公式:

$$U_{in} = U_{CC}(1 - e^{-0.693K})$$

计算出相应的 U_{in} 理论值,组成一个表格(该过程可以用 C 语言或 BASIC 语言编写一个简单的程序来完成)。表格中相邻两项之差称为增量,在进行插值运算时需要使用。为了加快运算速度,我们将增量表也事先计算出来。

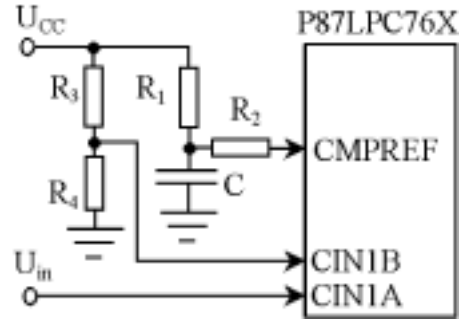


图 1.17-8 直流电压检测电路

6.程序设计

和前一种方案相比,本方案的程序设计有以下差别:

(1) 在检测未知信号 U_{in} 前,应该首先检测已知信号($U_{CC}/2$)的充电时间 t′,并记录下来。

(2) 然后检测未知信号的充电时间 t,从而计算出比值 K。

(3) 利用比值 K,进行查表插值运算,得到未知信号 U_{in} 的数值。

P87LPC76X 片内包含两个模拟比较器,最多可以检测 4 路模拟信号。由于本方案已经使用其中 1 路用来输入固定信号($U_{CC}/2$),故最多只能检测 3 路模拟信号,具体方法可参考前一方案。

7 . 系统标定

由于采用相对测量方法,只要分压电阻 R_3 和 R_4 的相对误差小于 0 .2%, U_{CC} 的数值稳定在设计表格时预定的数值,基本上不用标定就可以取得满意的效果,在 1 .00～4 .00 V 之间,检测精度优于 1%。

四、模拟比较器用于交流信号的 A/ D 转换

1 . 电路原理

对于交流信号的大小,通常是用它的有效值来衡量。为了检测有效值,常用的方法是首先进行整流滤波,得到它的平均值,然后采用与检测直流信号一样的方法检测它的平均值,最后折算成有效值输出。由于经过整流滤波环节,当交流信号源出现短暂变化时,这种传统的检测方法就不能及时发现。当精度要求不是很高时,用比较器来检测交流信号,不但具有电路简单、廉价的优点,而且能够发现信号源的短暂变化,分辨率达到 1 个周期。交流信号的 A/ D 转换原理电路如图 1 .17 - 9 所示。

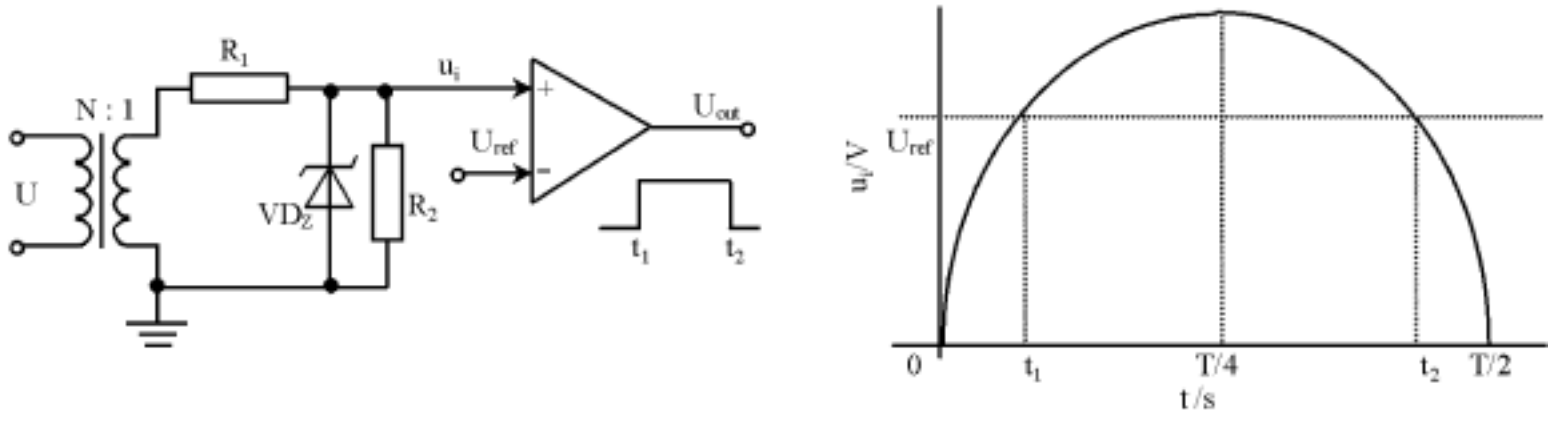


图 1 .17 - 9 交流信号的 A/ D 转换

设交流信号的有效值为 U,周期为 T,初始相位为 0。交流信号经过采样变压器进行隔离和降压(降压比为 N∶1),再由 R_1 和 R_2 分压,分压后形成的 u_i 输入比较器的正向端;基准电压 U_{ref} 输入比较器的反向端;稳压二极管 VD_Z 起保护作用,使比较器的正向端不必承受反向电压和过高的正向电压。设比较器输出低电平为 U_L,高电平为 U_H,再设 u_i 上升到 U_{ref} 的时刻为 t_1, u_i 下降到 U_{ref} 的时刻为 t_2,则比较器的输出电平为:

$$U_{out}=\begin{cases}U_L & 0\le t\le t_1\\ U_H & t_1<t<t_2\\ U_L & t_2\le t\le T\end{cases}$$

在 t_1 时刻,比较器输出端出现上升沿;在 t_2 时刻,比较器输出端出现下降沿。

在正半波任意时刻 t,输入到比较器正向端的信号电压的瞬时值 u_i 为:

$$u_i=\sqrt{2}U[R_2/N(R_1+R_2)]\sin(2\pi t/T)$$

即

$$u_i=\sqrt{2}UR_2\sin(2\pi t/T)/N(R_1+R_2)$$

在时刻 t_1,则有:

$$U_{ref}=\sqrt{2}UR_2\sin(2\pi t_1/T)/N(R_1+R_2)$$

从中可以求解出交流信号源的电压有效值为:

$$U=NU_{ref}(R_1+R_2)/\sqrt{2}R_2\sin(2\pi t_1/T)$$

由于 R_1、R_2 和 U_{ref} 均为系统固定参数,我们定义 1 个系统常数 K:

$$K = NU_{ref}(R_1 + R_2)/\sqrt{2}R_2$$

则待测交流信号的电压有效值为:

$$U = K/\sin(2\pi t_1/T)$$

从上式可以看出,只要检测 t_1,就可以计算出 U,而检测 t_1 必须从信号过零点开始计时,增加了检测难度。从波形示意图中可以看出,在交流信号的正半周里,只有从 t_1 到 t_2 这段时间之内 $u_i > U_{ref}$,比较器输出电压 U_{out} 为高电平,其他时间比较器都是输出低电平。当信号电压增加时,由于 U_{ref} 保持不变,t_1 到 t_2 这段时间间隔就增加,即比较器输出端维持高电平的时间就增加;当信号电压减小时,比较器输出端维持高电平的时间就减少,而检测比较器输出高电平的维持时间间隔($t_2 - t_1$)是一件比较容易的事情。为此,设 $\Delta t = t_2 - t_1$,则:

$$t_1 = [(T/2) - \Delta t]/2 = (T/4) - (\Delta t/2)$$

从而

$$U = K/\sin\{2\pi[(T/4) - (\Delta t/2)]/T\}$$

化简后得到

$$U = K/\sin[(\pi/2) - (\pi\Delta t/T)]$$

由于

$$\sin[(\pi/2) - x] = \cos x$$

最后得到的计算公式为:

$$U = K/\cos(\pi\Delta t/T)]$$

从公式中可以看出,当比较器输出端的高电平持续时间 Δt 缩短时,说明信号电压的有效值 U 在下降;当比较器输出端的高电平刚刚消失($\Delta t = 0$)时,U 正好等于 K;如果信号电压 U 继续下降,比较器输出端的高电平完全消失,无法进行检测。因此,系统常数 K 的物理意义为:系统能够正确检测到的最小信号电压的有效值。

从公式中还可以看出,仅仅检测比较器输出端的高电平持续时间 Δt 还不够,还必须知道交流信号的周期 T 才能计算出交流信号电压有效值 U。如果交流信号的周期(或频率)是固定的常数,就可以直接代入公式中进行计算;如果交流信号的周期(或频率)是未知的,可以通过检测比较器输出端相邻 2 个上升沿(或相邻 2 个下降沿)之间的时间间隔来得到周期 T。

2. A/D 转换过程

由于用比较器进行交流信号的检测实际上是对时间间隔的检测,故必须采用定时器来完成:

(1) 监视比较器的输出电平 U_{out},当 $U_{out} = U_L$ 时,进入第 2 步。

(2) 监视比较器的输出电平 U_{out},当 $U_{out} = U_H$(出现上升沿)时,开始计时,并进入第 3 步。

(3) 监视比较器的输出电平 U_{out},当 $U_{out} = U_L$(出现下降沿)时,读取定时器,得到高电平持续时间 Δt,如果需要检测周期 T(采用片内 RC 振荡器或信号周期未知),则进入第 4 步;否则,直接进入第 5 步。

(4) 监视比较器的输出电平 U_{out},当 $U_{out} = U_H$(出现上升沿)时,读取定时器,计算 2 次上升沿之间的时间间隔,得到周期 T。

(5) 计算检测结果 U。如果信号频率是固定的,且采用片外晶体振荡器,则只要检测 1 个实时数据 Δt,然后简单地通过查表和插值运算的方法计算出信号电压有效值 U。如果采用片内 RC 振荡器或信号周期未知,就需要检测 2 个实时数据 Δt 和 T,然后用下述第 8 部分介绍的方法算出信号电压有效值 U。

3 . 误差分析

引起检测误差的主要原因有:

(1) 系统时钟频率。如果使用外部晶体,其时钟精度自然满足要求。当采样单片机内部的 RC 振荡器时,其振荡频率误差较大,离散性也较大,高电平持续时间 Δt 的检测精度必然下降。从计算公式可以看出,高电平持续时间 Δt 的误差可以通过信号周期 T 的误差来进行补偿,因而,必须对信号周期 T 进行同步检测。

(2) 波形失真度。在分析电路原理时,假设被检测的交流信号是 1 个无失真的正弦波。这个假设在很多情况下都是不满足的,基于这个假设而推导出来的计算公式必然是不准确的。当信号的失真度不大,系统的精度要求也不是很高时,使用这个公式进行信号检测,完成一般的监控任务还是足够的。

(3) 采样变压器的降压比 N。变压器的降压比 N 理论上等于变压器初级线圈的匝数和次级线圈的匝数之比,当变压器初级和次级的工作电流不为零时,就会出现误差。为了降低系统成本,不可能选用价格昂贵的专用电压互感器,只能选用几元钱 1 只的廉价微型变压器。为了获得满意的稳定性,采样变压器的次级不能接入其他负载,使之处于“ 空载 ”状态。如果系统对电压检测精度的要求较低(10% 左右),检测的对象是交流市电,也可以不要独立的采样变压器,直接从系统电源变压器的某个次级线圈获取市电信号。

(4) 基准电压 U_{ref}。如果基准电压 U_{ref} 是用 2 个电阻从 U_{CC} 分压而得到的,则 U_{CC} 的波动也会带来新的检测误差。为了减少 U_{CC} 波动的影响,最好将耗电较大的部件(例如显示部件)分开供电。

(5) 定时器的时间分辨率。由于定时器的分辨率有限,当信号频率较高时,定时器的量化误差对检测结果的影响不可忽视。因此,交流信号的频率不可太高,以 1000 Hz 以下为好。

(6) 时间间隔检测。在检测时间间隔 Δt 和 T 的过程中,软件指令的判断和执行过程会造成一些延误。当信号频率较低时,这些延误可以忽略不计;当信号频率较高时,就会给检测结果带来一些误差。

(7) 分压电阻 R_1、R_2 的稳定性。考虑环境温度和湿度的变化对电阻值的影响,应该采用稳定性比较好的金属膜电阻。

基于以上原因,用比较器检测交流信号时,应该对检测结果的精度有一个基本估计。

4 . 电路参数选择

基准电压 U_{ref} 可以用 4 种方法产生:第 1 种方法是选用专用基准电压源芯片,可以获得很高的精度和稳定性,但与廉价、系统精度要求不很高的出发点矛盾,故不可取。第 2 种方法是用 1 个稳压二极管和 1 个电阻产生基准电压,稳定性尚可,成本也可以接受。第 3 种方法是用 2 个电阻直接从 U_{CC} 分压,基准电压的稳定性与 U_{CC} 的稳定性相同,成本低,当检测精度要求为 5% 左右时,实为一个廉价方案。第 4 种方案是选用 P87LPC76X 单片机内含的基准电压源,此为最佳方案。

基准电压 U_{ref} 数值的选择:1 2～2 .5 V 为好。

稳压二极管 VD_Z 的选择:比 U_{ref} 高出 2 V 左右,但不超过 5 .2 V。

系统常数 K 的选择:首先确定信号电压范围的最小值 U_{min},然后将 K 选择得比 U_{min} 再小一些。

采样变压器可选用廉价微型变压器,然后在空载的条件下实测其降压比 N。如果需要检

测的交流信号不需要隔离,也可以将采样变压器省略(这时应将计算公式中的 N 去掉)。

R_1 和 R_2 的选择:R_2 可选择 1～5.1 kΩ 的数值,然后,利用已经选定的 K、N、U_{ref} 和 R_2,通过系统常数公式求解 R_1。

下面以一个具体的应用项目作为例子来说明电路参数的设计过程。

在一个应用项目中需要监测市电的波动情况。已知市电的频率基本稳定在 50 Hz,理论有效值是 220 V,允许最小值为 160 V,最大值为 250 V。为了使检测电路安全可靠运行,用一个降压变压器对市电进行隔离采样。变压器选用廉价微型变压器,初级电压为交流 220 V,次级为 2 个 7.5 V,接成 15 V 使用,求得变压比 N = 220∶15 = 14.7。R_3 选用 7.5 kΩ 普通电阻,R_4 选用 2.4 kΩ 普通电阻。当 U_{CC} = 4.95 V 时,算得 U_{ref} = 1.20 V。选用 5 V 左右的稳压二极管进行保护。由于允许最小值为 160 V,选定系统常数 K = 150 V,R_2 选用 2.4 kΩ 普通电阻,从而计算出 R_1 = 26.5 kΩ(由 1 个 20 kΩ 普通电阻和 1 个 10 kΩ 可调电阻组成)。

5.表格设计

前面已经提到,当系统采用外部晶体振荡器且被检测信号的频率基本固定时,信号电压的有效值由比较器输出高电平的持续时间 Δt 来确定,两者之间存在一一对应关系,特别适合查表和插值运算处理。不同的应用项目决定表格有着不同的数据,但表格的设计过程是相同的。我们以前面介绍的市电监测作为例子,来说明表格设计的过程。

将 K = 150 V,市电的周期 T = 20 ms = 20 000 μs 代入检测公式,可以得到:

$$U = K/\cos(\pi\Delta t/T) = 150/\cos(3.1416\Delta t/20\,000) = 150/\cos(0.000\,157\Delta t)\ \mathrm{V}$$

式中 Δt 的单位是 μs。当 $U = U_{min}$ = 160 V 时,从公式可以得到 Δt_{min} = 2 263 μs;当 $U = U_{max}$ = 250 V 时,从公式可以得到 Δt_{max} = 5 906 μs。因此,实际运行过程中,Δt 的检测范围控制在 0～6 144 μs 就足够了,即表格的覆盖范围确定为 0～6 144 μs。为了计算方便,将 6 144 μs 按 256 μs 的间隔进行分段,共 24 段。然后用 C 语言编写 1 个简单的程序,计算出每一个分段的起点对应的电压值和每个分段的电压增量值,组成 2 个表格。由于市电不超过 255 V,故表格的内容可以用 1 个字节的无符号整数来表示。

6.程序设计

PHILIPS 公司的 P87LPC76X 单片机内含 2 个模拟比较器,每个模拟比较器可以检测 2 路交流信号。当只需要检测 1 路交流信号时,相关电路如图 1.17-10 所示。交流信号从 CIN1A(模拟比较器的正向输入端)输入,采用片内 1.28 V 的基准电压源,从 CMP1 的 CO1 位上可以检测到模拟比较器的输出信号。程序设计的方法和直流信号检测类似,主要区别是时间间隔的长短和获取的方法:

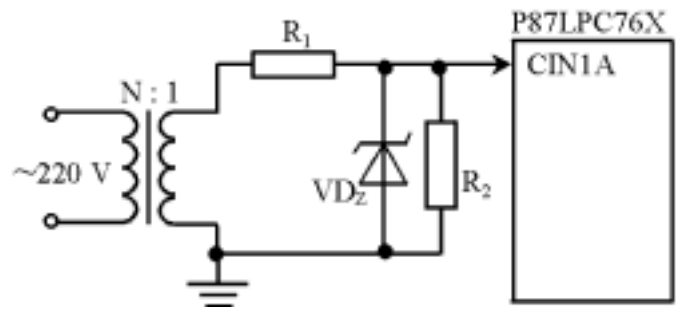


图 1.17-10　交流信号的 A/D 转换电路

① 等待低电平。

② 初始化定时器。

③ 等待上升沿,开始计时。

④ 等待下降沿,读取高电平维持时间 Δt。

⑤ 如果需要,继续计时,直到再次出现上升沿,读取周期 T。

⑥ 查表和插值运算,求出交流电压值。

P87LPC76X 片内包含 2 个模拟比较器,最多可以检测 4 路交流信号,每路信号必须分别进行相关硬件电路的设计;如果各路信号的频率不同,还必须分别进行表格设计;如果各路信号频率相同,可以共用 1 个表格。最后,根据各路系统常数 K 的不同,分别进行比例变换。程序设计和检测单路信号时基本相同。

7 .系统标定

从计算公式可以看出,检测结果与 3 个数据有关:系统常数 K、市电频率(周期 T 的倒数)和比较器输出高电平的持续时间 Δt。其中市电频率的稳定性我们无法控制,不予考虑;高电平持续时间 Δt 的检测精度由软件解决;硬件电路的任务是保证系统常数 K 的稳定性和准确性。系统常数 K 的稳定性和准确性由采样变压器的降压比、4 个电阻和 U_{CC} 共同决定,并不要求每一个元器件的参数都准确无误,U_{ref} 也不要求准确(但要求稳定)。可以按近似值选定电路中 R_2、R_3 和 R_4,将 R_1 和 1 个可调电阻串联,用来对系统进行标定。标定过程如下:用 1 个调压变压器模拟市电的变化,将调压变压器的输出调到 210 V,加到采样变压器的初级;调整可调电阻,使显示器显示值为 210 V;然后将输入电压在 140～270 V 之间变化,记录对应的显示值,便完成了标定任务。标定结束后,可用 1 个合适的固定电阻代替可调电阻。

8. 采用片内 RC 振荡器的交流信号有效值检测方法

当采用片内 RC 振荡器时,高电平持续时间 Δt 的检测精度必然下降,上面的单纯查表方法难以达到精度要求,必须采用相对测量的方法,即第 2 部分中的步骤④必须执行,对周期 T 进行同步检测。由于系统时钟的误差对 Δt 和 T 产生的影响(相对误差)相同,故它们的比值与系统时钟的误差无关。

设 k = Δt/ (T/ 2),即高电平持续时间 Δt 和周期的一半之比,则 k 值与系统时钟的误差无关。

由于

$$U = K/\cos(\pi\Delta t/T)$$

从而可以得到:

$$U = K/\cos(\pi\Delta t/T) = K/\cos(k\pi/2)$$

由于高电平持续时间 Δt 不会超过半个周期,故 k 值小于 1,可用 1 个字节十六进制纯小数来表示 k。仍然以检测交流市电为例,设 K = 150 V,当市电达到 250 V 时,对应的 k 值为 0 .590(1 个字节十六进制小数为 0 .97H),依次取 k = 0/ 32,1/ 32,2/ 32,3/ 32,4/ 32,…,18/ 32,19/ 32,代入公式,便可得到 1 个基数表格和对应的增量表格。k 值的范围为 0 .00H～0. 9FH,超出范围时用 k = 0 .9FH 进行封顶,区间间隔为 0 .08H(1/ 32)。系统工作步骤如下:

① 检测高电平持续时间 Δt。

② 检测信号周期 T,取其一半,得到 T/ 2。

③ 计算 k = Δt/ (T/ 2),结果取 1 个字节的十六进制纯小数,当 k > 9FH 时,令 k = 9FH。

④ 用查表和插值算法计算交流市电有效值。

五、模拟比较器用于参量信号的 A/ D 转换

1 .电路原理

无源传感器的电气参数随着被测物理量的改变而改变。可以通过检测传感器的电气参数来间接测量该物理量,热敏电阻、湿敏电阻、气敏电阻等传感器就是最常见的例子。以热敏电

阻为例,说明用模拟比较器完成温度检测的方法。

图 1 .17 - 11 所示,由 R_1 和 R_2 组成分压电路,生成的参考电压 U_{ref} 加到模拟比较器的反向输入端;由热敏电阻 R_T 和电容器 C 组成充电电路,充电电压 U_C 加到模拟比较器的正向输入端,K 为放电开关。

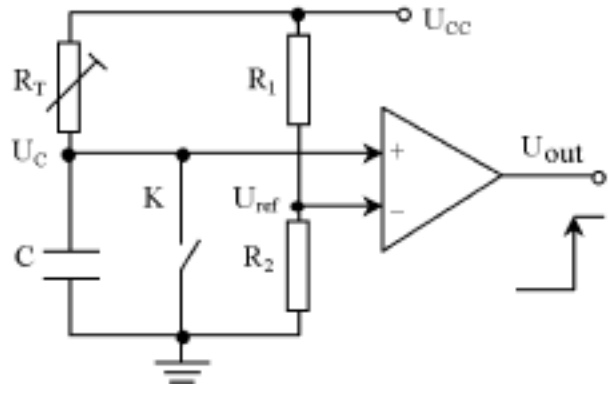


图 1 .17 - 11　参量信号 A/ D 转换原理

根据电路知识可以得到:

$$U_{ref} = U_{CC} R_2 / (R_1 + R_2)$$

$$U_C = U_{CC} (1 - e^{-t/R_T C})$$

当比较器输出端出现上升沿时,说明充电电压与参考电压相等,由此得到:

$$1 - e^{-t/R_T C} = R_2 / (R_1 + R_2)$$

即

$$e^{-t/R_T C} = R_1 / (R_1 + R_2)$$

等式两边分别取对数,整理后得到:

$$t / R_T C = \log(1 + R_2 / R_1)$$

从而解出热敏电阻的电阻值:

$$R_T = t / C \log(1 + R_2 / R_1)$$

令

$$K_{RC} = 1 / C \log(1 + R_2 / R_1)$$

即可得到热敏电阻的电阻值:

$$R_T = K_{RC} t$$

式中:t 为充电时间;系数 K_{RC} 只与电路中的 3 个元件(R_1、R_2、C)有关,与 U_{CC} 无关。这是因为参考电压和充电电压取自同一个 U_{CC} 的结果。这种检测电路适合电源电压不稳定的场合,特别是用电池供电的系统。

我们的目的并不是要检测热敏电阻的电阻值,而是要知道当时的温度值。为此,还必须确定热敏电阻的电阻值 R_T 和当时的检测温度 θ 之间的关系。当温度变化范围不大时,常用热敏电阻的电阻值与温度之间的近似关系如下:

$$R_T = R_0 e^{-\theta / k}$$

式中:R_0 为热敏电阻在 0℃时的电阻值;k 为热敏电阻的材料系数;θ 为当时的温度(℃)。由此得到:

$$R_0 e^{-\theta / k} = K_{RC} t$$

即

$$e^{-\theta / k} = K_{RC} t / R_0$$

等式两边分别取对数,整理后得到:

$$\theta = k \log(R_0 / K_{RC}) - k \log(t)$$

令

$$K = k \log(R_0 / K_{RC}) = k \log[R_0 C \log(1 + R_2 / R_1)]$$

最后得到温度的近似计算公式为

$$\theta = K - k \log(t)$$

式中系数 K 和 k 都是只与电路元件有关的常数。从近似计算公式可以看出,被检测的温度和电容器充电时间的对数近似线性相关:温度越低,热敏电阻的电阻值越大,充电时间越长;反之,温度越高,充电时间越短。

2．A/D 转换过程

充电时间 t 可以使用单片机中的定时器来检测。

(1) 开机上电时首先将 K 闭合,电容器 C 进行放电,为第 1 次检测作好准备。

(2) 当确保电容器放电结束后,将 K 断开,电容器 C 通过热敏电阻 R_T 开始充电,同时定时器开始计时。

(3) 当比较器输出端出现下降沿时,立即从定时器中读取充电时间 t。然后将 K 闭合,使电容器 C 开始放电,给下一次测量作好准备。

(4) 根据检测到的充电时间 t,通过数据处理计算出温度值 θ。由于这种检测方法的精度不可能很高,通常采用查表和插值运算的方法求得结果。

(5) 由于第 3 步已经将电容器放电,以后各次检测可以从第(2)步开始。

3．误差分析

引起检测误差的主要原因:

(1) 系统时钟频率。如果使用外部晶体振荡器,其精度自然满足要求。当采用单片机内部的 RC 振荡器时,其振荡频率误差较大,离散性也较大,充电时间 t 的准确性也就较低。从计算公式可以看出,充电时间 t 的误差可以通过调整其他元器件的参数进行补偿,故使用单片机片内 RC 振荡器必须增加调整电路(见图 1.17-13)。

(2) 电容器放电残留电压 U_{C0}。其影响的机理在前面已经介绍过。为了降低 U_{C0} 的影响,参考电压 U_{ref} 不能太低,最好取 U_{ref} 等于 U_{CC} 的一半左右。

(3) 电容器充电终止电压 U_{CC}。如果参考电压 U_{ref} 从 U_{CC} 分压而来,则 U_{CC} 对检测结果没有影响;如果参考电压 U_{ref} 由稳压管产生或者常用单片机内部参考电源,则 U_{CC} 的波动将直接影响充电时间 t,使温度检测结果出现新的误差。

(4) 充电时间的检测。冻结定时器的实际时刻比出现下降沿的时刻延误了 2～6 μs,导致从定时器中读取的数据比实际充电时间多出几个 μs。在温度较低时,热敏电阻的电阻值比较大,充电时间比较长,几个 μs 的延误可以忽略不计;在温度较高时,热敏电阻的电阻值比较小,充电时间比较短,几个 μs 的延误引起的误差不可忽略。

(5) 热敏电阻特性的差异。由于材料和制造工艺的问题,热敏电阻的电阻值和温度之间的关系与理论上的关系会有所差异,即使同一批热敏电阻也存在差异,这种差异使公式计算的误差增加。

(6) 计算公式的近似性。当温度变化范围较大时,计算公式将出现较大误差。

(7) 分压电阻 R_1、R_2 和电容器 C 的稳定性。在温度和湿度等环境参数变化时,R_1、R_2 和 C 的数值多少都会产生一些变化,从而对检测结果的精度产生一定的影响。为此,应该选用稳定性比较好的电阻和电容器,例如聚苯乙烯薄膜电容器(注意:不要把涤纶薄膜电容器当成聚苯乙烯薄膜电容器)和金属膜电阻。

4．电路参数选择

为了减少电容器放电残留电压 U_{C0} 的影响,参考电压宜选高一些;为了提高检测速度,充电过程不宜太长,参考电压宜选低一些。两者兼顾,取 $U_{ref} = 0.5U_{CC}$,即取 $R_1 = R_2 = 4.7$～10 kΩ。这时,充电时间约等于 $0.7R_TC$。

在温度范围的高端,由于热敏电阻的电阻值最小,充电时间也就最短。为了保证一定的检测精度,充电时间必须长于某个最小值(例如 70 μs),从而估算出温度范围高端的时间常数约

为 100 μs。然后实测热敏电阻在温度范围高端的电阻值,就可以估算出电容器的电容量。

热敏电阻的选择要根据温度检测范围和成本来考虑。在温度范围的高端,其电阻值不能太低,否则,必须加大电容器的电容量才能保证一定的检测精度。电容器的电容量越大,放电时对单片机端口的冲击也越大,有可能损害单片机。为了保护单片机,在电容器和单片机端口之间应该接入 1 个 100 Ω 左右的电阻 R_3。

作为一个例子,用模拟比较器为家用热水器设计一个电子水温温度计。温度检测范围是 10～90℃,要求在 30～55℃的范围内误差不超过 1℃;这个范围之外,误差在 2℃左右。按图 1 .17 - 12(省略显示部件和其他电路)制作了 1 个电子温度计,图中 $R_1 = R_2 = 7.5$ kΩ, $R_3 = 100$ Ω。根据现有热敏电阻的特性,选定 C = 0 .02 μF。

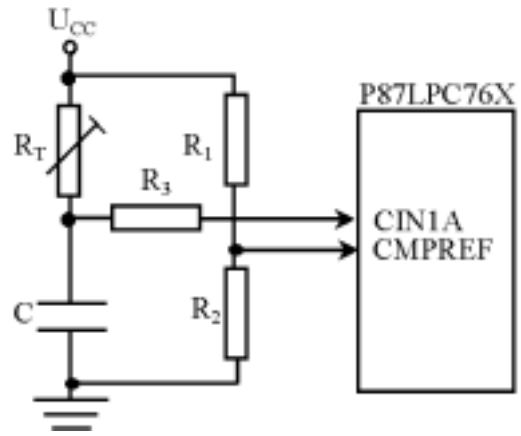


图 1 .17 - 12　P87LPC76X 单片机温度计

5 . 表格设计

为简化程序设计,仍然采用查表和插值运算的方法来得到温度值。有以下 3 种方法可以得到表格所需的数据。

(1) 完全依靠公式计算。从近似计算公式可以看出,温度是充电时间对数的线性函数。理论上只要在 2 个不同的温度下实测对应的充电时间,通过解联立方程,就可以求出公式中的 2 个系数 K 和 k 的数值。然后再用计算机依次算出各个充电时间对应的温度值,就可以得到所需的表格。这样得到的表格在实际使用中效果并不好,原因是热敏电阻的电阻值和温度的关系与理论公式有差异,温度与充电时间的对数并不是完全的线性关系。虽然在温度范围的两端被校准,但在温度范围的中段误差超过允许值,而在正常情况下设备都是工作在温度范围的中段。

(2) 完全依靠实际标定。放弃计算公式,利用可调式恒温装置进行实际标定:从低温端到高温端每隔 2～3℃标定 1 个点,从而得到比较完整的温度与充电时间关系表格。用这种方法得到的表格在实际使用中精度比较高,因为这个表格完全反映了真实情况。这种方法的缺点是:必须采用可调恒温设备进行标定,对测试设备要求比较高,否则,数据的精度没有保证。

(3) 实际标定与公式计算相结合。在没有恒温设备的情况下,利用普通的水银温度计和电热水杯来获取热敏电阻的表格数据。

在上面 3 种方法中:第 1 种方法最简单,但实际使用效果不好,测温误差较大,不予考虑;第 2 种方法实际使用效果最好,测温精度最高,在条件许可的情况下,尽可能采用;第 3 种方法不需要专用设备,标定条件要求最低。下面详细讨论第 3 种方法的操作过程和表格生成步骤。

由于没有恒温设备,充电时间肯定是测不准确的。为了在简陋的条件下获得尽可能好的结果,采取以下措施。

(1) 采用对称测试方案。在进行了 1 次升温测试和 1 次降温测试后,每一个温度点都有 2 个充电时间记录,一个偏大,一个偏小,能起到一定的相互抵消作用。

(2) 控制升温和降温速度。在升温的过程中,用增加水容量,降低加热功率的方法尽可能降低升温速度,使升温不超过每分钟 5℃。在降温的过程中,降温速度应尽可能和升温速度相同,如果自然冷却降温的速度偏慢,可以用滴冷水的方法加快降温。当降温速度和升温速度基

本相同时,两组数据的误差就能较好地相互补偿。

(3) 非均匀设置测点。以热水器为例,大多数场合下,水温在 36～50℃之间。为此,在 36～50℃之间增加测试点(间隔 2℃左右),在低温区和高温区减少测试点(间隔 5～10℃)。

(4) 测点质量的可视化处理。将测试的所有数据填入对数坐标纸中(最好显示在计算机屏幕上),这些测试点应该近似组成 1 条直线,误差大的数据偏离这条直线就远。通过观察,可以剔除那些质量差的数据。

(5) 线性回归处理。剔除误差太大的数据以后,将剩余的数据进行线性回归处理,就可以得到线性方程的 2 个系数,求出最终的数学模型。由于热敏电阻的实际特性与理论特性有差异,将整个温度范围用 1 条直线来拟合难以满足精度要求,故将整个温度范围分为 4 段,分别进行回归处理,最后用 1 条折线来表示热敏电阻的数学模型。

(6) 生成表格。利用最终数学模型计算出各种充电时间对应的水温,组成表格。

(7) 调整表格。由于最终数学模型不可能完全准确,应该进行实际标定。对各个温度段的误差进行核实,找出误差规律,再对表格进行修正,使这些误差得到一定程度的补偿,从而得到一个比较满意的表格。

为了实现以上方案,先将测试程序写入图 1.17-12 电路的 P87LPC76X 单片机中。这个程序会将充电时间以十六进制的方式显示出来,以便实测充电时间与温度的关系,得到基数表格。为了插值运算方便,将表格各个分段增量的绝对值预先计算出来,就可以得到该分段的增量绝对值,由此得到相关的增量表。由于整个表格是由折线组成的,在连接处不够平滑,应该将基数表中的对应项作微量调整,使增量表中的数据严格按递减顺序排列。

6. 程序设计

在程序设计中注意以下几点。

(1) 信号输入端口和基准电压输入端口的初始化。必须关闭输入端口的数字 I/O 功能,使其处于高阻状态。

(2) 系统时钟。对于批量较大、有一定精度要求的产品,为了降低调试成本,应该选用外部晶体振荡器。对于批量较小、精度要求不高的产品,为了降低成本或者节省引脚,也可以采用片内 RC 振荡器。这时应该对每一台设备单独进行标定。

采用图 1.17-12 的电路,以充电时间测试程序为基础,增加查表插值运算子程序和相关表格,就可以完成程序设计任务。

P87LPC76X 片内包含 2 个模拟比较器,最多可以检测 4 路参量信号。程序设计和检测单路参量信号基本相同。

7. 系统标定

如果只是制作一台设备,可以根据实际运行中的误差对表格数据进行修改,进一步提高检测精度。如果是大批量生产,不可能为每一台设备单独进行表格调整,只能采用相同的表格数据。由于各台设备的元器件不可能完全一致,尤其是电容器的实际电容量有比较大的离散度,导致各台设备的读数和样机不一样。从计算公式可以看出,最终读数取决于各个元器件的综合效果,某个元器件的误差可以通过其他元器件来进行补偿。因此,在最终产品里采用图 1.17-13所示电路,增加 1 个可调电阻 R_W (电阻值约为$\frac{1}{4}R_1$)。标定时,将水温稳定在设备典型工作温度上(例如 45℃)。通过调节可调电阻 R_W 抽头的位置,使显示的温度值正好为实际

温度。经过标定后,基本上可以达到样机的检测精度。

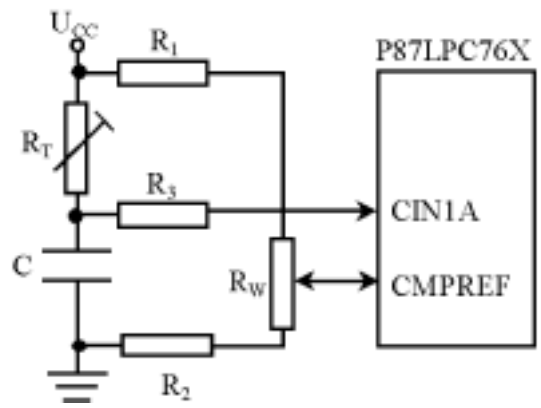


图 1 .17 - 13　电子温度计的标定

选自《单片机与嵌入式系统应用》月刊,2001 年第 2、3、4、5 期

第二章

综合应用技术

2.1 闪速存储器硬件接口和程序设计中的关键技术

哈尔滨工业大学通信技术研究所 312 信箱(150001)
胡永庆 田日才

闪速存储器(Flash Memory)以其集成度高、制造成本低、使用方便等诸多优点广泛地应用于办公设备、通信设备、医疗设备、家用电器等领域。利用其信息非易失性和可以在线更新数据参数特性,可将其作为具有一定灵活性的只读存储器使用。

在单片机应用系统中,经常遇到大容量的数据存储问题。闪速存储器由于容量大、存储速度快、体积小、功耗低等诸多优点,而成为应用系统中数据存储器的首选。但是,由于单片机的资源有限,而闪速存储器的种类和工作方式又千差万别,因而在单片机与闪速存储器的接口电路和程序设计中,有许多关键技术问题需要解决。

单片机与闪速存储器的接口电路应注意的问题有:

(1) 很多单片机的工作电压为 +5 V,而很多闪速存储器却工作在 1.8～6 V 之间,有些闪速存储器的擦除电压又工作在 12 V。

(2) 8 位的单片机很多,而闪速存储器很多是 16 位的。

(3) 同一型号的闪速存储器由于厂家不同,引脚的定义是不一样的,例如 Intel 公司的 28F008BV 与 AMD 公司的 29LV008 有很多引脚是不一样的。

单片机与闪速存储器的程序设计应注意的问题有:

(1) 不同厂家的闪速存储器使用不同的操作命令集,软件要根据不同厂家的闪速存储器使用不同的操作命令集。

(2) 很多闪速存储器内部存储结构和时间参数是不同的。由于闪速存储器内部都分成不同大小的存储块,在对闪速存储器进行擦除操作时,软件要根据不同型号的闪速存储器调整被擦除存储块的大小等参数。同时,由于不同型号的闪速存储器时间参数是不同的,软件要根据闪速存储器的时间参数来调整读写和擦除操作的时间。

针对上面遇到的问题,我们从硬件和软件两个方面来考虑单片机与闪速存储器应用系统中应解决的关键技术问题。

一、单片机与闪速存储器硬件接口的关键技术

生产闪速存储器的半导体公司主要有美国的 Intel、AMD 公司和日本的 Sharp、Fujitsu 公司,这四家公司生产的闪速存储器在市场占有份额相当大。表 2.1－1 列出了四家公司生产的主要型号的闪速存储器的性能指标。

表 2.1-1 Intel、AMD、Sharp、Fujitsu 闪速存储器的性能指标表

性能 / 型号	工作电压 V_{cc}/ V	擦除电压 V_{pp}/ V	容量/ 数据位长度
28F008SA(Intel)	5	12	8 M 位/ 8
28F016SA(Intel)	5	12	16 M 位/ 8
28F008S3(Intel)	2.7/ 3.3	2.7/ 3.3 或 12	8 M 位/ 8
28F008S5(Intel)	5	5/ 12	8 M 位/ 8
28F016SC(Intel)	2.7/ 3.3 或 5	3.3/ 5 或 12	16 M 位/ 8
28F008BV/ BE(Intel)	2.7/ 3.3/ 5	5/ 12	8 M 位/ 8
28F800BV/ CV/ CE(Intel)	2.7/ 3.3/ 5	5/ 12	8 M 位/ 16
28F008B3(Intel)	2.7～3.6	2.7～3.6 或 12	8 M 位/ 8
28F400B3(Intel)	2.7～3.6	2.7～3.6 或 12	4 M 位/ 16
28F160B3(Intel)	2.7～3.6	2.7～3.6 或 12	16 M 位/ 16
28F400B5(Intel)	5	5/ 12	4 M 位/ 16
29LV008(AMD)	2.7～3.6	2.7～3.6	8 M 位/ 8
29LV160(AMD)	2.7～3.6	2.7～3.6	16 M 位/ 16
LH28F008SC(Sharp)	3.3/ 5	3.3/ 5 或 12	8 M 位/ 8
LH28F400BG(Sharp)	2.7/ 3.3 或 5	2.7/ 3.3/ 5 或 12	4 M 位/ 16
LH28F160S3(Sharp)	2.7 或 3.3	2.7/ 3.3 或 5	16 M 位/ 16
MBM29LV008(Fujitsu)	2.7～3.6	2.7～3.6	8 M 位/ 8
MBM29LV800(Fujitsu)	2.7～3.6	2.7～3.6	8 M 位/ 16

从表 2.1-1 中可以看出,不同厂家的闪速存储器的工作电压和编程擦除电压是不一样的,同时数据位的长度也是不一样的。由于目前国内应用最广泛的单片机仍然是 8 位的 MCS—51 系列单片机,16 位的单片机种类比较少,而且工作电压在低电压(2.7～3.6 V)的单片机又是寥寥无几。能否用市场上常见的普通 8 位单片机来设计一个与大多数闪速存储器接口的电路呢?答案是肯定的。我们用普通的 8 位单片机 AT89C52 设计了一个与闪速存储器 TE28F160B3 的接口电路。AT89C52 是 ATMEL 公司生产的与 MCS—51 系列单片机兼容的 8 位单片机,内部有一个 16 KB 的 E^2 PROM 程序存储器,工作电压是 5 V。TE28F160B3 是 INTEL 公司生产的容量为 16 M 位、数据总线宽度为 16 位的闪存存储器,工作电压为 2.7～3.6 V。需要指出的是,虽然 TE28F160B3 的工作电压为 2.7～3.6 V,但是其各引脚的最大工作电压范围却为 -0.5～5.0 V,各引脚高电平最高工作电压不能超过5.5 V,这样就使得我们可以使用 AT89C52 来设计与 TE28F160B3 的接口电路。该接口电路如图 2.1-1 所示。

由于 AT89C52 是 8 位单片机,而 TE28F160B3 是 16 位数据总线,我们使用了两片 74HC244 和两片 74HC373 来完成 8 位和 16 位的数据转换。当 AT89C52 往 TE28F160B3 写数据时,单片机首先将高 8 位数据写入锁存器 74HC373-1 中。其中 74HC373-1 锁存信号 W373 由译码器 GAL16V8 输出,然后单片机开始执行对 TE28F160B3 写数据操作。低 8 位数据由 AT89C52 的 P0 口直接写入 TE28F160B3,而锁存在 74HC373-1 中的高 8 位数据通

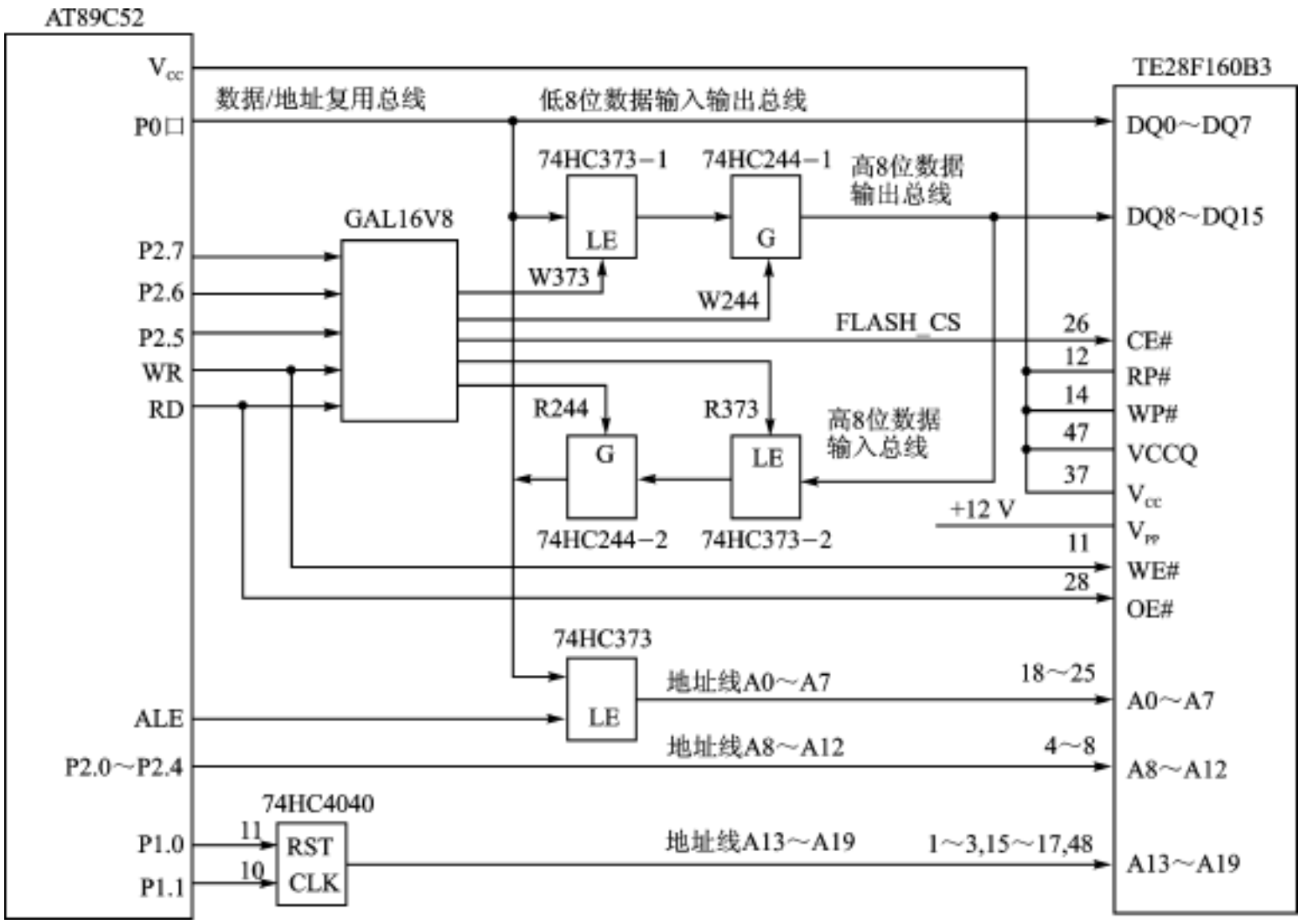


图 2.1－1 AT89C52 与 TE28F160B3 的接口电路

过缓冲器 74HC244 - 1 写入到 TE28F160B3 的 DQ8～DQ15 总线上。当 AT89C52 从 TE28F160B3 读数据时，读出的高 8 位数据先锁存到 74HC373 - 2 上，然后通过缓冲器 74HC244 - 2 读入到 AT89C52 中。TE28F160B3 的存储容量为 16M 位，有 20 根地址线 A0～A19，而 AT89C52 一共才有 16 根地址线。因此利用 AT89C52 的地址线 A15、A14 和 A13 经译码作为两片 74HC244、两片 74HC373 和 TE28F160B3 的锁存信号和片选信号。这样地址线只剩下 A0～A12，为此利用一片计数器 74HC4040 作为地址线 A13～A19，从而就解决了 AT89C52 的寻址问题。

TE28F160B3 的供电电源 V_{CC} 与 AT89C52 一样，均接 + 5 V 直流电源。但是 TE28F160B3 的编程电压和擦除电压 V_{PP} 必须接 + 12 V。

图 2.1－1 的单片机使用了市场上常见的 AT89C52，但在设计中我们推荐使用宽电压范围工作的单片机 AT89LV52 和地址译码器 ATF16LV8，这样就可以使用 + 3 V 左右的供电电源。

在生产闪速存储器的半导体公司 Intel、AMD、Sharp 和 Fujitsu 中，Intel 和 Sharp 公司的闪速存储器的引脚是一样的，AMD 和 Fujitsu 公司的闪存存储器的引脚是一样的。所以 Intel 和 AMD 公司的闪速存储器是不能互换的，如果要互换必须经过一个接口板进行转接。

二、单片机与闪速存储器程序设计的关键技术

由于生产闪速存储器的半导体公司众多,即使是同一公司的闪速存储器也是型号众多、千差万别。为使程序设计尽可能地适用于大多数的闪速存储器,需注意以下几个关键技术。

1.器件自动识别

器件自动识别要识别出器件使用的命令集、内部阵列结构参数、电气和时间参数及器件所支持的功能。器件自动识别的方法有两种:如果闪速存储器支持 CFI 功能,可以直接通过 CFI 获得器件的各种参数;如果闪速存储器不支持 CFI 功能,可以写器件识别命令,然后从器件中读取产品的生产厂家和器件代码,根据生产厂家和器件代码从程序中建立的器件参数表中读取器件的各种参数。器件自动识别的流程图如图 2.1-2 所示。

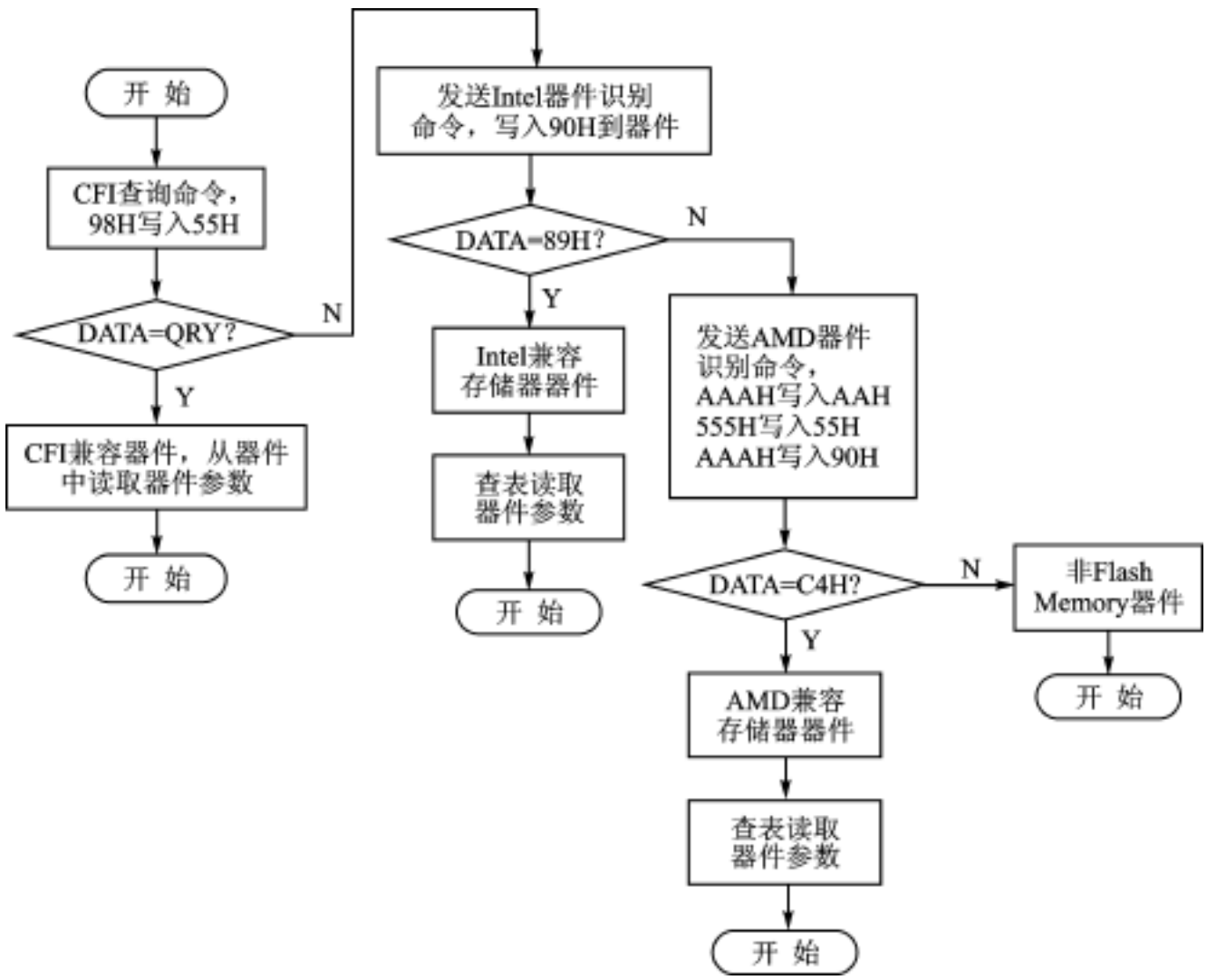


图 2.1-2 闪速存储器芯片自动识别流程图

正确识别器件之后,就可以根据器件的命令集对器件进行各种操作。对闪速存储器的所有操作都是通过芯片的命令用户接口 CUI 实现的。通过 CUI 写入不同的控制命令,闪速存储器就从一个工作状态转移到另一个工作状态。其主要的工作状态是:读存储单元操作、擦除操作和编程操作。

2.读存储单元操作

在闪速存储器芯片上电以后,芯片就处于读存储单元状态,也可以通过写入复位命令进入读存储单元状态。读存储单元的操作与 SRAM 相同。

3.擦除操作

在对闪速存储器芯片编程操作前,必须保证存储单元为空。如果不空,必须对闪速存储器

芯片进行擦除操作。由于闪速存储器采用模块分区的阵列结构,使得各个存储模块可以被独立地擦除。当给出的地址是在模块地址范围之内且向命令用户接口写入模块擦除命令时,相应的模块就被擦除。要保证擦除操作的正确完成,必须考虑以下几个参数:(1) 该闪速存储器芯片的内部模块分区结构;(2) 擦除电压 V_{PP};(3) 整片擦除时间和每个模块分区的擦除时间参数。上面三个参数在器件识别中获得。

4.编程操作

闪速存储器芯片的编程操作是自动字节编程,既可以顺序写入,也可指定地址写入。编程操作时注意芯片的编程电压 V_{PP} 和编程时间参数,这两个参数也可以在器件识别中获得。

上面,我们给出了单片机与闪速存储器硬件接口电路和软件编程设计中应注意的关键技术问题。硬件上主要考虑芯片的工作电压和编程电压,软件上要考虑到器件的内部结构、使用命令集和时间参数等因素。随着闪速存储器器件朝着容量越来越大、工作电压越来越低、支持共同的接口标准的方向发展,将会使闪速存储器硬件接口和软件编程设计越来越容易,也会使闪速存储器的应用更加广泛。

参 考 文 献

1 ntel 公司. Intel Products CD-ROM. 1998.5

2 何立民.MCS—51 系列单片机应用系统设计——系统配置与接口技术.北京:北京航空航天大学出版社,1990

3 窦振中.单片机外围器件实用手册——存储器分册.北京:北京航空航天大学出版社,1998

选自《电子技术应用》月刊,2001 年第 11 期

2 .2　51 单片机节电模式的应用

天津大学自动化学院(300072)　李　刚　李巧真　孙宏军

MCS—51 系列单片机是目前国内市场上主流的 8 位单片机计算机。从 20 世纪 70 年代中期至今经历了 20 多年的发展、完善,同时也出现了几近完备的各种 8 位外围芯片,使得 MCS—51 系列单片机的应用领域更为宽广。在智能化仪器仪表方面,用单片机改造原有的测量、控制装置,使其向数字化、智能化、多功能化发展,实现测量装置的误差修正、线性化处理,使传统的仪器仪表发生了根本性的变革。基于 MCS—51 系列单片机开发出的各种智能化仪器仪表目前几乎应用于国民经济的各个领域。

随着智能化仪表应用领域的不断扩大,新的矛盾也会随之产生,单片机的功耗问题即为其中之一。本文结合作者的工作实际,介绍了一种基于 51 单片机构成的应用系统降低功耗的设计方案。

在一些特殊场合,对一些特殊用途的仪器仪表除了要实现其功能外,其本身的能量消耗—功耗也是一个必须考虑的问题,有时还必须优先考虑。比如,对于那些靠电池供电的便携式仪器,对于那些安装在不适宜交流供电的场合,不方便铺设电线电缆的地方进行数据采集、测量的仪表等,应尽量降低其自身的功率消耗,使它们在不更换电池的情况下能可靠地工作几个月甚至几年。这是智能化仪表开发者们当前面临的一项重要课题。

一、低功耗单片微机系统设计的一般原则

以单片机为核心构成的智能化仪器仪表也可称之为一个单片微机系统。单片微机系统的功耗是由很多因素决定的。比如系统的技术指标、电路芯片和器件的参数以及系统的工作方式等等。要设计出低功耗的单片微机应用系统,就必须对这些因素全面考虑。简单地归纳起来,低功耗单片机系统设计的有如下几条原则。

1 . 采用低功耗集成电路

采用静态功耗几乎为零的 HCMOS 或 CMOS 集成电路。这种电路的最大优点是微功耗,另外,这种电路的输出逻辑电平摆幅较大,因而抗干扰能力强;同时它的工作温度范围也比较宽,从而适用面更广,更有利于现场仪表的使用。因此 HCMOS 及 CMOS 集成电路是低功耗单片机系统设计时的首选器件。

2 . 用低电压供电

单片微机系统的功耗与系统的供电电压之间存在着一定的函数关系。供电电压越高,系统的功耗也就越大。对于纯电阻电路,由欧姆定律可知其功耗是与供电电压的平方成正比的。低功耗单片微机系统大多是靠电池供电的,为了延长电池的使用寿命,应尽量降低供电电压。

3 . 尽量采用高速低频的工作方式

低功耗单片微机应用系统中全部采用 CMOS 和 HCMOS 器件,这一类器件因自己的结构所决定,静态功耗几乎为零,仅在逻辑状态发生转换时电路中才有电流流过。因此,它的动态

功耗与它的逻辑状态转换频率成正比,与电路的逻辑转换时间成正比,换言之,只要电路在进行逻辑转换,其功耗就相对比较大。于是从降低功耗的角度考虑,应使 CMOS 电路工作时工作速度加快——缩短工作时间;同时尽量延长 2 次工作之间的间隔时间——降低工作频率。这就是所谓高速低频的工作方式。另外,CMOS 电路中不用的输入端不要悬空,以免输入逻辑电平不定引起电路来回翻转而增大系统功耗。

4 . 合理地选择系统技术指标

在一个单片机应用系统中,往往有很多技术指标是与其功耗联系在一起的,像工作速度、驱动能力、稳定性、线性度等,这些技术指标的提高往往是以提高电路的功耗来换取的。因此,对于低功耗仪器仪表,应结合其应用特点合理地选择、确定技术指标,有时甚至采取降低某些非关键性指标的措施以达到降低系统功耗的目的。

5 . 采用分区分时供电方式

在低功耗仪器仪表的设计过程中,有时不可避免地要用到功耗相对较大的器件或电路。如果这些器件或电路只是相对短时间的工作,则可将其设计为分区域、分时段供电方式,一旦需要它们参与工作才为之供电,而在大部分时间里,可把这部分电路的供电电压降到功耗最低的维持状态甚至对其断电,以降低系统的整体功耗。

6 . 采用低工耗的工作方式

厂商不仅生产了各种各样低功耗的器件,而且为一些器件设计了降低功耗的各种工作方式,如单片机的“ 待机 ”、“ 掉电 ”工作方式,存储器的维持工作方式等。在设计单片机系统时,应当对之加以充分利用,使这些器件尽量在这些节电方式下工作。

二、51 单片机的低功耗运行

51 系列单片机中的 HCMOS 型由于采用了 HCMOS 工艺制造,因而其功耗低、抗干扰能力强,并具有“ 待机 ”和“ 掉电 ”两种节电运行模式,从而为设计低功耗单片机系统创造了条件。表 2 .2 - 1 给出了 51 系列单片机中的 89C52 在不同运行模式下的功耗情况。

表 2 .2 - 1

正常运行	3 .6 V 供电	600 kHz 时钟	功耗 3 .0 mA
待机运行	3 .6 V 供电	600 kHz 时钟	功耗 0 .6 mA
掉电运行	3 .6 V 供电	600 kHz 时钟	功耗 16 μA

1 . 待机运行模式

将 89C52 单片机特殊功能寄存器中的功耗寄存器 PCON 的 D_0 位(IDL)置位后,89C52 单片机即进入待机运行模式。如图 2 .2 - 1 所示,一旦进入待机运行模式,虽然片内时钟发生器仍在工作,但通往 CPU 的内部时钟已被门控电路切断,CPU 处于睡眠冻结状态。在进入“ 待机 ”运行前一瞬间,CPU 及 RAM 的状态被完整地保存下来,如堆栈指针、程序计数器、程序状态字、累加器及其他所有的寄存器均保持为待机前的状态。各 I/ O 口也都保存着待机前的逻辑状态。ALE 和 PSEN 均进入无效状态。从图 2 .2 - 1 可见,虽然 CPU 在“ 睡眠 ”,但是单片机的内部时钟仍旧供给中断电路、定时/ 计数器及串行口等。所以上述电路及串行口均可继续工作。

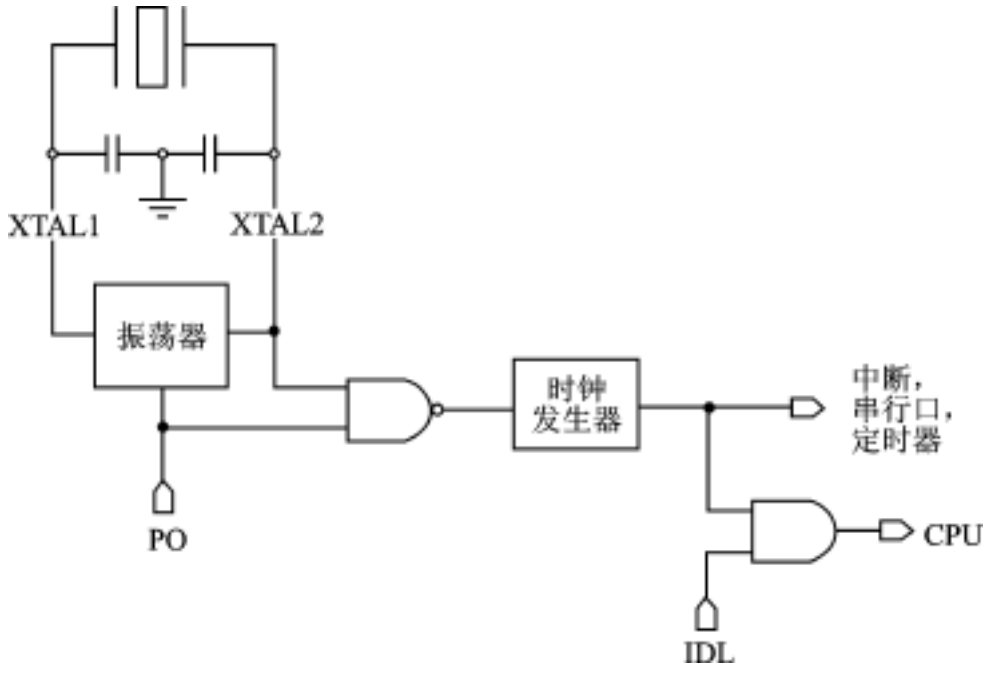


图 2.2-1　待机和掉电硬件

结束“待机”状态的方法有两种:中断和复位。任何在“待机”之前已开放的中断矢量在待机过程中接到中断请求,都会引起硬件对 IDL 位的清零,从而终止“待机”运行状态。于是中断得到响应,程序进入中断服务程序;当执行完中断返回指令 RETI 后,单片机返回执行使单片机进入“待机”状态的那条指令的下一条指令。结束待机运行的另一种方法是硬件复位。由于“待机”方式下振荡器仍在运行,硬件复位只需保持 2 个机器周期(24 个振荡周期)有效就能完成复位操作。

2.掉电运行模式

一条把 PD(PCON.1 PCON 寄存器中的 PD 位)置位的指令执行完后,89C52 即进入掉电运行模式。由图 2.2-1 可见,在“掉电”方式(PD=1)时,振荡器被冻结起来,时钟信号发生器停止工作,单片机的一切功能都停止。只有片内 RAM 的内容仍被保持,各片脚的输出值仍为缓冲器“掉电”前一瞬间的状态。掉电运行模式主要被用来以很小的功耗保存片内 RAM 的数据,以为下一正常运行周期作准备。

结束“掉电”方式的惟一途径是硬件复位。复位电平应保持足够长的时间,以保证振荡器启振并达到稳定。

三、51 单片机的复位状态

HCMOS 型 51 系列单片机的复位操作,只需在其复位引脚(RESET)端保持一个足够长时间(不短于振荡器建立起振荡的时间加上两个机器周期)的高电平。复位后,程序从地址 0000H 处开始执行,单片机中的所有寄存器(除串行数据缓冲寄存器 SBUF 之外)均清零,全部 I/O 口线均输出高电平(置 1)。由于单片机具有这种复位特性,在设计使用其掉电工作模式时要格外小心。

四、低功耗单片微机应用系统的硬件设计要点

要降低单片机的功耗,使之满足低微功耗的要求,可以考虑以下两条途径之一。选择新型的微功耗单片机,这是最有效、最彻底的解决办法。如近年来美国 Microchip 公司推出的 8 位 CMOS 单片机 PIC16C5X 系列,在供电电压为 3.7～6.0 V 范围内,正常运行时耗电不超过

1 mA,待机状态下的耗电仅有 10 μA,以这样低的功耗一般地说是可以满足工业用低功耗仪器仪表的需要了。选择新机型这条路在设计新型节能型仪器仪表时理应成为首选方案。然而对于已形成系列产品的生产厂家来说,为了降低功耗而改变应用多年的单片机机型,重新熟悉、适应新机型,可能会给开发、生产、维修等诸多方面带来意想不到的麻烦,甚至得不偿失。在这种时候,作为一个过渡手段,选择以下第二条路也是行得通的,即采用单片机的节能工作方式进行电路及程序设计。

51 系列单片机目前的低功耗产品之一是 AT89C52,其内部有 8K EPROM(闪速可改写程序存储器);24 条口线在不需要扩展外部程序存储器时,可全部用作 I/O。AT89C52 的功耗情况如表 2.2-1 所列。从表中可见当单片机进入掉电模式时其功耗(3.6 V 供电、600 kHz 时钟时)只有 16 μA,差不多是正常工作时功耗的 1/200。试想,如果能使单片机除短暂的工作时间外都处于“掉电”模式下,则其功耗将会大大降低。另外,AT89C52 的市场价格仅为每片十几元人民币,具有很高的商用价值。单片机的工作速度取决于其时钟频率,若以 600 kHz 为其时钟频率,则单片机的一个机器周期仅为 20 μs。假设一个工作程序由长度为 1000 个机器周期的指令构成,则单片机执行该程序的时间仅为 20 ms。而对于工业过程中的热工量(压力、物位、温度等)一般可视为缓变量,几秒内一般不会发生很大变化,于是当我们对这些量进行测量时,就可以采用间断的方式,每隔 1 s 或几 s 检测一次,而在两次检测的间隔时间里,使单片机处于十分省电的“掉电”状态,于是整个电路就有可能成百、上千倍地降低功耗了。但是,凡事有一利就会有一弊,用单片机的“掉电”模式能大大降低功耗,同时也带来了许多麻烦和问题:单片机掉电后内部时钟停止工作,内部计时器、计数器都不能使用了;另外,“掉电”后的单片机要恢复工作,只能通过给“Reset”端加高电平的方式——“复位”来实现,而“复位”又会使单片机的全部口线输出高电平,内部寄存器、计时/计数器清零等等,因此要使用单片机的“掉电”模式必须妥善解决好这些问题,兴利除弊,以达到我们既降低功耗又实现功能的目的。

用单片机进行测量,不论是对何种热工量,也不论输入信号是脉冲还是其他形式的,很多情况下是利用计数或计频的方法,经过一系列的数学运算及标度变换,显示或输出相应的信息。但是,在单片机处于“掉电”状态时,机内的定时/计数器停止工作,对于需要用累积脉冲数进行测量的场合,必须外接计数器。外接计数器的计数容量可根据单片机所测信号的最高频率而定,以一个复位周期(单片机相邻两次正常工作之间的时间间隔为一个复位周期)内最高信号频率时不会使计数器溢出为原则。复位周期的长短取决于对被测参数测量实时性的要求,一般可以是 1 s 至几 s 甚至几十 s。在单片机处在“掉电”状态时,机内时钟停振,各 I/O 口线处于“掉电”前的状态,单片机恢复工作只能靠外部的“复位”脉冲触发。因此,使用掉电模式时,单片机外必须有一套可靠的、低功耗的定时复位脉冲发生电路,该电路产生的复位脉冲的时间间隔即为单片机的工作周期(或称复位周期)。单片机“复位”后,全部 I/O 口线均输出高电平,只有内部 RAM 中的内容保持进入“掉电”前瞬间的状态。因此,在使用“掉电”模式时,单片机所连接的输出设备,包括显示报警等,一般应具有静态锁存功能,且应为单片机输出低电平时选通,否则就可能产生误动作。

五、应用方案介绍

作者最近设计了一套以 51 单片机为核心的气体流量测量装置,用途是累积流过管路某一截面的理想气体的标准状态下的体积量(标方值)并将它显示出来,同时还可显示出工作状态

下的温度、压力等信息,显示状态的变换是靠设计在面板上的显示转换键实现的。用户要求仪表现场就地显示且无需外供电,电池寿命不少于 2 年。根据用户以上要求,作者选用了目前国内可批量供货的容量/ 价格比较高的 10 Ah 锂电池。为了延长电池的使用寿命,就要设法降低仪表的功耗,于是设计了基于掉电模式的单片机工作方案。由用户提供的流量传感器是信号频率最高只有 100 Hz 左右的螺茨流量传感器;而压力、温度均采用半导体集成传感器测得的模拟信号,经 A/ D 转换送入单片机。整个装置的原理如图 2.2-2 所示。

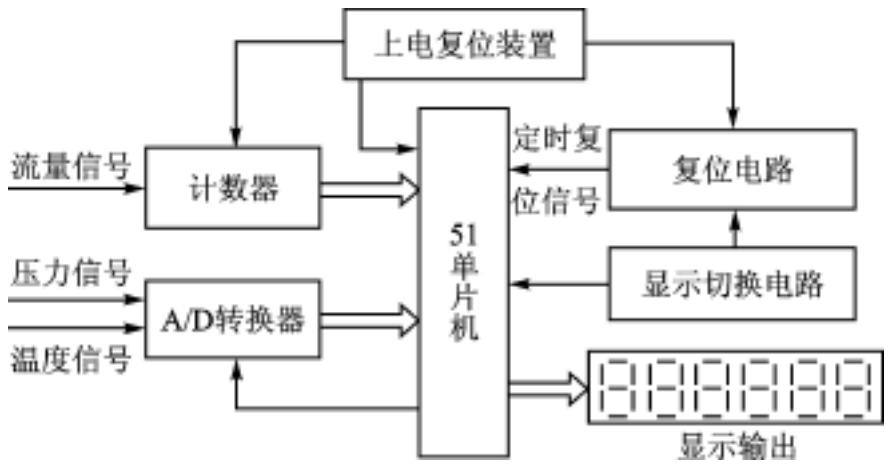


图 2.2-2　硬件框图

单片机的定时复位信号由机外的复位电路产生;显示输出选择静态锁存方式的液晶显示屏;上电复位装置可实现单片机的上电复位及计数器、累积总量的清零;单一的显示切换键可实现流量总量、工、压力等值的循环显示切换。在无人操作,无人查看时,仪表屏幕清屏,处于屏幕保护状态。

硬件电路的设计为单片机系统实现低功耗创造了物质条件,能否最大限度地发挥硬件电路的节电功效,还有赖于应用软件的设计、运行。首先,根据实际需要可确定仪表的工作周期为 2 s,即每隔 2 s 读一次累积流量脉冲的计数器,而对于被测介质的压力、温度,由于它们相对于被测介质的流量来讲是一个缓变参数,因此,更可以隔若干个工作周期测量一次,实际工作中定为每隔 2 min 测一次温度、压力值。这样在不对温度、压力进行测量时,可用软件控制不给温度、压力传感器及相应的 A/ D 转换电路供电,实现“分区供电”,从而进一步降低功耗。另外,在长时间无人按键时使仪表显示“清屏”也是通过软件设计实现的。图 2.2-3 是该装置的程序流程。

经实验测试,该电路平均功耗仅为 80 μA,用 10 Ah 的锂电池供电,预期寿命可达 10 年之久,完全可满足用户 2 年的要求。

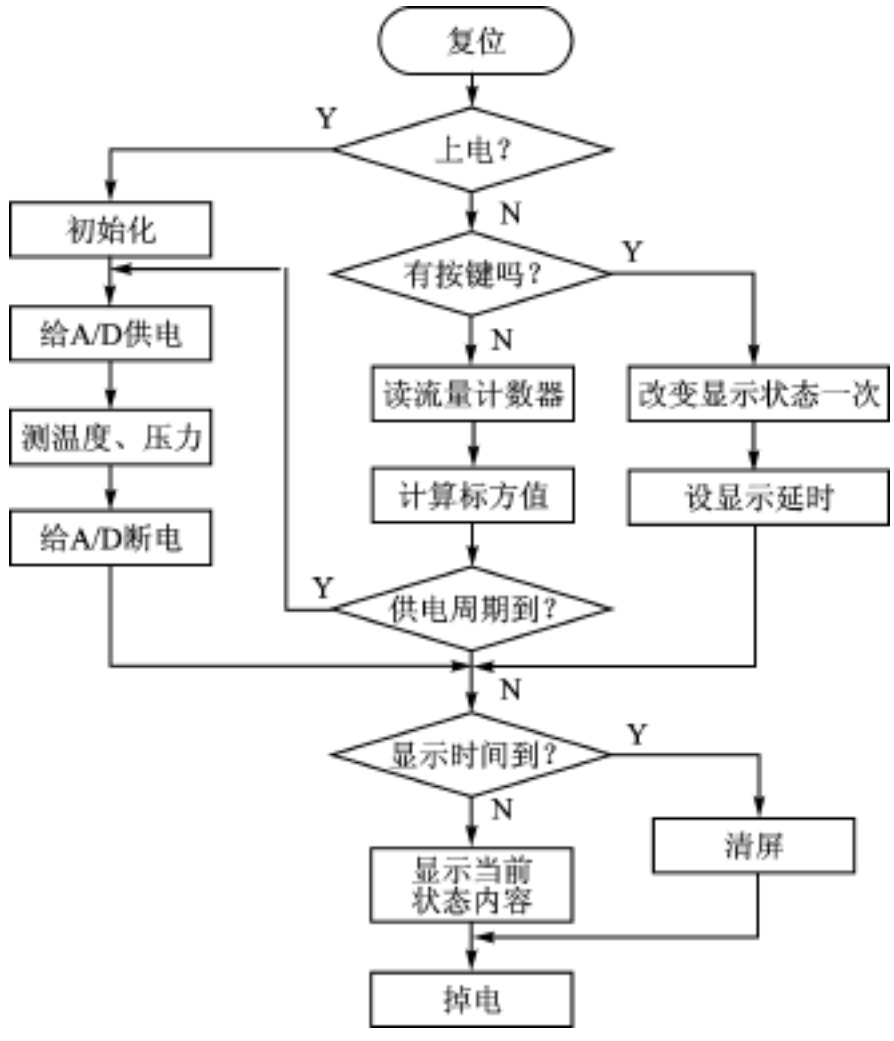


图 2.2-3 软件流程图

参 考 文 献

1 何立民.低功耗单片微机系统设计[M].北京:北京航空航天大学出版社,1994

2 孙涵芳,徐爱卿.MCS-$\frac{51}{96}$系列单片机原理及应用[M].北京:北京航空航天大学出版社,1994

选自《制造业自动化》月刊,2000 年第 11 期

2.3 分布式实时应用的两个重要问题

华中理工大学计算机学院 胡贯荣 谢美意

一、引 言

分布式实时应用是这样一类任务[1],任务的活动是物理分布的,整个任务的正确性不仅依赖于执行结果,而且依赖于任务的完成时间,即有时间限制(dead-line)。分布式实时应用有广泛的应用领域[2,3],例如在工业自动控制、航空等领域,计算机通过特定的外部设备如传感器感知外部事件,并立即作出响应,这里的活动都要求实时性,并且根据现场信息,启动或终止活动。

通常,依据实时应用的性质,实时应用有两种类型:硬实时(hard real-time)和软实时(soft real-time)。硬实时应用指任务在指定期限内必须完成,否则整个实时应用的效益立即降为零,有时需要终止,以防灾难性后果;软实时应用指任务在指定期限内必须完成,否则整个实时应用的效益可能成线性下降,直至为零。一般地,分布式实时应用有以下特性:

(1) 时间期限的限制;

(2) 应用的功能是相对确定的;

(3) 应用的活动是分布的,活动之间有同步关系或消息通信;

(4) 应用的行为一般是事先不可准确感知的。

当前有很多研究工作围绕实时应用展开,其中实时应用的可预测性(predictability)和分布式实时 IPC(Inter Process Communication)是分布式实时应用的两个重要问题。本文主要讨论分布式实时应用的可预测性概念,提出分布式实时 IPC 的抽象模型,并分析影响分布式实时 IPC 性能的主要因素。

二、可预测性

定义 1 可预测性 在一定特定环境(硬件和软件平台)下,一个实时应用是否能够执行成功即满足正确性的一种预先感知,称为可预测性。

该定义体现了实时应用的最本质特性,也是所有实时应用共同的最为重要的一个需求特性,通常可预测性包括静态可预测性和动态可预测性。

定义 2 静态可预测性 如果一个实时应用在启动之前,某些特性是可预测的,称这种事先可感知性为静态可预测性。

一个分布式实时应用实际上有一些静态可预测的特性,主要包括:

(1) 活动的物理分布及资源需求;

(2) 各个活动所需 CPU 开销,即理想情况下,程序代码对应的 CPU 指令周期总数;

(3) 多个实时应用并发时,能否被调度运行。

定义 3 动态可预测性 如果一个实时应用在启动之后,某些特性可以根据一定的条件加以预测,称这种动态感知性为动态可预测性。

一个分布式实时应用的动态特性主要包括:

(1) 活动的启动时间;

(2) 任务启动后某一时刻到最后执行完所需时间,由尚需 CPU 开销、I/O 开销和分布式 IPC 开销组成。

定义 2 和定义 3 从实现的角度分析了分布式实时应用可预测性,为满足分布式实时应用的可预测性,必须从这两个方面加以保障和支持。

三、分布式实时 IPC

1 . 分布式实时 IPC 模型

分布式实时应用一般由多个活动组成,各个活动可能位于同一站点,也可能位于不同站点,活动之间往往有消息通信,称之为分布式实时 IPC。分布式实时 IPC 的主要特征就是消息必须在一个时间期限内到达,该消息称为有效[4]。

不失一般性,一个分布式实时 IPC 机制可抽象为如图 2 .3 - 1 所示的模型,模型中各模块的功能如下:

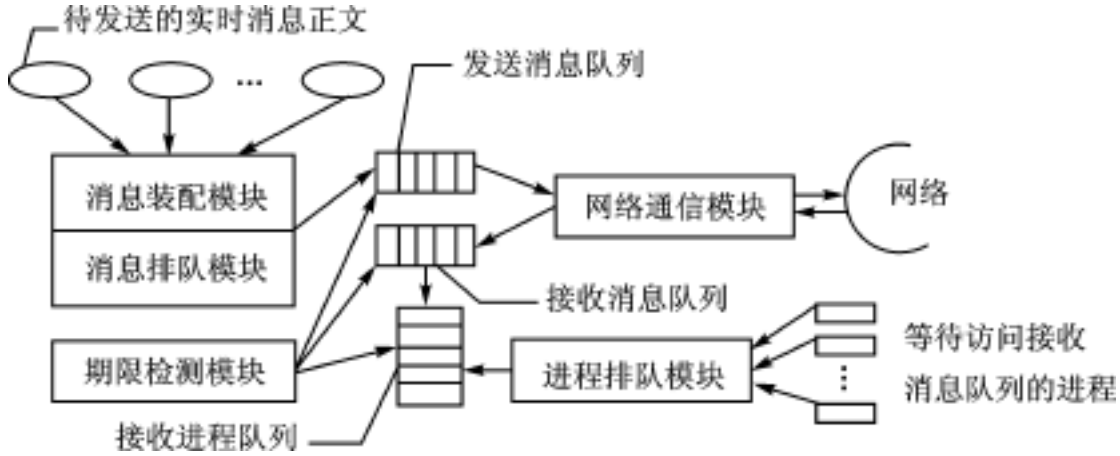


图 2 .3 - 1 分布式实时 IPC 模型

(1) 消息装配模块为实时消息分配消息缓冲区,并给消息正文加上必要的消息首部,形成系统定义的消息结构。其中,消息首部的各域应包括:消息类型、消息发送者、消息接收者、优先级、时限约束信息等。

(2) 消息排队模块按照某种排队策略,将消息插入消息队列中。典型的排队策略有 FIFO 策略、优先数策略、最小限期优先策略等。

(3) 进程排队模块的功能与消息排队模块类似,按某种策略将提出接收消息请求的进程插入进程队列,等待访问接收消息队列。

(4) 期限检测模块定期扫描发送消息队列、接收消息队列及进程接收队列,检测队列中是否有失效(miss dead-line)的消息或进程,并及时提供相应的异常处理。

(5) 网络通信模块负责网络上的消息发送和接收。只要发送消息队列不为空,它就将消息从队列中取出并向网络发送。同时,它也从网络上接收消息并放入接收消息队列中。

2 . 分布式实时 IPC 性能分析

分布式实时 IPC 是影响分布式实时应用可预测性的因素之一,下面我们就从图 2 .3 - 1 所

示的模型来分析影响分布式实时 IPC 性能的主要因素。

根据图 2.3-1 中的模型,一个消息从发送到接收要经历以下过程:

(1) 由消息装配模块分配缓冲区,加上消息首部;

(2) 由消息排队模块插入发送消息队列,等待发送;

(3) 由网络通信模块向网络发送,经网络传送到目的结点;

(4) 由远程网络通信模块接收后,入接收消息队列,等待接收。

因此,整个过程的时间开销 Ttransfer 可表示为:

$$Ttransfer = Talloc\_buf + Tin\_sendq + Tnet\_delay + Tin\_rcvq$$

其中,Talloc_buf 为缓冲区资源不足时的等待时间;Tin_sendq 为消息在发送队列中的等待时间;Tnet_delay 为网络传输时延(Tnet_delay≤τ),Tin_rcvq 为消息在接收列列中的等待时间。

由于 Tnet_delay 由网络性能决定,与 IPC 的实现机制无关,以下只讨论 Talloc_buf、Tin_sendq 和 Tin_rcvq。

(1) Talloc_buf

影响 Talloc_buf 值的因素是消息装配模块采用的缓冲区分配策略。IPC 通常采用消息到来后临时分配缓冲区的方法,这样可提高系统资源的利用率,但同样也提高了消息因等待缓冲区而阻塞的概率。在实时系统中,为缩短因缓冲区资源不足的等待时间,需要为实时消息预先分配缓冲区。

(2) Tin_sendq

Tin_sendq 值的大小取决于某时刻消息到达的密度及消息排队模块采用的排队策略。在到达密度相同的情况下,无论采用什么排队策略,所有消息的等待时间之和是相同的,但是对某一具体的消息,不同的排队策略导致的等待时间则可能相差很大。由于排队的目标是保证尽可能多的消息在死限前到达,选择策略时要综合考虑消息的类型、优先数、时间约束等。

(3) Tin_rcvq

影响 Tin_rcvq 值的因素包括接收进程的执行状态、进程排队模块采用的排队策略等。接收进程的状态分析起来较复杂,与远程结点的进程调度、资源分配等情况有关,这超出了我们的讨论范围。如果只考虑消息到达时,接收进程正在等待接收的情况,则 Tin_rcvq 只与排队策略有关,不同的排队策略将导致不同的等待时间。同样,进程排队的目标是保证尽可能多的消息在死限前被接收,选择策略时也应该综合考虑进程的优先数、时间约束等。

四、结束语

本文探讨了分布式实时可预测性的概念和分布式实时 IPC 模型,并分析了影响分布式实时 IPC 性能的主要因素,对分布式实时应用的设计有一定的指导作用。

参 考 文 献

1 anenbaum A S . Distributed Operating Systems [M] . 北京:清华大学出版社,1996

2 Tokuda H, et al . ARTS: A Distributed Real-Time System [J] . ACM Operating System Review, 1989, 23 (3)

3 Kandler D D, et al . HARTOS: A Distributed Real-Time Operaing System [J] . ACM Operating System Review, 1989, 23(3)

4 Kitayama T, et al . RT-IPC: An IPC Extension for Real-Time Mach [M]

5 都幸帆,等 .UNIX 的进程通信及应用[J] .小型微型计算机系统,1990,20(1)

6 郭锐峰 .分布式实时操作系统中实时通信的研究[J] .小型微型计算机系统,1995,20(8)

7 穆兵 .实时 UNIX 系统主要性能指标的测量方法 [J] .计算机工程与设计,1992,(3)

选自《计算机工程与科学》双月刊,2000 年第 1 期

2.4 分布式运算单元的原理及其实现方法

南京 724 所信号处理室(210003) 蒋亚坚 张庆雷

随着 FPGA 集成度的不断提高,在单片 FPGA 中完成复杂的数字信号处理过程变成了现实。譬如:FIR 滤波器、FFT 以及雷达信号处理中的数字脉冲压缩、数字鉴相等,都可以在单片 FPGA 中实现。在基于 Xilinx XC4000 系列 FPGA 设计的 DSP 中,分布式运算单元 DA 扮演着重要的角色。本文介绍其原理及其实现方法。

一、分布式运算单元原理

DA 的运算原理非常简单,但是它的应用却十分广泛。

一个线性时不变网络的输出可以用下式表示:

$$y(n) = \sum_{k=1}^{K} A_k X_k(n) \tag{1}$$

其中,$y(n)$为第 n 时刻网络的输出;$X_k(n)$为第 n 时刻的第 k 个输入变量;A_k 为第 k 个输入变量的权值。

在线性时不变系统中,对于所有 n 时刻,A_k 都是常量。如果该网络表现为滤波器,常量 A_k 即为滤波器系数,变量 X_k 为单一数据源的抽样数据(如 A/D 的输出)。而在时-频转换系统中(如离散傅里叶变换及快速傅里叶变换),常数 A_k 即为旋转因子值,变量 X_k 为单一数据源的数据块(多源数据的例子可以在图像处理系统中发现)。

仔细观察式(1)可以看出,单个输出 y(n)需要将 k 个乘积累加。在以 XC4000 系列 FPGA 中的可配置逻辑功能块(CLB)的查找表(Look-Up Table)结构[1]为基础的 DA 中,这种乘积累加可以由查找表来实现。XC4000 系列的 CLB 结构特点使得它很容易被高效地配置。

为了使得乘法之后的数据宽度不至于展宽,先把数据源数据格式规定为浮点数 2 的补码形式。需要注意的是,常数 A_k 不一定要进行格式转换来匹配输入数据的格式,它可以根据所要求的精度进行定义。

变量 X_k 可以用下式表示:

$$X_k = -X_{k0} + \sum_{b=1}^{B-1} X_{kb} 2^{-b} \tag{2}$$

其中,X_{kb}为二进制数,即取值为 0 或 1;X_{k0}为符号位,X_{k0}为 1 表示数据为负,为 0 表示数据为正。

将式(2)代入式(1)可以得到:

$$\begin{aligned} y &= \sum_{k=1}^{K} A_k \left[-X_{k0} + \sum_{b=1}^{B-1} X_{kb} 2^{-b} \right] \\ &= -\sum_{k=1}^{K} X_{k0} \cdot A_k + \sum_{k=1}^{K} \sum_{b=1}^{B-1} X_{kb} \cdot A_k 2^{-b} \end{aligned} \tag{3}$$

将求和符号展开,可以得到式(4):

$$\begin{aligned} y = -&[X_{10}\cdot A_1 + X_{20}\cdot A_2 + \cdots + X_{K0}\cdot A_K] + \\ &[X_{11}\cdot A_1 + X_{21}\cdot A_2 + \cdots + X_{K1}\cdot A_K]2^{-1} + \\ &[X_{12}\cdot A_1 + X_{22}\cdot A_2 + \cdots + X_{K2}\cdot A_K]2^{-2} \\ &\quad\vdots \\ &[X_{1(B-2)}\cdot A_1 + X_{2(B-2)}\cdot A_2 + \cdots + X_{K(B-2)}\cdot A_K]2^{-(B-2)} + \\ &[X_{1(B-1)}\cdot A_1 + X_{2(B-1)}\cdot A_2 + \cdots + X_{K(B-1)}\cdot A_K]2^{-(B-1)} \end{aligned} \tag{4}$$

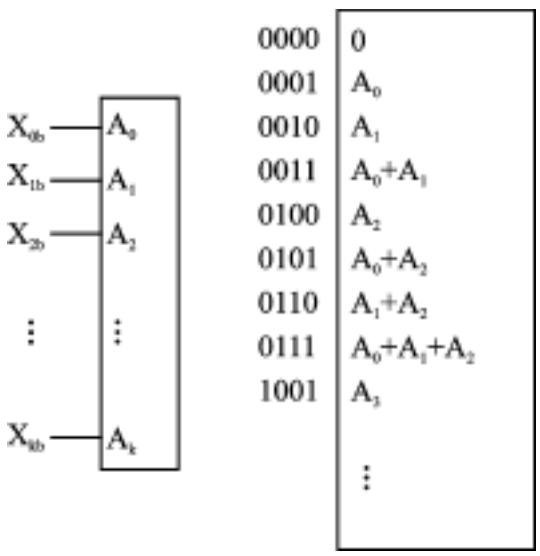


图 2.4-1 DA 查找表的寻址方式及其内容

可以看出,每个方括号中进行的是输入变量的某一个数据位和所有常数($A_1 \sim A_k$)的每一位进行位"与"并求和。而指数部分则说明了求和结果的位权。现在就可以建立查找表来实现方括号中的操作了,其查找表用所有输入变量的同一位进行寻址,如图 2.4-1 所示。

图 2.4-1 中所示的 DA 查找表,其宽度为对常数 A_k 定义的宽度,深度为 2^K。K 是能够对数据源抽样数据进行处理的数据长度。对于滤波器就表现为滤波器阶数;对于 FFT 就表现为 FFT 点数。

这样,式(1)所表示的方程就可以由加法、减法和二进制除法来实现了。但是,DA 仅仅是运算方程(1)的核心,要完成式(1)还需要根据系统对时间以及 FPGA 资源的考虑,选择相应的方法。

二、几种实现方法

1. 全并行实现方法

市场上已经有大量的通用 DSP 芯片,这些芯片以并行的乘法、加法运算,地址产生器和片内存储器为主要特点,如 TMS320C620x、ADSP2106x 及各种通用的 FFT 芯片(如 PDSP16510)。为什么还要选择 FPGA 呢?主要是考虑速度。要实现一个 64 阶 FIR 滤波器,如果采用全并行方式,则 FPGA 可做到 50 MHz 的数据率,可以和系统时钟相匹配,这是通用 DSP 芯片无法做到的。下面就举出全并行的例子。

若将式(4)每个方括号之间的加法并行执行,即将每个 DA 查找表的输出采用并行的加法,就得到了全并行结构。现将式(4)中的某个方括号重写如下,并缩写为 sum:

$$[X_{12}\cdot A_1 + X_{22}\cdot A_2 + \cdots + X_{K2}\cdot A_K] = [\mathrm{sum2}]$$

同样道理,

$$[X_{1(B-1)}\cdot A_1 + X_{2(B-1)}\cdot A_2 + \cdots + X_{K(B-1)}\cdot A_K] = [\mathrm{sum}(B-1)]$$

设 B = 16,则由式(4)得式(5)为:

$$y = -[\mathrm{sum0}] + [\mathrm{sum1}]2^{-1} + [\mathrm{sum2}]2^{-2} + \cdots + [\mathrm{sum14}]2^{-14} + [\mathrm{sum15}]2^{-15} \tag{5}$$

将式(5)改写如下:

$$y = -[sum0] + [sum1]2^{-1} + \{[sum2] + [sum3]2^{-1}\}2^{-2} +$$
$$\{[sum4] + [sum5]2^{-1} + \{[sum6] + [sum7]2^{-1}\}2^{-2}\}2^{-4} +$$
$$\{[sum8] + [sum9]2^{-1} + \{[sum10] + [sum11]2^{-1}\}2^{-2} +$$
$$\{[sum12] + [sum13]2^{-1} + \{[sum14] + [sum15]2^{-1}\}2^{-4}\}2^{-8} \quad (6)$$

利用式(6),可以得到一种直观的树形阵列,如图 2.4-2 所示。

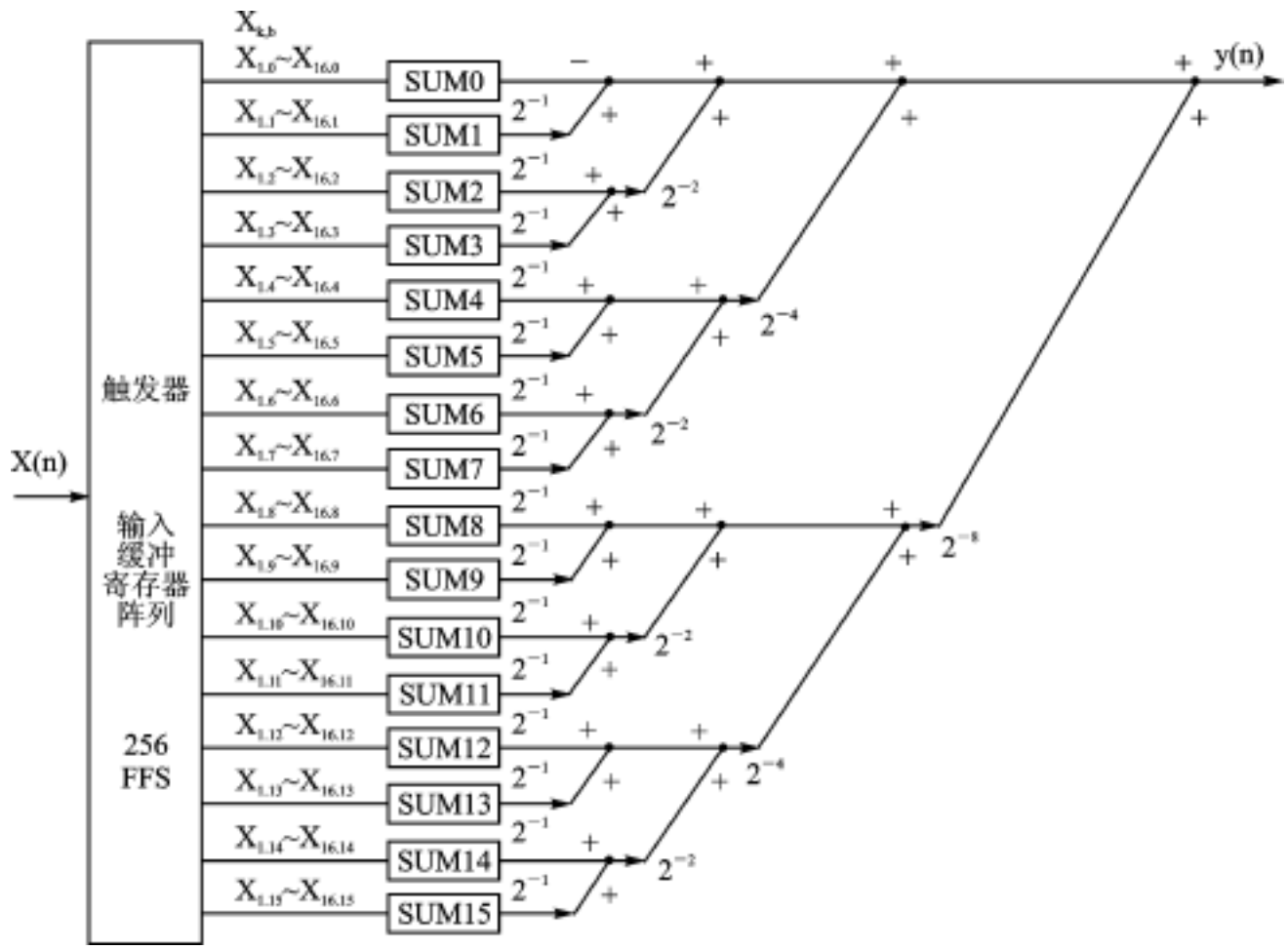


图 2.4-2 全并行 DA 模式的示例(K = 16,B = 16)

图 2.4-2 中,首先要建立一个 K×B 位的寄存器阵列,将其输出进行排列,用所有 K 个输入数据的相同位,对 DA 查找表寻址。从图中可以看出,当 B = 16 时,输入到输出所需的路径最长,该路径为关键路径,影响着电路处理的速度,在进行设计时应该注意到这点,所以应该采用流水线设计方法[1],并进行适当的约束,其数据率可以达到 50 MHz。图中的 15 个节点代表着 15 个并行的加法器,中间过程的数据宽度既可以保持双精度(B + C)位数据(C 是常数 A_k 的宽度),也可以采用截尾的办法得到单精度 B 位数据,可根据系统所要求的精度确定。

2. 全串行实现方法

当系统对速度的要求不是很高的时候,可以用全串行设计方法,即利用一个 DA 查找表。一个并行的加法器以及简单少量的寄存器就可达到目的,这样能够节省大量的 FPGA 资源。同样,设 K = 16,B = 16,将式(4)改写如下形式:

$$y = \{[sum15]2^{-1} + [sum14]\}2^{-1} + [sum13]\}2^{-1} +$$
$$[sum12]\}2^{-1} + [sum11]\}2^{-1} + [sum10]\}2^{-1} + [sum9]\}2^{-1} +$$
$$[sum8]\}2^{-1} + [sum7]\}2^{-1} + [sum6]\}2^{-1} + [sum5]2^{-1} +$$
$$[sum4]\}2^{-1} + [sum3]\}2^{-1} + [sum2]\}2^{-1} + [sum1]\}2^{-1} - [sum0] \quad (7)$$

为了实现式(7),先从最低位开始,用所有 K 个输入变量的最低位对 DA 查找表进行寻

址,得到了[sum15],将[sum15]右移一位即将[sum15]乘 2^{-1} 后,放到寄存器中,设为[tem15];同时,K 个输入变量的次低位已经开始对 DA 查找表寻址得到[sum14],右移一位后,与[tem15]相加,重复这样的过程,直至得到[sum0],并用前面得到的累加结果减去[sum0]。要实现上述过程,需要 K 个长度为 B 的串并行转换移位寄存器、一个容量为 $2^K \times C$ 的 DA 查找表和一个累加器。该全串行电路的数据率为输入数据抽样频率的 1/ B,即完成一次运算需要 B 个时钟周期。由此可以得到全串行 DA 模式,如图 2 .4 - 3 所示。

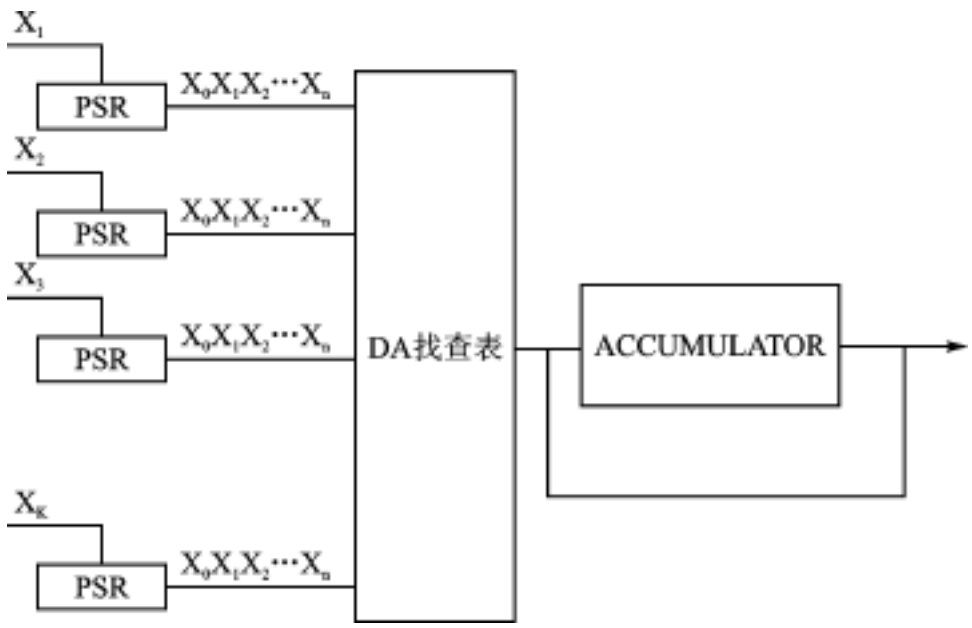


图 2 .4 - 3 全串行 DA 模式的示例

3 . 串并行相结合实现方法

以上介绍的全串行方式是每个时钟周期对所有 K 个变量的一位进行串行处理,全并行方式是每个时钟周期对所有 K 个变量的所有 B 位进行并行处理,这两种方法是针对速度优化和资源优化设计的两种极限情况。在有些情况下,我们可以对这两种情况进行折中考虑,获得最佳资源利用和系统速度。我们可以从每个时钟周期对 K 个变量的两位进行处理开始着手,回顾一下式(5),并将该式改写如下:

$$\begin{aligned} y = -[sum0] + \{[sum1] + [sum2]2^{-1}\}2^{-1} + \\ \{[sum3] + [sum4]2^{-1}\}2^{-3} + \\ \{[sum5] + [sum6]2^{-1}\}2^{-5} + \\ \{[sum7] + [sum8]2^{-1}\}2^{-7} + \\ \{[sum9] + [sum10]2^{-1}\}2^{-9} + \\ \{[sum11] + [sum12]2^{-1}\}2^{-11} + \\ \{[sum13] + [sum14]2^{-1}\}2^{-13} + [sum15]2^{-15} \end{aligned} \tag{8}$$

完成该式功能的功能框图如图 2 .4 - 4 所示。

将图 2 .4 - 4 与图 2 .4 - 3 进行比较后就可以发现,图 2 .4 - 3 中的 DA 查找表由 16 个输入变量的同一位进行寻址,而图 2 .4 - 4 中的 DA 查找表的寻址是由 16 个输入变量的连续两位进行的,即寻址的位数由 16 位变成了 32 位。这样,查找表的内容也需要相应地改变;而且完成一次运算也由原来的 B 个时钟周期变成了需要 B/ 2 + 1 个时钟周期。

下面介绍一种更易于理解的串并行混合设计方法。

将式(5)改写成如下形式:

$$y = \begin{vmatrix} -[sum0] \\ +[sum1]2^{-1} \\ +[sum2]2^{-2} \\ +[sum3]2^{-3} \\ +[sum4]2^{-4} \\ +[sum5]2^{-5} \\ +[sum6]2^{-6} \\ +[sum7]2^{-7} \end{vmatrix} + \begin{vmatrix} [sum8] \\ +[sum9]2^{-1} \\ +[sum10]2^{-2} \\ +[sum11]2^{-3} \\ +[sum12]2^{-4} \\ +[sum13]2^{-5} \\ +[sum14]2^{-6} \\ +[sum15]2^{-7} \end{vmatrix} \cdot 2^{-8} \tag{9}$$

从式(9)得到流程图如图 2.4-5 所示。

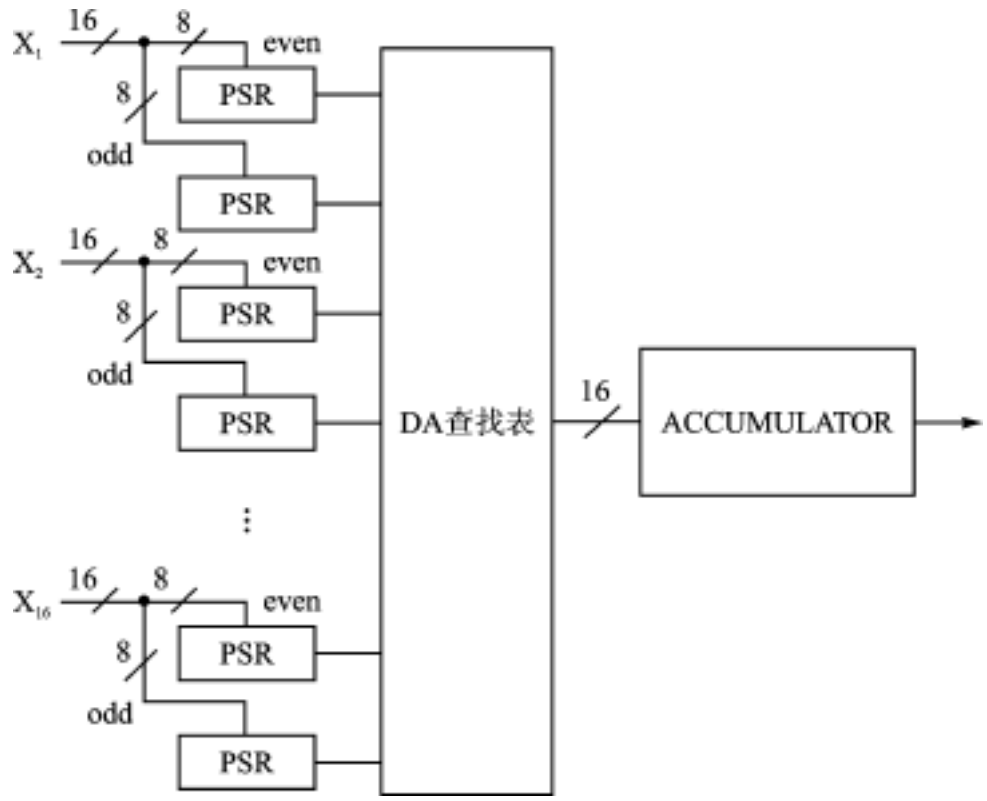


图 2.4-4 每个时钟周期处理两个数据位的 DA 流程图

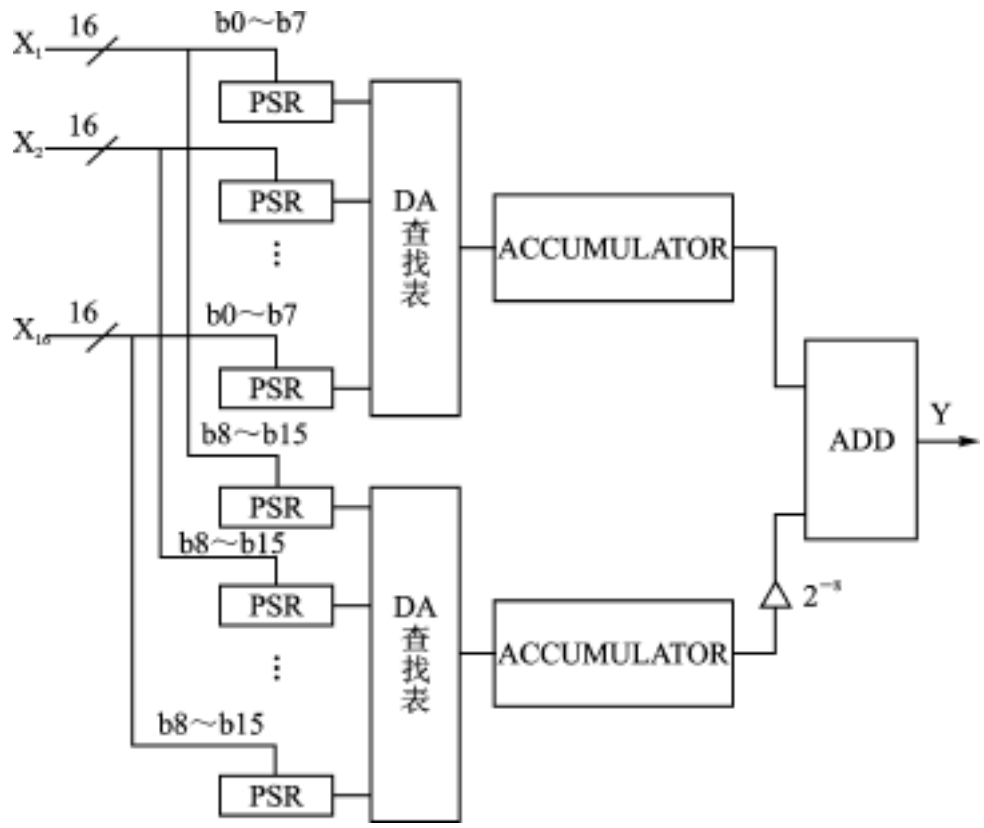


图 2.4-5 单时钟处理两个数据位的 DA 流程图

实现过程中应该注意 DA 查找表的内容,累加之前要乘 2^{-1},注意进位等。

从以上给出的两种串并行结合的设计方法可以看到,只要将式(5)进行适当的变换。还有其他的硬件实现方法,这里就不一一叙述了。

下面给出在 K = 8、B = 16 的情况下,不同的 DA 查找表所占用的资源。Xilinx 公司的 XC4000 系列 FPGA 的一个 CLB 可以实现 32×1 大小的 RAM,在图 2.4-4 中所描述的 DA 查找表占用 2 048 个 CLB,而在图 2.4-5 中所描述的两个 DA 查找表只占用 256 个 CLB。用一片 XC4025 即可完成后者,其数据率可达到 16 MHz。

综上所述,由于分布式运算单元的应用,改变了传统的设计观念,为基于 FPGA 的 DSP 设计提出了新的思路,必将在高速的 FIR 滤波器设计、高速 FFT 设计中得到广泛的应用。随着 FPGA 集成规模的不断提高(Xilinx 公司 Virtex 系列已经达到了百万门级),许多复杂的数学运算已经可以由 FPGA 来实现,单片 FPGA 实现系统的设想即将变为现实。

参 考 文 献

1 蒋亚坚,沈桂明.FPGA 在雷达信号处理器中的应用研究.雷达与对抗,1999(2):57～63

2 Xilinx. The Programmable Logic Data Book. 1999

选自《电子技术应用》月刊,2000 年第 2 期

2.5 用 PLD 器件设计逻辑电路时的竞争冒险现象

中国科学技术大学 金 革 聂际敏

一、引 言

由于采用可编程逻辑器件(PLD)能方便、灵活地实现逻辑功能,而且集成度高。因此,基于芯片的设计方法正在成为电子系统设计的主流,越来越多的人在电路设计中使用 PLD 器件实现复杂的逻辑。常见的 PLD 器件有 CPLD、EPLD、GAL、PAL、PLA、EPGA 和 PROM 等。

虽然 PLD 器件使用方便灵活、集成度高,但也正是由于这个原因,使其内部结构比较复杂,输入输出之间的延迟时间随着使用的功能单元的在 PID 器件内部位置不同、使用的内部连线的类型和长短不同、以及输入输出单元的位置不同而不同。一些功能较强的 PLD 开发平台虽然能提供时序模拟仿真,但由于 PLD 器件结构比较复杂和一些设计平台的局限性,很少有人能直接控制信号在器件内部的实际路径,从而无法保证各路信号保持同样的时间延迟,因此,在利用 PLD 器件设计组合电路时常常会遇到竞争冒险现象。

二、竞争冒险概念

所谓竞争冒险,是指在 n 个输入信号 $x_1, x_2, \cdots, x_n$ 中,如果其中有 p 个输入信号 $x_1, x_2, \cdots, x_p$,发生变化$(1 < p < n)$,即输入信号由 X 变到 Y,其中

$$X = (x_1, x_2, \cdots, x_n, x_p, x_{p+1}, x_{p+2}, \cdots, x_n)$$

$$Y = (\overline{x_1}, \overline{x_2}, \cdots, \overline{x_n}, \overline{x_p}, x_{p+1}, x_{p+2}, \cdots, x_n)$$

如果在稳定时,组合电路的输出函数 $F(X) = F(Y)$,在输入信号由 X 变到 Y 的瞬间,组合电路输出有过渡性干扰脉冲现象叫做竞争冒险。这种过渡性干扰脉冲的宽度很窄,幅度一般也达不到正常的电平幅度,像毛刺一样,所以有时称其为毛刺。如果负载电路对脉冲十分敏感,如计数器的输入端和清除端,在接收到由竞争冒险产生的毛刺信号时,就会产生计数错误,因此需要采取必要措施消除竞争冒险。

三、PLD 器件中产生竞争冒险的原因

逻辑电路的竞争冒险可分为两种类型。第一种是当有两个或两个以上的输入信号发生变化时,由于可能经历不同变化途径而产生的竞争冒险现象,称为功能竞争冒险。如逻辑电路的函数表达式为:

$$F = A\overline{C} + CD + B\overline{D}$$

当输入信号 ABCD 由 0100 变化到 1101 时,在稳定情况下,$F(0,1,0,0) = F(1,1,0,1) = 1$。在 A、D 两个输入信号发生变化时,可能出现 A 先于 D 或 D 先于 A 的情况,如果 D 先于 A,则输入信号要经历 0100→0101→1101 的途径,由于 $F(0,1,0,1) = 0$,因此在输出中将会出

现 1→0→1 的情况,F(0,1,0,1)是一个短暂过程,输出 1 中的 0 也是短暂的,这就是竞争冒险。若 A 先于 D,则输入信号要经历 0100→1100→1101 的途径,而 F(1,1,0,0) = 1,因此在输出中将不会有毛刺产生。另一种类型是由于门和路径的传输延迟不同,可能产生的冒险现象,成为逻辑竞争冒险。

在使用 PLD 器件设计逻辑电路时,通常是采用一些与器件无关的硬件描述语言(VHDL)来进行逻辑电路设计,如 ABEL、CUPL 等,然后由设计平台进行逻辑优化、器件内部资源分配和连线,设计者无需也很难直接控制逻辑电路在器件内部的布局和连线。由于无法控制逻辑的优化过程、无法控制逻辑电路的器件内部的分布和指定信号的路径,因此无法控制信号的延迟时间。当信号在器件经历不同的路径时,就会产生不同的延迟,因此,在 PLD 器件中两种类型的竞争冒险都有可能产生。

四、如何判断竞争冒险的出现

在完成了 PLD 器件的逻辑设计之后,应采取一些必要的手段检查是否会出现竞争冒险现象,当可能出现的竞争冒险现象会影响到后续电路的逻辑和电路的可靠性时,就需要采取必要的措施加以消除。具体的判断规则如下。

1.功能竞争冒险

如果逻辑电路中有 n 个输入变量,当有 p 个变量发生变化时(p > 1),变化前后稳定时,F(X) = F(Y)。若由不变的(n - p)个输入变量组成的 2^{n-p} 个乘积项中,既有 1 又有 0,则 p 个变量发生变化时,就有可能发生功能竞争冒险。

2.逻辑竞争冒险

当 p 个变量发生变化时(p≥1),如果判断出不会发生功能竞争冒险,但在函数的最简“与”-“或”表达式中不包含由(n - p)个不变变量组成的乘积项,则有可能发生逻辑竞争冒险;如果包含有该乘积项,则不会发生逻辑竞争冒险。

五、消除竞争冒险的方法

1.封锁法

为了消除竞争冒险所产生的毛刺,可以引入一个负脉冲,在输入信号发生竞争的时间内,把可能产生毛刺的门封锁住,如图 2.5-1(a)所示。

2.选通法

采用选通脉冲,如图 2.5-1(b)所示。由于选通脉冲的作用时间是在电路达到新的稳定状态之后,所以输入的过渡状态产生的竞争冒险不会在选通时间内出现。计算机总线上的读写脉冲$\overline{RD}$和$\overline{WR}$就是选通脉冲,当地址和数据线上信号稳定之后,$\overline{RD}$或$\overline{WR}$出现,将数据读出或写入存储器/寄存器。

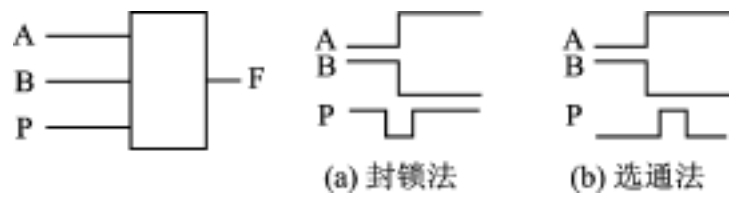


图 2.5-1 消除竞争冒险的方法

3 . 滤波电容法

因为竞争冒险产生的干扰脉冲一般很窄,所以可以在输出端和地之间接一个几十到几百皮法的滤波电容来消除干扰脉冲,如图 2 .5 - 2 所示。但 PLD 器件内部产生的竞争冒险无法用此种方法来消除干扰。另外,由于输出端接入了滤波电容,会使输出信号的上升和下降沿变慢。

4 . 增加冗余项法

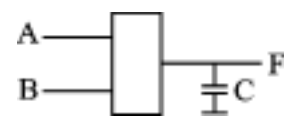


图 2 5 - 2　采用滤波电容 C 消除毛刺

引入封锁脉冲和选通脉冲虽然可以有效地防止竞争冒险,但需要有适当的 P 信号,当电路中没有适当的信号可以用来作为 P 信号时,就需要在 PLD 器件的内部或外部产生这个 P 信号,这无疑会增大设计的复杂性。由于 PLD 器件门的资源比较丰富,因此可以通过增加冗余项的方法来防止竞争冒险。其方法是逻辑函数由全部主要项构成,而不是最简式。对于函数:

$$F = A\overline{C} + CD + B\overline{D}$$

可修改为:

$$F = A\overline{C} + CD + B\overline{D} + AB + BC + AD$$

其中 AB、BC、AD 是冗余项。在 PLD 器件中,只要资源够,增加冗余项不是一件困难的事情,但需要保持信号在路径上的延迟一致,防止出现功能竞争冒险。需要注意的是,有些开发平台在编译时会对逻辑进行优化,常常会将冗余项优化掉,因此应把编译的优化选项关掉。

六、结　束　语

在使用 PLD 器件设计实践中,只要我们对可能出现竞争冒险的地方进行仔细地分析,采取适当的措施,竞争冒险带来的毛刺干扰就有可能被抑制。

选自《电子测量技术》双月刊,2001 年第 3 期

2.6 IRIG-B 格式时间码解码接口卡电路设计

北京航天指控中心 张向荣

一、IRIG-B 格式码的格式与规范

图 2.6-1 为 B(DC)码示意图。它是每秒一帧的时间串码,每个码元宽度为 10 ms,一个时帧周期包括 100 个码元,为脉宽编码。码元的“准时”参考点是其脉冲前沿,时帧的参考标志由一个位置识别标志和相邻的参考码元组成,其宽度为 8 ms;每 10 个码元有一个位置识别标志:P_1,P_2,P_3,…,P_9,P_0,它们均为 8 ms 宽度;P_R 为帧参考点;二进制“1”和“0”的脉宽为5 ms 和 2 ms。

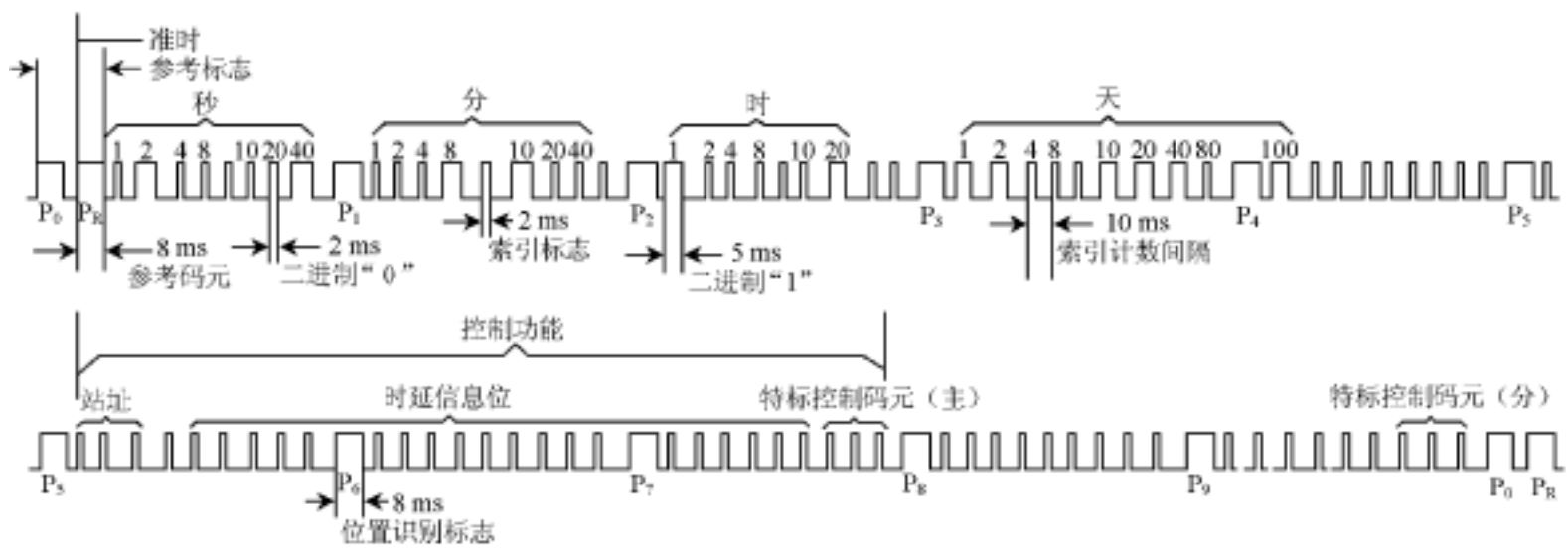


图 2.6-1 B(DC)码示意图

一个时间格式帧从帧参考标志开始。因此连续两个 8 ms 宽脉冲表明秒的开始,如果从第二个 8 ms 开始对码元进行编码,分别为第 0,1,2,…,99 个码元。在 B 码时间格式中含有天、时、分、秒,时序为秒 - 分 - 时 - 天,所占信息位为秒 7 位、分 7 位、时 6 位、天 10 位,其位置在 P_0～P_5 之间。P_6～P_0 包含其他控制信息。其中“秒”信息:第 1,2,3,4,6,7,8 码元;“分”信息:第 10,11,12,13,15,16,17 码元;“时”信息:第 20,21,22,23,25,26,27 码元;第 5,14,24 码元为索引标志,宽度为 2 ms。时、分、秒均用 BCD 码表示,低位在前,高位在后;个位在前,十位在后。

二、B 码解码接口卡设计方案

B 码解码接口卡功能框图如图 2.6-2 所示。

对 B 码进行解码就是将 B 码中所包含的时、分、秒信息提取出来,转换成主计算机能够识别的形式,同时以秒的准时点为参考,生成毫秒信息,一同送入主计算机中。解码的关键在于检测 B 码中各个码元的高电平宽度,首先要检测连续两个 8 ms 宽的码元出现的位置,然后再检测随后的 30 个码元脉冲宽度,以确定时、分、秒。这里不检测天的值,天可以直接在主计算

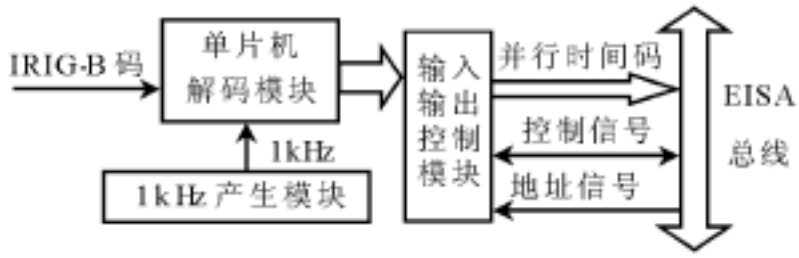


图 2.6-2　B 码解码接口卡功能框图

机上设置。

检测高电平宽度:将 B 码送入单片机 89C51 的$\overline{\text{INT0}}$端,在$\overline{\text{INT0}}$引脚由低电平变为高电平时,启动单片机的内部定时器 T0,开始计数;在$\overline{\text{INT0}}$引脚由高变低时(即下降沿),触发$\overline{\text{INT0}}$中断,读取计数器的值,脉冲宽度等于计数值乘以计数周期。

形成毫秒值:将 1 kHz 信号接到单片机的$\overline{\text{INT1}}$端,每毫秒产生一次中断。$\overline{\text{INT1}}$中断处理程序完成毫秒计数,当计到 1 000 ms 时完成秒加 1。

三、电路设计

1. 电路设计方案 1

方案 1 中,输入输出模块由 5 片锁存器构成,接口控制简单,如图 2.6-3 所示。

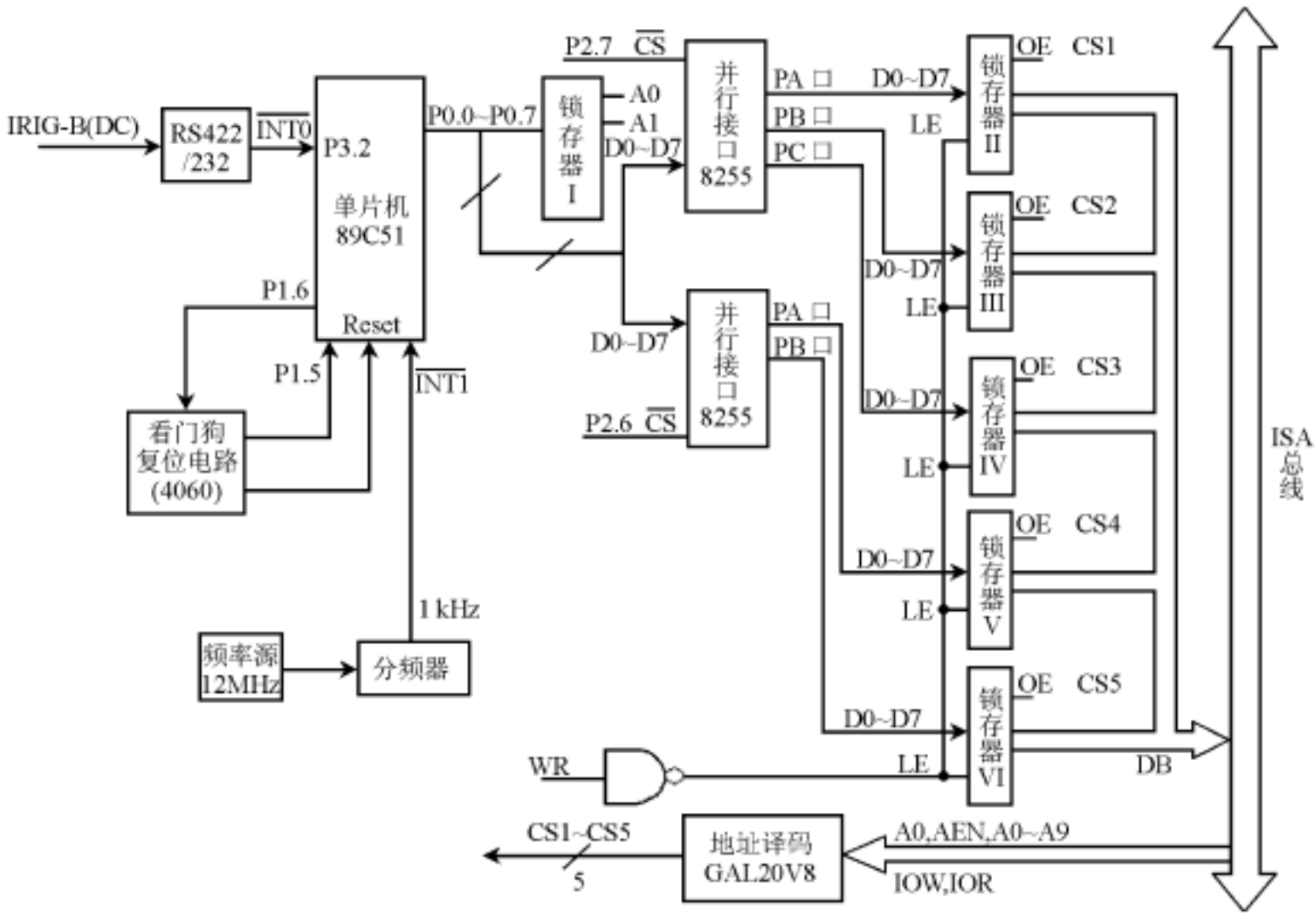


图 2.6-3　B 码解码接口卡电路

来自时统设备的 IRIG-B(DC)码,为 RS422 接口信号,经转换后变为 TTL 电平,送至单片机 89C51 的$\overline{\text{INT0}}$端。单片机初始化设置中,定时器 T0 工作在模式 1,采用内部时钟。当

$\overline{\text{INT0}}$引脚由低到高时,定时器 T0 开始计时;当$\overline{\text{INT0}}$引脚由高变低时,触发$\overline{\text{INT0}}$中断,执行中断服务程序,计算$\overline{\text{INT0}}$引脚的高电平宽度。根据宽度对 B 码各脉冲进行解码,形成秒、分、时的 BCD 码,存入单片机的内部 RAM 中。同时,由频率源产生的 12 MHz 的信号经分频器后,输出 1 kHz 信号,送至单片机$\overline{\text{INT1}}$引脚,使 1 ms 产生一次$\overline{\text{INT1}}$中断,执行$\overline{\text{INT1}}$ 中断处理程序,对毫秒进行计数。毫秒计数到 1 000 时,进行秒加 1,毫秒初值在 B 码的准时点进行赋值。

单片机的 P0 口经锁存器Ⅰ输出地址线 A0、A1,以控制两个并行接口芯片 8255 的输出端。单片机解码和计数输出的毫秒(2 字节)、秒(1 字节)、分(1 字节)、时(1 字节)BCD 码,在单片机的写指令控制下,分五次送到两片 8255 的不同端口。各端口经锁存器Ⅱ~Ⅵ,将数据锁存。单片机送出一组时间码后,向主计算机发送中断。主计算机响应中断后,依次读取各锁存器的值,即为当前的时统时间。

锁存器Ⅱ~Ⅵ的数据输出端直接挂在 EISA(ISA)总线上,其片选信号 CS1~CS5 通过地址译码产生。

主计算机读时应能保证数据不处于变化中,因此,须将单片机的写信号 WR 经延时反向后,送到各锁存器的 LE 端,使数据被锁存。主计算机内设一存储单元,存放前一秒的数据。当前读取的数据如果比前一秒多 1,则认为时间正确。

为避免板内程序死循环,在该板上设计了看门狗复位电路。在单片机板内程序中,每隔一定时间对 P1.6 口进行置 1。计时器的最高位输出端接至单片机的 Reset 端。在正常情况下,P1.6 口总能执行置 1 操作,不会对单片机复位;但若程序中有死循环,则 P1.6 口不被置 1。当计时器计到最高位输出端为 1 时,就会对单片机复位。看门狗复位电路采用 14 位二进制计数器 4060,并具备上电复位和手动复位的功能,其振荡周期由外接电阻、电容的大小决定。

2. 电路设计方案 2

方案 2 如图 2.6-4 所示。解码原理同方案 1。该电路采用 FIFO(IDT7201)加状态寄存器和缓冲器,实现单片机与主计算机之间的数据传输,无需 8255 和锁存器,器件较少,但相应地会增加软件控制的工作量。单片机解码后,在每帧数据前加上标志码,输出时、分、秒、毫秒信息,在写信号的控制下,经缓冲器送入 FIFO 中。主计算机查询状态寄存器,了解 FIFO 的状态(空、满、半满)后,读取 FIFO 中的数据。这里用 FIFO 的 8 位数据线。

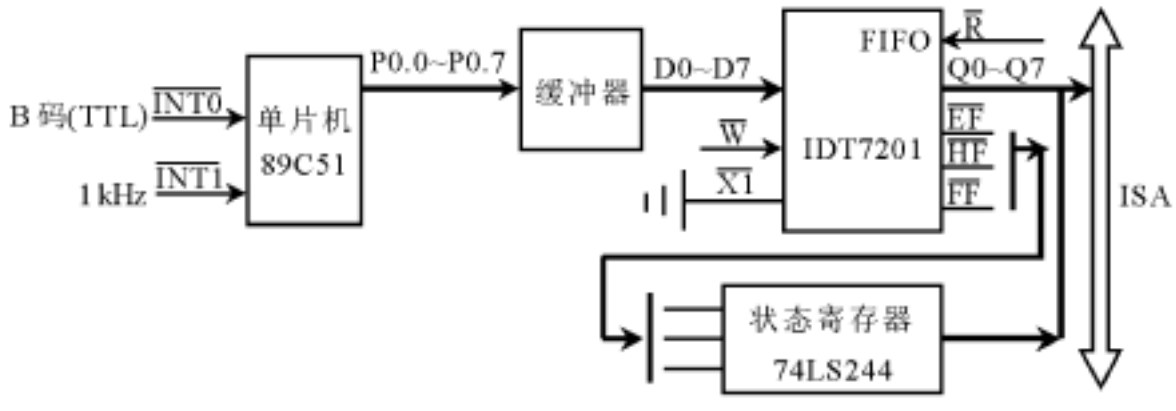


图 2.6-4 FIFO 实现输入输出控制原理图

IDT7201 为先进先出双口存储器。内部 RAM:512×9,设有空标志($\overline{\text{EF}}$)、满标志($\overline{\text{FF}}$)和半满标志($\overline{\text{HF}}$),以避免数据溢出和空读。读、写数据通过内部循环指针,无需地址信息存取数据。

IDT7201 复位时,空标志($\overline{EF}$)置 0;满标志($\overline{FF}$)和半满标志($\overline{HF}$)置 1;读、写指针设到初始位置。当写信号(瑏)由高变低时,若满标志($\overline{FF}$)为 1,则开始写循环,将数据写入 RAM 中,不受任何读操作的影响;当 RAM 半满时,$\overline{HF}$置 0;当写指针比读指针小 1 时,表明缓冲区已满,$\overline{FF}$置 0,禁止写操作。当读信号(瑐)由低到高时,如果$\overline{EF}$为 1,则开始读循环,数据以 FIFO 的方式读出;当所有数据均读出,读指针等于写指针,缓冲区已空,$\overline{EF}$置 0,禁止读操作。在缓冲区空或满时,瑐、瑏信号的外部变化,不影响 FIFO。

四、板内程序设计流程

在单片机内部 RAM 中,用可位寻址的 21H、22H、23H、24H 单元分别存放“秒”、“分”、“时”BCD 码和帧标志等,如表 2.6-1 所列。

表 2.6-1 时间码存放表

		D0	D1	D2	D3	D4	D5	D6	D7
单片机内部 RAM	21H	08H	09H	0AH	0BH	0CH	0DH	0EH	
		秒的个位				秒的十位			
	22H	10H	11H	12H	13H	14H	15H	16H	
		分的个位				分的十位			
	23H	18H	19H	1AH	1BH	1CH	1DH		
		时的个位				时的十位			
	24H	20H	21H	22H	23H	24H	25H	26H	27H
				帧标志					

$\overline{INT0}$、$\overline{INT1}$中断处理程序如图 2.6-5 和图 2.6-6 所示。

参 考 文 献

1 何立民.MCS—51 系列单片机应用系统设计.北京:北京航空航天大学出版社,1990

2 李华.MCS—51 单片机实用接口技术.北京:北京航空航天大学出版社,1993

选自《单片机与嵌入式系统应用》月刊,2001 年第 9 期

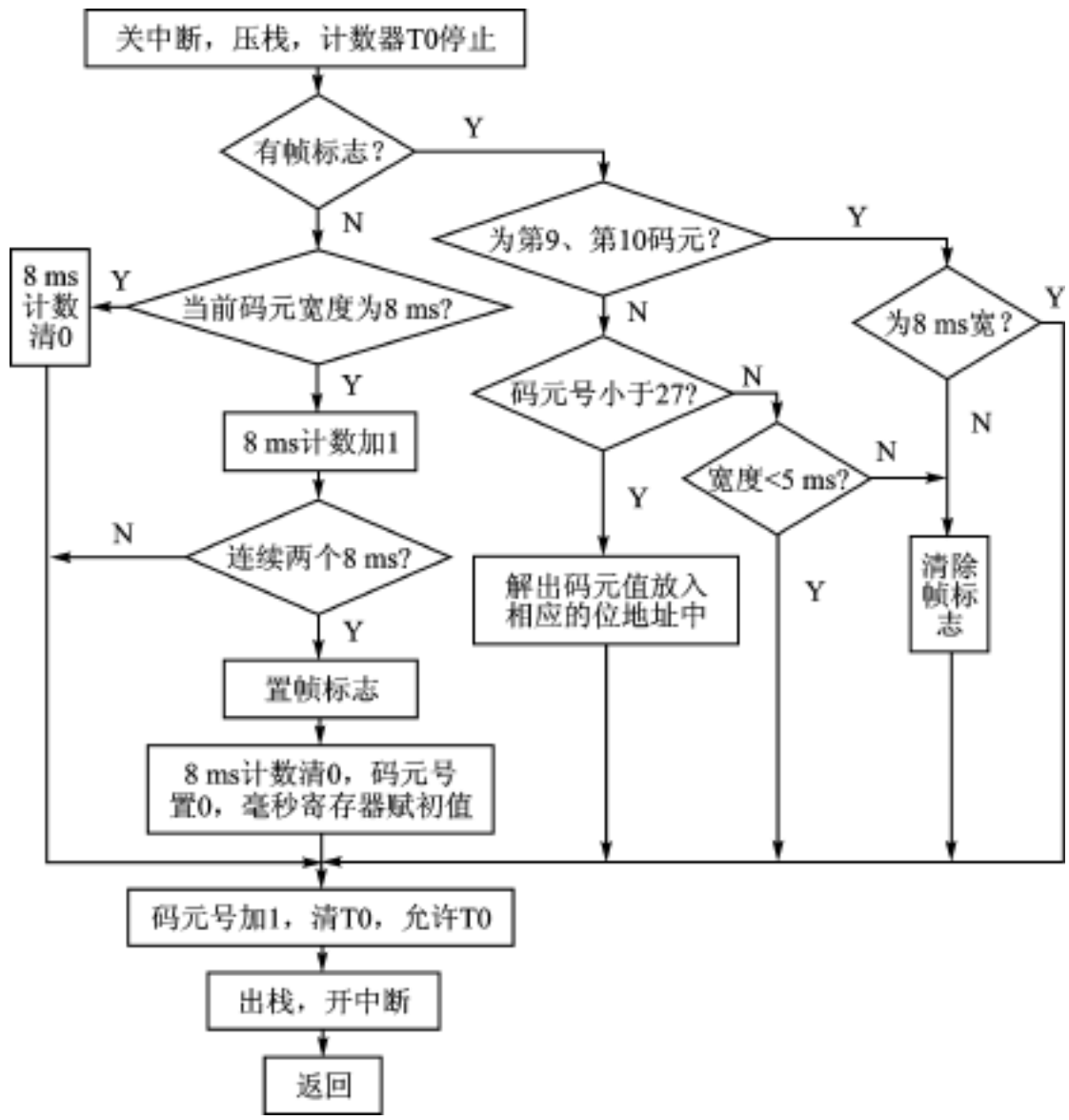


图 2.6-5 $\overline{\text{INT0}}$中断处理程序流程图

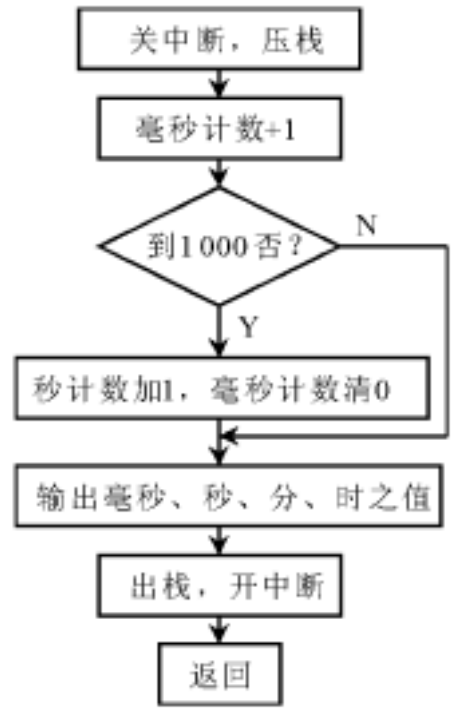


图 2.6-6 $\overline{\text{INT1}}$中断处理程序流程图

2.7 一种基于单片机时频信号处理的实用方法

周 渭 周 晖 李志勇

在稳定频率源的基础上精密时间信号的产生、稳定的相位步跳和改正、频率的改正及合成、以及精密的时间和频率测量具有很广泛的应用。借助于计算机来完成上述工作已经是普遍应用的途径。但是把计算机作为信号处理的工具时,其机器周期的大小成了影响时间分辨力的关键。虽然一些辅助技术可以应用,但是应用新原理来寻求更简单的途径是改善相应设备的有效方法。

一、计算机时钟信号周期与相位变化的影响

由计算机所产生的周期性信号的周期值,通过软件设计可以为

$$T = nt \tag{1}$$

式中,t 是计算机的机器周期;n 是受计算机软件控制的整数值。计算机的机器周期一般是其时钟周期的若干倍,如常用的单片机是 12 倍的关系。这样,获得的时间分辨力是很有限的。以 t_1 表示计算机的时钟周期,那么,一般情况下,也就是 t_1 平稳变化的情况下

$$T = n(mt_1) \tag{2}$$

其中,m 是机器周期与时钟周期之间的固定整数比(如 12 等),要提高分辨力也只能从 t_1 本身和 m 下功夫。显然,用扣时钟脉冲的方法,我们可以改变 m,其效果等效于小数分频。而用对时钟信号移相的方法,我们又可以改变 t_1。

如果仅仅借助于计算机的机器周期,它能够产生的频率信号为 f_0。当对其时钟信号采取了扣脉冲(周期处理)和移相(如用量化时延的方法),这样,所引起的输出信号周期的变化而会产生频差 Δf。也就是频率值成为 $f_X = f_0 + \Delta f_0$

$$\frac{\Delta f}{f_0} = \frac{\Delta T}{\tau} \tag{3}$$

即获得的频率值为

$$f_X = f_0 + \Delta f = f_0\left(1 + \frac{\Delta T}{\tau}\right) \tag{4}$$

该式的意义是通过对频率信号在一定时间 τ 内的时间(相位)移动 ΔT,就产生了新的频率信号,当用计算机产生频率信号时,f_0 可以理解为用计算机软件在机器周期的基础上所产生的频率值,而 f_X 则是通过对计算机时钟信号移相或扣脉冲后获得的值。在这种情况下,τ 完全可以是所输出信号的周期值,而 ΔT 是在该时间内的信号相位变化。这种变化既可以是连续的,又可以是步跳的。为了保证输出信号的稳定性,这种变化应该连续发生在每个周期的内部。这对于方波输出的信号,是不会表现出相位扰动的。ΔT 是以时间为单位的,既可以是时钟的周期,又可以是其相位等。

f_X 信号周期的构成为

$$T = nt + n_1 t_1 + n_1 t_1 + n_2 t_2 \quad (5)$$

式中，t_2 是延迟单元的延迟时间，而 n, n_1, n_2 均是受计算机软件控制的正整数。可见，用本方法来处理时间或频率信号时，都是以时间(或相位)的处理为基础的，这里 $t > n_1 t_1, t_i > n_2 t_2$。

量化时延的方法可以被用来对计算机时钟信号移相，并且可以利用门电路等构成延时单元[1]。如果能对一个高频信号进行和脉冲处理后再分频作为单片机的时钟信号，那么就能够获得更高的分辨力。

从式(4)和式(5)可以发现，如果延时和扣脉冲处理是周期性进行的，也就是在微机输出信号的每个周期的时间间隔内不断地对时钟信号进行周期和相位处理，这样它所输出的信号的周期值就是一个稳定的值，也就起到频率(或周期)改正的作用。如果上述处理是单步进行的，则起到了对输出信号相位移动的效果。

总之，通过对计算机的时钟信号扣脉冲或延时等时间处理，可以把小范围内时钟信号的相位变化通过软件控制，扩展为在一个很宽的范围内频率或时间信号的频率与相位的变化。这方面另一个可以采用的方法是用一个简单的窄频率范围的频率合成器来代替计算机的时钟，在软件的支持下，它的频率的调整及变化就会产生范围的计算机输出信号的频率变化；或者是计算机对其他信号处理的节奏变化。

由计算机直接产生的周期性信号，由于受其指令或软件执行时间的影响，其频率值不可能很高。而且用简单的方法处理，计算机也只产生方波信号。但是这种方法产生的信号在周期和相位方面的分辨力和可调节的步进值被大大提高。这在许多控制信号的产生和应用方面是很有价值的，也得到了证明[2]。

另外，也可以通过计算机处理及一定硬件的配合，通过相位延迟处理的方法来完成对一定的频率信号的额外频率改正及合成进行控制。此时，新的频率信号并不是由计算机产生的，计算机所给出的是控制信号，通过对一个给定信号施加周期或非周期性的相位延迟而产生精密的频率改正信号或相位移。这样做常常可以获得更高的输出频率值。

该方法也能够和锁相的压控振荡器及可变分频器结合，在宽频率范围产生更高的频率信号。如图 2 .7 - 1 所示。

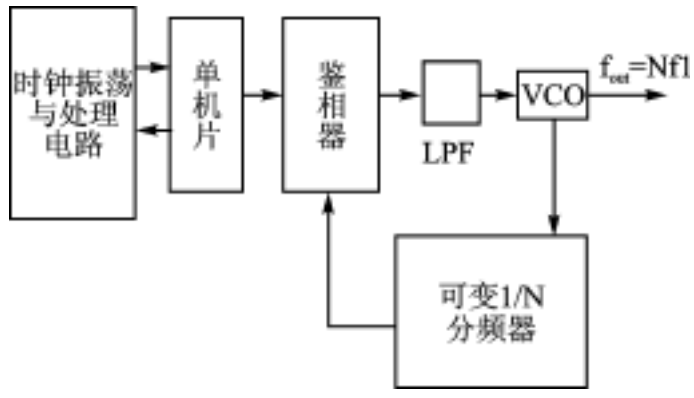


图 2 .7 - 1 高频信号的产生方框图

二、具体实施方法及其存在的问题和解决方法

为了提高由计算机所产生的频率信号的分辨力，可以对一个稳定的高频信号采取扣脉冲措施后再把它分频成为合适的频率来作为计算机的时钟信号。用这样的方法，式(5)中的 t_1 就是这个高频信号的周期。

图 2.7－2 是实现扣时钟脉冲的简单线路图,图 2.7－3 是其波形图。

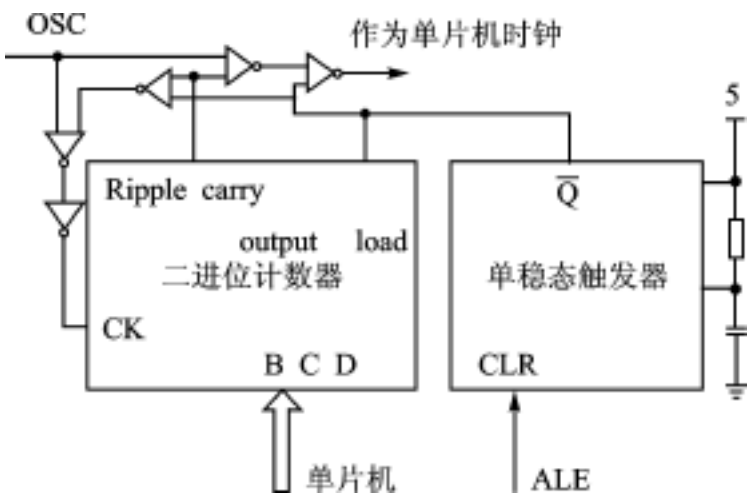


图 2.7－2　实现扣时钟脉冲的简单线路图

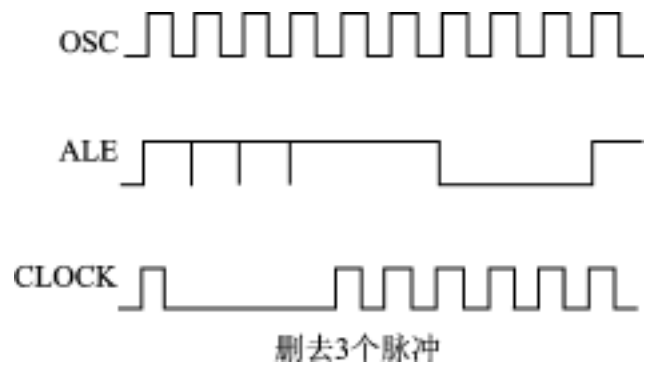


图 2.7－3　图 2.7－2 所对应的波形图

对时钟信号扣脉冲的周期处理方法提高分辨力的作用是有限的,它不会高于一个对应的时钟周期。为此,采用对时钟信号相位处理的方法就能够得到更高的所产生信号的周期或相位的分辨力。对于一个周期性的信号,相位的处理和周期处理在本质上是一样的,只是有更高的分辨力。对信号周期性的移相也可以起到频率或周期改变的作用。对时钟信号的周期和相位处理都是以时间处理为基础的。

图 2.7－4 示出了通过对计算机时钟信号周期性移相的方法获得修改它所产生频率信号频率值的效果。

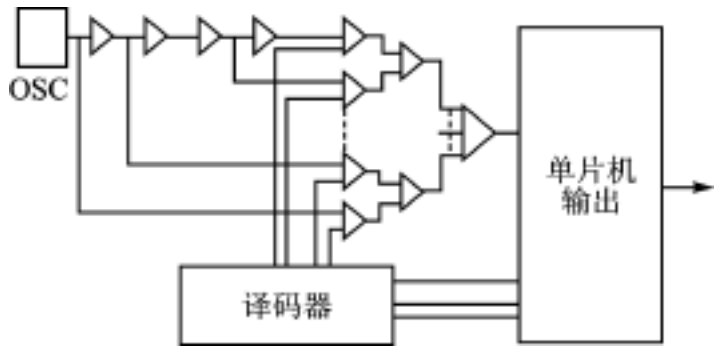


图 2.7－4　用延时移相的方法高分辨力修改计算机产生信号的频率值

这里的门电路链是起延迟作用的线路,它们起到了对时钟信号移相的效果。其中的选择电路等要求对各路延迟信号能够完全对称,并且有较高的延时稳定性。一般来说,延时级数较多且延迟单元的延迟时间短时,所控制的准确度会更高。但此时对器件的要求更高,要注意的地方也更多。延迟链可以用串联的方法也可以用串并联的方法,以提高使用的灵活性。

对于时钟信号进行延迟、移相处理,会获得较之扣脉冲更高的测量分辨力,但是也会附带引入一些误差和问题,也就是付出了降低频率稳定性的代价。一个是在延迟链中延迟单元本身有延迟误差(可以用一定的软硬件自校措施来改善),这会影响输出信号的频率稳定性。另一个是在计算机输出频率的产生和改正过程中,延迟总是按输出信号的周期而周期性的进行的,这样,延迟单元总是在延迟数值严格等于一个时钟信号的周期值时,完成一个延迟转换而重新从延迟链的始端开始向后进行延迟(与此同时,也附加有一个时钟周期扣除的处理)。这时,就可能出现时钟信号周期与延迟链的总延迟数值不可能完全重合的‘不重合误差”。这会对输出信号进行周期性相位调制,也影响其频率稳定性。另外,对延迟级数控制时,所用的线

路本身时延的不稳定性与不对称也有可能引起附加误差。所以这种方法应该针对性的在实际中应用,一般所取的延迟级数不能太多。延迟级数多时还要占用计算机较多的端口。当用门电路作为延迟单元时,延迟门的数量一般不宜超过 20 个。

我们也正在探索用有一定差频的双频互换的时钟信号来解决提高输出信号分辨力的问题,也就是通过它们不同的周期数组合最终实现满足输出的细分辨力。这里要求加入附加的计数电路、相位重合检测电路等。

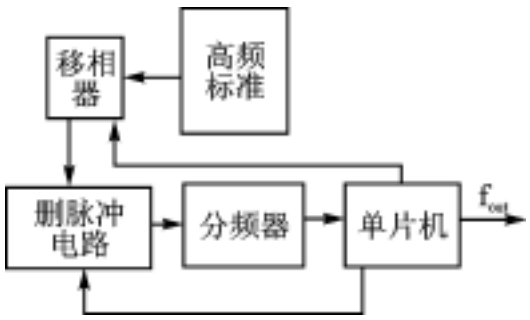


图 2 .7 - 5 一种时钟信号处理线路图

通过对高频频标移相、扣脉冲再分频产生计算机时钟的方法来获得较高分辨力的频率信号,其原理如图 2 .7 - 5 所示。

这种方法可以避免使用太多的延迟器件而得到较高的时间信号处理的分辨力。标准信号的频率可以选择 50～100 MHz 以上。用这样的方法已经可以由计算机产生 1 ns 的时间信号的分辨力。当把这种方法与带有频差倍增功能的压控晶体振荡器结合在一起时,按照不同的倍增级数,还可以获得数十皮秒的时间分辨力。

三、实验以及结果

用图 2 .7 - 2 的方法作了扣脉冲情况下的计算机输出信号频率稳定性实验。这里,用高稳定性的 5 MHz 晶体振荡器作为一个单片计算机的时钟信号,并用高分辨的频率计来测量由计算机所产生的频率信号,频率计的标频信号也使用了同一个晶体振荡器。通过软件的控制,计算机产生了几个相邻的频率信号。实验中,信号从 ALE 端输出。对该信号的测量结果列于表 2 .7 - 1。

表 2 .7 - 1 对加入扣时钟脉冲功能的输出信号的实验结果

被删除的时钟脉冲数目	1	2	3
输出信号频率/ kHz	624 .9999989	555 .55555549	500 .00000001
	625 .0000001	555 .55555556	499 .99999993
	624 .9999996	555 .55555558	499 .99999993
	624 .9999994	555 .55555563	499 .99999993
	624 .9999995	555 .55555545	499 .99999998
	625 .0000002	555 .55555553	500 .00000006
	624 .9999993	555 .55555542	500 .00000001
	625 .0000004	555 .55555544	499 .99999991
	625 .0000003	555 .55555554	500 .00000001
	624 .9999995	555 .55555543	500 .00000004
	625 .0000000	555 .55555544	499 .99999993
	625 .0000004	555 .55555543	499 .99999998
	624 .9999996	555 .55555554	500 .00000004
	624 .9999990	555 .55555557	500 .00000001
	625 .0000002	555 .55555543	499 .99999991
最大误差	$\pm 1.2\times10^{-10}$	$\pm 1.9\times10^{-10}$	$\pm 1.5\times10^{-10}$

采用相同的方法,但是不扣计算机的时钟脉冲,再测量输出频率信号。对紧邻频率信号的测量结果列于表 2.7-2。

表 2.7-2 没有扣脉冲功能的输出信号的实验结果

	基于计算机软件的连续输出频率	
输出信号频率/ kHz	104.16666668	83.333333326
	104.16666662	83.333333353
	104.16666669	83.333333337
	104.16666664	83.333333346
	104.16666666	83.333333344
	104.16666666	83.333333351
	104.16666662	83.333333359
	104.16666662	83.333333354
	104.16666663	83.333333353
	104.16666665	83.333333363
最大误差	$\pm 3.3\times10^{-10}$	$\pm 2.2\times10^{-10}$

这里,所用的高分辨力频率计的分辨力是 1×10^{-10}/ s。当该仪器用于 100 kHz 以下频率的测量时,测量准确度也能保证 3×10^{-10} s 以上。测量时门时选为 1 s。因为所用频标、测量仪器的准确度等方面的限制,这样的测量结果并不能反映计算机的软件和硬件等对信号稳定度影响的极限值。但是从测量获得的数据来看,它们的影响至少优于 0.1 ppb。从实验结果来看,通过对时钟信号扣脉冲处理,计算机所输出的相邻频率信号的间隔可以更小,也就是在频率分辨力上获得了提高。

在时钟信号处理线路中采用了延迟单元为 10 ns 的延迟链,再测量计算机的输出信号。其结果列于表 2.7-3。

表 2.7-3 带时钟相移功能的输出信号的实验结果

	基于删时钟脉冲和时钟相位处理的连续输出频率		
输出信号频率/ kHz	502.51256260	505.05050535	507.61421311
	502.51256320	505.05050499	507.61421297
	502.51256281	505.05050561	507.61421289
	502.51256275	505.05050478	507.61421267
	502.51256295	505.05050496	507.61421355
	502.51256289	505.05050505	507.61421311
	502.51256239	505.05050521	507.61421347
	502.51256242	505.05050555	507.61421319
	502.51256244	505.05050511	507.61421363
	502.51256322	505.05050500	507.61421301
	502.51256298	505.05050440	507.61421289
	502.51256277	505.05050498	507.61421337
最大误差	$\pm 8.2\times10^{-10}$	$\pm 1.2\times10^{-9}$	$\pm 9.4\times10^{-10}$

从表 2 .7 - 3 的延时相移实验也表明了,所获得的分辨力(能够产生的频率信号的间隔值)被大大提高,但是由于延时单元的准确度和延时链延时值之和与时钟周期之间的误差,所产生信号的稳定性会受到一定程度的影响。

四、结 论

利用单片计算机来产生或处理时间与频率信号是一种简单和方便的方法。而采用了对计算机时钟信号扣脉冲和移相处理的方法,能从其输出处获得更细的频率间隔和更高的时间及频率分辨力。很多关于相位或时间处理的方法、频率微调技术和周期处理技术可以被用于计算机的时钟线路,并获得不同程度的分辨力。这种方法尤其适合于用单片计算机兼有多种功能的情况下。我们已经将这种方法用于微机补偿晶体振荡器中产生频率修正信号,而省略了原来必须要用的 DDS 集成电路芯片,使装置造价大大降低。同时,这种方法也用于和测量相结合的设备中,构成微机倍频器、时频信号处理器等,有广阔的应用前景。

参 考 文 献

1 ei Zhou, et al . Some new methods for precision time interval measurement . The proceedings of the 1997 IEEE International frequency control symposium, pp .418～421, 1997

2 Wei Zhou, et al . An improvement method of MCXO . The 1999 IEEE international frequency control symposium .

3 Wei Zhou, et al . A practical method to process time and frequency signal . The 1999 IEEE international frequency control symposium .

选自《宇航计测技术》双月刊,2000 年第 1 期

2.8　射频接收系统晶体振荡电路的设计与分析

东南大学 ASIC 工程技术研究中心　孙华芳　吴建辉

随着通信技术的快速发展,振荡器的研究、设计和技术得到了很大的发展。为了适应无线寻呼接收机、FM-SCA 股票机、PDA 等通信产品的小型化,在射频接收电路中一本振采用了晶体振荡电路。

一、射频接收中晶体振荡电路的设计及工程估算

1.振荡电路的确定

对振荡电路的选择取决于对工作频率、频率稳定度的要求,同时还要考虑射频接收小型化、低功耗及其他要求。晶体振荡电路应设计成结构简单、功耗小、调试方便并且频率可以微调的电路。经过分析,确定采用如图 2.8-1 所示的结构。该电路为电容三点式振荡,是串联式晶体振荡电路。

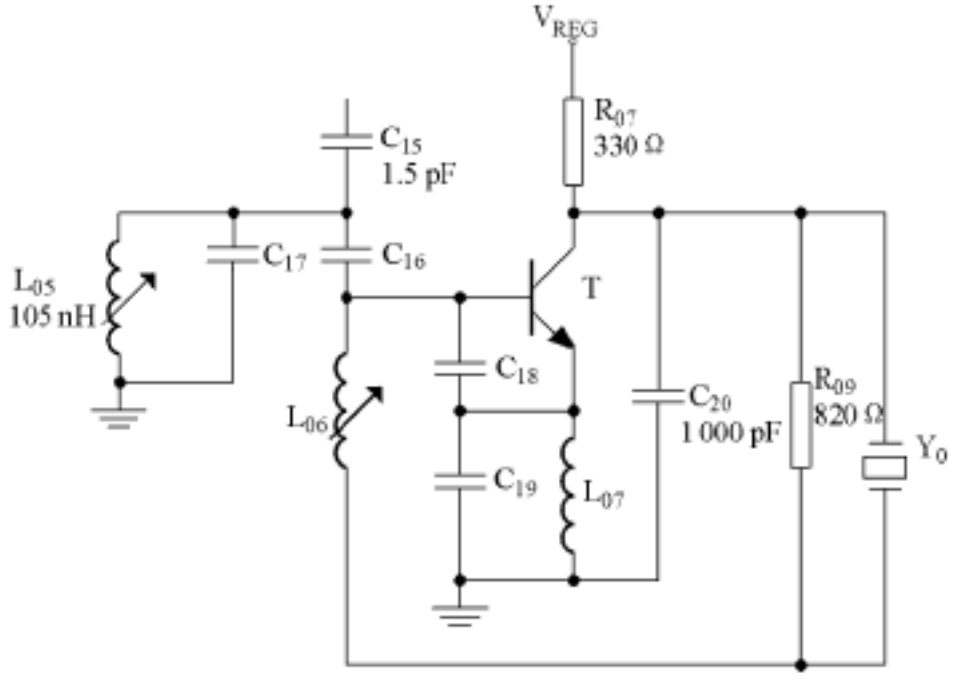


图 2.8-1　一本振石英晶体振荡电路原理图

(1) 振荡电路原理分析

图 2.8-2 为图 2.8-1 的交流等效电路,其中 C_{16}、C_{17}、L_{05} 组成的支路相对于三次泛音晶体的基频开路,在晶体的标称频率振荡时可以不考虑。由于振荡器的输出负载和振荡器之间是弱耦合,故也可以忽略不计。晶振工作在串联谐振频率上,且晶振发生串联谐振时,该振荡器电路的正反馈最强,只有这时才能满足振幅条件而使电路启振。一旦晶振工作点偏离串联谐振点,由于晶体的动态电感很大,而 R_{09} 较小,则等效并联在电感和晶振两端的电阻较小,大大影响了谐振网络的 Q 值,使电路无法工作。

在此电路中,电路进入集电结的饱和区而发生饱和限幅失真,集电极电流因此包含丰富的

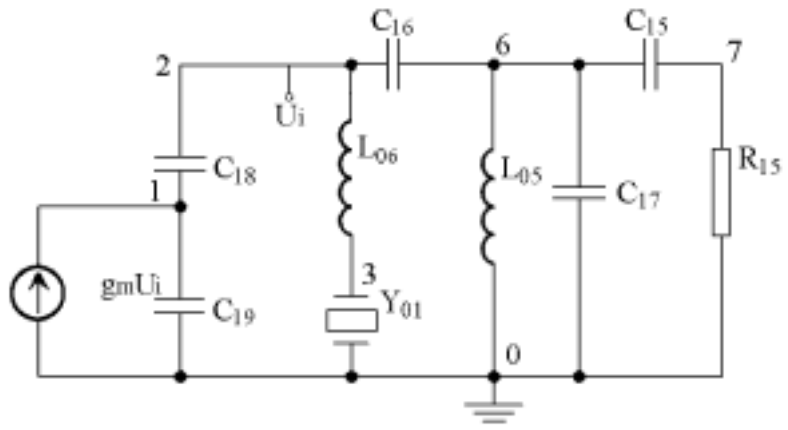


图 2.8-2 交流等效电路

谐波分量,输出负载网络调谐在振荡回路的二次谐波上(L_{06} 和晶振支路相对于二次谐波开路),而有效抑制基波及其余各次谐波。

此种类型的振荡器可以用如图 2.8-3 所示的模型描述,即一个线性时不变网络(LTIN)将振荡器电路分成三部分:一个非线性有源器件、一个基波谐振网络和一个二次谐波网络。

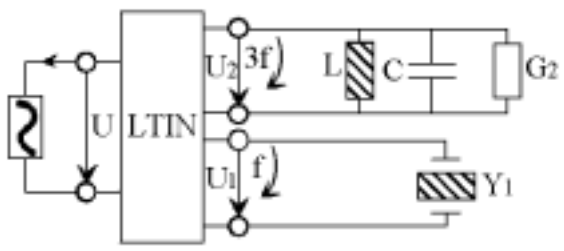


图 2.8-3 一本振石英晶体振荡器等效电路模型

其中,有源器件的电流-电压转移关系可描述为:

$$I = g_{m1} U + g_{m3} U^3$$

由于除了基波和二次谐波外,其他各次谐波没有可以流通的谐振回路,故为简化起见,分析时仅列出这两项(如上式所示)。在本电路中,二倍频选频回路与基波振荡回路之间属弱耦合,且二次谐波分量对振荡回路电流 I 的影响甚小,故 $U \approx U_1$。因此,在分析此电路时,可先忽略二倍频选频回路,仅分析基波谐振回路与有源器件形成的网络回路,求出各点的基波电压幅度。由于该电路也会出现集电极饱和,因此集电极-基极电压被限幅,然后由图中基极-发射极电压与集电极各次谐波电流的关系得到集电极二次谐波电流,由此电流与二倍频选频网络组成的回路得出其二次谐波输出电压幅度。

(2) 频率稳定度分析

图 2.8-4 为由交流等效电路图简化后得到的振荡回路中基波回路的交流等效电路图。振荡器的环路增益可近似表示为 $A_L = g_m Z_L (C_2/C_1)$,其中 Z_L 是 L、C_1、Y_1 和 C_2 的并联阻抗,而 Y_1 为晶体的等效阻抗,则频率稳定度为:

$$S_f = \frac{dA_L}{d\omega} = \frac{dZ_L}{d\omega} \tag{1}$$

环路增益的极点有两对共轭解,频率为:$\omega_{01} = 1/\sqrt{LC}$,$\omega_{02} = 1/\sqrt{L_0 C_0}$(式中 C 为 C_1 与 C_2 的并联电容),其中频率为 ω_{01} 的极点位于右半平面,ω_{02} 的极点位于左半平面,故频率稳定度为

$$S_f = \sum 2Q_i = 2Q_2 - 2Q_1 = 2\omega_0\left(\frac{L_0}{R_0} - \frac{L}{R}\right)$$

($\omega_0 = \omega_{01} = \omega_{02}$ 时) (2)

其中,Q_2 为晶体的品质因数,Q_1 为回路的品质因数,$Q_2 \gg Q_1$,所以 $S_f \approx 2Q_2$。由式(2)可知,采用上述振荡电路的频率稳定度极高,即振荡电路的振荡频率很稳定,有利于保证振荡频率的一致性。

调整 L、C_1、C_2 或由于温度等的变化使 L、C_1、C_2 值发生变化时，振荡频率的相对变化率为

$$\frac{\Delta\omega}{\omega_0} = \frac{d}{2Q_2} = \frac{Q_1}{2Q_2}\left(\frac{dC}{C} + \frac{dL}{L}\right) \qquad (3)$$

式(3)表明：晶体振荡器的频率在由晶振和外部调整网络共同构成的回路中，外部 LC 网络的 Q 值应该选择得比较低，才有益于晶体振荡器的频率稳定。

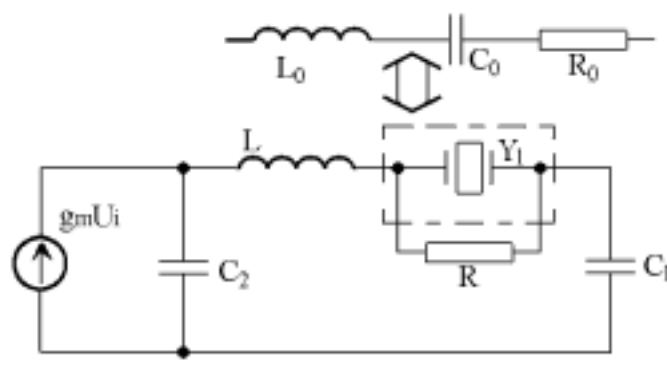


图 2.8－4　振荡器的交流简化图

2．晶体管的选择与静态工作点的确定

对小功率振荡器，晶体管的选择原则是能在工作频率范围内稳定地振荡。此外，管子除了应有足够大的跨导外，对于高频振荡器，其特征频率 f_T 应大于振荡频率 f_0。静态工作点的设定除了考虑启振和波形失真两方面要求外，还应考虑射频接收的低功耗要求。Motorola 的晶体管 MMBR911 等即符合要求。图 2.8－5 为振荡电路的静态偏置电路。R_{07}、R_{09} 分别为集电极和基极偏置电阻，静态工作点为 U_b = 0.7 V、U_c = 0.62 V，集电极电流为 1.30 mA。

图 2.8－5　静态偏置电路

3．LC 回路参数与二倍频电路参数的确定

(1) 工程估算

以实际开发中的 SCA 股票机为例，假设接收频率为 101.1 MHz，第一中频为 10.7 MHz，则本振信号频率为 90.4 MHz。本振信号取自晶体振荡器的二次倍频信号，则该晶体的标称频率为 45.2 MHz。根据石英的特点，这种频段的晶体一般采用石英的三次泛音制成。

作为工程计算，当回路谐振时，图 2.8－1 中石英晶体可以近似地等效为纯电阻。振荡回路与二倍频回路的耦合是弱耦合，因此，在计算回路谐振频率时，可以忽略二倍频回路对振荡回路的影响。设 C_{18} = 39 pF，C_{19} = 10 pF，则回路串联电容值主要决定于 C_{19}，计算时可以忽略晶体管参数 C_{be}。

下面计算回路谐振频率为 45.2 MHz 时可调电感的中心值：

$$L_{06} = \frac{1}{(2\pi f)^2 C_0} \approx 1.55\ \mu H$$

式中

$$C_0 = \frac{C_{19}C_{18}}{C_{19} + C_{18}} \approx 8\ pF$$

在实际使用中，由于寄生参数等的影响，L_{06} 的值还须实验校正或适当调整 C_{19} 的值。同时，在各频段的电路调试中，若电容 C_{18} 和 C_{19} 的值不变，则在不同频段 L_{06} 的值应有所不同。电感值应由工程估算后，经实验确定。

在振荡电路中，发射极电感 L_{07} 作轭流圈使用。它与 C_{19} 组成的回路应谐振在三次泛音晶体的基频 15 MHz 左右，以抑制振荡回路的基频分量，故可取 L_{07} = 10 μH，此时回路的谐振频率为

$$f_0 = \frac{1}{2\pi\sqrt{L_{07}C_{19}}} \approx 16\ MHz$$

采用电容耦合取出振荡信号,通过二倍频网络的选频作用,在混频管基极得到所需的本振频率信号。二倍频电路由并联 LC 回路组成。设可调电感 L_{05} 的中心值为 105 nH,当回路谐振在 f = 90 .4 MHz 时,电容 C_{17} 近似计算如下:

$$C_{17} = \frac{1}{(2\pi f)^2 L_{05}} \approx 29 \text{ pF}$$

以上为振荡回路与选频回路的主要元器件的工程估算值,精确值还须实验调试确定。

(2) SPICE 模拟

采用 SPICE 分析时,由于回路中有高 Q 值的石英晶体,在串联谐振点附近,电路对微分方程的不精确性格外敏感,最坏情况是正弦波在谐振点附近而又不在谐振点。这种情况下,SPICE 误差控制公式不够严密,因此结果会不准确。

在进行 SPICE 模拟时,起初以工程估算值进行模拟,结果不太理想。后来,经过不断调整谐振回路参数,发现在 L_{06} = 1 .4 μH、C_{17} = 24 pF、C_{18} = 39 pF、C_{19} = 10 pF、L_{07} = 10 μH 时效果最好,其瞬态分析结果如图 2 .8 - 6 所示(在模拟中晶振采用了近似短路模型)。

由图 2 .8 - 6 可知:该振荡电路在约大于 100 ns 的时间内就能稳定地振荡,完全满足射频接收中的预升起工作时间的要求;另外,二次谐波的频率较准,幅度远大于 50 mV,也能满足混频器对它的要求。

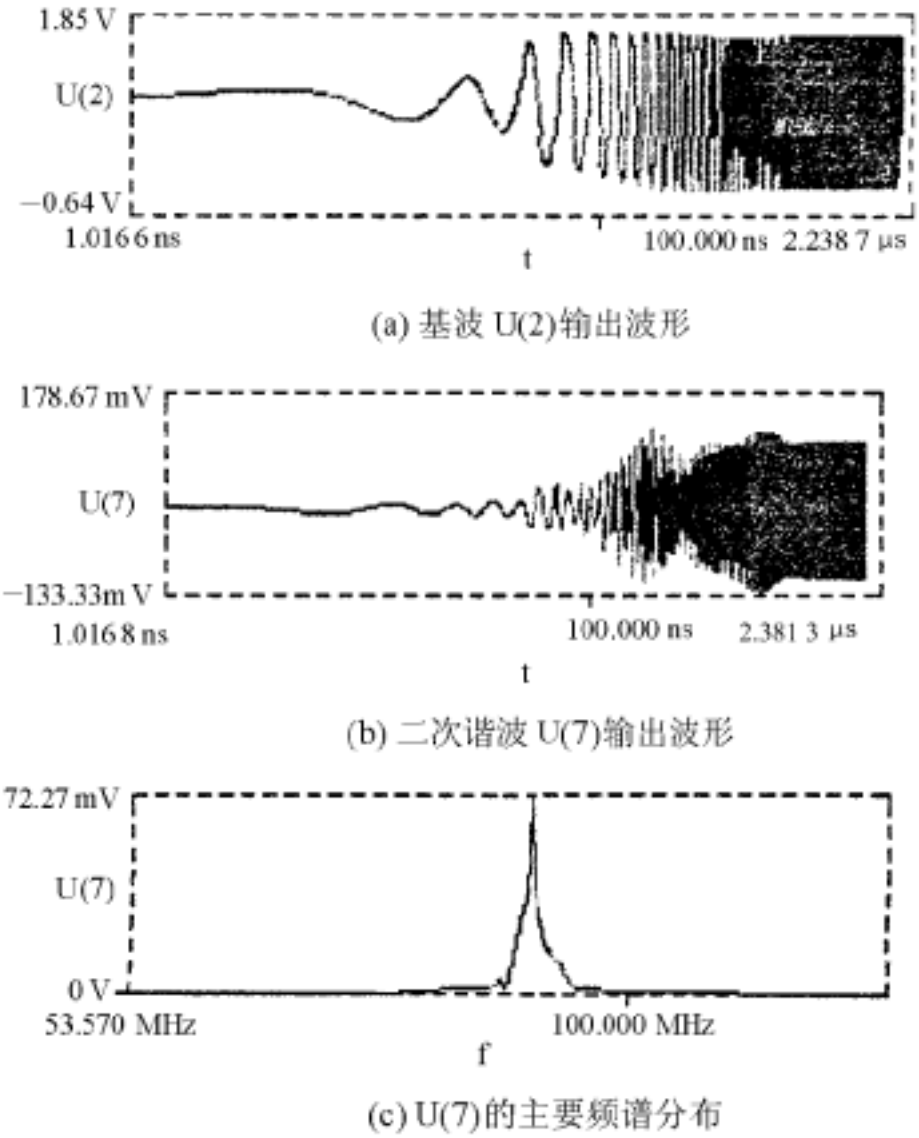


图 2 .8 - 6　一本振晶体振荡器瞬态分析结果

二、实际电路中的振荡元器件参数

由以上分析所得的振荡回路及二倍频选频回路元器件参数运用到实际电路中进行测试,发现由于寄生效应的影响,这些元器件值并非最好:在确定了电感值的基础上,通过改变电容值使该电路的二倍频输出的频率为所需频率,并且其频谱较纯、幅度最大(在无高频信号输入的情况下用多功能综合测试仪测试混频输入级)。实验结果表明:当 C_{17} = 22 pF、C_{16} = 2.2 pF、L_{06} = 1.2 μH 时,一本振的振荡频率为晶体的标称频率,且二倍频选频输出为 100 mV 左右的二次谐波频率的正弦波。所以,在实际运用中射频接收中的一本振的电路我们选用了如图 2.8-1 所示的电路。

三、提高振荡电路性能指标的方法

本地振荡电路的性能直接影响射频接收电路的总体性能,因此,从电路总体性能的要求出发,对本振电路提出的相应设计要求主要包括振荡信号的强度、杂波抑制、频率准确度、稳定度等。提高振荡电路性能指标的方法如下:

· 选用寄生响应比较高的泛音晶体。

· 提高振荡回路及选频回路的选择性。

· 在使用晶体时,应规定较大的寄生响应电阻比。一般三次泛音晶体应大于 4∶1。

· 当晶体串联谐振频率 ω 偏离回路谐振频率超过一定数量时,振荡信号强度变小,且出现畸变,杂波分量加大,这种情况应避免。

· 二倍频选频电路的频率响应也是影响振荡信号的强度及杂波分量大小的因素。因此,应选用高 Q 值电感,以提高回路增益及减小带宽。

最后,经实际测试,该振荡电路起振时间短、振荡幅度稳定且振荡频率可实现在晶振的标称值附近进行微调,其频率稳定度高且二倍频回路选频效果良好,幅度可达 50～100 mV,符合混频器对本振信号的功率要求。

参 考 文 献

1 王国裕,陆时莹.SPICE-Ⅱ通用电路分析程序使用方法及其应用.南京:东南大学出版社,1988

2 高文焕,汪蕙.模拟电路的计算机分析与设计——Pspice 程序应用.北京:清华大学出版社,1999

3 Ferking M E 著.晶体振荡器设计与温度补偿.杜丽冰,詹汉强译.北京:人民邮电出版社,1985

选自《单片机与嵌入式系统应用》月刊,2001 年第 1 期

2.9 揭开 Σ-Δ ADC 的神秘面纱

Maxim 公司 Franco Contadini 徐继红 编译

越来越多的应用,例如过程控制、称重等,都需要高分辨率、高集成度和低价格的 ADC。新型 Σ-Δ 转换技术恰好可以满足这些要求。然而,很多设计者对于这种转换技术并不十分了解,因而更愿意选用传统的逐次比较 ADC。Σ-Δ 转换器中的模拟部分非常简单(类似于一个 1 位 ADC),而数字部分要复杂得多,按照功能可划分为数字滤波和抽取单元。由于更接近于一个数字器件,Σ-Δ ADC 的制造成本非常低廉。

一、Σ-Δ ADC 工作原理

要理解 Σ-Δ ADC 的工作原理,首先应对以下概念有所了解:过采样、噪声成形、数字滤波和抽取。

1.过采样

首先,考虑一个传统 ADC 的频域传输特性。输入一个正弦信号,然后以频率 f_s 采样——按照 Nyquist 定理,采样频率至少两倍于输入信号。从 FFT 分析结果可以看到,一个单音和一系列频率分布于 DC 到 $f_s/2$ 间的随机噪声。这就是所谓的量化噪声,主要是由于有限的 ADC 分辨率而造成的。单音信号的幅度和所有频率噪声的 RMS 幅度之和的比值就是信号噪声比(SNR)。对于一个 N 位 ADC,SNR 可由公式:SNR = 6.02N + 1.76 dB 得到。为了改善 SNR 和更为精确地再现输入信号,对于传统 ADC 来讲,必须增加位数。

如果将采样频率提高一个过采样系数 k,即采样频率为 kf_s,再来讨论同样的问题。FFT 分析显示噪声基线降低了,SNR 值未变,但噪声能量分散到一个更宽的频率范围。Σ-Δ 转换器正是利用了这一原理,具体方法是紧接着 1 位 ADC 之后进行数字滤波。大部分噪声被数字滤波器滤掉,这样,RMS 噪声就降低了,从而一个低分辨率 ADC,Σ-Δ 转换器也可获得宽动态范围。

那么,简单的过采样和滤波是如何改善 SNR 的呢?一个 1 位 ADC 的 SNR 为 7.78 dB(6.02 + 1.76),每 4 倍过采样将使 SNR 增加 6 dB,SNR 每增加 6 dB 等效于分辨率增加 1 位。这样,采用 1 位 ADC 进行 64 倍过采样就能获得 4 位分辨率;而要获得 16 位分辨率就必须进行 4^{15} 倍过采样,这是不切实际的。Σ-Δ 转换器采用噪声成形技术消除了这种局限,每 4 倍过采样系数可增加高于 6 dB 的信噪比。

2.噪声成形

通过图 2.9-1 所示的一阶 Σ-Δ 调制器的工作原理,可以理解噪声成形的工作机制。

Σ-Δ 调制器包含 1 个差分放大器、1 个积分器、1 个比较器以及 1 个由 1 位 DAC(1 个简单的开关,可以将差分放大器的反相输入接到正或负参考电压)构成的反馈环。反馈 DAC 的作用是使积分器的平均输出电压接近于比较器的参考电平。调制器输出中“1”的密度将正比于输入信号,如果输入电压上升,比较器必须产生更多数量的“1”,反之亦然。积分器用来对误

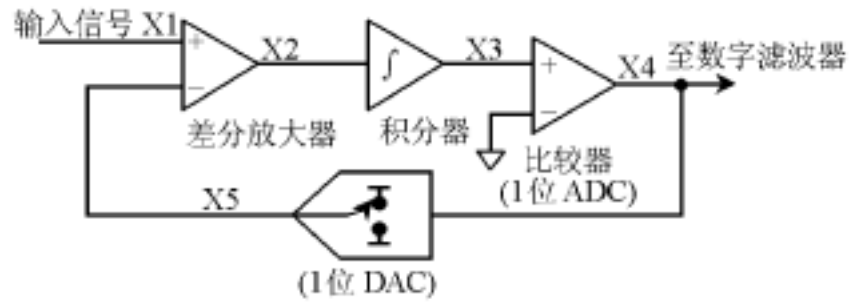


图 2 9－1　Σ－Δ 调制器

差电压求和,对于输入信号表现为一个低通滤波器,而对于量化噪声则表现为高通滤波。这样,大部分量化噪声就被推向更高的频段。和前面的简单过采样相比,总的噪声功率没有改变,但噪声的分布发生了变化。

现在,如果对噪声成形后的 Σ－Δ 调制器输出进行数字滤波,将有可能移走比简单过采样中更多的噪声。这种调制器(一阶)在每两倍的过采样率下可提供 9 dB 的 SNR 改善。

在 Σ－Δ 调制器中采用更多的积分与求和环节,可以提供更高阶数的量化噪声成形。例如,一个二阶 Σ－Δ 调制器在每两倍的过采样率下可改善 SNR 15 dB。图 2 9－2 显示了 Σ－Δ 调制器的阶数、过采样率和能够获得的 SNR 三者之间的关系。

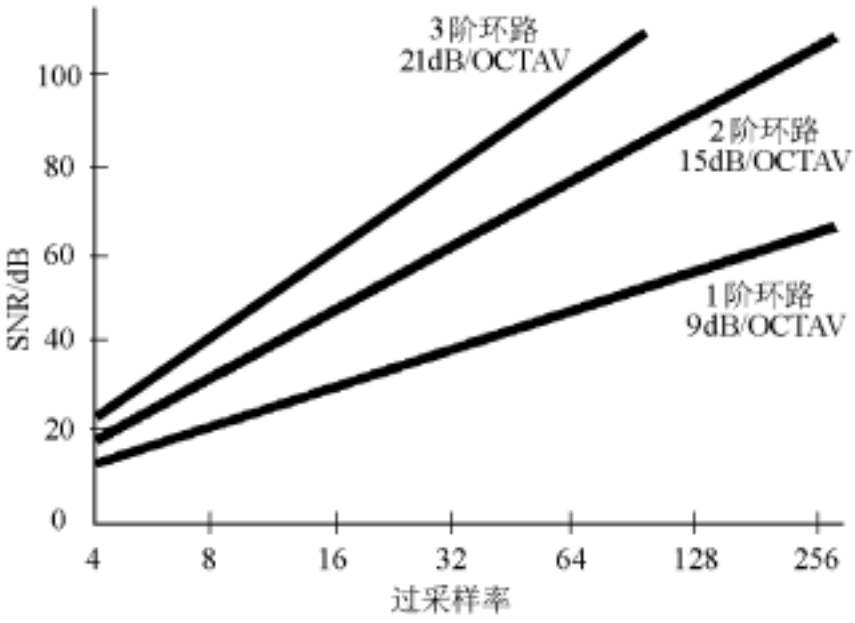


图 2 9－2　SNR 与过采样率的关系

3 . 数字滤波和抽取

Σ－Δ 调制器以采样速率输出 1 位数据流,频率可高达 MHz 量级。数字滤波和抽取的目的是从该数据流中提取出有用的信息,并将数据速率降低到可用的水平。

Σ－Δ ADC 中的数字滤波器对 1 位数据流求平均,移去带外量化噪声并改善 ADC 的分辨率。数字滤波器决定了信号带宽、建立时间和阻带抑制。

Σ－Δ 转换器中广泛采用的滤波器拓扑是 $SINC^3$,一种具有低通特性的滤波器。这种滤波器的一个主要优点是具有陷波特性,可以将陷波点设在和电力线相同的频率,抑制其干扰。陷波点直接相关于输出数据速率(转换时间的倒数)。$SINC^3$ 滤波器的建立时间三倍于转换时间。例如,陷波点设在 60 Hz 时(60 Hz 数据速率),建立时间为 3/ 60 Hz = 50 ms。有些应用要求更快的建立时间,而对分辨率的要求较低。对于这些应用,新型 ADC 诸如 MAX1400 系列允许用户选择滤波器类型 $SINC^1$ 或 $SINC^3$。$SINC^1$ 滤波器的建立时间只有一个数据周期,

对于前面的举例则为 1/ 60 Hz = 16.7 ms。由于带宽被输出数字滤波器降低,输出数据速率可低于原始采样速率,但仍满足 Nyquist 定律。这可通过保留某些采样而丢弃其余采样来实现,这个过程就是所谓的按 M 因子“抽取”。M 因子为抽取比例,可以是任何整数值。在选择抽取因子时应该使输出数据速率高于两倍的信号带宽。这样,如果以 f_s 的频率对输入信号采样,滤波后的输出数据速率可降低至 f_s/ M,而不会丢失任何信息。

二、MAXIM 的新型 Σ－Δ ADC

新型高集成度 Σ－Δ ADC 正在得到越来越广泛的应用,这种 ADC 只需极少外接元件就可直接处理微弱信号。MAX1402 便是这种新一代 ADC 的一个范例,大多数信号处理功能已被集成于芯片内部,可视为一个片上系统,如图 2.9－3 所示。该器件在 480 sps 工作速率下可提供 16 位精度,4 800 sps 时精度达 12 位,工作模式下仅消耗 250 μA 的电流,掉电模式仅消耗 2 μA。信号通道包含一个灵活的输入多路复用器,可被设置为 3 路全差分信号或 5 路伪差分信号、2 个斩波放大器,1 个可编程 PGA(增益 1～128)、1 个用于消除系统偏移的粗调 DAC 和 1 个二阶 Σ－Δ 调制器。调制器产生的 1 位数据流被送往一个集成的数字滤波器进行精处理(配置为 $SINC^1$ 或 $SINC^3$)。转换结果可通过 SPI™/ QSPI™ 兼容的三线串行接口读取。另外,该芯片还包含有 2 个全差分输入通道,用于系统校准(失调和增益);2 个匹配的 200 μA 电流源,用于传感器激励(例如可用于 3 线/ 4 线 RTD);2 个“泵出”电流,用于检测选定传感器的完整性。通过串行接口访问器件内部的 8 个片内寄存器,可对器件的工作模式进行编程。输入通道可以在外部命令的控制下进行采样或者连续采样,通过 SCAN 控制位设定,转换结果中附加有 3 位“通道标识”位,用来确定输入通道。

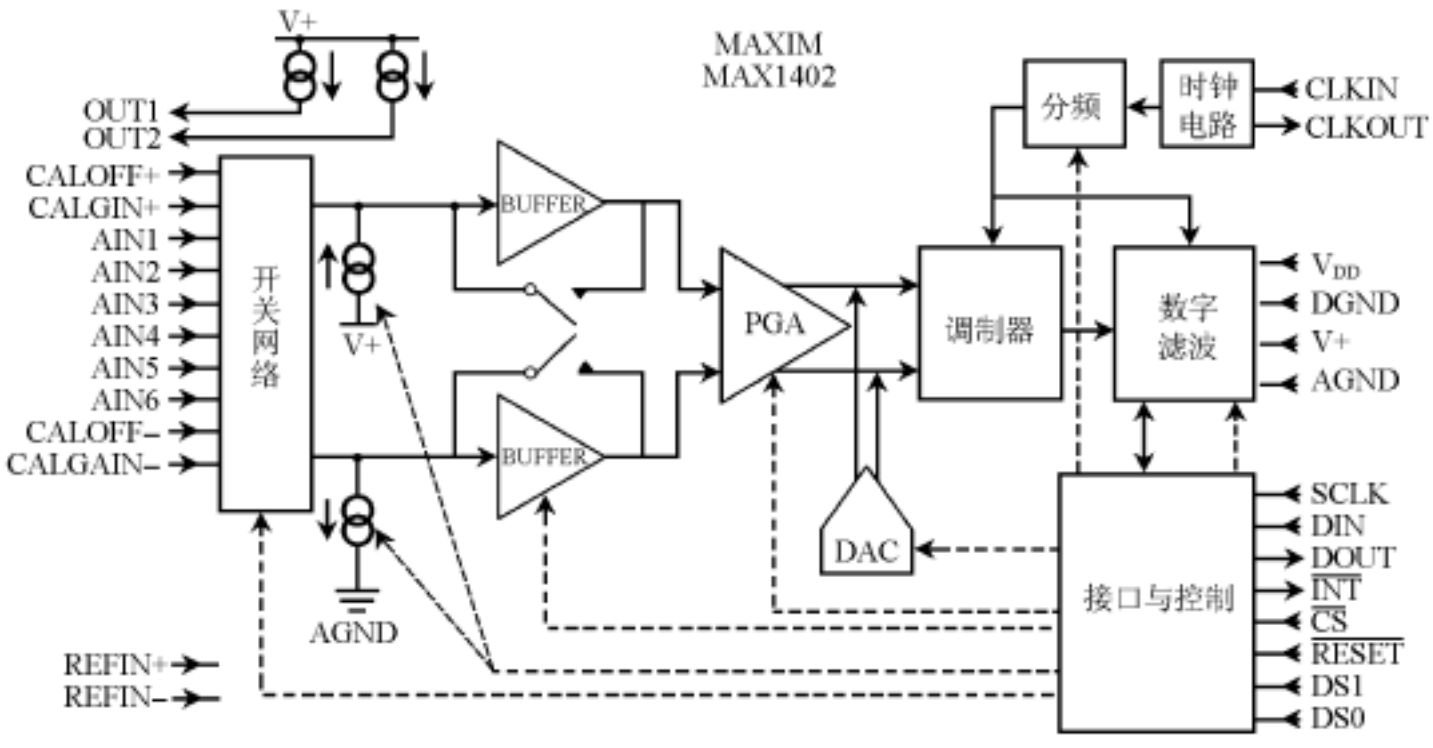


图 2.9－3 MAX1402 原理框图

两个附加的校准通道 CALOFF 和 CALGAIN 可用来校准测量系统。此时可将 CALOFF 输入连接到地,将 CALGAIN 输入连接到参考电压。对上述通道的测量结果求取平均后可用来对测量结果进行校准。

三、Σ－Δ ADC 的应用

1 .热电偶测量及冷端补偿

如图 2 .9－4 所示,在本应用中,MAX1402 工作在缓冲方式,以便允许在前端采用比较大的去耦电容(用来消除热电偶引线拾取的噪声)。为适应输入缓冲器的共模范围,采用参考电压对 AIN2 输入加以偏置。在使用热电偶测温时,要获得精确的测量结果,必须进行冷端补偿。热电偶输出电压可表示为

$$V = \alpha(t_1 - t_{ref})$$

其中 α 是与热电偶材料有关的 Seebeck 常数,t_1 是待测温度,t_{ref} 是接线盒处的温度。为了对 t_{ref} 造成的误差进行补偿,可以在热电偶输出端采用二极管补偿;也可以测出接线盒处的温度,然后用软件进行补偿。在本例中,差分输入通道 AIN3、AIN4 被用来测量 P－N 结的温度(用内部 200 μA 电流源加以偏置)。

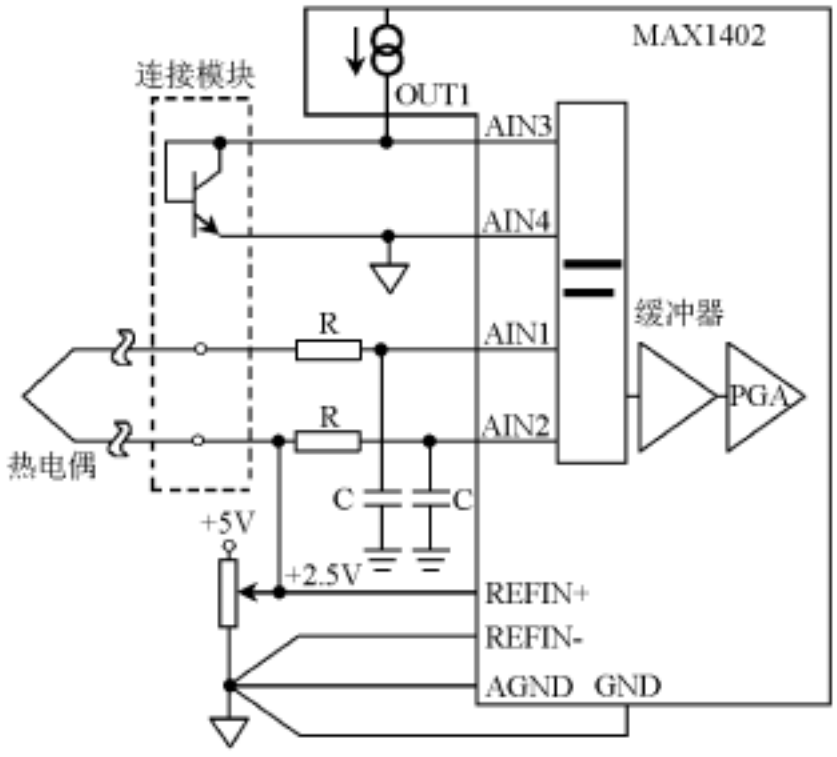


图 2 9－4　热电偶测量及冷端补偿

2 .3 线和 4 线 RTD 测量

铂电阻温度传感器(RTD)被许多需要测量温度的应用所优选,因为它们具有优异的精度和互换性。一个在 0℃时具有 100 Ω 电阻的 RTD,到 + 266 ℃时电阻会达到 200 Ω,灵敏度非常低,约为 ΔR/ Δt = 100 Ω/ 266 ℃。200 μA 的激励电流在 0 ℃时可产生 20 mV 输出,+ 266 ℃时输出 40 mV。MAX1402 可直接处理这种低电平的信号。

根据不同应用,引线电阻对于测量精度会产生不同程度的影响。一般来讲,如果 RTD 靠近转换器,采用最简单的两线结构即可;而当 RTD 比较远时,引线电阻会叠加入 RTD 阻抗,并给测量结果引入显著误差。这种情况通常采用 3 线或 4 线 RTD 配置,如图 2 .9－5 所示。

MAX1402 内部两个匹配的 200 μA 电流源可用来补偿 3 线或 4 线 RTD 配置中引线电阻造成的误差。在 3 线配置中,两个匹配的 200 μA 电流源分别流过 R_{L1} 和 R_{L2},这样,AIN1 和 AIN2 端的差分电压将不受引线电阻的影响。这种补偿方法成立的前提是两条引线材质相同,并具有相同的长度,还要求两个电流源的温度系数精确匹配 (MAX1402 为 5×10^{-6}/ ℃)。

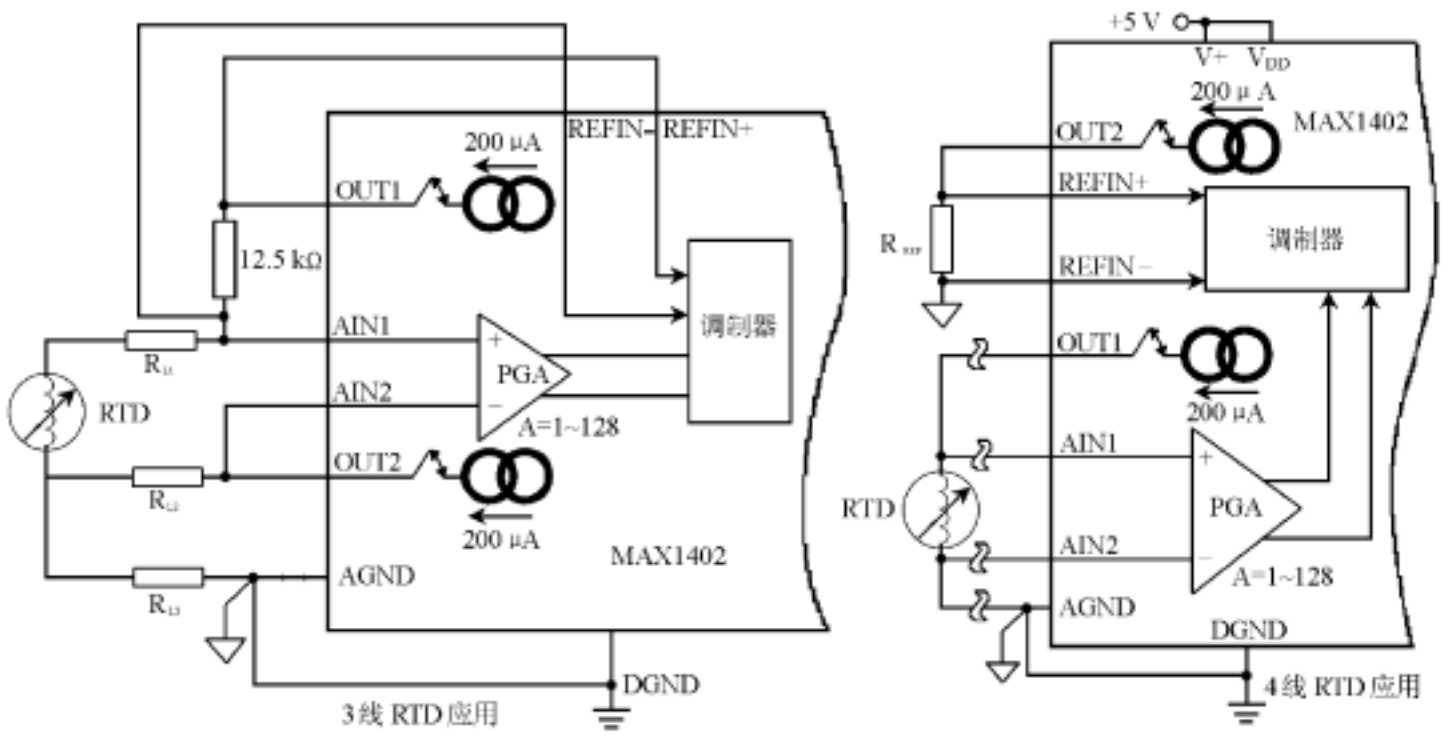


图 2 9－5 3 线和 4 线 RTD 测量

4 线配置中引线电阻将不会引入任何误差,因为在连接到 AIN1 和 AIN2 的测量引线中基本上没有电流流过。在此配置中,电流源 OUT1 被用来激励 RTD 传感器,电流源 OUT2 被用来产生参考电压。在这种比例型配置中,RTD 的温漂误差(由 RTD 激励电流的温漂引起)被参考电压的漂移补偿。

3 . 智能 4～20 mA 变送器

老式的 4～20 mA 变送器采用一个现场安装的敏感元件感测一些物理信息,例如压力或温度等,然后产生一个正比于待测物理量的电流,电流的变化范围标准化为 4～20 mA。电流环具有很多优点:测量信号对于噪声不敏感;可以方便地进行远端供电。第二代 4～20 mA 变送器在远端进行一些信号处理,通常采用微控制器和数据转换器,如图2 .9－6所示。这种变

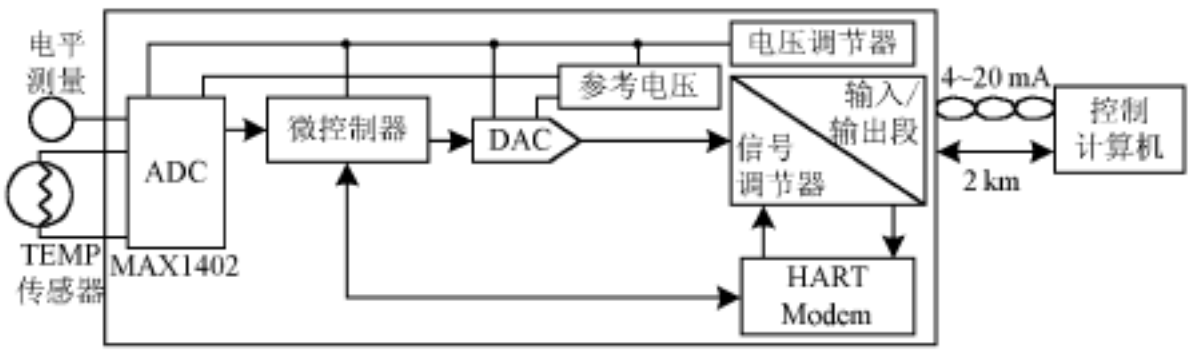


图 2 9－6 智能 4～20 mA 变送器

送器首先将信号数字化,然后采用微控制器内置的算法进行处理,对增益和零点进行标准化,对传感器进行线性化,最后再将信号转换到模拟域,作为一个标准电流通过环路传送。第三代 4～20 mA 变送器被称为“灵巧且智能”,实际上是在前述功能的基础上增加了数字通信(和传统的 4～20 mA 信号共用同一条双绞线)。利用通信信道可以传送一些控制和诊断信号。MAX1402 这样的低功耗器件对于此类应用非常适合,250 μA 的功耗可以为变送器中的其余电路节省出可观的功率。智能变送器所采用的通信标准是 Hart 协议。这是一种基于 Bell 202 电信标准的通信协议,工作于频移键控方式(FSK)。数字信号由两种频率组成:1 200 Hz

和 2 200 Hz,分别对应于数码 1 和 0。两种频率的正弦波叠加在直流模拟信号上,通过同一条电缆同时传送。因为 FSK 信号的平均值总是零,因此 4～20 mA 模拟信号不会受到影响。在不干扰模拟信号的前提下,数字通信信号具有每秒更新 2～3 个数据的响应速度。通信所需的最小环路阻抗是 23 Ω。

四、小　结

在高集成度调理系统出现之前,过程控制通常采用多个独立的芯片实现信号调理和处理。Σ－Δ 技术降低了这部分电路的成本、空间需求和功率需求(事实上多数应用只需要 + 3 V/ + 5 V单电源)。这种特性尤其适合于电池供电的便携系统。元件数量的降低同时还改善了系统的可靠性。

选自《单片机与嵌入式系统应用》月刊,2001 年第 10 期

2.10 过采样高阶 A/D 转换器的硬件实现

西北工业大学航海工程学院(710072) 邱宏安 朱治富 王英哲

过采样 Δ-Σ 调制技术应用于 A/D 转换器的研究,始于 20 世纪 80 年代初,到 20 世纪 80 年代中期,理论研究已基本成熟,目前在理论上主要探讨 Δ-Σ 调制器中的非线性特性及对 A/D 转换结果的影响。国内在这方面的研究仅仅处于起步阶段,国外高水平的研究主要集中在美、日两国,日本在高分辨 Δ-Σ A/D 转换器(24 位以上)应用技术上处于领先地位。就目前发展水平来看,实现过采样 Δ-Σ A/D 转换器的难点主要在技术和工艺方面。

传统 A/D 转换器,主要有积分型、逐次比较型和并联型三种,当分辨率超过 16 位时,目前的大规模集成电路技术都难以实现。例如,要实现 16 位 A/D 转换器,则需使用多达 65 000 个电容,而每个电容的容量偏差都必须控制在 0.2%的标准偏差内,这对生产制造来说是相当困难的。解决这一问题的办法就是采用过采样 Δ-Σ 调制技术。

本文提出利用多阶并联(MASH)方案实现 A/D 转换调制器。研究了三阶单比特调制器硬件实现方法,介绍了该方法的实现过程,给出了调制器的输出信号频谱图和时域波形。它可用于声信号的数据采集中。

一、过采样 Δ-Σ 调制原理

在模-数转换中,对模拟信号的量化必然带来量化噪声。量化噪声总功率与量化间隔(Q)的平方成正比,具体计算公式为

$$\sigma_e^2 = \frac{Q^2}{12} \tag{1}$$

量化噪声可假设为白噪声。从频域上看,它均匀分布在(0～f_s/2)的范围内,其谱密度为

$$e(f) = \frac{\sigma_s^2}{(f_s/2)} \tag{2}$$

式中,f_s 为采样频率。由于常规 A/D 转换器的量化间隔有限,因而量化噪声总功率难以进一步降低。但是,从上式可以看出,如果提高采样频率 f_s,则谱密度可降低,从而降低信号频带内的量化噪声功率,这就是过采样技术的基本出发点。

如果输入信号中心频率为 f_0,则过采样比(Oversampling Ratio,OSR)为

$$OSR = \frac{2f_s}{f_0} \tag{3}$$

如单纯将采样频率提高一倍,则信号频带内的量化噪声功率将降低 3 dB,也就是相当于增加 0.5 比特的分辨率。

Δ-Σ A/D 调制的思想最初来源于通信学科中的 Δ 调制和 Σ 调制两者的综合。因此,Δ-Σ 调制器又被称为“噪声整型器”。通过增加采样频率提高过采样比和提高调制器阶数来降低量化噪声谱密度。根据计算,N 阶调制器输出量化噪声总功率与过采样比 OSR 和调制器阶数 N 的关系是

$$\sigma^2 = \sigma_s^2 \frac{\pi^{2N}}{(2N+1)}(OSR)^{-(2N+1)} \tag{4}$$

这样,Δ－Σ 调制器输出信噪比为

$$\begin{aligned} SNR_Q &= 10\lg\frac{\sigma_s^2}{\sigma_e^2} \\ &= SNR_C + (2N+1)\lg(OSR) + 10\lg(2N+1) - 9.94N \\ &\approx 6L + (2N+1)\lg(OSR) + 10\lg(2N+1) - 9.94N \end{aligned} \tag{5}$$

式中,SNR_C 为常规量化器决定的信噪比,由量化级数 L 决定。也就是说,Δ－Σ 调制器输出信噪比与量化级数、调制器阶数成正比,与过采样比 OSR 的对数成反比。

将 Δ－Σ 调制器输出的信号经抽取器抽取后,就可得到高信噪比的数字信号,因此,Δ－Σ 调制器与抽取器串联即可形成过采样 Δ－Σ A/D 转换器。从应用的观点来看,Δ－Σ A/D 转换器的主要性能指标有信噪比和谐波失真两种。

Δ－Σ A/D 转换器的噪声包括调制器量化噪声、抽取器量化噪声和电路噪声。调制器量化噪声是整个转换器噪声的主要部分,可由式(4)计算出来,它直接由调制器阶数、量化间隔及过采样比决定;抽取器量化噪声由抽取滤波器(一般为 FIR 滤波器)长度和滤波器权系数字长决定。如滤波器长度为 20,权系数字长为 24 位和 32 位,则输出噪声分别为 -118 dB 和 -172 dB。电路噪声与具体电路设计方案和实现方法有关,由转换器动态范围决定。

Δ－Σ A/D 转换器产生谐波失真原因有两个:一是调制器反馈环节,二是量化间隔。调制器反馈环节由调制器实现方式决定;量化间隔大于 1,则有非线性现象出现,量化间隔越大,非线性越严重。

二、Δ－Σ 调制器的硬件实现

实现高阶调制器的方案分两大类:多阶串接和多阶并接(MASH)。多阶串接方案的优点在于分析方便,电路实现简单,不要求各阶电路之间保持严格匹配,但它的最大弱点在于难以保持稳定(当阶数超过三阶时尤其如此);设计多阶并接方案的目的就是为了保证高阶调制器的稳定,但它的不足之处在于各阶电路参数之间的严格匹配。本文选用 MASH 方案实现三阶调制器,见图 2.10－1。

下面具体分析其信号输入-输出关系。

对一阶 Δ－Σ 调制器,其模拟电路图及等效离散域框图如图 2.10－2、图 2.10－3 所示。设第 i 时刻输入信号,积分器输出,量化噪声和输出信号分别为 x_i,w_i,e_i 和 y_i。第 i 时刻积分器输出可表示为

$$w_i = x_{i-1} - y_{i-1} + w_{i-1} \tag{6}$$

而 Δ－Σ 调制器输出与积分器输出有以下关系:

$$y_i = w_i + e_i \tag{7}$$

所以有

$$y_i = x_{i-1} + e_i - e_{i-1} \tag{8}$$

同理,可得图 2.10－1 中三阶 Δ－Σ 调制器的等效离散域框图(图略),并可推出其输出与输入信号有以下关系

$$y_i = x_{i-1} + e_i - 3e_{i-1} + 3e_{i-2} + e_{i-3} \tag{9}$$

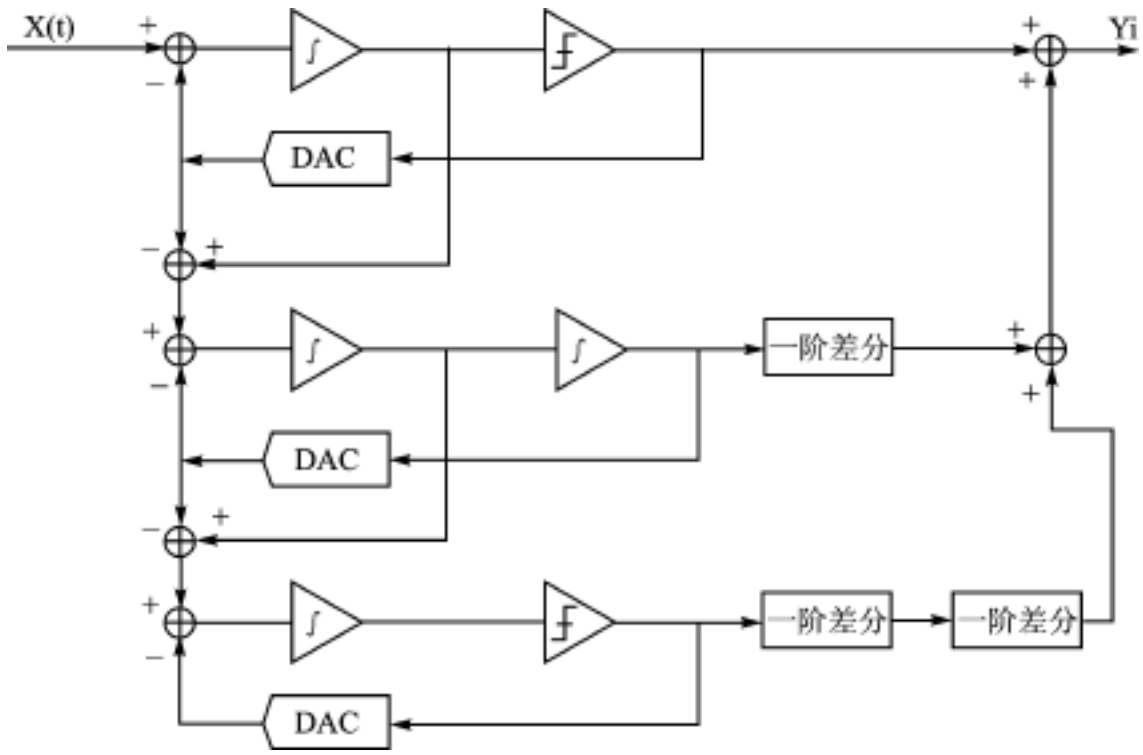


图 2 .10 - 1 三阶并接 Δ - Σ 调制器实现方案

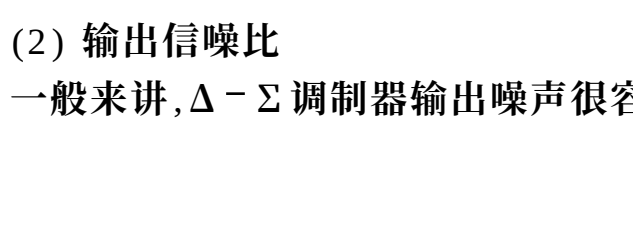


图 2 .10 - 2 一阶 Δ - Σ 调制器模拟电路

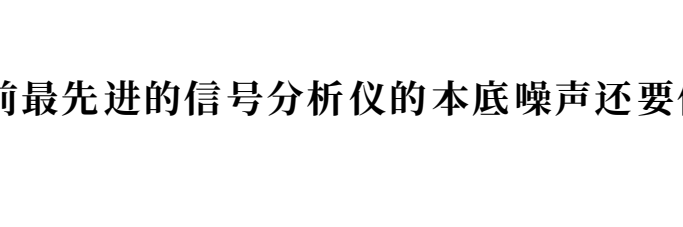


图 2 .10 - 3 一阶 Δ - Σ 调制器等效电路

可见,实现 MASH 方案的关键在于保证各阶调制器量化噪声能相互抵消。下面给出硬件实现 MASH 方案的关键技术:

(1) 实现一阶调制器的精确度

三阶 MASH 调制器的核心是三个一阶 Δ - Σ 调制器,它们决定了整个调制器的基本性能,为此要做到以下几点:

① +5 V 的参考电源要非常稳定(参考电源中的谐波分量和波动将直接反映到调制器输出中)。

② 由输出控制的反馈电压要保持对称稳定,否则将引起积分器输出饱和。

③ 积分器的积分时间和各阶滤波器的放大倍数要适当,具体调节要看积分器输出波形。

(2) 各阶调制器参数要严格保持一致

Δ - Σ 调制器输出信号是 PCM 信号,其质量决定整个转换器输出信号的质量。一般来说,对于三阶以下的抽取器,如中间抽取频率等于奈奎斯特频率的 4 倍,则抽取器引起的附加噪声不会超过 2 .75 dB。因此,我们可通过分析调制器输出信号,预测整个 A/ D 转换器性能。分析内容包括:

(1) 积分器输出波形

在连续域,理想的积分器输出为经输入信号调制的三角波。

(2) 输出信噪比

一般来讲,Δ - Σ 调制器输出噪声很容易比目前最先进的信号分析仪的本底噪声还要低,

为了验证设计的 Δ－Σ 调制器的实际信噪比,可以从下面两个方面着手分析:

① 将采样比降低,提高量化噪声,直至能在分析仪上观察出来;

② 将过采样比提高,观察量化噪声的变化,按式(5)推算 Δ－Σ 调制器实际信噪比。

三、测试结果

按上述方案,我们采用若干运算放大器和可编程 EPLD 芯片等构成的一台三阶单比特 Δ－Σ调制器,并调试成功,测试由 CF—350FFT 分析仪完成,输出信号由 B&K1027 信号发生器产生。输入信号中心频率为 70 Hz,调制器采样频率选用 8 kHz。图 2 .10－4 和图 2 .10－5 分别表示积分器输出和调制器输出(频域,16 次平均值)。调制器中设有滤波器,截止频率为 8 kHz,当采样频率为 8 kHz 时,过采样率为 16,按式(4)计算,三阶单比特调制器输出信噪比理论值为 60 .5 dB。由于信号发生器输出信号信噪比为 57 dB,实测信噪比为 53 dB,两者相差不大,因而可以认为符合理论设计要求。上述结论还可通过观察积分器输出得到验证。另外,当输入为宽带信号时,调制器亦能正常工作。在电路改进方面,主要有两点: ① 各级调制器中反馈电压由独立电源供给; ② 差分放大器的组成应进一步完善。

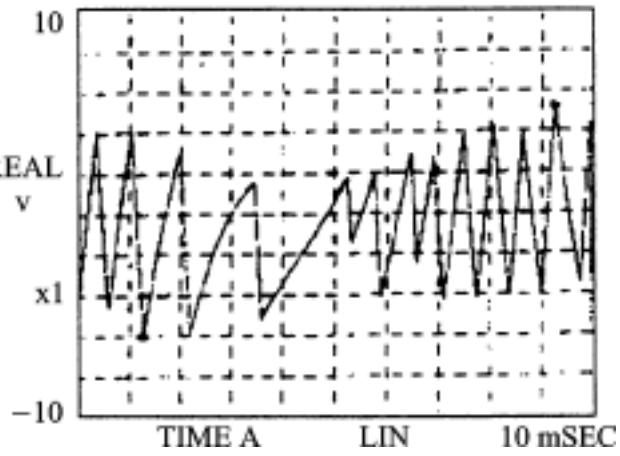


图 2 .10－4　积分器输出

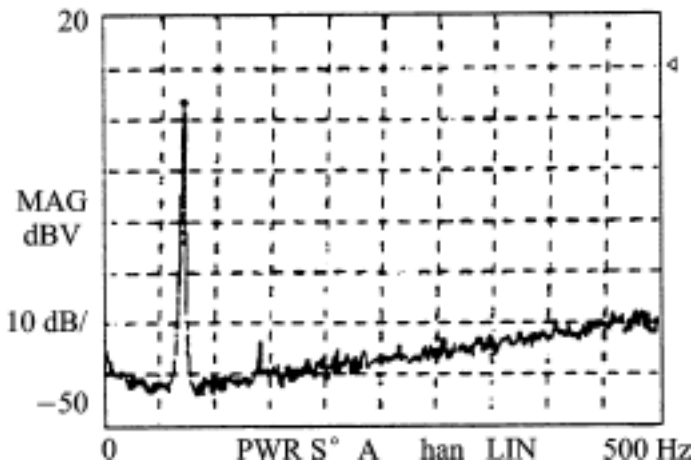


图 2 .10－5　调制器输出

四、结　论

测试结果表明:本文提出的高阶调制器及硬件实现方案是成功的。调制器主要用于实现高阶 A/ D 转换器和高精度 D/ A 转换。另外,它还可以用于 A/ A 转换,保证高质量数据传输,具体论述见文献[2]。

参　考　文　献

1　C Candy, C T Gabor . Oversampling delta—Signma data Converters [C] . IEEE Press, New York, 1992

2　D Harres . Delta—sigma analog—to—analog converter solves through design [J] . problem, EDN, 1995, 27:111～124

3　罗来根 . Σ－Δ ADC 分析与改进[J] .测控技术,1997(2):48～50

选自《探测与控制学报》季刊,2000 年第 2 期

2.11 A/D转换的计算与编程

河北省科学院自动化研究所(050081) 耿振鸿
河北省环保局信息中心(050051) 杜凡远

近年来计算机的硬件和软件都得到突飞猛进的发展,工程技术人员利用高级语言编制的商业软件,可方便地组成实现各种功能的系统软件。但是由于汇编语言运行速度快,占用硬件资源少,对软件环境依赖低,在许多场合仍然具有高级软件不可替代的优点。特别是用于智能仪表、家用电器等领域里的微电脑控制系统。对大型的工业控制系统,多半是将系统分成多级控制,其最低层又是由许多任务明确而简单的子系统组成。对于这些基层子系统,最适宜采用低级语言实现自己的各项功能。本文就工业控制中常用的A/D转换,汇编语言编程技巧及计算问题进行探讨,力图给出一个计算精度高、运算速度快、通用性强的实现程序设计方法。

一、A/D转换的基本原理

在微机工业控制、电测技术、智能仪表、数字通信等场合,输入系统的信息绝大多数是模拟量。为了让计算机能对这些模拟量进行处理,就得通过A/D转换技术将这些模拟量转换为数字量。最常用的A/D的转换方法是电压/数字转换技术,分为计数式、逐次逼近式、并行式和双积分式等方式。其中的逐次逼近方式,由于转换速度快、精度高、电路简单而被普遍采用。本文将着重讨论这种A/D转换的编程与计算。

对于A/D转换的编程通常采用4种方式,即程序查询方式、中断方式、同步方式和DMA方式。DMA方式多用于高速A/D转换,其速度超过CPU的控制速度,CPU无法对其进行控制,只能由硬件逻辑电路实现。对于A/D转换时间较长,并且有几件事需要CPU同时处理时,采用中断方式为宜。如果A/D转换时间较短,那么由于响应中断,保留现场,恢复现场,退出中断等环节的时间开销甚至超过A/D转换的时间,这时最好采用同步方式或查询方式。A/D转换的编程主要包括:A/D转换,对转换结果进行滤波,然后将最终结果经一定的运算处理后,输出供显示或是对设备进行控制。

二、A/D转换过程计算

就其本质而言,A/D转换就是利用有限的数字输出代码进行组合后用来表示某物理量在规定范围内的全部模拟值。假设某物理量的取值范围为(A_0,A_r),其所对应的数字量为(D_0,D_r),那么介于(D_0,D_r)中的任意数字量D_x与所对应的模拟量A_x有:

$$A_x = (A_r - A_0) \times \frac{(D_x - D_0)}{(D_r - D_0)} + A_0 \qquad (1)$$

式中数字代码D_0、D_x、D_r是理论值。实际上由于系统热噪声的存在,对于同一模拟信号的采集会出现±2LSB甚至更大的误差。这种随机误差,通常通过数字滤波予以降低。在计算A_x时,我们取D_r、D_0、D_x为实际测量并经滤波后的数值,且在滤波时取P+2采样值,在求均值计

算时去掉其中的最大值和最小值。假设去掉极值后的 D_r、D_0、D_x 的滤波的有效样品值之和分别为 S_0、S_x、S_r。

则由式(1)容易导出:

$$A_x = K \times 2^z (S_x - S_0) + A_0 \tag{2}$$

式 $K = \dfrac{2^l \times R}{2^z (S_r - S_0)}$ 中 R 的取值范围为 8 000 < R < FFFF(十六进制码),l 是通过对 $A_r - A_0$ 移位得到的对应于 R 的指数。这样做的目的是为了减少计算的舍入误差,增强软件的通用性。2^z 是为避免在计算 K 值时产生溢出而作的恒等变换,对 8 位到 16 位的不同 A/D 转换系统根据 P 的取值不同,Z 的取值范围应为 -4～8,如在 12 位的系统里,可取 Z=0,P=16。

通过上述的推导过程我们可以从式(2)获得如下好处:

(1) K 值具有很强的通用性,其取值范围为 2^{-128}～2^{127}。

(2) 直接用滤波器的和 S_x 进行运算求取 A_x,减少了除法运算从而节约了 CPU 的运行时间,增强了 A/D 转换的实时性。作为特例,当选用 12 位分辨率的 A/D 芯片并取 P=16,且 A=0,S=0 时有 $A_x = K \times S_x$。

对于逐次逼近式的 A/D 转换器,其转换过程含有两个用于等待的时间段,一段时间用于前置放大和采样延时,另一段时间用启动和 A/D 转换。CPU 可以利用这两个时间段对上一人采样值进行滤波处理,以加快 A/D 检测过程。

三、采用多 A/D 转换系数减少系统误差

进行 A/D 检测时存在 3 种误差,即随机误差、计算误差和系统误差。随机误差和计算误差可通过数字滤波和改进算法得到有效处理。下面通过软件方法来减少系统误差。

A/D 转换过程中的系统误差主要由 3 个环节产生:(1) 传感器的相对误差;(2) 变送器的相对误差;(3) A/D 转换电路的相对误差。可以通过采用先进工艺、提高硬件的产品质量来减小系统误差。但不管硬件产品质量多高,其相对误差总是无法避免的,这就是说利用软件进一步改善 A/D 测量中的系统误差仍是必要的。采用软件措施可以降低成本,也给系统的维护和参数的标定带来了极大方便。

由式(2)可以看出,降低系统的非线性误差的关键是如何获得准确的 A/D 转换系数 K。为此我们把模拟量 A_x 与数字量 D_x 的对应曲线分成若干段,如图 2.11－1 所示。图中曲线 1 为理想曲线,曲线 2 为实测曲线,(D_{i-1},D_i)是其中的一段。直线 BC 的斜率就是根据测量值 D_{i-1},D_i 及 A_{i-1},A_i 通过公式 $K = \dfrac{2^l \times R}{2^z (S_r - S_0)}$ 求得的 A/D 转换系数 K_i。对于 D_{i-1} 到 D_i 上的任意一点 D_x 求其所对应的 A_x,显然使用 K_i 比使用 K 所引起的线性误差要小些。

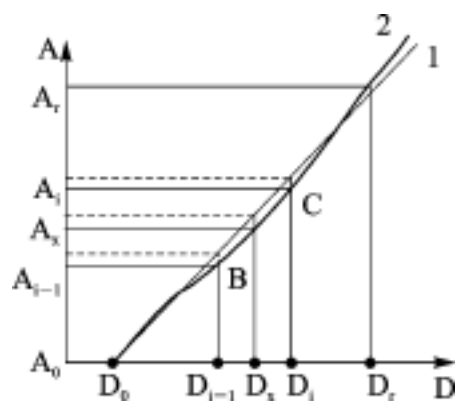


图 2.11－1 数字量与模拟量的关系曲线

对于在实际测量中到底将曲线分成几段才能取得最理想的效果,主要取决于 A/D 转换系统的分辨率,一般分辨率越高,分段越细。对于 12 位的 A/D 转换系统可分为 3、4 段,如果分辨率为 16 位,则可分为 5、6 段,甚至更细。

四、编程与实践

根据前面的分析与讨论,可对 A/ D 转换系统编制一个通用性强、精度高、运行速度快、对硬件环境要求低的测量软件。该软件可分为 3 个模块。

1 . A/ D 采样模块

该模块用于对给定模拟量 A_x 求其 A/ D 转换值,完成采样及滤波处理。首先对系统进行初始化,包括进行采样次数、滤波器的极大值、极小值及采样和的设置。之后便是采样与滤波计算,滤波计算是在采样时 A/ D 转换的延时中完成的,其流程如图 2 .11 - 2 所示。

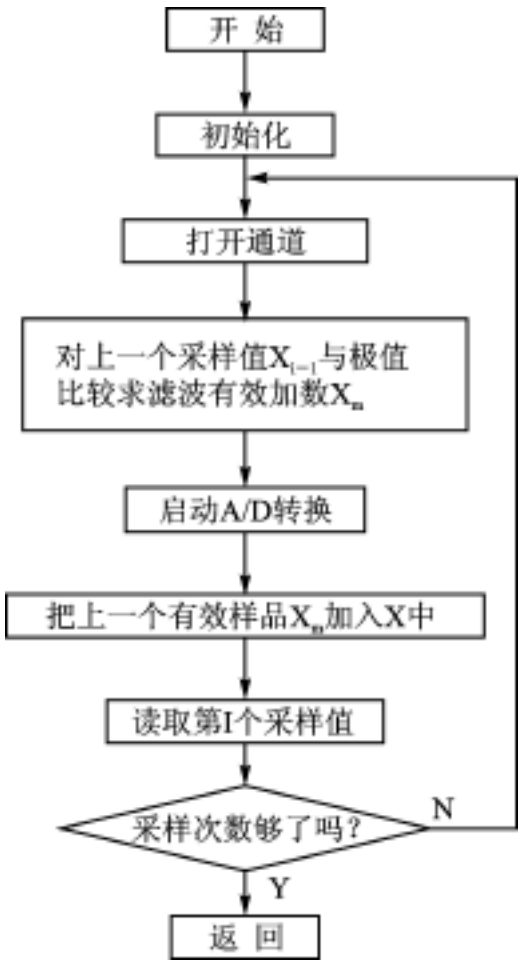


图 2 .11 - 2 A/ D 采样流程图

2 . 系统标定模块

根据实际情况可以在 A_0 到 A_r 之间选若干点进行标定,首先利用模块 1 在 A 点进行 A/ D 检测求得 S_0,然后逐个检测,对于每个由相邻检测点所形成的区间,分别计算出 K,并将各点的测量值和 K 值存入 E^2 PROM中作为实时 A/ D 转换的计算根据,用 3 个字节存放 K 值,其中一个字节用于存放 1,两个字节存放 $R/(S_i - S_{i-1}) \times 2^z$。由测量得到的 A_i 及 S_i 各占用两个字节存放。从而形成一张参数表。

3 . 实时 A/ D 检测模块

在实时检测中,利用模块 1 对某模拟量 A_x 进行 A/ D 转换求得 S_x,利用查表的方式在由模块 2 形成的参数表中查找 S_i,使得 S_x 介于 S_{i-1} 与 S_i 之间,然后再利用(4)求得 A_x。

经在玻璃微机配料控制系统及纸张克重水分微机控制系统的应用实践证明,本文所介绍的方法具有精度高、实时性强的特点,取得了良好的检测效果。特别是在玻璃配料系统中的称重检测中,我们对本文所介绍的方法与通常单一 A/ D 转换系数的测量方法进行了对比,在同样的硬件环境下,其称重测量精度几乎提高了一个数量级,虽然我们采用的是国产称重传感器和自制的变送器,但其测量精度仍达到了 Philips 同类产品的水平。

参 考 文 献

1 AXIM . NUW RELEASES DATA BOOK . 1996

2 雷丽文,朱晓华,等 .微机原理与接口技术[M] .北京:电子工业出版社

3 蔡征宇 .AD9007 12 位 ADC 用于 10 MHz 采集系统的设计[J] .电子工程师,1996,1

选自《制造业自动化》月刊,2000 年第 6 期

2 .12　一种提高单片机内嵌式 A/ D 分辨力的方法

清华大学精密仪器系(100084)　陈非凡　尤　政　于世洁

一、引　言

对于智能测试仪器的设计,单片机(CPU)和 A/ D 转换器都是必不可少的两个部件。随着微电子技术的发展,目前越来越多的单片机都带有内嵌式 A/ D 集成电路,这无疑给现代仪器的设计提供了一条缩小体积、简化电路、提高模拟通道信噪比、降低系统成本的有效方法与途径。可是目前常用的 8 位和 16 位单片机中,其内嵌式 A/ D 转换器的分辨力都只有 8 位或 10 位,而对于许多精密测试仪器来说,分辨力为 10 位的 A/ D 不能满足测量任务的要求。例如 3 位半(“ ±1999 ”)数字电压表,就要求其 A/ D 分辨力至少为 12 位。所以,许多情况下由于分辨力的原因,设计者只好“ 忍痛割爱 ”,放弃使用单片机内嵌式 A/ D 通道。本文介绍一种提高单片机内嵌式 A/ D 分辨力的有效实用方法,能更合理利用单片机内部资源,通过简单外围电路和单片机内部软件计算可以将原 A/ D 分辨力提高 1～3 倍。

二、原理与应用

通过对 Intel、Motorola、Philips、Nec 等世界著名大公司的几百种 8 位和 16 位单片机[1] 的内嵌模拟通道的分析,发现其内嵌式 A/ D 转换器几乎都具有如下相同的特征:

(1) A/ D 分辨力为 8 位或 10 位;

(2) A/ D 转换器提供多路输入通道;

(3) A/ D 转换器的信号输入端要求单极性输入;

(4) A/ D 转换器多为逐次比较型结构,且外设 A/ D 参考电压。

基于以上特点,下面以常用的 Philips 单片机 80X552 为例来说明如何利用简单外围电路将其内部原 10 位分辨力的 A/ D 提高到 11 位和 12 位使用。80X552 的内嵌式 A/ D 是单极性的(0～V_{max},一般情况下 V_{max} = 5 .0 V),8 路输入通道(ADC0～ADC7),10 位分辨力的转换器[2]。考虑一般性,假设表示原物理量的信号是双极性的模拟信号,其幅度 V_i 变化范围为 - V_m ～ + V_m,即

$$\begin{cases} V_i^{TOP} = V_m \\ V_i^{BOT} = -V_m \end{cases} \quad (V_m > 0) \tag{1}$$

式(1) 中的上标“ TOP ”和“ BOT ”分别表示该变量的上限和下限值。A/ D 转换的参考电压为 V_{ref} ($V_{ref} > 0$),对于单极性的 A/ D 转换器,最优工作条件是端口输入信号 V_x (下标 x 为 A/ D 的输入通道号)应该满足

$$\begin{aligned} V_x^{TOP} &= V_{ref} \\ V_x^{BOT} &= 0 \end{aligned} \tag{2}$$

为了满足以上条件,一般采用外围电路实现以下关系式

$$V_x = A \cdot V_i + V_{os} \tag{3}$$

其中,A 表示端口输入信号 V_x 对原始输入信号 V_i 的放大倍数,V_{os} 是为了满足条件式(2)而人为引入的直流偏置。

典型的实现电路如图 2.12-1 所示。V_0 是 80X552 的 ADC0 号 A/D 输入通道。显然图中电路满足条件式(2),此时,$V_{os} = V_{ref}/2$,A = 1,且由式(3)可以得出

$$V_x^{TOP} - V_x^{BOT} = A(V_i^{TOP} - V_i^{BOT})$$

x = 0,A = 1 时

$$V_{ref} = V_0^{TOP} - V_0^{BOT} = V_i^{TOP} - V_i^{BOT} = 2V_m \tag{4}$$

于是,A/D 转换器对原始输入信号 V_i 能分辨的最小电压 VR 可以计算为

$$VR = \frac{V_{ref}/2^N}{A} = \frac{V_{ref}}{2^N} = \frac{2V_m}{2^N} = \frac{V_m}{2^{N-1}} \tag{5}$$

上式中 N 表示 A/D 转换器本身的二进制分辨力位数,对于 80X552 的内嵌式 A/D(10 位),N = 10。

为了说明问题,假设图 2.12-1 中原始输入信号 V_i 和参考电压 V_{ref} 的大小都保持不变,现在简单改变一下其外围电路的设计,修改以后的电路如图 2.12-2 所示,此时运放 A_1、A_2 的作用是对输入信号 V_i 中正负信号分别进行半波放大(2 倍),并分别与两个不同的 A/D 输入通道(ADC0、ADC1)连接,显然 V_0、V_1 均满足 A/D 转换器最优工作条件式(2)。值得说明的是图 2.12-2 中二极管 D 的连接方式很重要,它既起到了负向电压嵌位的作用,同时又不影响信号零点附件的灵敏度,特别适合精密测量,因为实践表明如果采用全波整流,二极管的非线性导通特性会明显影响信号零点附件的灵敏度。

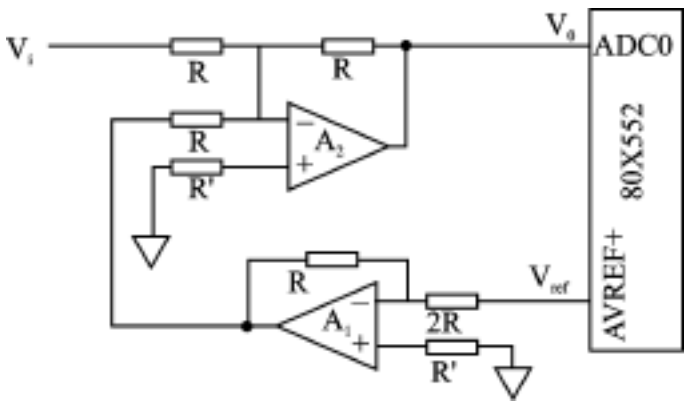


图 2.12-1

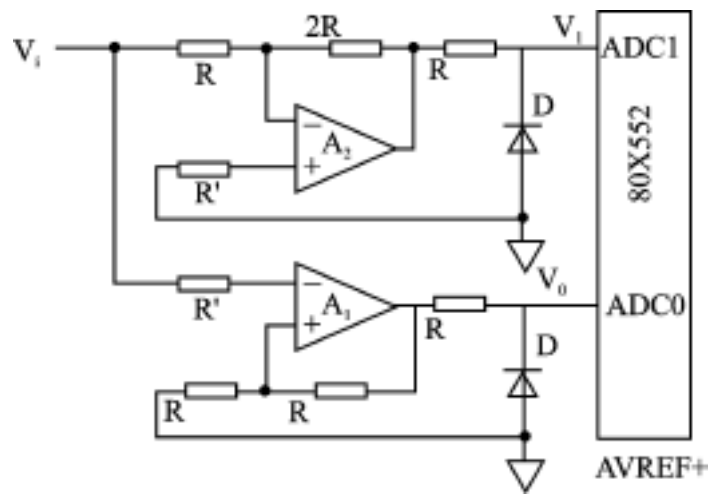


图 2.12-2

利用软件可以在单片机内部将两个输入通道的 A/D 结果 V_0、V_1 合成为一个结果 V。其软件处理过程如下:

同时采集通道 ADC0,ADC1

$$\begin{cases} \text{当 } V_i \geq 0 \text{ 时}, V = V_0 = 2V_i \\ \text{当 } V_i < 0 \text{ 时}, V = V_1 \times (-1.0) = -2V_i \end{cases}$$

A/D 结果合成后 $V = 2V_i$

于是单片机内部合成信号 V 相对于原始输入信号 V_i 有

$$\begin{cases} A = 2 \\ V^{TOP} = 2V_m \\ V^{BOT} = -2V_m \end{cases} \tag{6}$$

此时 A/ D 转换器对原始输入信号 V_i 能分辨的最小电压 VR′可以计算为

$$VR' = \frac{V_{ref}/2^N}{A} = \frac{V_{ref}}{2 \cdot 2^N} = \frac{2V_m}{2 \cdot 2^N} = \frac{V_m}{2^N} \tag{7}$$

比较式(5)可以看出,VR′是 VR 的 1/ 2,数字化分辨力明显提高 1 倍。也就是相当于 A/ D 转换器本身的二进制分辨力位数 N 提高了 1 位(相当于 N = 11)。相对于图 2 .12 - 1电路,图 2 .12 - 2 电路并不需要增加很多的外围元件。

在图 2 .12 - 2 的基础上,再对其外围电路修改一下可以将 A/ D 分辨力再提高 1 倍,修改后的电路如图 2 .12 - 3 所示,类比于以上分析过程,不难导出图 2 .12 - 3 中 A/ D 转换的分辨力是图2 .12 - 2的 2 倍,或图 2 .12 - 1 的 4 倍(即相当于 N = 12)。限于篇幅,在此不对图中 A/ D 电路的分辨力做详细的推导分析。惟一需要说明的是,图 2 .12 - 3 中参考电压 AV_{ref}的大小应该为图2 .12 - 1、图 2 .12 - 2 中参考电压 V_{ref}的 1/ 2。

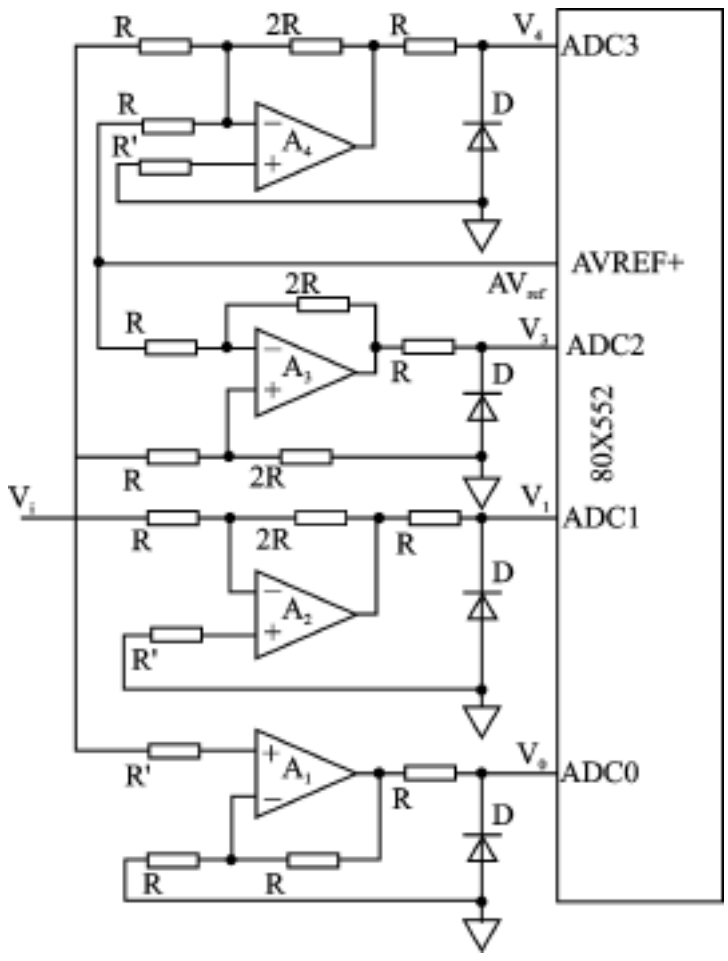


图 2 .12 - 3

以上各图中的 R 可以取相同的电阻,而 R′分别为对应运放单元的匹配电阻。如果采用单片运放的集成电路,如 AD713、LF444、LM324 等,图 2 .12 - 3 相对于图 2 .12 - 1、图 2 .12 - 2 的外围电路只需简单增加几个电阻和二极管,这也正是该方法的实用之处。目前,该方法已经成功地应用在我们开发的分辨力为 0 .01′的高精度自准直系统中,获得了理想的应用效果。

本文介绍了一种提高单片机内嵌式 A/ D 分辨力的有效实用方法,通过简单外围电路和单片机内部软件计算将原分辨力提高 1～3 倍。该方法能更合理利用单片机内部集成资源,对需要考虑成本和电路板大小的精密测量电路设计与研制,提供了一套简单、实用、有效的技术改进途径。

参　考　文　献

1　纪宗南 .单片机外围器件实用手册:输入输出器件分册 .北京:北京航空航天大学出版社,1998
2　张友德 .飞利浦 80C51 系列单片机原理与应用手册 .北京:北京航空航天大学出版社,1992

选自《航空计测技术》双月刊,2000 年第 2 期

2.13 单片微型计算机多字节浮点快速相对移位法开平方运算的实现

长春吉林大学 赵 伟 王晶芝

一、引 言

在计算机做高精度数学运算时,通常采用浮点运算。其中除了加、减、乘、除运算外,开平方运算是最常用的一种运算。目前,在开方运算中,通常采用的数据结构有二种:一种是采用“减奇”算法;另一种是采用“牛顿迭代”算法。在定点数开方中常使用“减奇”算法。但对于一个较大数开平方要花大量运算时间,且精度低;“牛顿迭代法”虽然简单,但是初值不易选择,迭代次数不定,收敛速度慢且不易确定,还要不断使用浮点除法程序,所以对于快速高精度的数据处理不够理想。本文介绍一种可以在微型计算机上实现多字节浮点快速开平方运算的方法。

二、二进制数开方原理

二进制数开平方基本方法与十进制开方基本类似,试根总是两位一试,所以,用上一次的“根”乘 2 后加 1 就产生了新的试根。下面以 $(144)_{10}$ 开方为例:$(144)_{10} = (10010000)_2$ 而 $(10010000)_2$ 开平方的结果为:

$$\sqrt{(10010000)_2} = (1100)_2 = (12)_{10}$$

$$\sqrt{(144)_{10}} = (12)_{10}$$

$(144)_{10}$ 开平方结果为 $(12)_{10}$,与二进制开平方结果相同。

因此,二进制开方与十进制开方方法上完全相似,也是在试根过程中,够减时“根”记 1;不够减时“根”记 0。

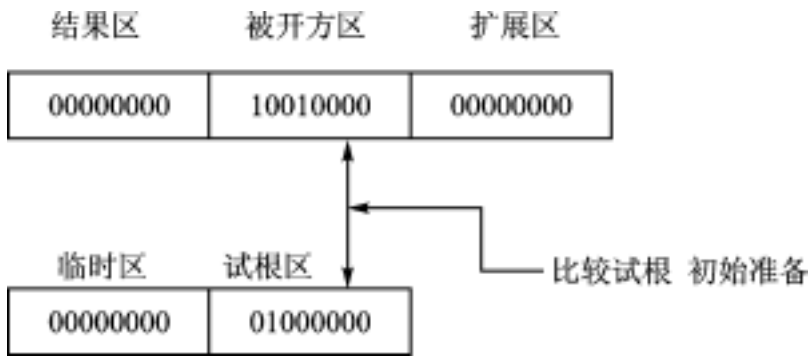
三、快速相对移位法开平方运算的实现

本文根据二进制数开平方的基本原理导出快速相对移位法的基本模型。根据“分缝法”原理推出了“相对移位”法,它将浮点二进制数分为阶码处理与尾数处理两部分,其重点放在尾数处理上;应用相对移位,在保证二位一节正常开方的基础上,巧妙地使相减的试根区长度缩短了一倍,从而使运算时间大大缩短。

1.尾数处理

对于 n 位二进制数开方,通常设立 2n 位区域,前 n 位为被开方区,后 n 位为扩展区,这样才能保证结果为 n 位的精度。下面以具体实例说明相对移位法的具体实现过程。假设 n = 8。

对尾数 $(10010000)_2$ 进行开方,将它放入被开方区,后面加上 8 位扩展区,结果区(8 位)清 0,临时区与试根区清 0。



(1) 被开方区与试根区比较

00000000	01010000	00000000

够减, CY = 0, 被开方区高位为 0

00000000	10100000	00000000

结果与被开方区左移 1 位

00000001	10100000	00000000

$\overline{CY}$→送结果低位

试根区处理

00000001	1000000

结果→临时区, 试根区置 01

00000000	10100000

临时区与试根区右移 1 位

(2) 被开方区与试根区再比较

00000001	00000000	00000000

够减, CY = 0, 被开方区高位为 0

00000010	00000000	00000000

结果与被开方区左移 1 位

00000011	00000000	00000000

$\overline{CY}$→送结果低位

试根区处理

00000011	01000000

结果→临时区, 试根区置 01

00000000	11010000

临时区与试根区右移 2 位

(3) 被开方区与试根区比较

00000011	00000000	00000000

不够减,CY = 1,被开方区恢复为原值,高位为 0

00000110	00000000	00000000

结果与被开方区左移 1 位

00000110	00000000	00000000

$\overline{CY}$→送结果低位

试根区处理

00000110	01000000

结果→临时区,试根区置 01

00000000	11001000

临时区与试根区右移 3 位

(4) 被开方区与试根区比较

00000110	00000000	00000000

不够减,CY = 1,被开方区恢复为原值,高位为 0

00001100	00000000	00000000

结果与被开方区左移 1 位

00001100	00000000	00000000

$\overline{CY}$→送结果低位

以上为相对移位法的简要图示,结果区为$(00001100)_2$,正是$(10010000)_2$的开方结果,从上述处理过程我们可以看出,尾数处理主要有二个过程: ① 被开方区与试根区比较; ② 处理试根区。第二个过程是得到浮点尾数的关键。下面仍以上述过程为例进行详述。

首先,结果区清 0,临时区清 0,试根区放入 40H(01 试商);被开方区减试根区,若够减(CY = 0),并且被开方区高位为“0”,则结果区与被开方区一起左移 1 位,$\overline{CY}$送结果低位,表示产生当前临时结果;若被开方区高位为“1”,被开方区就不能移位了。否则将丢失有效中间结果,所以这时仅结果区左移 1 位,$\overline{CY}$送结果低位,由于此次被开方区未左移,下次试根右移时仅右移 1 位,与相对移位不符,而且不符合开方原理。因此处理方法为:在移位时若被开方区高位为“1”,则不动被开方区,仅移结果区,特殊计数增 1,以便使试根区在下一次试根时多下移 1 位,与被开方区正确对齐比较。

其次,被开方区减试根区,若不够减(CY = 1)时,则先恢复被开方区,其他处理方法与够减时类似。

试根区处理时,先将当前结果区送入临时区,然后临时区与试根区一起右移,将已得到的当前结果移入试根区,此时若特殊计数不为 0,则临时区与试根区继续右移特殊计数的次数,向被开方区对齐,进行下一次比较,直至结束。

由于被开方区每次左移 1 位,试根区又右移 1 位,相当于一次向下对 1 节(2 位)。这就是相对移位法与正常开方的相似之处,但它节省了移位次数和空间。

2 . 阶码处理

当阶码为 n = 2k 时(k 为整数),不论正负一律折半,保留符号,即为当前结果阶码。因为:

$$\sqrt{M \cdot 2^{n}} = \sqrt{M \cdot 2^{2k}} = \sqrt{M} \cdot 2^{k}$$

尾数 M 右移一位空出一节准备得到的结果符号位,开方方法见尾数处理。

当阶码为 n = 2k + 1 时,分两种情况处理。阶码为正时,阶码减 1,成为偶数,此时尾数 M 有一位为整数了,开方完毕以后,在结果中仍有一位为整数(1.* * * 开方以后仍为 1.* * *),阶码除以 2 后再进行结果规格化处理;阶码为负时,亦如此处理,M 不动,开方少计 1 位空出符号位,若负阶码用原码表示,须增 1(实际是减少 1 的处理),反之亦然。

四、讨 论

当被开方区小于试根时,并且由于高位为"1"不能左移,特殊计数增 1;设相比以前结果为 x 位,产生新结果为 x + 1 位,形成的试根也多了 1 位,由于特殊计数也增加了 1,试根还要多右移 1 位,比上一次试根值多下移 2 位,相当于试根下移 1 节,与被开方区中间结果相比,由于被开方区一次增加 1 节,且高位为"1",试根一次仅增 1 位,因此下次肯定够减,不会出现连续不够减且被开方区不能左移的情况。

若连续不够减,则被开方区高位为"0",可以继续左移,不受影响。

此外亦注意临界值,如 00(H)减 1 为 81(H),81(H)增 1 为 00(H)等。

在分析试根移位总次数时,我们看到共需 $N = 1 + 2 + \cdots + n(n+1)/2$ 次移位,若想缩短此时间,可以采用在开方过半以后,将试根变右移为从右向左对齐,这样相当于对两个 n/2 位的被开方数进行操作,那么移位次数为

$$N' = 2(1 + 2 + \cdots + n/2) = 2[(1 + n/2)n/2]/2 = n(n+2)/4$$

N′大约为 N/4,因此试根移位时间大大缩短,可以一试。

五、小 结

在 MCS—51 单片机上,当晶振为 7.3728 MHz 时,用牛顿迭代法对五字节浮点二进制数进行开平方,当数为小阶码正数时(小于 100(D),大于 0.01(D)),精度要求为 10^{-8} 时,耗时 120～250 ms,若阶码更大,迭代次数也更大,耗时很可观;用快速开方法实现同样的开平方运算,由于对尾数采用了手工相对移位法,不论数值大小,执行时间在 44 ms 左右,且精度达 10^{-9} 同样的条件,在 MCS—96 单片机上执行时间只有 12.5～13 ms。

因此,快速相对移位法开平方运算方法对单片微型机数据处理能力的提高大有好处。

参 考 文 献

1 周明德.微型计算机硬件软件及其应用.北京:清华大学出版社,1988
2 黄铠.计算机算术运算原理、结构与设计.北京:科学出版社,1980
3 何立民.MCS—51 系列单片机应用系统设计.北京:北京航空航天大学出版社,1990
4 薛均义,等.MCS—51/96 系列单片机微型计算机及其应用.西安:西安交通大学出版社,1993
5 赵伟,王晶芝.单片微型计算机多字节浮点除法快速扫描运算的实现.微型电脑应用,2000,4

选自《微型电脑应用》月刊,2000 年第 10 期

2.14 单片微型计算机多字节浮点除法快速扫描运算的实现

长春吉林大学 赵 伟 王晶芝

单片微型计算机的应用越来越广泛,它不仅应用在各种控制上,有时也用来做数据处理运算,因此,在保证高精度的情况下提高运算速度是很有必要的。

一、对传统的标准边减边移位除法的分析

标准浮点二进制除法分为阶码运算与尾数运算两部分,其主要部分为尾数运算。标准尾数相除是采用边减边移位的办法来实现的,即被除数向左移位进入余数区产生中间余数,中间余数与减数比较,若大于减数,则余数减去减数,商区未位上商"1"否则,恢复余数,商区未位上商"0"然后余数、被除数及商区一起左移,进入下一次操作,直至得到完整的商。

例如:被除数 除数

01010000 1000

余数区	被除数	商区	说 明
0000	0101	0000	
0000	1010	0000	被除数、商区、余数区左移一位
1000	1010	0000	减除数 d
0000	1010	0000	不够减,恢复余数,商上 0
0001	0100	0000	被除数、商区、余数区左移一位
1001	0100	0000	减除数
0001	0100	0000	不够减,恢复余数,商上 0
0010	1000	0000	被除数、商区、余数区左移一位
1010	1000	0000	减除数
0010	1000	0000	不够减,恢复余数,商上 0
0101	0000	0000	被除数,商区、余数区左移一位
0101	0000	0000	减除数、不够减恢复余数,商上 0
1010	0000	0000	被除数、商区、余数区左移一位
0010	0000	0000	减除数
0010	0000	0001	够减,商上 1
0100	0000	0010	被减数,商区余数区左移一位
0100	0000	0010	减除数,不够减,恢复余数,商上 0
1000	0000	0100	被除数,商区、余数区左移一位
0000	0000	0100	减除数
0000	0000	0101	够减,商上 1
0000	0000	1010	

这种方法对于两个 n 位二进制尾数进行操作,被除数扩展为 2n 位的操作数后才能进行,需要 n 个减法器。

若中间余数小于除数时,则需恢复余数,需进行一次加法,若中间余数小于除数与大于除数的机会均等,对于标准算法,进行了 2n 次 3n 位的移位,以及 2n 次 n 位的减法和 n 次 n 位的加法,因此,若对标准算法进行改进,则必须设法缩短它的移位时间和减法时间。

二、快速扫描除法原则

快速扫描除法在每个移位周期不是仅形成一位商,而是形成多位商,这样就可以减少移位周期,从而提高执行速度。

三、快速扫描除法的实现

在实际过程中,要多开设几个缓冲区来供现实除法使用。由于除法是乘法的逆运算,因此,可用乘法扫描方法的逆推来考虑,首先,被除数比较除数若大于除数则上商‘1”然后被除数、商区左移半个字节(4BIT)否则直接将被除数,商区左移半个字节,现在与除数比较的余数区扩展 N + 4 位,包括串出的高 4 位,由于被除数串出以前已小于除数,因此它左移 4 位相当于扩大 16 倍。以后再加上移上来的低 4 位,此数小于 16,这样,现在余数区对除数的倍数为 T,O≤T < 16,因为 T 若大于等于 16,证明余数区在移位前不小于除数,因此不可能。

这样,若确定 T 值以后,那么本次就可一次产生一个 4 位的商值 $Y = [YN_3, YN_2, YN_1, YN_0]_4$,(0≤Y≤16);这样就完成了一次循环。

那么,如何确定 T 是一个关键问题,这里采用二分比较的方法;因为 $Y = [YN_3, YN_2, YN_0]_4$”代表 16 种可能的状态,因此 0≤Y≤15;令除数为 C,在开始时用移位方法快速求得 2 * C,4 * C 与 8 * C 并保存在缓冲区,形成的 N + 4 位余数也在缓冲区,先从中间入手,余数与 8 * C 比较,若大于则利用上面结果顺利地与 12 * C 比,否则与 4 * C 比较;由于 $2^4 = 16$,这样在最坏的情况下经 4 次相减即可得到 Y 值,本文循环结束。在操作时由于有缓冲区保存中间结果,不够减时不用恢复余数,传送过来即可,这种循环重复[N/4]次。

快速扫描算法进行了[N/4]次 2N 位的移位,移位时间为标准算法的 1/2;进行了[N/4] * 次 N + 4 位的减法,比标准算法也大为减少,速度提高了一倍,且精度与标准算法一样。

四、讨　论

采用半个字节移位和一次扫描形成 4 位商的扫描方法大大减少了运算时间,不过由于判断复杂,增加了程序长度,而且对程序结构带来非规范化影响,在程序空间上作出牺牲;另外占用内存太多,使之利用率下降,这些也是有待解决的。

若想进一步扩大扫描位数,将使程序更加复杂化,这有待于实际情况的需要而进行进一步探讨。

参　考　文　献

1　周明德 .微型计算机硬件软件及其应用 .北京:清华大学出版社,1988

2　黄铠 .计算机算术运算原理、结构与设计 .北京:科学出版社,1980

3　何立民 .MCS—51 系列单片机应用系统设计 .北京:北京航空航天大学出版社,1990

4　薛均义,等 .MCS—51/96 系列单片机微型计算机及其应用 .西安:西安交通大学出版社,1993

选自《微型电脑应用》月刊,2000 年第 4 期

2.15 DSP芯片与触摸屏的接口控制

武汉理工大学(430074) 施保华 金晓波 秦娟英

以DSP(数字信号处理)芯片和FPGA(现场可编程逻辑门阵列)为核心的无线数字扩频通信平台是无线扩频通信的一个开放式平台,可用于无线接入、无线图像和音频传送、移动INTERNET、精确区域定位LPS、智能遥控探测等高科技领域。我们在此基础上增加了液晶显示和触摸屏控制,从而实现文字和图形信息的编辑和无线传送,使该产品用途更加广泛。

ADS7843是专用于4线电阻式触摸屏的12位模/数采样转换器,具有单一电源供电、完全降功耗模式、转换速度快的特点。ADS7843大量用在电池供电PDA(Personal Digital Assistants)和手持便携式装置中。

一、液晶触摸屏控制产品设计简介

液晶采用Microtips Technology Inc公司的型号为MTG-32240X的中小规模液晶显示器(图形方式为320×240点)。触摸屏控制器采用香港BURR-BROWN公司的ADS7843。对FPGA进行逻辑编程可实现液晶显示控制器的功能。用FPGA定时中断DSP(约占DSP工作时间的5%)获取外部RAM的显示数据,然后FPGA内部逻辑将显示数据送到液晶屏上进行显示。DSP对触摸屏的转换信号进行处理和计算以及完成各种文字和图形的编辑任务,然后将需要显示的数据送入显示缓冲区(外部数据RAM)。也就是说,用DSP和FPGA来控制液晶的显示和文字、图形信息的编辑处理,并能接收触摸屏信号,从而实现笔输入掌上电脑的功能。再结合无线数字扩频通信平台就能实现文字和图形信息的无线传送。该产品能用于记者在通信不便的现场进行文字和图形的实时编辑和采访报道以及在移动过程中需要进行文字和图形信息无线传送的地方。

二、ADS7843的工作原理

ADS7843的工作电压V_{cc}为2.7～5 V,参考电压V_{ref}在1 V至V_{cc}之间均可。转换器有4路模拟信号输入通道和独特的在线低阻开关,它允许由未选中的输入通道为外部线路提供电源和地。靠这样的输入结构和参考电压设定,ADS7843能极大地消除内部开关电阻带来的转换误差,这种误差是这类转换器的主要误差来源。+REF至-REF(见图2.15-2)之间参考电压的大小决定了模拟输入电压的范围。参考电压设置有两种工作模式:SER(single-ended reference)和DFR(differential reference)。SER模式需要低噪声、低波动的稳定电源,转换器内X+、X-和Y+、Y-的低阻开关对转换精度有一定影响。DFR模式不管内部开关电阻如何变化,其转换结果总是与外部电阻成比例,它完全克服了内部开关的影响,但是当转换器的转换频率很高时增加了功耗。

图2.15-1中,X+、X-和Y+、Y-之间横排和竖排的电阻代表电阻式触摸屏。开关S模仿PDA笔尖按压触摸屏时动态短接X、Y轴两层电阻的位置。Y位置进行转换时,X+作

为模拟输入(见图 2 .15 - 2)。

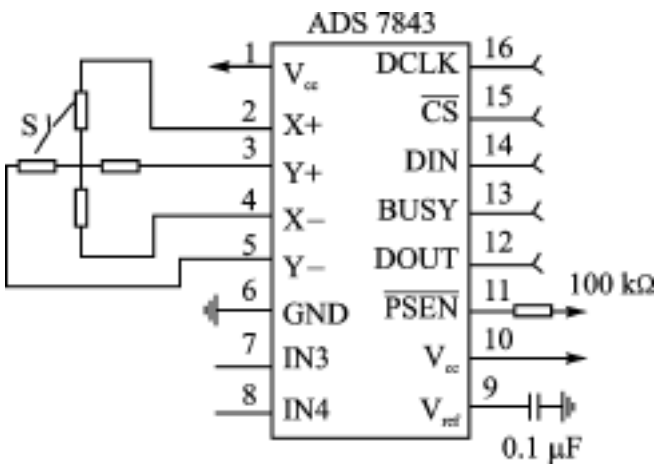


图 2 .15 - 1　ADS 7843 接线图

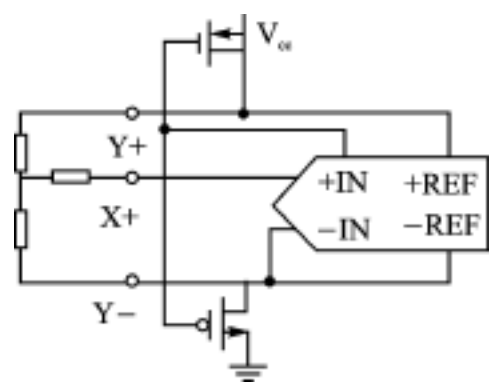


图 2 .15 - 2　DFR 模式, X + 模拟输入

ADS 7843 的控制字见表 2 .15 - 1。

表 2 .15 - 1　ADS 7843 的控制字

位	名　称	说　明
7	S	S = 1, 开始位
6～4	A2～A0	通道选择, 001 为 Y 通道, 101 为 X 通道。010 为 IN3, 110 为 IN4
3	MODE	12/ 8 位转换选择, 0 为 12 位, 1 为 8 位
2	SER/ DFR	1 为 SER 模式, 0 为 DFR 模式
1～0	PD1～PD0	00 为笔中断降功耗方式, 11 为连续工作方式

三、DSP 与 ADS7843 的接口控制

我们采用美国 TI 公司性价比很高的 TMS320F206 作为数字信号处理器(DSP), 用 DSP 的同步串行口与 ADS7843 转换器相连接。DSP 与 ADS7843 的连接如图 2 .15 - 3 所示。DSP 的同步串口有两个 4 字深度的先入先出缓冲器, 突发模式或连续模式进行数据收发。当使用外部时钟时, 具有宽范围的操作速度。如果使用内部时钟, 其速度固定为 DSP 时钟频率的 1/ 2。

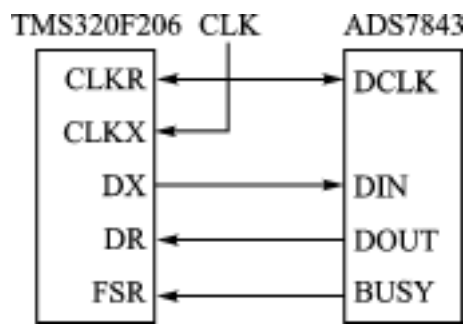


图 2 .15 - 3　DSP 与 ADS7843 的接线图

经测量, 触摸屏 X 方向的转换值为从大到小(X_{max} 至 X_{min}), Y 方向的转换值为从小到大(Y_{min} 至 Y_{max})。触摸屏 X、Y 方向的转换值必须与 320 × 240 的液晶显示相对应, 因此 X、Y 方向的转换值必须按下式计算:

$$y = (y - Y_{min}) \times 320 / (Y_{max} - Y_{min})$$

$$x = (X_{max} - x) \times 240 / (X_{max} - X_{min})$$

DSP 同步串口每次发送 1 个字(16 位), 而 ADS7843 的控制字为 8 位, 从图 2 .15 - 4 转换时序可以看出 DIN 的后 8 位是零。因此, Y 值转换的控制字为 # 9300H, X 值转换的控制字为 # 0D300H。BUSY 信号作为 DSP 同步串口的同步信号, BUSY 信号下跳沿启动串口输入。ADS7843 转换器的转换值为 12 位, DSP 同步串口一次接收 1 个字(16 位), 右移 4 位即可得到

转换值。DSP 程序框图如图 2 .15 - 5 所示。

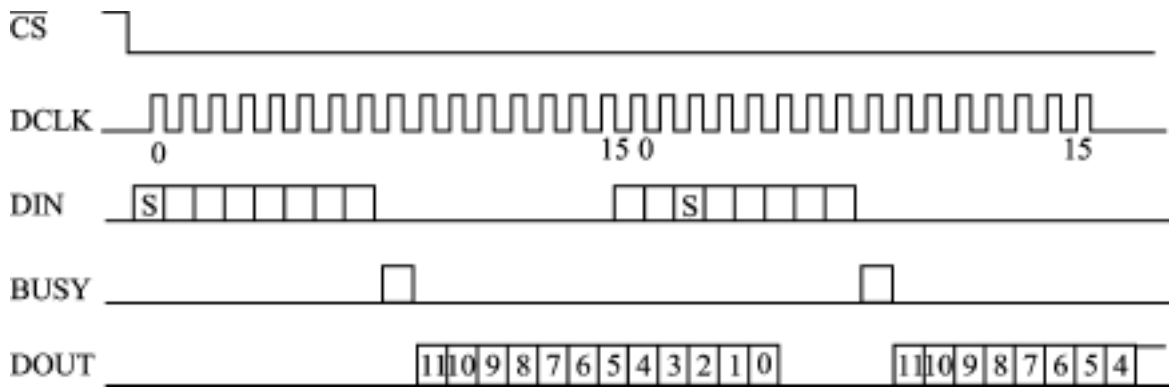


图 2 .15 - 4 16 个时钟转换时序图

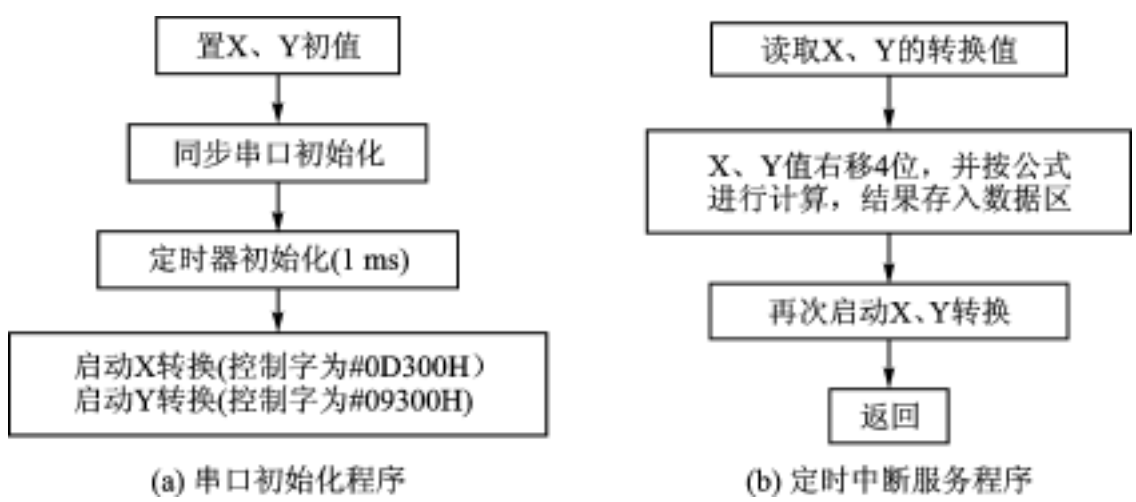


图 2 .15 - 5 DSP 程序框图

四、ADS7843 的笔中断控制

ADS7843 采用笔中断控制时能极大地降低功耗,从 750 μW 到低于 0 5 μW。控制字中最后两位 PD1、PD0 全为零时,ADS7843 进入笔中断降功耗模式。当新的转换开始时,转换器立即恢复正常工作,不需要延时等待电源上升,每次转换都是有效的。ADS7843 进入降功耗模式后,按压触摸屏时 Y - MOSFET 提供电流通道,其他三个 MOSFETS(X + 、X - 、Y +)处于高阻态。电流通过 100 kΩ 电阻和中断二极管从 Y - MOSFET 到地。这时/ PEN 变成低电位,DSP 必须检测到这个信号并发出启动转换命令。在转换期间,DSP 必须维持/ PEN 一直为低。

五、触摸屏与液晶显示的配合

当 PDA 笔按压触摸屏时,ADS7843 的转换值必须在液晶上相应位置正确显示。由于转换器每次开机的转换值略有偏差,因此,为了准确定位,每次开机时重新得到 X、Y 值范围。方法是开机时依次在液晶屏四角显示定位点,将得到的触摸屏的转换值作为基准。

参 考 文 献

1 彭启棕,李玉柏 .DSP 技术 成都:电子科技大学出版社,1997
2 张芳兰 .TMS320C2XX 用户指南 .北京:电子工业出版社,1999

选自《电子技术应用》月刊,2001 年第 1 期

第三章

操作系统与软件技术

3.1 嵌入式系统中的实时操作系统

中国科学院软件研究所(100080)
叶以民 赵会斌 耿增强 李小群 郑良辰 罗从难

所谓“实时”,对于计算机系统而言意味着不但要求逻辑结果正确,而且有时间的要求,即这个结果必须产生截止期限之前。对于实时而言,时间期限的要求是必须得到满足的,但是区分具体应用场合,这种要求的严格程度又有所不同。如果这种要求是绝对的,任何一次不满足就能造成灾难性后果,那就称之为强实时;否则,偶尔的不满足并不足以造成严重后果,是可以接受的,则称为弱实时。相应地,具有这两种特性的操作系统就分别被称为强实时和弱实时操作系统。

在操作系统领域,实时操作系统属于一个很特别的“另类”,绝大多数的人们对此较为陌生。人们常见的是所谓分时系统。最为经典的分时系统当推 UNIX 操作系统,已经具有近百年的历史,广泛应用于研究、教育及商业领域。UNIX 系统的优点在于它的高效率及开放性,在这方面,同为分时系统的 Windows 是望尘莫及的。由于其开放性,人们依托 UNIX 制定了一系列关于开放系统的标准,UNIX 系统逐渐成为开放系统的代名词。近来“窜红”的 Linux 操作系统[2] 是 UNIX 的诸多变种之一,它是“自由软件”,更把 UNIX 的开放性发扬到了极致。分时操作系统的设计目标在于通过对最一般情形的优化,最大限度地有效利用硬件资源,从而达到最大的处理能力,即“吞吐率”;而实时操作系统的设计思想,则与此大相径庭,对于实时系统而言,更重要的是响应时间,而且要保障最坏的情形下的响应时间。

一、实时操作系统的实现方法

由于指导思想南辕北辙,因而,分时系统与实时系统是很难“整合”的。不过,由于分时系统应用广泛,被称为通用系统,在它上面有很好的开发环境、调试工具,并且这些环境与工具为开发者所熟悉,所以人们还是在进行着这种努力。这方面,几乎所有的工作都集中在依托 UNIX 系统对它进行改造,从而实现实时系统上面。增强 UNIX 系统实时性,是一个很早就有的想法,为此,20 世纪 90 年代初人们就制定了相关标准,即 POSIX1003.1b。这个标准规定了“开放系统”对实时应用提供的服务,包括较高精度的时钟,实时调度策略,进程页面驻留等。目前几乎所有 UNIX 类操作系统都实现了这个标准所规定的大部分内容[5]。然而,仅仅实现了 POSIX1003.1b,通用分时系统所获得的实时性是很“弱”的,弱到对很多应用而言几乎没有任何实际意义。

通用分时系统,简单地说就是 UNIX 系统成为实时系统的重要障碍,大致有以下几点[4]:

(1) UNIX 的核心是不可切换的。当一个进程运行于核心态,例如执行系统调用的时候,它将不被换出,直到退出核心态,即从系统调用返回,或因某种原因而阻塞,如等待 I/O 设备完成动作。就是说,这段时间之内是不能进行进程切换的,这样,实时事件的即时响应自然也就无从谈起了。

(2) UNIX 核心中为了保证核心数据的完整性,在进入对关键核心数据结构进行修改的所谓‘临界区’时惯常采用‘关中断’的办法。此时系统无法对中断做出响应。我们知道,非周期的实时进程大多是由中断触发的;对于周期性实时进程而言,也需要调度模块来调度运行,而调度模块的执行恰恰是由时钟中断所触发的。所以频繁的关中断导致实时任务不能被及时调度执行。

(3) 分时系统针对一般情形,为提高资源利用率,提高系统整体处理能力所做的优化,也在响应时间上带来了很多不确定性。如请求分页的内存管理机制使得代码、数据被调入的时间成为不确定的;而磁盘操作的缓冲机制又给读取文件的时间带来不确定性。而在实时系统中,这些不确定因素是不能容忍的。

若基于分时系统建立实时操作系统,必须要解决上述问题。事实上,这方面的工作也是按照这个思路进行的,已有一些比较成功的例子。不过,一般而言,彻底的改造很困难,绝大多数情形是在一个方面,或者几个方面做了不同程度的改造,从而获得不同程度的实时性。但是,这只能得到弱实时性。实际情况也是如此,通过改造通用分时操作系统得到的实时操作系统几乎不可能取得强实时特性。

由于上述方法虽然相对简单易行,但存在相应的局限性。于是人们就想到另一种方法,即从头开始设计一个全新的实时操作系统。目前,林林总总已经出现了不少产品。依据这种设计思想生成的实时操作系统,在达到强实时性方面不存在问题,但通用性差,开发环境简陋。不难看出,两种设计思路各有千秋,使用者应当根据具体需求,做出合理选择。

二、基于 Linux 核心的实时操作系统实例

人们对于实时操作系统的研究几乎与操作系统的出现同时起步,迄今已经取得了长足进步,其应用领域不如分时系统广泛,但也早已有了各种各样的不同产品。有商业化产品,也有非商业化产品。一般而言商品化产品功能比较完善,大多伴随着一整套开发、调试的工具、手段;缺点是‘二次开发’的潜力有限,而且价格大多不菲。非商业化产品一般是由大学或其他研究机构研制出来的,没有盈利目的,因而很多源码开放,其优点是可以免费得到,并且可以此为基础进行更深入的研究开发;其缺点是比较简陋、不够完善等。在商业化产品方面,较有代表性的当推 QNX, VxWorks 及 LynxOS 等,本文将着重介绍非商业化的实时操作系统。

Linux 操作系统近年来突然流行起来,成为 UNIX 类操作系统抗拒 MS Windows 的中坚力量;事实上, Linux 也已经对 MS Windows 构成了威胁,特别是在服务器方面,已经对 Win NT 形成了冲击。随着 Linux 的风行,很多大学、研究机构开始基于 Linux 构建实时操作系统。这样做的好处是:Linux 源码开放,是完全自由的。所以,可以预料,基于 Linux 构建实时操作系统将会成为一种越来越流行的趋向。这里,准备对两个有代表性的基于 Linux 建立的实时操作系统作简单介绍。

1. KURT[6]

首先是 University of Kansas 计算机及电子工程系信息及远程通信技术中心的 KURT (Kansas University Realtime OS)。该技术中心研制 KURT 的目的在于满足他们在 ATM 及多媒体方面的研究工作需求。这些研究工作对操作系统提出了特殊要求,其特殊性在于:既要求有很高的实时性,如与时间相关的 QoS(服务质量)保证,又要求全面的操作系统服务。分时系统无疑可以满足后者,但无法满足前者,即便实现了 POSIX1003.1b,其实时性依然显得

太弱;而专用实时系统可以满足前者,对于后者却又无能为力。因而 KURT 的研制者认为,弱实时或强实时操作系统均不符合他们的要求,而他们的要求兼具两方面的特性,为此专门提出了一个新概念:firm realtime。

KURT 的研制者是通过对 Linux 核心进行改造来实现他们这个所谓的 firm realtime 操作系统的。他们采用的方法比较简洁,没有大动干戈,却基本达到了目的。KURT 对 Linux 核心做了如下两点改动:

(1) 修改了时钟中断机制,在以 X86 为处理器的 PC 上,系统时钟可以提供的最高频率超过了 1 MHz,但 Linux 通过对它的编程,将时钟频率设定为 100 Hz,即时钟中断间隔为 10 ms。对于实时操作系统而言,这种时钟粒度太粗,无法满足实时响应的需求。如何解决这个问题呢?最容易想到的莫过于提高时钟频率了。然而简单地提高时钟频率意味着时钟中断的相应处理过程将占用更多的处理器时间,从而使得整个系统的有效利用率急剧下降,所以这不是一个好办法。KURT 的办法非常巧妙,它改变了时钟中断的固定频率模式,通过重新设定使得时钟得以 μs 为单位在任何需要的时候产生中断。这样,既保证了响应时间,又避免了不必要的开销。

(2) 增加了新的实时调度模块。KURT 核心可以有 3 种调度状态:正常态、实时态及混合态。当处于正常态时,KURT 核心的进程调度机制与常规 Linux 核心无异;实时态时则只有 KURT 实时进程可以被调度;混合态是二者的折衷:当没有实时任务可以被调度时将允许一般进程被调度。表面看来混合态似乎也能保证实时任务被优先调度,其实不然,因为此时核心不可切换的性质依然没有改变,所有一个运行中的一般任务一旦进入核心,则很可能妨碍实时任务的及时调度,所以混合态引入了很大的风险。

经改动之后,KURT 获得了很强的实时性能。这时原则上说调度可以精确到 μs。但考虑到核心内部的开销如时钟管理、进程切换等所耗时间约为 200 μs,所以,真正有意义的最小实时任务周期约为 500 μs。显然,随着处理器能力的提高,这个周期会因之缩短。此外,KURT 并没有对核心做更多的改动,所以不难看出这个标称 firm realtime 的操作系统实质上依然是弱实时操作系统。不过,KURT 的设计者通过精巧的方法,以不大的代价达到了设计目的。

2 . RTLinux[1]

既然 KURT 是一个弱实时操作系统,这样在强实时应用方面,它是无能为力的。而另一个基于 Linux 核心的实时操作系统 RTLinux 恰恰填补了这个空白。基于通用分时系统核心构造强实时操作系统,这听起来不可思议,那么 RTLinux 又是如何实现这一点的呢?RTLinux(realtime Linux)是由 New Mexico Institute of Technology 计算机系研制开发的。

RTLinux 的设计别有机杼。其设计者充分意识到通过改造通用分时系统以实现强实时操作系统困难重重,几乎是不可能;这也就意味着强实时操作系统很难提供通用分时操作系统所能提供的丰富的系统服务和完善的开发环境及调试手段等,而这些都是实时应用本身及其开发工作所希望获得的支持。RTLinux 的设计者认真分析了实时应用的整体特征后发现,绝大多数实时应用其实都可以分成实时和非实时两部分。例如:对于实时数据采样分析而言,其数据采样过程必须是实时的,而分析之后的图表显示却可以是非实时的。而且这两部分分别具有如下特性:实时部分较少需要操作系统支持,甚至可以不需要这样的支持,至少应用程序的设计者可以做到这一点;而非实时部分一般则需用相应的操作系统支持,有这样的支持将会

极大地方便应用的开发。基于这种认识,RTLinux 的设计者便确定了如下设计思想:构造一个简单的强实时内核,而应用的实时部分作为实时进程直接运行在这个强实时内核之上;原来的常规 Linux 核心,这时作为一个优先级最低的任务也为这个实时内核所调度,而应用的非实时部分作为非实时进程运行 Linux 核心之上,从而可以获得 Linux 核心所提供的一切服务。为做到这一点,RTLinux 采用了如下手段:

(1) 对 Linux 核心进行改动,将其与中断控制器隔离,不再允许它任意关中断。核心中所有的中断操作指令都被替换为相应的宏,可以简单地理解为这时关中断、开中断的指令实际上仅仅变更一个中断状态标志的值,而不真正改变中断状态。此时,中断控制器由实时核心来控制。所有的中断首先被实时核心所截获,实时核心首先进行相关的中断处理,然后才把中断"传"给 Linux 核心。这样,Linux 核心的一切活动都无法导致中断被关闭,也就无从影响实时核心的任务调度,从而保证了核心的强实时性;同时,容易看出,这种方法依然维持了 Linux 核心中数据结构的完整性。

(2) 改变时钟中断机制。与 KURT 一样,RTLinux 也需用粒度更细的时钟。事实上,RTLinux 采用的方法与 KURT 是一样的,这里不再重复。

(3) 提供实时调度。人们对实时任务调度问题进行了长期深入的研究,实际上,这是实时操作系统领域内人们投入最多、研究最为透彻的一个方面。目前已经提出了形形色色的调度算法。这些算法适用于不同类型的实时应用场合,各有千秋[3]。但是要在一个具体实时操作系统内部实现一个"万用"的调度模块也不是不可能的,只是效率低。RTLinux 在这方面采用了非常灵活的做法,它把调度模块的编写任务交给了系统的使用者,使用者编制的模块可以很容易地加入到系统之中。这样,使用者可根据自己的不同需求,采用适宜的调度算法。当然,为方便起见,RTLinux 自身也提供了两个实时调度模块。

(4) 实时进程与非实时进程间的通信。如前所述,RTLinux 的全部设计思想基于实时应用的划分。这里,一个实时应用被划分成了一个运行于实时核心之上的实时进程及运行于 Linux 核心上的分时进程;于是,实时进程与非实时进程间的通信就成了一个必须解决的问题。RTLinux 通过建立特殊的命名管道来解决这个问题的。这样便在实时进程与非实时进程之间建立了数据传输机制。

理论上说,RTLinux 提供了与 KURT 一样的时钟粒度。但是,考虑到 RTLinux 中实时进程是运行在核心空间中的,这样进程切换的开销远比常规的进程切换的开销要小得多。所以 RTLinux 上有意义的最小的实时进程周期可以达到 100 μs。

从某种意义上说,KURT 与 RTLinux 的出发点都在于整合常规分时系统核心与强实时特性,只是 KURT 侧重于前者,而 RTLinux 着力于后者;它们也分别达到了自己的目的,各自实现了弱实时及强实时操作系统,以满足各自不同的应用需求。二者的设计思想都比较巧妙,在不同的方面也表现得非常优异。使用者可以结合它们的不同特点,结合自己的需要在这两个优秀的实时操作系统之中做出选择。笔者对以上两个系统都进行过较为全面深入的剖析与研究,并予以改进和增强。航天部某研究所与笔者在开发仿真控制计算机系统方面进行了友好合作。在这个仿真控制计算机系统中,选择了基于 RTLinux 的经过改进、增强的强实时操作系统,从而以比国外进口相应产品低一二个数量级的价格,在功能、性能方面达到乃至超过了他们的水平。

参 考 文 献

1 ichael, Barabanov, Victor Yodaiken . Introducing Real-time Linux . Linux Journal, Issue 34, 1997

2 Michael Bech, et al . Linux Kernel Internals, 2nd Edition . Addison-Wesley, 1998

3 Alan Burns . Scheduling hard real-time systems: A review . Software Engineering Journal, 1991, 6(3): 116～128

4 Borko Furht, et al . Real-time UNIX systems: design and application guide . Kluwer Academic Publishers Group, Norwell, MA, USA, 1991

5 Marshall K .McKusick, et al . The Design and Implementation of the 4 .4BSD Operating System . Addison-Wesley, 1996

6 Balaji, Srinivasan . A Firm Real-Time System Implementation Using Commercial Off-The-Shelf Hardware and Free Software . Technical Report 11510-02, Information and Telecommunication Technology Center, University of Kansas, 1998

选自《测控技术》月刊,2000 年第 4 期

3.2 嵌入式系统的开发利器——Windows CE 操作系统

北京航空航天大学电子工程系(100083) 彭 飞 柳重堪 张其善

一、Windows CE 系统简介

Windows CE(以下简称 CE)是美国微软公司专门为各种移动和便携电子设备、个人信息产品、消费类电子产品、嵌入式应用系统等非台式或笔记本电脑领域设计的一种 32 位高性能操作系统。它具有一个简捷、高效的完全抢先式多任务操作核心,支持强大的通信和图形显示功能,能够适应广泛的系统需求。CE 操作系统的主要特点包括:

· 兼容于微软公司的视窗(Windows)PC 电脑操作系统,支持超过 1 000 个常用的 32 位视窗应用程序接口函数(Win32 API),支持高分辨率真彩色显示,为应用软件提供了强大的运行平台。

· 对硬件没有任何特殊要求,允许系统设计者根据所开发产品的要求自由选择硬件,同时提供最广泛的硬件设备支持,包括通信接口、显示和打印设备、输入输出设备、音频设备、网络和存储设备等。

· 支持多达数十种不同的 32 位微处理器芯片,包括 Intel 和 AMD 公司的 X86 系列、摩托罗拉公司的 PowerPC、日立公司的 SH3 系列、东芝公司的 MIPS 系统以及 PHILIPS、NEC 公司的处理器产品等。

· 采用模块化结构,配置灵活,运行时仅需很少的存储器(RAM)资源,并且是目前惟一的可以从 ROM(只读存储器)中直接启动的 32 位操作系统,能够满足具有严格硬件资源限制的系统要求。

由于其本身具有的出色性能,CE 系统自 1996 年底面世之后,迅速在国外最新一代的工业和家用电子设备中得到了广泛应用。在美国,仅基于 CE 系统的掌上电脑产品销量就已超过了 200 万台。本文将从系统结构、硬件要求、设计开发和应用领域等 4 个方面对 CE 系统作全面的介绍。

二、Windows CE 的系统结构

CE 操作系统由一些独立的模块组成,每一个模块提供特定的系统功能,大的模块又可以分成为几个组件。这种组件式结构能使 CE 系统变得非常紧凑,仅需使用很少的硬件资源就可运行。最主要的系统模块有 4 个,分述如下。

1.内 核

内核是整个操作系统的核心部分,它负责最基本的操作系统功能,包括内存管理、进程管理和必需的文件管理。CE 系统的内核继承了微软公司视窗操作系统的大部分出色性能,能够提供虚拟内存管理、进程调度、多任务管理、多线程管理以及中断处理、异常处理等系统级服务。CE 系统强大的抢先式多任务处理机制允许多达 32 个独立的应用程序(即所谓的进程)同

时运行,而多线程机制又支持每个进程拥有自己下属的多个运行分支(即所谓的线程)。此外,CE 系统还允许每个进程拥有不同的优先级,优先级高的进程可以比优先级低的进程拥有更高的系统资源使用权限。这种多任务特性使 CE 系统非常适用于需要实时处理功能的嵌入式系统,它使应用程序能够同时运行几个分支以处理不同的突发任务。

2.持久性存储模块

持久性存储模块通过对 CE 系统中对象存储(object store)功能的支持,为用户和应用程序提供数据存储服务。对象存储包含三种类型:文件系统、CE 数据库和系统注册表。文件系统使用文件分配表(FAT)来管理用户安装或创造的可执行程序或数据文件,最多支持 9 个 FAT 分区,并具有镜像处理和安全功能,能预防在掉电或其他紧急情况下丢失数据。数据库提供结构化的数据存储和访问功能。系统注册表用于存储应用程序需要快速访问的系统配置数据以及其他信息。

3.绘图、窗口、事件子系统(简称 GWES)

GWES 是用户、应用程序和操作系统之间的图形化操作界面,负责提供系统与用户之间的人机交互接口。GWES 处理输入的方式是将按键、手写式输入装置的移动、菜单或按钮等控件的选择等用户输入事件转换为消息,然后再传递给应用程序;处理输出的方式是在显示设备或打印机上显示或打印窗口、图形、文本等信息。GWES 的核心是窗口,所有的应用程序都需要通过窗口从操作系统中接收信息,而所有的输入事件都要经操作系统翻译为消息后通过窗口传递给应用程序。图 3.2-1 为 GWES 的结构。

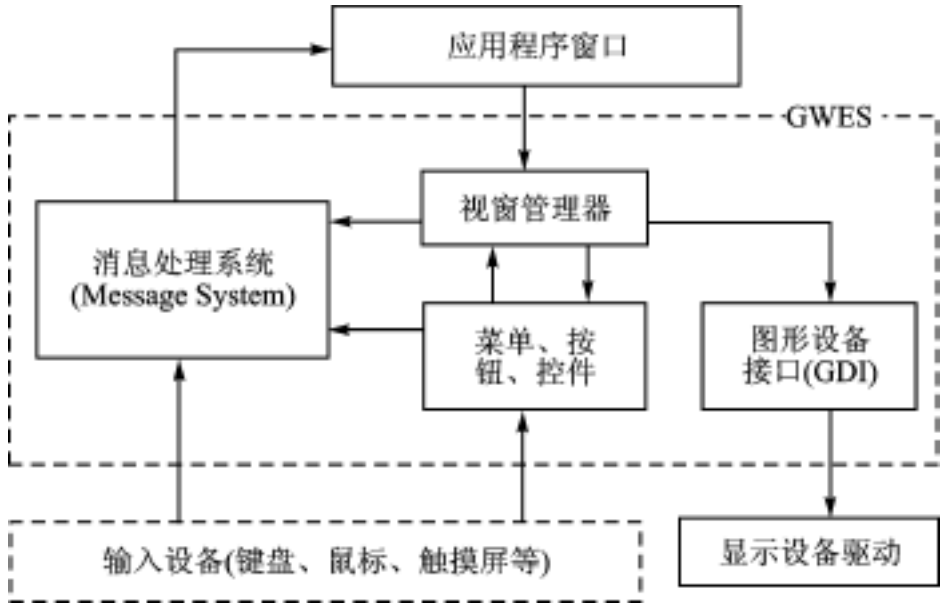


图 3.2-1　GWES 结构示意图

4.通信模块

通信模块为运行 CE 系统的设备提供对多种通信硬件和数据传输协议的支持,包括串/并行数据端口、红外数据端口、电话应用程序接口(TAPI)以及网络通信协议。CE 系统的网络功能十分强大,支持局域网(LAN)、传输控制协议/Internet 协议(TCP/IP)、远程访问服务(RAS)、网间控制报文协议(ICMP)、超文本传输协议(HTTP)、文件传输协议(FTP)等所有流行的网络协议。

以上介绍的 4 个模块提供了最主要的操作系统功能。降此之外,CE 系统还有其他一些可选择的模块,包括:设备管理器和可安装型设备驱动模块、多媒体(声音)支持模块、组件对象模型(COM)支持模块、系统外壳(Shell)等。对于 CE 系统而言,每一个模块都提供一种完整的

功能。在最终的应用系统中,这些功能不一定都会被使用。CE 系统不要求用户包含所有的功能,如果某项功能不需要的话,整个模块就可以被省略。

这种基于模块和组件的结构使 CE 系统具有了高度的可伸缩性,它允许用户在设计自己的操作系统时自由选择需要的模块或组件。这样,系统开发者就能够方便地根据具体的硬件要求定制出合适的操作系统,使其中仅包含设备运行所必须的功能模块,从而不必为支持其他不需要的硬件或功能付出额外代价。通过采用这种方式,CE 系统在支持目前市场上绝大多数硬件设备及接口标准的同时又保证了最大的灵活性和系统效率,因此能够适应广泛的用户系统的要求。

三、Windows CE 系统的硬件要求

CE 操作系统不需要任何特定的硬件结构,实际的硬件系统完全由用户根据需要自由设计。从理论上说,一台以 CE 系统为基础的设备必须有一个处理器,有内存及内部时钟以便处理进度,除此之外再没有其他的硬件要求。不过在实际应用中,每个系统都需要有周边设备才能完全其功能。如前所述,CE 系统提供了广泛的硬件兼容性,支持几乎所有的外围设备、其他设备和网络设备。为了进一步减少对内存资源的需求,CE 系统在采用模块化设计的同时还大量使用了动态链接库(Dynamic-Link Library),许多功能函数都包含在动态链接库中,仅当应用程序需要时才调入内存,平时则以文件形式保存在存储器内。

作为一个小而灵活的操作系统,以 CE 为基础的系统所需内存大小完全依赖于系统选用的模块和组件。表 3.2-1 列出了几种有代表性的基于 CE 的操作系统配置以及对存储器资源的要求。其中,MinInput 表求最小输入采集系统,MinGDI 表示最小图形系统,MinComm 表示最小通信系统,H/PCUI 表求手持电脑用户操作系统。表格中,ROM 栏表求系统需要的永久存储空间,RAM 栏表求系统运行时占用的系统内存,STACK 栏表求系统运行时占用的数据存储空间。

表 3.2-1 几种基于 CE 的操作系统配置与资源占用情况

系 统	包含模块	RAM/KB	ROM/KB	STACK/KB
MinInput	内核,最小文件系统,基本输入设备支持	27	352	2
MinGDI	内核,文件系统,基本图形 GWES,输入设备支持	711	670	19
MinComm	内核,文件系统,通信模块,输入设备支持	119	1 103	14
H/PCUI	内核,文件系统,注册表,GWES,通信模块,输入设备支持,图形外壳	857	1 806	35

可以看出,相对于它强大的功能,CE 操作系统对于硬件资源的要求非常低。一个只包含内核、通信端口,不需要显示功能的基本系统仅需不到 400 KB 的 ROM 和 30 KB 的 RAM 即可运行,而一个完整配置的掌上电脑操作系统也只需大约 2 MB 的 ROM 和 1 MB 的 RAM。当然,最终的系统配置还必须考虑应用程序运行和用户数据存储的需要。

四、基于 Windows CE 的应用系统开发与设计

设计一个运行 CE 的嵌入式系统需要经过以下步骤:

(1) 明确设计目的,确认系统功能,选择合适的微处理器芯片和周边设备;

(2) 完成硬件平台的设计;

(3) 选择 CE 功能模块,定制操作系统,并改编部分代码使之与硬件平台相配合;

(4) 编写应用软件。

典型的基于 CE 的嵌入式系统结构如图 3.2-2 所示。其中,设备管理器提供对可安装设备的支持,允许在系统中安装诸如 PC 卡存储器和调制解调器之类的设备以扩充功能;附加技术模块指由 CE 操作系统提供的一些可选择的专用功能模块,如 JAVA 语言(一种网络编程语言)支持模块、手写笔输入识别模块等;外壳模块为系统提供一个与 PC 电脑上的视窗操作系统类似的图形化操作界面。

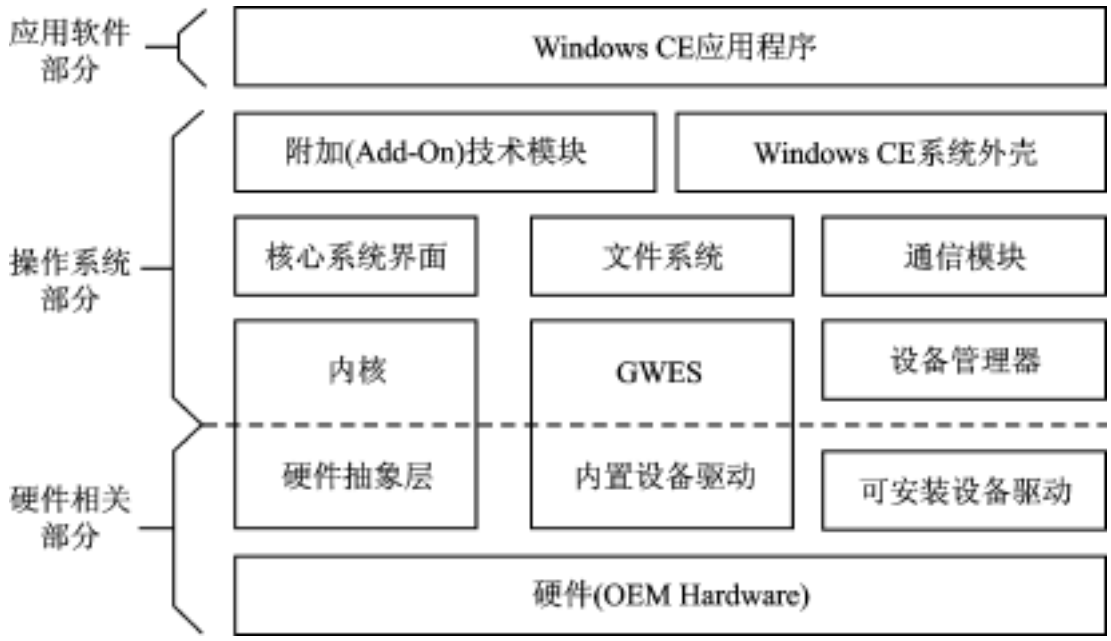


图 3.2-2　典型的基于 CE 的嵌入式系统结构

对于系统设计者而言,需要自行开发的是应用程序和直接与硬件有关的部分,包括硬件系统本身、硬件抽象层和设备驱动程序。其中,硬件抽象层是指建立在硬件设备与系统内核之间的一层代码,主要任务是为内核管理具体硬件设备的时钟、中断和实施电源管理提供支持。设备驱动程序负责支持操作系统对目标硬件的访问。硬件抽象层和驱动程序都需要针对具体的硬件设备编写。一般硬件设备制造厂商会为设备提供驱动软件支持,如果有专门为 CE 系统编写的驱动程序,只需将其加入到操作系统的相应模块中就可使用。如果没有,则应根据厂商提供的驱动程序开发包编写。通常这些开发包中已经包含了完成各种硬件操作的标准代码,开发者的工作只是将其与相应的 CE 系统 API 函数对应起来。在 CE 系统中,所有涉及硬件的操作都通过调用相应的 API 接口函数来完成,而硬件抽象层和驱动程序就是要为这些 API 函数提供支持,将其翻译为直接对目标硬件进行的底层操作。通过这种方式,CE 系统将应用程序与具体的硬件设备进行了隔离,应用程序只需调用 API 函数就可以实现对硬件的访问。这样,程序员在开发应用程序时就没有必要编写任何直接针对硬件的代码,因此不必考虑具体的硬件特性,而应用程序也具有了不依赖于具体硬件设备的独立性。

微软公司提供了专门的 Windows CE 平台编辑器软件来帮助开发者完成定制操作系统和编写硬件抽象层的工作。该编辑器是一个可以在 PC 电脑上运行的工具软件包,其中包含完整的带有集成开发工具的 Windows CE 操作系统,并附带了编程指南、应用程序接口(API)参考等资料和设备驱动程序工具箱。用户可以使用其中的操作系统编辑器、汇编编辑器和经过优化的 Visual C++ 编辑器进行开发,并将最终生成的操作系统编译为能够被处理器芯片识

别的二进制代码文件,以便在目标系统中使用。除启动和初始化部分外,CE 系统的其他部分可以直接以文件形式存储。CE 平台编辑器中还带有远程调度工具,只要硬件开发系统支持并留出调试端口,就可以将编辑好的操作系统从 PC 电脑中下载到硬件平台进行调度,并在 PC 电脑上观察系统的运行状态和内存资源使用情况等信息。

由于 CE 操作系统支持 1 000 多个与 Windows 系统兼容的 API 函数,使在基于 CE 的嵌入式系统中运行的程序可以拥有与 PC 电脑中的程序同样复杂和强大的功能。微软公司提供了专门用于编写 CE 程序的 Visual C + + 和 Visual Basic 附件工具包以及 CE 平台模拟器,让程序员能够直接利用这两种熟悉的编程语言在 PC 电脑上编写和调试 CE 系统中的程序。此外,尽管编写 CE 系统的应用程序与编写 PC 平台的 32 位 Windows 应用程序有一些区别,但其中进程、线程、视窗的概念以及 API 函数、资源、控件都与 Windows 系统非常相似,因此任何具有 Windows 编程经验的程序员都能够很容易地掌握 CE 编程。

五、Windows CE 系统的应用领域

目前国外已经有大量基于 Windows CE 系统的设备面市,所涉及的领域包括家用电器、娱乐设备、专用计算机系统、便携电脑以及个人通信产品等。微软公司于 1999 年 3 月在中国深圳提出了维纳斯计划,其核心内容就是推广基于 Windows CE 系统的信息家电产品。目前已经有许多符合这一概念的产品面世,如美国国家半导体公司生产的便携式网络终端 Web-Pad,能够随时与互联网连接,进行网络浏览和收发电子邮件。类似的产品还包括网络机顶盒和网络电话。在家庭娱乐市场方面,日本世嘉公司于 1998 年底推出了运行 CE 系统的 Dreamcast 电视游戏机,其拥有的强大的处理器运算能力和图形功能甚至超过了桌面电脑产品,除进行游戏之外还能播放 DVD、浏览互联网或运行基于 CE 系统的应用程序。在专用计算机系统方面,使用 CE 系统的有打包和邮件跟踪设备、超市收银机终端、数据采集设备、车载导航电子设备以及工业控制系统等。在便携电脑市场方面,运行 CE 系统的设备包括掌上电脑、个人商务通产品和车载移动电脑系统。掌上电脑是最近几年兴起的新型便携信息产品,一般都配备笔输入装置,具有文字处理、电子记事和网络浏览功能,可以与桌面电脑进行对接交换数据,而质量不到传统笔记本电脑的一半。在国外,流行的掌上电脑操作系统有 Palm OS 和 Windows CE 两种,而国内则主要是中文 Windows CE。常见的惠普、联想等公司的产品都使用了最新的中文 CE2 .11 版本,其中还引入了汉王笔输入识别技术和金山词霸英汉双向翻译系统。在个人通信市场方面,已经有大量内置 CE 系统、能够进行网络访问和收发电子邮件的移动电话产品出现。

总结以上介绍可以得出结论,由于 CE 操作系统具有强大的功能、广泛的兼容性、灵活的适用性、最小的硬件资源要求和完善的开发工具支持,因此是设计嵌入式系统的理想选择。21 世纪将是人类社会全面走向网络化、信息化的时代,电子产品小型化、智能化、网络化的趋势已不可避免。面对这一浪潮,将先进的操作系统性能和强大的通信能力高度集成在一起的 CE 系统无疑具有美好的应用前景。

参 考 文 献

1 ranklin Fite Jr .Embedded Development with Microsoft Windows CE 2 .0 .Microsoft Developer Network Library, 1997,9

2 John Murray . Microsoft Windows CE Memory Use . Microsoft Developer Network Library, 1997,9

3 Jason Black, Jon christianson . Microsoft Windows CE Display Driver and Hardware . Microsoft Developer Network Library, 1997,9

4 Douglas Boling . Programming Windows CE . Microsoft Press, 1998,10

选自《电子技术应用》月刊,2000 年第 9 期

3.3 介绍一种实时操作系统 DSP/ BIOS

北京建国门外大街 19 号国际大厦 7 楼 A-D(10004)
德州仪器(中国)有限公司应用工程师

德州仪器新推出的 DSPC 6400 系列最高运行时钟可以达到 1.1 GHz,运算能力可以达到 8 800 MIPS。如何充分发挥 DSP 的这种性能优势,对软件提出了很高的要求。首先为了降低系统成本,就要求将许多以前用硬件实现的功能软件化,原来由多个 DSP 完成的工作由一块 DSP 完成,DSP 要能够同时完成多个相同或不同的任务而相互之间没有影响。其次为了产品的维护和升级,要求软件尽可能地模块化,使用高级语言如 C 来编程,有统一的接口 API。所有这些新的要求,都需要使用实时操作系统。以往直接将应用程序运行于裸机之上的作法显然已经不再适用了。德州仪器推出的 DSP/ BIOS 技术正是基于其多年从事 DSP 研制生产的经验,为开发者量身定做的一个优秀软件平台。更可贵的是该操作系统是免费的,这为众多的中小用户使用 DSP 打开了方便之门。

一、实时操作系统

简单地说,实时操作系统与一般意义上的操作系统(如 Windows, Unix 等)的主要差别就在于实时操作系统提供了一种机制,使得运行于其上的应用程序都能够满足实时性的要求。在 Windows 中常见的沙漏现象(用户等待现象)在实时系统中是绝对不允许的。因为这可能造成通信中断、马达损毁等灾难性的结果。DSP/ BIOS 是特别针对实时系统,运行于德州仪器 C5000、C6000 系列的 DSP 之上的一个实时操作系统。DSP/ BIOS 实际上是一个可调用的系统模块 API 的集合。以下就各个模块分别加以介绍。

二、LOG

在开发的时候通常需要使用 printf()来显示当前状态。但是 printf()是非常花费时间的函数,而且不具有实时性。因为 DSP 需要对显示的数据进行分析,整理成合适的显示格式,并调用输出显示模块。所以在一个实时性要求很高的应用中,对 printf()的调用可能会使系统根本无法满足实时要求。在 DSP/ BIOS 中引入了一个相应的函数 LOG_printf()。该函数是 LOG 对象的一个方法(或调用函数)。LOG 对象本质上是一个 32 位的整形数,其高低 16 位分别代表要显示的两个数据。例 1 是分别使用 printf()和 LOG_printf()作比较的示范程序:

```
#include <stdio.h>

/* Header files needed for DSP BIOS */
#include <std.h>
```

```
#include <log.h>

/* functions */
void func_printf();
void func_LOG_printf();

/* Objects created by the Configuration Tool */
extern LOG_Obj logTrace;

/*
 * ========  main  ========
 */
void main()
{
  return;
}

void func_printf(int time)
{
  printf("Strart printf demo\n");
  printf("Current time = %d\n",time);
  printf("End printf demo\n");
  return;
}

void func_LOG_printf(int time)
{
  LOG_printf(&logTrace, "Strart LOG_printf demo\n");
  LOG_printf(&logTrace, "Current time = %d\n"time);
  LOG_printf(&logTrace, "End LOG_printf demo\n");
  return;
}
```

func_printf()和 func_LOG_printf()由 DSP 内时钟控制每 100 ms 周期性地分别调用一次。通过对 printf()和 LOG_printf()运行时间作比较发现,在 C6211 运行在 150 MHz 的情况下,printf()需花费 4 000 个周期约 26.7 μs, LOG_printf()只花费 36 个周期约 0.24 μs。printf()比 LOG_printf()多开销 100 倍以上的时间,因此 LOG_printf()对于实时地显示一些运行状态是非常有帮助的。而且对于熟悉 C 语言的开发者来说,LOG_printf()的调用格式几乎与 printf()完全一样。

三、STS

对一个软件进行分析优化时,通常会用到 profile 的功能。但是在实时运行的 DSP 的环

境中使用 profile 等效于加入了多个程序断点。由于现在的 DSP 通常具有很深的流水线结构来保证 DSP 的高运算能力,如德州仪器的 C6000 系列的流水线长度为 12 级,程序断点需要排空所有已经进入流水线的指令。这样也就破坏了真正的运行环境。同时 profile 还必须调用输出模块向主机传递时间信息。因此在 profile 的情况下真正的实时运行环境是没有办法得到保护的。DSP/ BIOS 针对这种情况引入了一个统计模块 STS。STS 对象只有 4 个数据 Previous、Count、Total 和 Max。调用的方法(API)也只有 4 个,即 STS_add()、STS_set()、STS_delta()和 STS_reset()。这些 API 对数据的操作功能如表 3.3-1 所列。

表 3.3-1 STS 模块方法(API)的功能

	STS_add(x)	STS_set(y)	STS_delta(z)	STS_reset()
Previous		y	z	
Count	+1		+1	0
Total	+x		+(z - Previous)	0
Max	If x > Max, Max = x		If z - Previous > Max,最大负整数 Max = z - Previous	

如果要对某一段程序进行分析时,只需在其前后调用 STS_set 和 STS_delta 就可以了。如例 2 使用 STS 测试程序段执行周期如下:

```
/ * Header files needed for DSP BIOS */
 # include < sts .h >
 # include < clk .h >

/ * functions */
 void func_load();

/ * Objects created by the Configuration Tool */
 extern STS_Obj stsLoad;

/ *
 * = = = = = = = = main = = = = = = = =
 */
   void main()
 {

/ * fall into DSP/ BIOS idle loop */
   return;
 }

 void func_load()
 {
   STS_set( &stsPrintf, CLK_gethtime());
```

```
/ * 测试程序段 */
…
  STS_delta(&stsPrintf, CLK_gethtime());
}
```

func_load()为一个中断服务程序(ISR)。在 C6211, 150 MHz 的情况下,仅插入 33 个周期,约 0 .22 μs。

四、任务调度(HWI/ SWI/ TSK)

一个操作系统的核心永远都是任务的调度。在 DSP/ BIOS 中任务的调度是通过 HWI、SWI 和 TSK 三个模块来实现的。这三个模块分别对应于不同的调度方法。HWI 即硬件中断。在 DSP/ BIOS 中硬件中断主要负责从外部设备中读写数据。由于硬件中断直接与硬件打交道,所以对应的中断服务程序 ISR 应该尽可能地短小精悍。需要注意的是 HWI 并不引起任务调度,因此在 ISR 的入口和出口成对地调用_HWI_enter()和_HWI_exit()这两个宏是必须的。HWI 在处理完数据的输入输出后调用 SWI_post()来调度相应的软件中断,SWI 来完成数据处理工作。SWI 是 DSP/ BIOS 任务调度的核心,共有 14 个优先级,每个优先级可以有多个任务。SWI 任务是抢断式的,即高优先级的任务可以抢断低优先级的任务。但是 SWI 任务是不可阻塞的。它的运行状态如图 3 .3 - 1 所示。

所有 SWI 任务共享一个堆栈,SWI 任务只能在程序编制时预先定义好。DSP/ BIOS 中对任务的动态产生和对阻塞状态的支持是通过 TSK 模块来实现的。TSK 有 15 个优先级,也是可以抢断的,但是每个 TSK 任务使用独立的堆栈。TSK 任务是通过 TSK_create()和 TSK_delete()来动态生成和结束的。它的运行状态如图 3 .3 - 2 所示。

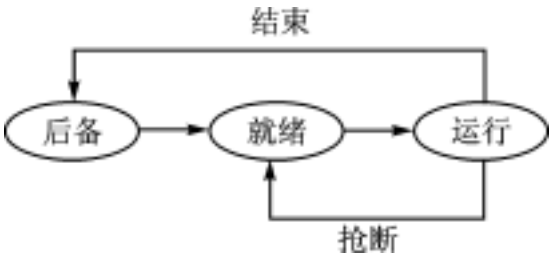


图 3 .3 - 1 SWI 的运行状态

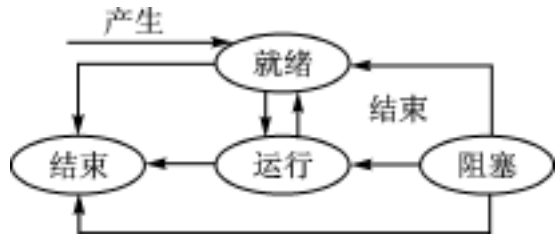


图 3 .3 - 2 TSK 的运行状态

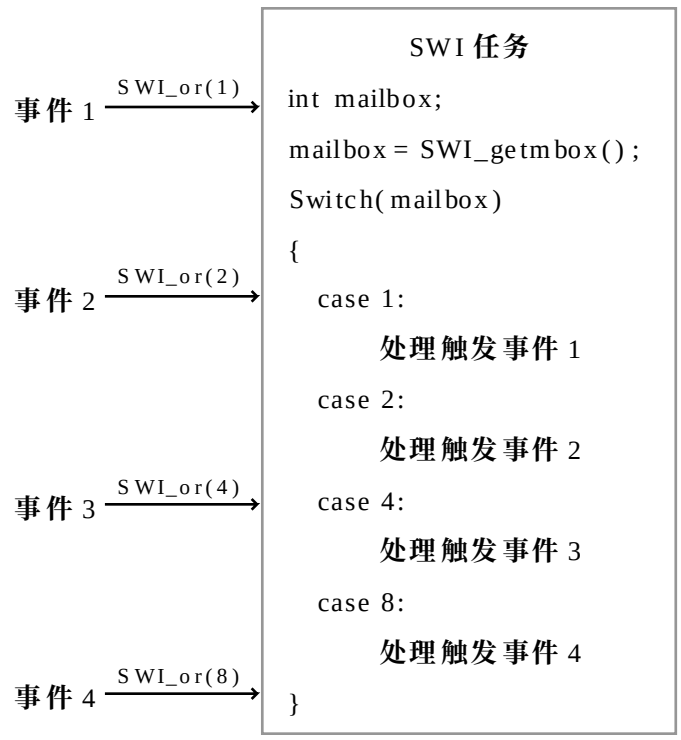
五、同步(SEM/ ATM/ QUE/ MBX)

多任务系统中多个任务之间的协调同步工作可以通过多种方法来实现。常用方法如信号量、原子量、队列和邮箱等。在 DSP/ BIOS 中对这些方法的支持分别通过模块 SEM、ATM、QUE 和 MBX 来实现。由于这些方法的使用与一般的操作系统完全一样,在这里就不再赘述了。仅就最灵活的在 SWI 中使用 Mailbox 的方法来加以简单地说明。每个 SWI 任务都带有一个 Mailbox,对它的操作可以是计数型的 SWI_inc()、SWI_dec();也可以是比特位操作型的 SWI_or()、SWI_andn()。Mailbox 控制 SWI 任务被调度的条件。这些操作的功能如表 3 .3 - 2所列。

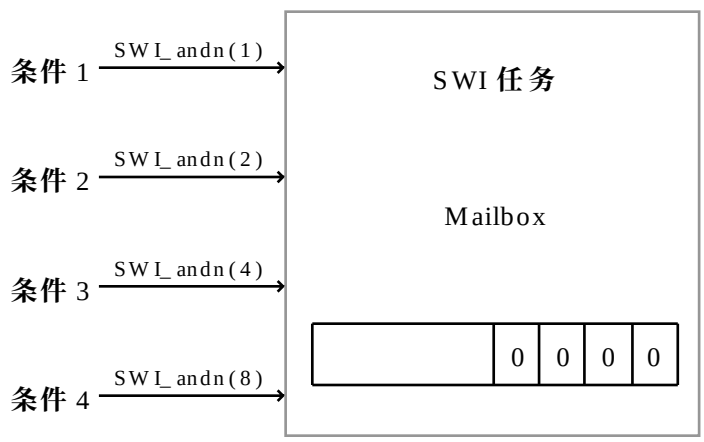
表 3.3-2 SWI 中 Mailbox 的操作功能

调 度	Mailbox 是比特操作	Mailbox 是计数操作
总是引起任务调度	SWI_or	SWI_inc
仅在 Mailbox = 0 时引起任务调度	SWI_andn	SWI_dec

or 操作是将 Mailbox 中的某一位置 1,同时引起 SWI 任务的调度。当一个 SWI 任务可能由多个事件触发时,使用 or 操作可以方便地表示出触发的事件。如例 3 使用 or 操作指示触发事件:



andn 操作是将 Mailbox 中的某一位清 0,如果 Mailbox 为 0,则引起 SWI 任务的调度。一个 SWI 任务需要多个条件都满足时才运行的情况下,使用 andn 操作可以方便地表示出这些条件的状态。如例 4 用 andn 操作来表示多条件时 SWI 任务调度:



inc 和 dec 操作则更加灵活,用户可以借此实现多种应用。惟一需要注意的是,inc 操作总是引起任务调度,而 dec 操作仅在 Mailbox 减到 0 时才引起一次任务调度。

六、通信(PIP/ SIO/ HST)

一个系统如何从外部设备中取得数据,向外部设备输出数据,如何在两个任务之间进行数据正常交换是多样灵活的。但是这种多样性也给软件的维护升级以及模块化工作带来许多不

利因素。因此在保持多样性的同时,保持接口的一致性对于一个软件来说是非常有帮助的。考虑到 DSP 大多数是通过某种类型的串行接口如中继线 E1、IIS、SPI、同步串行口等与外部设备进行数据交换的,所以在 DSP/ BIOS 中提供了两种非常有用的接口对象 PIP 和 SIO。

PIP 对象包含一个缓冲队列,与之对应的有两个任务读和写。图 3 .3 - 3 很好地说明了 PIP 的逻辑关系和操作方式。例 5、例 6 分别是一个 PIP 对象对应的读任务和写任务的示范程序。

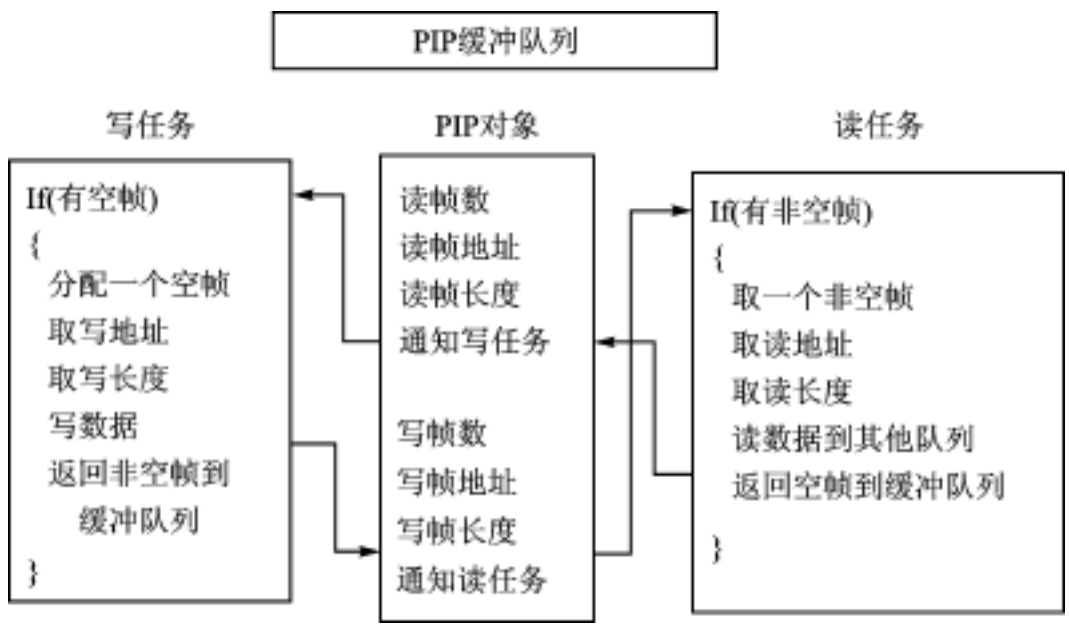


图 3 3 - 3 PIP 对象逻辑关系和操作方式

例 5 是 PIP 对应的读任务:

```
extern far PIP_Obj pip;
reader()
{
    Uns size;
    Ptr addr;
    if( PIP_getReaderNumFrames( &pip ) > 0)
    {
        PIP_get( &pip) ;
        addr = PIP_getReaderAddr( &pip) ;
        size = PIP_getReaderSize( &pip) ;

        / * Code to empty the frame */

        PIP_free( &pip) ;
    }
    else{
        LOG_error( "no frames available") ;
        / * or you could just return; */
    }
}
```

例 6 是 PIP 对应的写任务:

```
extern far PIP_Obj pip;

writer()
{
    Uns size;
    Ptr addr;
    if (PIP_getWriterNumFrames(&pip) > 0){

        PIP_alloc(&pip);
        addr = PIP_getWriterAddr(&pip);
        sixe = PIP_getWriterSize(&pip);

       /* fill the frame up to size */
        PIP_put(&pip);
    }
    else{
            LOG_error("no frames available");
           /* or you could just return; */
    }
}
```

由逻辑关系可以看到,通过使用 PIP 应用程序可以保持一个简单统一接口而不必关心具体的硬件操作,因此当该软件移植到不同环境中时,至多只需要改写设备驱动程序。使用 PIP 的一个具体实例就是 HST 模块。HST 模块在主机和 DSP 之间建立起一条数据链路,该链路就是一个 PIP 对象。对 HST 的操作方式与 PIP 一致。其差别仅仅在于 HST 在初始化时指向了预定义的 DSP 上的 HPI 接口而已。

SIO:从 PIP 的逻辑关系可以看出,读写 PIP 就是一个数据拷贝的过程。这在某些应用中,如实现网络协议 TCP/IP 时,不是非常有效。因为数据每向上传递一层就需要进行一次数据拷贝,其效率非常差。如果采用 SIO 来实现就会有很大的改善。SIO 的操作只有 get()和 put 两种。与 PIP 不同的是 SIO 没有自己的缓冲队列。每次 get()或 put()操作时都会在应用程序和设备驱动程序之间交换缓冲的指针。所以 SIO 操作的实质是数据地址的交换。由于没有数据拷贝,其运行效率就很高。SIO 的运行逻辑如图 3.3-4 所示。

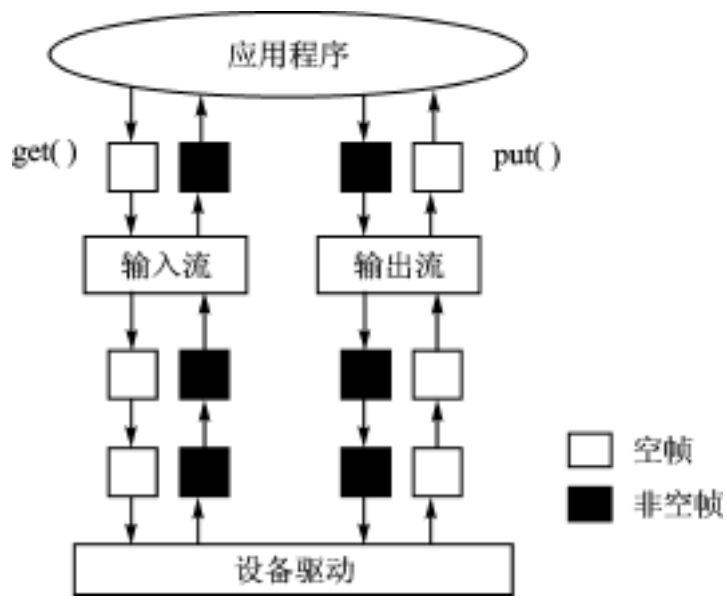


图 3.3-4 SIO 中 get()和 put()操作的逻辑关系

七、RTDX

实时数据交换 Real-Time-Data-eXchange 是 DSP/ BIOS 提供的一个全新的功能。在很多应用中要求 DSP 不能够停下来,而需要从主机中实时地读取数据或者向主机实时地输出数据。德州仪器的 C5000,C6000 系列的 DSP 都可以通过 JTAG 接口来实现这个功能。其逻辑结构如图 3.3-5 所示。

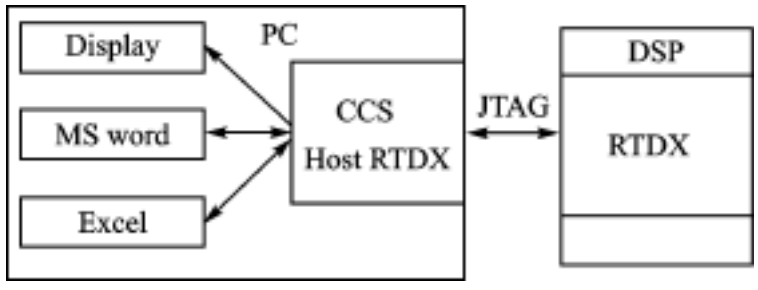


图 3.3-5 RTDX 的逻辑结构图

RTDX 在主机端可以与任何符合 OLE 接口的应用程序交换数据。例 7 是一个使用 RTDX 在主机和 DSP 之间进行数据传递的例子。主机端是一个基于 VB 的小程序。

例 7 DSP 程序:

```
#include<rtdx.h>
RTDX_CreateInputChannel(writeload);
RTDX_CreateOutputChannel(readload);

int main()
{
    RTDX_enableInput(&writeload);
    RTDX_enableOutput(&readload);
    return;
}

void doExchange()
{
    if(!RTDX_channelBusy(&writeload)){
        RTDX_readNB(&writeload, &loadVoal, sizeof(loadVal));
    }

    RTDX_write(&readload, &loadVal, sizeof(loadVal));
}
```

使用 VB 编制的主机端程序:

```
set r = CreateObject("RTDX")
status = r.open("readload","R")
set w = CreateObject("RTDX")
```

```
status = w .open(" writeload "," W ")
status = r .ReadI4(data)
status = w .WriteI4(value, bufferstate)
```

综上所述,DSP/ BIOS 针对 DSP 的应用环境,通过一系列的对象模块向开发者提供了一个实用优秀的实时操作系统。它可以帮助用户提高软件的模块化、并行性和维护性等,有利于降低系统成本和缩短开发周期。同时由于它是免费的,可以预计 DSP/ BIOS 将对 DSP 技术在中国的推广使用起到积极的推动作用。

选自《电子技术应用》月刊,2000 年第 12 期

3.4 实时操作系统用于嵌入式应用系统的设计

南京航空航天大学自动化学院测试工程系(210016)
蒋书波 张焕春 经亚枝 李 焱

我们经常从 ATM 中取钱,使用移动电话或是家用电器,这些设备就是本文所提到的嵌入式系统。每年都有几十亿的微处理器和微控制器销售用于嵌入式系统,这其中大部分用于消费类电子产品。那么嵌入式系统是如何定义的呢?嵌入式系统通常是指操作系统和功能软件集成于计算机硬件系统之中,简单地说就是系统的应用软件与系统的硬件一体化,具有软件代码小、高度自动化、响应速度快等特点,特别适合于要求实时的和多任务的系统。由此可以看出嵌入式系统的特性,是利用有限的资源来完成一系列预先定义或特定的功能。

随着竞争的日渐激烈,决定了对开发工具的需求是嵌入式市场的一个重要的方面。嵌入式系统开发工具的重要环节是实时操作系统,还包括支持它们编程的工具:源代码调试器、集成开发环境和编译器。任一个实时操作系统不可能为每个应用都提供优化的解决方案,这就是现在的实时操作系统如此之多的原因所在。

一、实时操作系统

实时操作系统(RTOS)是嵌入式应用软件的基础和开发平台。RTOS 是一段嵌入在目标代码中的软件,用户的其他应用程序都建立在 RTOS 之上。不但如此,RTOS 还是一个可靠性和可信性很高的实时内核,将 CPU 时间、中断、I/O、定时器等资源都包装起来,留给用户一个标准的 API,并根据各个任务的优先级,合理地在不同任务之间分配 CPU 时间。

“实时”用在操作系统中是什么意思?首要任务是调度一切可利用的资源完成实时控制任务,其次才着眼于提高计算机系统的使用效率。简单地说就是嵌入式系统在能够响应外部事件的同时,也能完成等待其他任务或进程的操作,重要特点是满足时限的要求。通常又分为“硬实时”和‘软实时’。硬实时是指在特定时间内完成一个事件,通常有十几微秒到几毫秒,例如视频数据流的处理,或是自动引擎上脉冲的处理;软实时应用于那些不要求硬实时的场合,当超过了截止时限也不会破坏系统的完整性,但对系统会产生不利的影响。例如,当 POS 机不能读出代码时,可能是刷卡太快,系统产生错误,可以再次查询识别。无论硬实时还是软实时的操作系统,与桌面或大型机的操作系统相比,区别主要在以下四点。

(1) 受限的中断服务:有一个允许系统转向处理中断的最大时间,中断服务程序必须是绝对最小化的处理方法。

(2) 基于优先级的调度:在实时系统中,所有任务都被安排在某个优先级,这个优先级可能基于一些准则(包括运行时间)。没有执行的任务是因为它们处于等待状态。

(3) 悬挂任务:所有的任务和队列能被一些已经准备就绪的高优先级的任务或队列悬挂。

(4) 可衡量性:操作系统所提供的服务不是单一的,而是作为一系列的调度模块和库。

除了这四点外,在实时和桌面 OS 之间还有其他的区别,就是处理更多终端应用的需要、

嵌入式开发的需要和利用有限资源的需要。重要的是对 RAM 的需求,考虑到大部分嵌入式系统的成本和体积,OS 必须能高效地使用内存,防止产生存储碎片;在任务完成时能够重用内存;当任务创建时用最小的内存,提供有效的堆栈结构。再一个重要的就是调度算法,因为这些是系统性能的核心。由于终端的应用对象不同,开发了各种各样的算法,开发者希望能选择一种算法在最节省资源的情况下满足相应的需要。现已使用的一些算法有:① 启发式调度算法;② 周期调度;③ 固定时间调度;④ 简单优先级调度;⑤ 单调调度;⑥ 截止时限单调调度;⑦ 优先级继承协议的调度。这些调度算法在任务间创建同步和通信机制方面都是很重要的。在非实时操作系统中也能发现这些机制,但由于响应时间和资源有限,使得这些机制在嵌入式系统中占有重要地位。

图 3.4-1 是集成系统公司的 pSOS 系统,一种商用操作系统的结构。pSOS 系统使用了有标准组件的结构,包括 pSOS 实时、多任务内核及其软件组件和库。这些组件被封装成“黑盒子”,不会随着应用系统的改变而改变,确保了终端用户的可靠性。

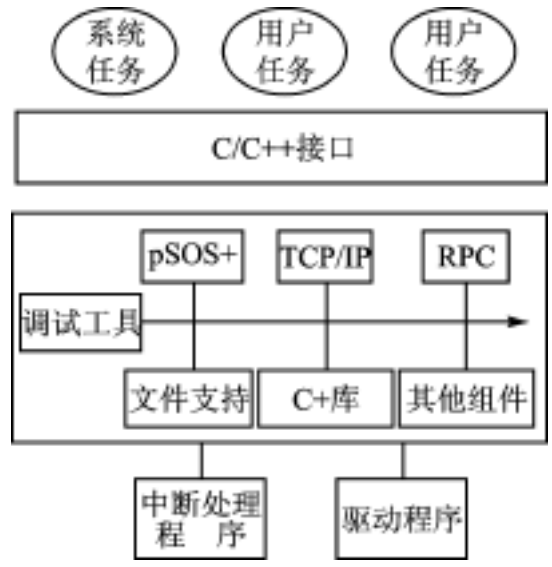


图 3.4-1 pSOS 系统基本结构

二、通用实时操作系统

据《实时》杂志统计,有 200 多种 RTOS 可供选择。把它们分成三大类:

(1) 非商用的 RTOS,主要适用于小型嵌入式系统。但是,这种操作系统支持的微处理器较少,并且提供的支持也很少;

(2) 实时扩展的 NT。它的基本原理与众不同,是在一个平台上运行两个操作系统;

(3) 商用 RTOS,被分成两类:小型嵌入式应用系统和大型复杂的实时应用系统。

据《嵌入式系统》杂志的最新报导,世界各国有 40 多家公司推出各种各样可供嵌入式应用的 RTOS。其中几个著名的操作系统有:Windows CE 既可用于硬实时又可用于软实时,VRTX 老牌实时操作系统,VxWork,曾用于美国火星探测上,pSOS 是世界上最早的实时操作系统之一,QNX 是 X86 上最好的实时操作系统,RT-linux 能够提供实时的 Linux 操作系统等等。而且至今仍有不少的开发者使用自己开发的 OS。我国也自主研制开发了嵌入式实时操作系统。

在“选择一种实时操作系统”的报告中,提出了按照嵌入式系统的工作环境分类(例如,考虑到:是否中断程序要比存储器的管理和队列重要?是否有许多基于时间的功能?),并且根据各种 RTOS 的特点来选择适合自己需要的 RTOS。在另一个报告中分析了是自己做一个 RTOS 还是买一个 RTOS,如果想要买的话,如何选择商用 RTOS 的供应商。这个报告中还讨论了选择的一些准则,如性能、软件组件、设备驱动、调试工具、标准兼容、技术支持和源代码的实用性等等。然而自己设计的 RTOS 也是有自己的好处,可以按照自己应用系统的需要裁减内核及各种驱动,使它更适用于小型嵌入式应用的需要。

三、嵌入式系统设计中的问题

嵌入式系统的开发受到系统响应时间、大小、性能、开销等等因素的制约,要优化或充分利用这些限制因素,这就使得设计一个嵌入式系统是一项比较困难的任务。在设计之前有一系列的问题要考虑。

(1) 对每一个任务来说,它的极差事件性能的需求是什么;

(2) 要被执行的任务的数量和复杂性如何;

(3) 为了使处理器的负载平衡,如何将这些任务分配在软件任务中(这样可以得到更好的性能价格比);

(4) 这些任务的耦合度是什么样(重要的是截止时限、任务间数据流的类型、事件的相关性);

(5) 需要为硬件提供多大的 RAM 和 ROM 空间;

(6) 有多大的 RAM 和 ROM 空间被分配给特殊任务,中断服务请求,队列等等;

(7) 什么时候要分配资源给队列和消息,这些队列和消息如何使用才能避免优先级倒置;

(8) 分配给堆栈多大的缓冲区空间。

对于嵌入式系统的设计者来说,上述问题是我们经常要面对的。下面总结一下设计中遇到的主要问题。

1 . 事件限制

RTOS 是专门为受时间限制的任务而设计的,例如有一些程序必须严格地每隔一段固定时间执行,而其他一些程序则不要求严格地时间安排。嵌入式编程者的重要任务是给那些要执行的每一个事件加以标记,这样它就知道了如何给那些事件安排优先级和资源,使得能够满足整个系统的性能目标。为了辅助这个任务,在嵌入式系统需要中断事件的执行,这样有益于下列的任务执行。

(1) 时间紧急任务程序是那些每隔一段固定时间就要发生,允许带有最小时延(例如 A/ D 转换器)的服务程序;

(2) 时间敏感的任务程序是不同于时间紧急任务程序,因为在被接受中断之前允许大的时延。象时间紧急任务程序一样,它们可能发生在固定时间或可能在任意时间内被初始化,但是保证一定被执行(例如,防抖动开关);

(3) 闲置任务程序是很重要的背景操作,当它们频繁地在随机时间间隔内执行任务时,是很方便的。(例如使用任务级查询 UART 传送缓冲器状态,来决定何时从软件队列传送字符到传送缓冲器);

(4) 干线任务程序中断用户命令,执行非实时功能,并调用时间敏感和闲置任务服务程序。

2 . 安全性

硬件的可靠性有了很大程度提高的同时,并且嵌入式系统的任务也很紧急时,嵌入式设计者一定把建造处理器测试和内存测试用到应用系统中,这有各种各样的方法能实现。可能嵌入式编程者学到的第一个和最简单的安全技术,就是中断结构填充的未使用的编程内存。这种技术可以禁止非法跳转出程序空间,并且提供初级的安全保证。另一种通常的保护措施是使用缓冲器,保证堆栈不会溢出和任务堆栈崩溃。许多商用 RTOS 现在含有的功能能够支持

堆栈检查。

为了证实存储在 ROM 中的程序和数据的完整性,应该包括一个简单的 ROM 测试,就像一个看门狗定时器用来防止软件进入一个死循环一样。

众所周知,无用的指针可能导致整个内存的崩溃。那么如何保护好编程数据防止崩溃呢?一种技术——重要变量的冗余存储。另一个就是重要变量的分组,可以保证每组的校验码。

3.设备驱动程序

众所周知,编写高效的设备驱动程序既要有软件又要有硬件的知识。因为设备驱动程序被重复的调用,使得它是嵌入式系统性能的重要因素,并且利用响应时间、内存和其他系统资源来支配实时性能。嵌入式系统中 10%的代码是设备驱动程序,为了能使编程更有效,对更复杂的系统编写设备驱动程序需要更多的开发时间,设备驱动程序应该充分利用微控制器的能力。

4.中断服务程序

使用中断服务程序通常要比用软件循环不断地查询外围设备要合理得多。然而,中断处理策略不是由编译器支配的,大部分的 RISC 处理器对中断响应很少。这些限制给嵌入式开发商带来了很大的压力。如何选择中断结构,有一些方法节省了内存堆栈的中断内容,而有一些则保存这些中断的内容在高速缓存器中,或在片上寄存器或是片外存储器。为简化调试,最好保证中断服务程序短一些。

5.存储分配

存储分配是考虑选择 RTOS 或嵌入式系统设计时一个很重要的特性。设计动态的存储分配是很浪费的,原因有二:第一,从堆栈中分配内存即慢又不确定,在内存管理中寻找存储空间的空闲列表使用的时间是有限的;第二,内存分配的故障可能是由于分段的堆栈引起的。典型的解决办法就是将所有目标挂起,并除掉动态分配。

6.最优化性能

编写运行有效的嵌入式代码带来了一整套的规则。为了优化速度和大小而不违背设计目标,建议使用三种技术:第一,在大多数嵌入式交叉平台编译器中使用优化选项(例如,减少冗余码,用方程代替操作,展开环,优化寄存器的使用,除掉编译器知道的但不能实现的代码部分);第二,定点和浮点混合操作;第三,利用大部分可利用的资源优化用户。

7.调试内存的问题

因为许多 RTOS 和嵌入式微处理器不支持内存保护,追踪软件内存故障就变成了严重的调试问题。在处理这些问题上,最好按照被影响的内存类型分类问题。通常分成三类:

(1) 全部内存故障:那些故障导致全部内存数据区的崩溃;

(2) 堆栈内存故障:这些故障完全导致程序运行失败,而当这些故障作为外部事件的函数时,则很难追踪;

(3) 动态分配内存故障:存储器服务分配堆栈内存,用到了分配过的内存块或是使用了不再分配的内存块而引起的问题。

四、结束语

作为一个嵌入式开发者,面临的不仅仅是决定使用什么样的 RTOS 和其他开发工具;而且必须要学会处理像资源有限、硬件设备驱动、中断队列、储存分配等等一些嵌入式开发中的事项。

参 考 文 献

1 ack Ganssle . Changes in Embedded Design Methodolody . Embedded Systems Programming [M], Sept 1998

2 David Stepner, Nagrajan Rajan, David Hui . Embedded Application Design Using a Real-Time OS .36th Design Automatic Conference, Proceedings 1999

3 Robert Richards . Designing RTOS for Embedded Microcontrollers . Embedded Systems Programming [M], May 1997

4 Doug Brown . Solving the Software Safety Paradox . Embedded Systems Programming [M], Dec 1998

5 Shaul Gal-Oz . The Hazards of Device Driver Design . Embedded Systems Programming [M], May 1997

6 Daniel Mann . An Uncommon ISR Technique . Embedded Systems Programming [M], Jan 1995

7 David Lafreniere . An Efficient Dynamic Storage Allocator . Embedded Systems Programming [M], Sept 1998

选自《电测与仪表》月刊,2001 年第 8 期

3 5　实时 Linux 操作系统初探

长沙国防科学技术大学计算机学院(410073)　夏一民　罗　军　邓胜兰

一、引　言

实时操作系统是一种能在边界时间内提供所需级别服务的操作系统。它能够明确说明它的每一个系统服务运行所需的最长时间,运行在它上面的任务的行为都必须是可预测的。正是由于这个特点使得实时操作系统在计算机应用的许多重要领域(例如国防、通信、航空航天、工业控制等)起着不可替代的作用。这些领域对每个关键任务所用的最长时间都有着严格的要求,只有采用实时操作系统才能够保证关键任务能在规定的时间内完成。这些应用的需求极大地促进了实时操作系统的发展。目前已开发了至少数十种性能各异的实时操作系统,其中 UNIX 作为一种成熟、开放、标准并且兼容 POSIX 1003 实时规范的操作系统,成为目前使用最广泛的实时操作系统;而 Windows NT 只部分兼容 POSIX 1003 规范,所以在实时领域使用得并不多[8]。此外还有许多基于微内核的商业实时操作系统,它们更多用于嵌入式系统中[9]。

伴随着 Linux 这种自由软件的迅速发展,用于实时领域的实时 Linux 操作系统也开始出现了。目前已有 RTLinux, RED-Linux, KURT, RTAI 等多种实时 Linux 操作系统。它们都以稳定、高效的 Linux 内核作为基础,通过扩展而建立开放、标准、高效、便宜的多任务实时操作系统。实时 Linux 操作系统能够在使用 Linux 的网络、X Windows 和开发环境等资源的同时,提供很好的实时能力。本文将进一步介绍它们的实时机制,特别是它们的实时调度、中断管理和实时通信机制。

二、实时操作系统的概念及开发方法

在介绍实时 Linux 操作系统之前,我们需要先明确有关实时的几个重要概念和开发实时操作系统的几种常用方法,以便更好地理解实时 Linux 操作系统的实现原理。

1 . 实时操作系统的定义及特征

每个实时操作系统一般都是为某一类实时系统的应用而设计的。实时系统是指:其正确性不仅依赖于计算结果逻辑上的正确,还依赖于计算结果产生的时机是否正确。如果系统的时间要求没有被满足,系统也失败。所以,对实时系统的要求是其行为的可预测并且能够满足系统的时间约束。例如一只机器手通过一个小窗口从一条移动的传送带上取一样物品,由于物品在不断移动,所以机器手必须在物体移走之前取走它。如果迟了,即使机器手移到那也不能取到此物体。

对于实时操作系统,POSIX 1003 .1 是这样定义的:实时操作系统必须有能在边界时间内提供所需级别服务的能力。设计小规模的实时系统可以采用一个很大的循环(Loop)来实现。这样做的优点是实现简单;缺点是:系统不能按优先级来响应事件、最大响应延时会很长(等于

整个循环的最长执行时间)、灵活性差(修改一个地方可能牵动整个循环)。所以对于中大规模的实时系统必须有实时操作系统的支持。实时操作系统一般具有如下基本特征:

(1) 支持多线程和可抢先调度;

(2) 有线程优先级的概念;

(3) 有一个可预测的线程同步机制;

(4) 具有优先级继承机制;

(5) 操作系统的行为应该被用户了解和掌握。例如:

· 中断延时,即从中断产生到处理任务被调度运行的时间。它必须是可预测并满足应用的要求,当然中断延时依赖于并发中断的数量。中断延时可分为两大部分:系统的中断响应和实时任务的调度,它与以下指标有关:中断分派时间(IDT)、中断服务时间(IST)、内核抢先时间(KPT)、调度延时(SD)、上下文切换时间(CST)、系统调用返回时间(RST)。

· 每个系统调用的最长时间。由于应用在设计时无法知道执行时的系统状态,所以该指标必须独立于系统中的对象数。

· 操作系统和设备驱动程序屏蔽中断的最长时间。

实时操作系统按对延时的约束要求分为‘软实时操作系统”和‘硬实时操作系统”两种。一般来说,硬实时操作系统是指如果计算结果输出的时间超过了时限(Deadline)将会引起灾难性后果的实时操作系统,如控制飞机发动机的实时操作系统。软实时操作系统是指如果计算结果输出的时间超过了时限虽不会引起灾难性后果,但是应尽可能满足时限的实时操作系统,如播放多媒体的操作系统。

对于一个优秀的实时操作系统来说,不仅要有一个好的内核、很短的中断延时或上下文切换时间,还要有很好的说明文档、很好的开发和调试工具,这样才能为开发者提供一个良好的开发环境。

2 . 开发实时操作系统的几种方法

设计实时操作系统常见的有三种方法。

第一种方法是从零开始,设计一个完整的实时操作系统,如 RT-IX。这种方法的优点是能够得到一个功能强大的实时内核,但工作量非常大。

第二种方法是设计一个实时操作系统的微内核,再在此微内核上逐步增加功能,如 Vx-works, QNX。这种办法的优点是工作量小,系统灵活。对于一些不要求硬实时的功能,如网络通信、图形显示,可做成软实时而不影响内核的实时能力。

第三种方法是改造已有的通用操作系统,使其具有实时能力。采用这种方法的除实时 Linux 外还有实时 UNIX,实时 Windows NT 等。这样做的好处是既有确实时能力又编程简单,充分发挥了原操作系统的功能和资源。

通用操作系统和实时操作系统有很大的不同,前者是追求最优的平均性能,尽可能多的服务和最大的吞吐量;而后者则致力于改善最坏情况下的性能,缩短中断延时和提供可预测的调度策略。要将一个通用操作系统改造成一个实时操作系统,必须改造原有的调度策略、中断管理、任务管理、内存管理等许多部分。

3 . Linux 作为实时操作系统的不足

虽然从 Linux 2 .2 开始兼容 POSIX 1003 .1b 实时规范,提供内存锁定(Mlock)、实时任务的特定调度策略(Sched_setsched)和 POSIX RT 信号,但 Linux(其他 UNIX 操作系统也一

样)的实时进程仍可能被内核阻塞相当长的时间[4],因为:

① Linux 采用的时钟中断的精度不高。Linux 为了提高系统的平均吞吐率,将时钟中断的周期设为 10 ms,这个时钟精度不能满足实时要求;

② Linux 在每个系统调用期间都关闭中断。有些系统调用的时间很长,这增加了中断延时和调度延时;

③ Linux 设立了许多缓冲机制使得实时任务不能及时执行。例如磁盘缓冲和各种 Cache 机制,这使得系统的行为难以预测;

④ Linux 采用粗糙的同步策略,系统可能长时间占用某个临界资源而阻塞企图使用统一资源的实时进程;

⑤ Linux 下优先级倒置时间没有上界,实时任务可能被阻塞非常长的时间。

总之,原始的 Linux 只能提供一般性能的软实时能力,还不能成为一个实时操作系统。目前有许多通过扩展 Linux 的实时性而形成的实时操作系统。以下介绍几种比较有代表性的实时 Linux 操作系统。

三、几种实时 Linux 操作系统的实现原理

1. RTLinux

RTLinux 由美国新墨西哥州的 Victor Yodaiken 提出设想, Victor Yodaiken 和 Michael Barabanov 共同开发,主要应用于仪器设备和嵌入式系统。目前只有用于 INTEL X86 体系结构的版本,支持 SMP 多处理机系统。它的目标是将 Linux 提供的标准 POSIX 服务与硬实时服务有机地结合起来,提供透明的、模块化、可扩展的实时操作系统。

RTLinux 通过修改 linux/arch/i386/ 下与体系结构有关的部分,在 Linux 与硬件之间创建一个硬实时内核。Linux 作为此内核的一个优先级最低的任务运行。硬实时任务都在硬实时内核上运行,而所有非实时任务都在 Linux 上运行。RTLinux 的数据和控制流程图见图 3.5-1。

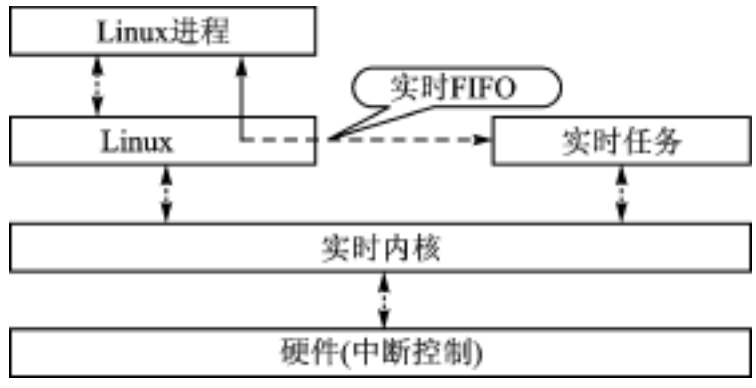


图 3.5-1 RTLinux 数据和控制结构图

RTLinux 接管了全部硬件中断(/linux/arch/i386/irq.c),并将 Linux 系统调用及驱动程序的开、关中断(sti, cli, iret)都用 RTLinux 的软中断(soft_sti, soft_cli, soft_iret)代替。Linux 的开中断和关中断只是在 RTLinux 的数据结构中作一个标记,并不真正地关闭中断。RTLinux 只在修改核心数据结构的关键代码处关中断,所以 RTLinux 的中断延时非常小。当发生中断时,实时内核首先检查是否有实时处理程序要处理这个中断,如果有则调用此实时处理程序,没有则设置相应的中断标志位,交给 Linux 处理。

RTLinux 带一个按纯优先级调度的简单调度器(rtl/schedulers/rtl_sched.c),用来调度 RTLinux 上的任务。用户可以根据需要编写自己的调度器(rtl_schedule())。为了使实时任务与非实时任务之间的通信不依赖于 Linux 的通信接口,RTLinux 开发了实时 FIFO、信号量、消息队列、共享内存等通信机制来实现实时任务和实时任务间、实时任务和非实时任务间

的通信。一个系统可以由实时任务采集信号,然后通过实时 FIFO 交给非实时任务显示和处理。RTLinux 所需的资源全部静态分配,以避免动态分配资源产生不可预测的延时。实时任务可编译成一个模块,并在运行时动态加载。

2 . KURT

KURT (KU Real-time Linux, KU: the University of Kansas Center for Research)由美国 Kansas 大学开发,目前 KURT 只提供 X86 版本。KURT 与 RTLinux 不同,它不在硬件与 Linux 之间增加一层内核,所以它只有一个内核。KURT 保留 Linux 中原有的中断管理,但和 RTLinux 一样,通过对时钟芯片 8254 编程达到微秒级的定时能力,而不是 Linux 的毫秒级的周期时钟,从而提供比 Linux 更好的软实时能力。换句话说,KURT 在提供很好的系统平均性能的同时又尽可能地提供好的实时性。KURT 主要应用于这样一类系统:采用一般操作系统则性能得不到保证,而采用硬实时系统又觉得代价又太大,如多媒体和高速 ATM 网络。

KURT 引入了两种操作模式:一般模式和实时模式。

在一般模式中,系统的行为和一般 Linux 系统一样,系统只调度非实时任务。当核心运行在实时模式时,它只执行实时进程。一个系统调用允许核心处于实时模式或一般模式之一。在实时任务初始化时,就已经指定了产生实时事件时被调用的处理程序。当实时任务执行结束时,核心可以切换回一般模式。

KURT 通过实时调度框架来处理所有实时事件和它们相应的处理程序。当产生一个实时事件时,实时框架调用相关的实时事件处理程序。这样就将实时任务的调度和执行分成两个独立的方面,从而简化了在实时系统的设计。KURT 采用循环调度策略,即根据“计划表”进行调度。计划表中包含全部所需调度的动作,如动作的执行时间,动作执行的处理程序等。此表在系统设计期间建立,在随后的运行中,调度的工作就是连续地从表中读取指令。当读到表尾时,又从头开始执行,因此叫循环调度。它的好处是实现简单、高效,计划制定后就可立即执行。主要的困难是如何确定计划,而且每次修改一个任务的参数都必须重建整个计划,而且表格也要占用很大的空间,当然可以根据需要将部分或全部表格放入内存以减少空间需求。

KURT 实时任务与实时任务之间、实时任务与非实时任务之间可通过实时 FIFO 进行通信。

3 . RED-Linux

RET-Linux (Real-time and Embedded Linux)的目标是建立一个用于嵌入式环境的实时操作系统内核,并提供较高的执行效率和灵活的调度策略。为提高效率采用一个很短的任务分派间隔(< 100 μs)、与 RTLinux 一样的高效定时器(μs 级)和软中断仿真;提供很好的实时调度策略支持以增加灵活性,即在一个统一的调度框架下支持各种实时调度方案,例如:优先级驱动、时间驱动和共享驱动等;在内核中插入抢先点以减少实时任务被阻塞的时间。

在大部分 UNIX 系统中,每个系统调用期间都要关闭中断。没有作业可以中断一个系统调用的执行,除非此系统调用因为等待资源而主动放弃 CPU。有时一个系统调用可能占用 CPU 数秒钟的时间。为了降低中断延时,RET-Linux 将一个内核服务程序分成许多小块(Blocks),在每个小块执行结束时检查是否有实时任务需要调度,如果有则执行调度程序。这种检查点就叫做抢先点(Preemption Point)。这种抢先方式是一种合作抢先方式(Cooperative Preemption)。系统的响应延时将决定于执行时间最长的那块内核程序。由于在一次系统调用期间有可能发生多次上下文切换,所以每次上下文切换都要保存许多状态。如果上下文切

换期间某些重要状态发生了改变,恢复运行时还要回滚(Rollback)到前一个检查点(Check Point)重新开始,所以块大小的选择对系统的运行效率有很大影响。另外,抢夺点在系统设计时已经固定,不能根据应用的需求改变。

RED-Linux 的软中断仿真解决了多个中断并发产生时的响应问题。当一个硬件中断发生时,系统只在事件表中设置这个事件的状态标志就立刻返回,并不搜索中断向量表,更不执行相应的中断服务程序,这有点类似管理中断的硬件(如 8259)。然后系统可以根据收到的状态标志决定是否处理中断事件。如果有更高优先级的实时任务在执行,就要等到此高优先级的实时任务执行完毕再执行中断服务。

RED-Linux 的调度器被分为两个构件:分配器和发送器(Allocator and Dispatcher)。Allocator 根据系统资源的使用情况决定是否给用户作业分配资源,Dispatcher 则具体实现实时任务的调度。全部非实时任务由原 Linux 调度。Allocator 是一个由用户创建的用户实时进程,它不是内核的一部分。实时任务先在 Allocator 中注册自己的调度参数,再由 Dispatcher 调度。用这种方法,调度策略易于修改而不必改变低级调度机制。这种结构也可以使好的应用调度器被其他应用重用。一个通用调度框架的示例见图 3.5-2。

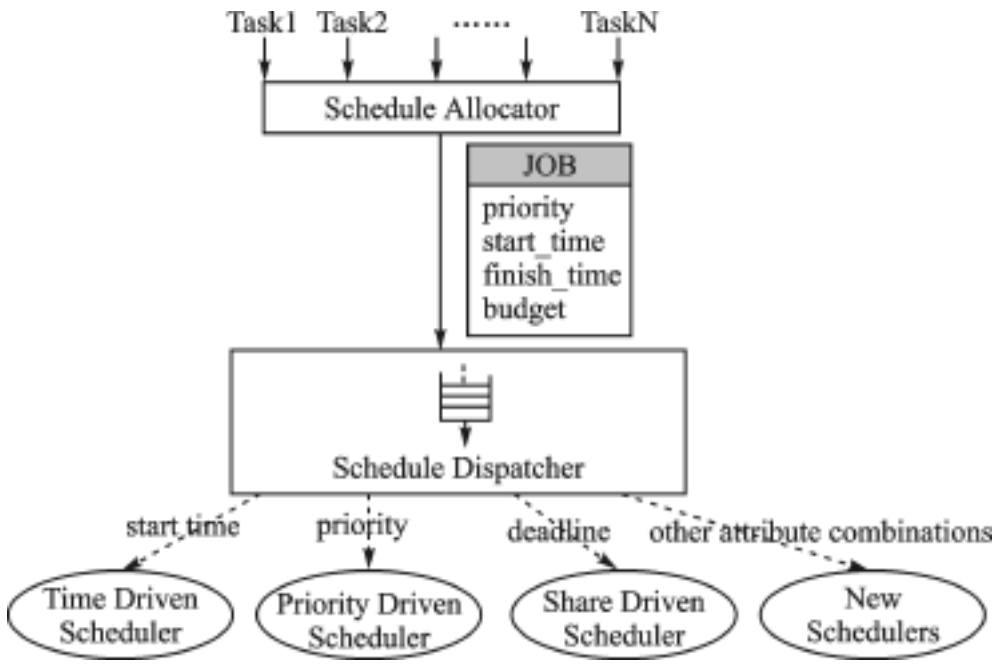


图 3.5-2 一个通用调度框架的示例

4. RTAI Linux

RTAI (Real Time Application Interface Linux)的目标是通过 RED-Linux 项目(the Real-time and Embedded Linux Project),在实时硬件抽象层(RTHAL)上建立实时应用接口(RTAI),为实时应用编程提供一个通用而功能强大的接口。RTAI 也支持 SMP 多处理机系统。

RTAI 修改了 Linux/arch/i386 中与体系结构相关的代码而形成了 RTHAL 层。RTHAL 的作用是使 RTAI 能够在实时任务需要运行的任何时刻中断 Linux。引入 RTAI 后,Linux 的功能没有改变,但运行于最低优先级下,当无实时任务执行时它才允许执行。

RAI 分为三个模块:(1) rtai(中断处理);(2) rtai_sched(调度模块);(3) rtai_fifos(fifo 模块)。

RTAI 能动态装入和卸载,实时任务在 RTAI 装入之后再装入。

RTAI 的任务调度器提供硬实时能力,采用基于固定优先级的抢先调度策略。全部调度可根据定时函数、实时事件(如信号获取)或异步事件来管理。RTAI 在 Pentium 和 486 类

CPU 上既支持单发定时器也支持周期定时器,但不能同时使用,可以在一个应用中交替使用它们。Pentium 类处理器支持单发任务的速率根据处理器的不同而不同,在 Pentium Ⅱ 233 上,超过 30 kHz。在 486 机器上,单发任务速率大约为 10 kHz(同时余下足够的 CPU 时间去完成标准 Linux 服务)。RTAI 的周期定时器支持速率超过 90 kHz,具体速率依赖于 CPU 和总线的速度以及芯片的性能。

实时任务中可使用浮点单元,当使用 FPU 的任务装入时被打上特殊的标志,并切换 FPU 的上下文。这种方法既允许实时任务和非实时任务同时使用浮点单元又减少了不必要的上下文切换时间。RTAI 和 RT Linux 一样,也提供直接高效的硬件存取调用,而不必通过标准 Linux 的中断管理层。

RTAI 的实时任务到实时任务的通信方法包括:信号、互斥、消息队列及远程过程调用。实时任务到 Linux 任务的通信方法包括实时 FIFO 和共享内存。

四、今后的目标

目前,各种实时 Linux 操作系统的功能还比较简单,同步、互斥和通信等机制还比较粗糙,需要进一步完善,需要对当前网络环境下的分布实时提供更好的支持,如高效可预测的通信协议、分布实时调度算法等。另外,对于恶劣条件下的应用应提供除定时狗以外更多的保护。为了增强系统的可靠性、减少错误,还应该提供一种好的测试工具以分析系统的最大中断延时、中断屏蔽时间、可调度性等指标,降低开发难度,提高软件的可靠性。

以上介绍的实时 Linux 操作系统都是自由软件,作者提供源代码下载。有了这些开放的实时操作系统,我们就可以方便地将实时理论的研究成果做成原型、范例或具体的实现,分析它们的实现对我们设计自己的实时操作系统也有很好的借鉴作用。有兴趣的读者可到以下站点下载相关源码和资料。

实时 Linux 主页:http:// www .realtimelinux .org

RTLinux 主页:http:// www .rtlinux .org

KURT 的主页:http:// www .ittc .ukans .edu/ kurt

RED-Linux 的主页:http:// www .ece .uci .edu/ red-linux

RTAI 的主页:http:// www .aero .polimi .it

参 考 文 献

1 smael Ripoll . Real-time Linux: RT-Linux [R]

2 Victor Yodaiken . The RT-Linux Approach to Hard Real-time [R]

3 Balaji S . KURT: The KU Real-time Linux [R]

4 P Mantegazza, E Bianchi . Introducing the Real Time Application Interface (RTAI) for Linux [J]

5 Wang Yu-Chung, Lin Kwei-jay . Providing Real-time Support the Linux Kernel [R]

6 Wang Yu-Chung, Lin Kwei-Jay . Implementing a General Real-time Scheduling Framework in the RED-Linux Real-time Kernel [R]

7 Wang Yu-Chung, Lin Kwei-Jay . Enhancing the Real-time Capability of the Linux Kernel [R]

8 Michael Barabanov . A Linux-based Real-time Operation System [R]

选自《计算机应用研究》月刊,2001 年第 1 期

3.6 Linux 网络设备驱动程序分析与设计

郑州信息工程大学电子技术学院(450004) 李炳龙 陈性元 杜学绘

一、引 言

Linux 系统作为一个开放的源代码操作系统,为人们深入了解操作系统的工作原理提供了极好的机会,同时它也为人们研制基于 Linux 系统的网络安全产品提供了一个理想平台。Linux 网络设备驱动程序是 Linux 操作系统内核的一个重要组成部分。对 Linux 网络设备驱动程序的原理与设计技术的掌握,有助于理解网络链路层的工作原理,它对我们从事基于 Linux 平台的网络安全产品开发具有一定的意义。本文在分析了 Linux 网络设备驱动程序的结构组成和工作原理之后,重点探讨了网络设备驱动程序的设计技术。

二、Linux 网络设备驱动程序的结构组成

从图 3.6-1 我们可以看出,Linux 网络设备驱动程序由四部分组成:网络设备接口对象、协议接口代码、网络设备驱动程序代码及网络设备和介质。网络设备数据包的发送和接收必须通过这四个部分的一系列协调的操作才能完成。

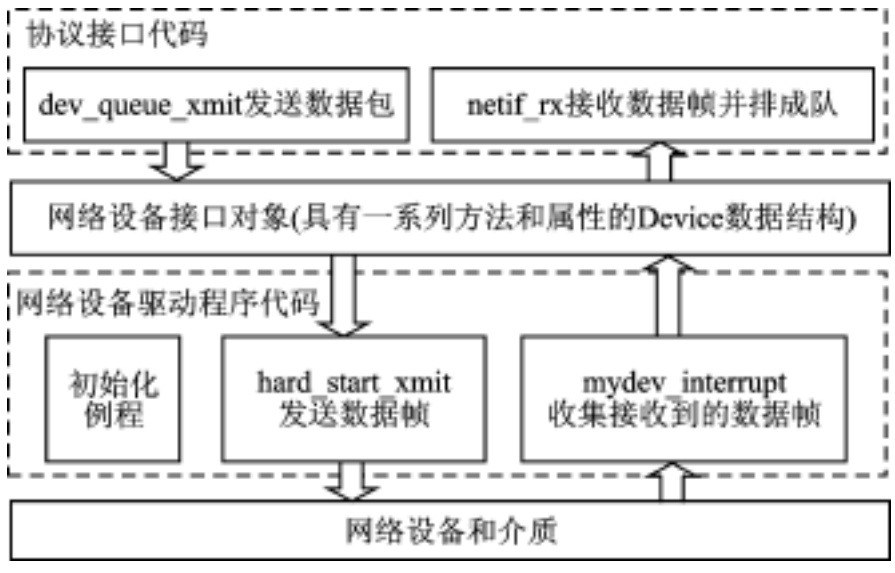


图 3.6-1 Linux 网络设备驱动程序的结构组成

1.网络设备接口对象

为了实现网络接口层软件与硬件的相对独立,在 Linux 系统中用 Device 结构来表示网络设备接口,用以保存所有与硬件无关的接口信息,并协调各个协议软件,使它们主要通过这个数据结构来完成与硬件的交互作用。在网络子系统内核中,全部网络设备接口组成一个以 dev_base 为头指针的设备链表。该设备链表由内核管理,它表示 Linux 系统中网络子系统部分要用到的所有网络设备,其中每个链表元素对应一个网络设备接口。该网络设备接口不仅可以为硬件接口,还可以为软件网络设备接口,如环路设备(Loopback)、隧道(Tunnle)等。接

口链表中的每个元素通过链表指针可被整个系统使用。Device 数据结构在域的意义上可以分为以下部分:

(1) 名称 name 指网络设备的名称。

(2) 总线接口参数 总线接口参数用来设置设备在设备地址空间的位置,如 irq(网络设备使用的中断请求)、base_addr(基地址)、dma(标志设备正在使用的 DMA 通道)等。

(3) 协议层参数 它是指为使网络协议层能够正确地执行任务网络设备驱动程序而需要协议层提供的一些性能标志和变量,其中主要是:网络接口的最大负荷(也就是网络可传输的最大数据包)、设备支持的地址族(常见的地址族是 AF_INET)、设备所连接的物理介质的硬件接口类型(它的值来自物理介质类型表)。

(4) 链接层变量 例如 MAC 头的长度。

(5) 接口标志 网络接口标志用来描述网络设备的特点和功能。例如 IFF_UP,它表示接口已经激活。

(6) 数据包发送队列 该队列包含了等待由该网络设备要发送的 sk_buff 数据包,这些数据包是由内核中协议层代码发送到该队列中。在 Device 结构中这样的队列共有三个,它们由数据结构 Qdisc 来进行管理。数据结构 Qdisc 是 Device 结构中的一个成员,它的作用是用来管理网络设备的数据包发送队列。

(7) 数据包发送队列长度 发送时最多缓存的数据包个数。

(8) 设备方法 网络设备作为一个对象,提供一些方法供系统访问,这些方法可以分为两类:实用性函数和标准接口例程。实用性函数是底层的基本函数,标准接口例程是供协议层调用的接口例程。

· 初始化(init)

init 函数在设备初始化和注册时被调用,它执行的是底层的确认和检查工作。网络设备驱动程序通过 init 函数可以完成对硬件资源的配置。init 函数是一个指针函数,它收到的惟一参数是一个指向正被初始化的网络设备接口指针(struct device * dev),其返回值是 0 或者一个负的错误代码。返回值为 0 表示探测设备成功,负值表示探测设备失败。

· 打开(open)

当网络设备被激活的时候调用 open 函数,此时网络设备的状态由 down 变为 up。

· 关闭(stop)

stop 函数与 open 函数的功能相反,可以释放某些资源以减少系统的负担。stop 使网络设备的状态由 up 变为 down。

· 数据包发送例程

所有的设备驱动程序都必须提供发送例程。其功能是把数据包发送到硬件网络设备。

· 中断处理例程(mydev_interrupt)

中断处理例程主要用来处理网络设备产生的中断,例如接收中断、发送完成中断和发送错误中断等。

2.协议接口代码

该代码提供了网络接口设备驱动程序的抽象接口。该接口为上层协议层提供服务,同时它隐藏了网络设备驱动程序的实现细节。相邻层的通信只需要通过特定函数调用实现,使通信过程相对透明,简化了开发的难度。这些接口代码定义在 </linux/net/core/dev.c> 文件

中,其中两个最为重要的函数是 dev_queue_xmit 和 netif_rx。

3.网络设备驱动程序代码

从图 3.6-1 可知,网络设备驱动程序代码主要分为三部分:初始化函数、数据包发送函数和数据包接收函数。网络设备驱动程序代码是整个驱动程序的主要部分。为了通过网络设备驱动程序与上层软件及下层硬件打交道,必须调用这些函数。

三、网络设备驱动程序代码组成

Linux 系统通过一个相当明确的 I/O 模型来处理与外部设备打交道的双向数据流,通过系统提供的 I/O 函数实现资源管理、初始化和清理进程。在 Linux 系统中,网络部分使用的是面向对象的设计方法,每一种设备都被看成一种对象。网络设备接口控制着网络数据由软件到硬件再到软件的过程,体现了网络和设备的关系,是网络传输的桥梁。因此在 Linux 的网络设备驱动程序源代码中,就用一种具有一系列操作方法的数据结构来描述网络设备。从图 3.6-1 可知,网络设备驱动程序代码的组成分为以下三部分。

1.初始化例程

网络设备驱动程序是以模块的方式加载到内核的。当一个模块被加载到运行的内核时,它要求一些资源,并初始化设备驱动程序。每一个设备驱动程序都有 init_module 函数和 cleanup_module 函数,网络设备通过它的 init_module 函数进行加载,其初始化过程是:

· 探测物理设备

初始化例程探测它的设备及其硬件位置(I/O 端口和 IRQ 线),并根据硬件的特征检查硬件是否存在,如果不存在,则初始化失败,否则继续执行。

· 配置资源

初始化例程可以完成对硬件资源的配置,比如即插即用的硬件就可以在这个时候进行配置(Linux 内核对 PnP 功能没有很好的支持,可以在驱动程序里完成这个功能)。配置或协商好硬件占用的资源,然后向系统申请这些资源。有些资源是可以和别的设备共享的,如中断;有些是不能共享的,如 IO,DMA 等。

· 初始化 Device 结构

初始化 Device 结构中的变量,用探测所得到的值填充 Device 结构。如果初始化失败,则开始下一个设备的注册,否则继续执行。

· 注册设备

设备驱动程序为新检测到的接口在一个网络设备全局链表中插入一个 Device 数据结构。每个网络设备驱动程序都试图注册它自己的设备,但只有确实存在的网络设备才能被链入该列表。当注册一个网络设备时,内核要求驱动程序初始化自己。网络设备驱动模块被加载到运行的内核中时,对网络设备的初始化可以分两种情况:一种是在加载时探测;另一种是根本就不探测,而是给网络设备指定一个确定的设备 I/O 地址。前者比较适合 PCI 总线的网络设备,后者比较适合 ISA 总线的网络设备。造成这种分离的原因是:ISA 体系结构在容错方面不如 PCI 好。在具体实现时,当前设备的 I/O 基地址(dev->base_addr)决定了要做什么:

① 如果 dev->base_addr 是一个有效的设备 I/O 地址,将不再探测其他 I/O 位置,而是使用这个值。如果这个值在加载时赋值,这种情况就会发生。

② 如果 dev->base_addr 是 0,那么探测设备是可以接受的。用户可以通过在加载时设置

这个 I/O 地址为 0 来请求探测。

③ 其他情况下,不进行探测。内核使用 0xff0 来阻止探测,但其实任何无效值都行。这需要依赖于驱动程序来无声地拒绝 dev->base_addr 中的一个无效地址,一个模块应该缺省地置这个地址为无效值来防止不期望的探测。探测函数要完成的主要工作是:填充 Device 结构。内核通过函数 ether_setup 填充结构 Device,它负责了一些以太网的缺省设置。

2.数据包发送函数

网络设备的主要功能是数据包的发送和接收。网络设备发送过程比较简单,其数据发送是依以下步骤进行的:

(1) 它调用 dev->hard_start_xmit 方法,该函数的功能是将套接字缓冲区发送到硬件设备。它是一个指针函数,提供给 hard_start_xmit 的套接字缓冲区含有物理包,它具有传输层的头。接口不需要修改被发送的数据,skb->data 指向被发送的包。skb->len 是它的长度,以字节为单位。该函数把套接字缓冲区链入网络设备结构管理的发送队列;随后唤醒网络设备,准备把该队列中的数据包发送到网络上。

(2) 判断从协议层传过来的数据包长度是否合法以及网络设备的状态是否可用,如果一切正常,则继续执行,否则返回错误码。

(3) 通过对控制寄存器的操作来执行设备接口发送数据包的程序代码。

(4) 当发送结束时,返回。

3.数据包接收函数

在 Linux 系统中,网络设备接口以中断处理程序的方式控制。实际上,网络设备驱动程序的接收函数就是网络设备驱动程序的中断处理函数。Linux 系统的中断处理程序分为两部分:“上半部”和“下半部”。所谓的“上半部”是通过 request_irq 函数注册的处理例程,而“下半部”(bottom half,简称“bh”)则是由上半部调度到以后在更安全的时间内执行的那部分例程。上半部和下半部处理程序最大的不同就在于在执行 bh 时所有的中断都是打开的,所以它是在“更安全”时间内运行的(详细情况可参考文献[1])。所有的中断处理程序都做了这样的划分。

我们知道多个设备可共享同一个硬件中断。设备驱动程序必须能够确认是否是它的服务设备产生了中断。如果不是,调用另一个中断处理程序来处理中断。中断处理的一般步骤如下:

(1) 读取中断线寄存器,获取中断号;

(2) 中断处理初始化;

(3) 调用中断服务程序。

Linux 系统提供了专门的数据结构 irqaction(有关该结构的详细情况可参考源代码 <linux/interrupt.h>)用于硬件中断的实现。它主要是通过虚拟某个 IRQ 端口,并处理 IRQ 端口上的硬件中断。在图 3.6-1 中,mydev_interrupt 就是网络设备的中断服务程序。为了使数据能够进行传送,首先要对中断进行处理。在接收中断消息之后,调用函数 mydev_interrupt 进行中断处理。该函数首先关闭接口设备的中断,在进行完数据的收发后,应尽可能快地调用函数 enable_interrupt 来重新打开中断。中断处理程序要做的工作很多,其要做的主要工作是数据包的发送和接收。网络设备数据包的接收比发送要复杂一些,因为要分配一个套节字缓冲区,并从一个中断处理程序中将其递交给高层协议层。数据帧的接收过程如下:

(1) 在硬件收到一个数据包后,网络设备的接收函数被调用,它通过一些底层的操作把数

据帧从网络设备接收缓冲区中读入内存中。

(2) 调用 netif_rx 函数,netif_rx 因此收到一个指向数据的指针和包的长度。该函数的惟一职责就是将包和一些额外信息发送到网络代码的高层。

(3) mark_bh 被调用,它的功能是进行标记操作,为内核调用 net_bh 函数(中断处理程序的下半部例程)做准备。

(4) 调用 net_bh,进行中断处理程序的下半部处理,对数据包进行实际的处理,最后将接收到的数据包链入到 backlog 接收队列中。以后的工作就由网络层代码进行处理。

四、Linux 网络设备驱动程序设计中的基本问题

设备驱动程序的设计是一项要求比较高的工作,它要求程序员应具有丰富的编程经验,才能生成高效的代码。Linux 网络设备驱动程序的设计过程中应注意的主要问题有以下几方面。

1. 网络设备硬件

网络设备底层发送和接收代码的实现是依赖于网络设备硬件的,对某一硬件设备必须编写特定的发送和接收代码。Linux 网络子系统为各种硬件设备如串行连接、并行连接及以太网连接提供了抽象接口,隐藏了通信媒体之间的差异。

2. 中断共享

Linux 系统运行几个设备可共享同一个中断。需要共享的话,在申请的时候利用系统提供的 request_irq()调用指明共享方式,该调用定义如下:

```
int request_irq (unsigned int irq, void ( * handler) (int irq, void * dev_id, struct pt_regs * regs),
const char * devname, void * dev_id);
```

如果共享中断,irqflags 设置 SA_SHIRQ 属性,这样就允许别的设备申请同一个中断。需要注意,所有用到这个中断的设备在调用 request_irq()时都必须设置这个属性。系统在回调每个中断处理程序时,可以用 dev_id 这个参数找到相应的设备。一般 dev_id 就设为 Device 结构本身。系统处理共享中断是用各自的 dev_id 参数依次调用每一个中断处理程序。

3. 硬件发送忙时的处理

主 CPU 的处理能力一般比网络发送要快,所以经常会遇到系统有数据要发,但上一帧数据网络设备还没发送完。因为在 Linux 里网络设备驱动程序一般不做数据缓存,不能发送的数据都是通知系统发送不成功,所以必须要有一个机制在硬件不忙时及时通知系统接着发送下面的数据。一般对发送忙的处理在前面设备的发送方法(hard_start_xmit)里已经描述过,即如果发送忙,置 tbusy 为 1。处理完发送数据后,在发送结束中断里清 tbusy,同时用 mark_bh()调用通知系统继续发送。

4. 流量控制(Flow control)

网络数据的发送和接收都需要流量控制。这些控制是在系统里实现的,不需要驱动程序做工作。每个设备数据结构里都有一个参数 dev- > tx_queue_len,这个参数标明发送时最多缓存的数据包。在 Linux 系统里以太网设备(10/ 100 Mbps)tx_queue_len 一般设置为 100,串行线路(异步串口)为 10。dev > tx_queue_len 并不是为缓存这些数据申请了空间,这个参数只是在收到协议层的数据包时判断发送队列里的数据是不是到了 tx_queue_len 的限度,以决

定这一帧数据加不加进发送队列。发送时另一个方面的流量控制是更高层协议的发送窗口(TCP 协议里就有发送窗口)。达到了窗口大小,高层协议就不会再发送数据。接收流量控制也分两个层次。netif_rx(()缓存的数据包有限制。另外高层协议也会有一个最大的等待处理的数据量。发送和接收流量控制处理在 net/ core/ dev .c 的 dev queue_xmit()和 netif_rx()中。

之外,还有一些在实际的设计过程中必须给予充分考虑的问题,如数据包的过滤、时钟检测硬件的挂起情况、驱动程序同步、支持多处理器和全双工等。

五、结束语

本文系统地分析了 Linux 操作系统网络设备驱动程序的组成及其原理,并给出了网络设备驱动程序的一般设计方法;最后讨论了在实际的设计过程中应着重注意的基本问题,为进一步开发实用有效的网络设备驱动程序奠定了基础。

参 考 文 献

1 lessandro Rubini Lisoleg . Linux 设备驱动程序[M] . 聊鸿斌等 . 北京:中国电力出版社,2000

2 陈莉君 . Linux 操作系统内核分析[M] . 北京:人民邮电出版社,2000

选自《计算机应用研究》月刊,2001 年第 10 期

3 .7　在 51 系列单片机上实现非抢先式消息驱动机制的 RTOS

厦门大学　许　俊　许克平

8051 单片机在中小型应用场合相当常见。在 20 世纪 80 年代中期, Intel 公司将 8051 内核使用权以专利互换或出售的形式转让给世界上许多著名的 IC 制造商, 使得 8051 成为众多厂商支持的、发展出上百种品种的大家族; 同时由于 8051 单片机是进入中国市场最早的单片机之一, 在国内有众多的仿真器开发商支持 51 系列单片机, 大多数的终端产品公司也大量应用 51 系列单片机作为产品的开发, 使得 51 系列单片机在国内成为开发中小嵌入式系统的首选单片机之一。由于一些知名的软件开发商开发出比较可靠的面向 51 系列单片机的 C 级编译器和 RTOS, 在 RTOS 的支持下, 可以应用 51 系列单片机开发出可靠性高、实时性强的较大型程序代码, 应用于更多要求较高的场合。

商品化的 51 系列 RTOS 中, 有代表性的是 Franklin 公司设计的 RTX51。它根据 RTOS 内核的大小分为迷你版本、标准版本和专业版本。RTX51 是一种抢先式的多任务操作系统, 可以设置 0～3 共 4 级优先级, 实时性很强, 功能齐全。可以供用户调用的服务(函数)有 3 类: 系统服务、信息服务和功能服务。各个任务之间的参数传递是通过邮箱操作(包含在功能服务中)来实现的。但是这样的一个操作系统对于大多数应用, 过于复杂, 特别是对于各个任务之间的参数传递必须使用邮箱操作来实现, 需要使用外部数据存储器和 300 字节左右的额外代码量。迷你版本(含标志传送功能)就有 700 字节左右的代码量, 而且分配给每个任务的内部 RAM 默认为 8 个字节。TINY 版本的 RTX51 最多可以允许 16 个任务运行, 这样对于内部 RAM 的分配变得非常困难。

笔者在某刊物上看到另外一种 51 单片机的 RTOS, 实际上是设置几个标志位。将中断产生的标志作为消息, 在主程序中不断地查询这些标志位。当查询到某个标志位改变时, 就调用相应的模块。首先, 这种方式不是严格意义上的消息驱动机制, 而且对于各个模块之间的参数传递没有定义, 不利于功能模块的划分; 此外, 由于标志位的查询是顺序进行的, 例如当某个中断发生, 设置了标志位 BIT0 和 BIT4, 假设标志位的查询顺序是从 BIT0 开始的, 本来按照正常次序要先执行模块 0 和模块 4, 但是在运行模块 0 时有另一个中断发生, 设置了标志位 BIT2, 造成模块 2 比模块 4 先执行, 这样就有可能造成不可预料的后果。

基于非抢先式消息驱动机制的 RTOS 是采用面向对象的思维方法, 把各个功能模块看成是不同的对象, 对象之间的通信称为发送消息。对象包含自己的数据和代码, 数据表征对象的特征, 代码用于相应消息, 使对象进行某些动作。对象响应消息进行处理时不被中断, 消息没有优先级之分, 除非是中断到来, 即消息驱动是非抢先式的。OS 严格按照消息队列的顺序来“唤醒”相应的对象。基于非抢先式消息驱动机制的 RTOS 的系统运行框图如图 3 .7 - 1 所示。

这样做有几个好处:

(1) 绝大多数的应用场合下,“实时性”只是体现在对外部事件的及时响应和对数据的即时接收或者发送。这可借助于 51 单片机的中断来实现。在中断处理程序中, 单片机只对外部

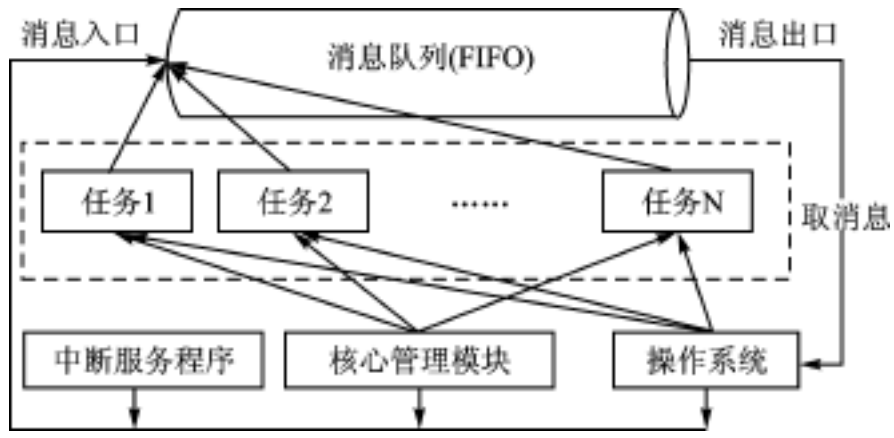


图 3.7-1 基于非抢先式消息驱动机制的 RTOS 的系统运行框图

事件作必要的处理或者只是将接收数据放到预定的缓冲区立即返回,同时向消息队列发送一条消息通知操作系统。

(2) 采用面向对象的思维方法有利于模块的划分。在多人协同编写 1 个软件的时候显得尤为重要。它使得负责单个模块的软件设计人员只需要关心自己的那一个部分(当然管理模块除外),各个功能模块的交互是透明的。

(3) 具备相当的通用性。如果需要增加新的功能模块,只需要编写新模块的代码,在 OS 中增加新模块的标志 ID 号,而其他部分可以保持不变。

下面详细阐述 RTOS 的构成。

一、消息的封装和消息队列

基于非抢先式消息驱动机制的 RTOS,消息由 4 个部分组成: TASKID、COMMAND、PARA1、PARA2。其中,TASKID 标志功能模块号,OS 中的消息循环根据 TASKID 来确定这个消息是发送给哪个功能模块的,占用 1 个字节;COMMAND 指示该功能模块执行什么样的操作,占用 1 个字节;PARA1 和 PARA2 是向该功能模块传递的参数(如果需要的话),各占用 1 个字节的空间。

这样一个消息占用 4 个字节的内部 RAM,消息队列可以容纳 10 条消息(消息队列的大小可以视具体情况而定),借助 2 个指针: WPTR、RPTR。当 WPTR = RPTR 时,表示此时消息队列中没有消息,一旦 WPTR≠RPTR 时,表明此时消息队列非空。消息队列的存储位置可以根据具体使用场合来确定:当需要使用的 RAM 空间较多时,例如进行大量数据采集一类的应用,那么消息队列可以位于外部 RAM 的空间,消息队列可以相应设置大一些;如果消息队列位于内部 RAM 区,消息队列就要尽量小,而且要合理安排所处的区域,对于 51 单片机来说可以位于30H～58H。

二、RTOS 主循环

RTOS 的主循环最主要的任务就是定时查询消息队列。若消息队列非空,则取出最“旧”的那一条消息,并唤醒相应的任务。为此,RTOS 还设置了 1 个“当前消息区”的区域 MSGARA,占用 3 个字节的内部 RAM 空间。这 3 个字节的内容由当前取出的那一条消息来填充,分别是 COMMAND、PARA1、PARA2,同时 TASKID 放在寄存器 A 中。当调用外部函数 TBL_JMP 时,外部函数 TBL_JMP 根据 A 中的 TASKID 唤醒相应的任务,同时把 MSGARA 区域中的参数传递给这个任务。OS 的主循环程序代码详列如下:

```
OS:
   LCALL     ASK_0ms                    清看门狗
   JB        F_1ms, OS_GETMSG           ;1 ms 定时到否 ?
   CLR       F_1ms                      ;1 ms 定时到
   LCALL     OS_CYCLE                   ;定周期处理
OS_GETMSG:
   ACALL     GET_MSG                    ;从消息队列中取消息
   JC        OS                         ;消息队列为空,则跳转回循环头部
   MOV       DPTR, # TASK_TBL           ;任务列表首地址
   LCALL     TBL_JMP                    ;唤醒相应的任务
   AJMP      OS                         ;跳转回循环头部
```

注 1:定周期处理函数 OS_CYCLE 在第四部分“RTOS 的定时处理”中有详细的介绍。

注 2: TBL_JMP 为外部函数,入口参数有两个:一个是寄存器 A,另一个是函数散转表的首地址。TBL_JMP 根据寄存器 A 中的内容执行相应的任务。

注 3:原则上 1 个任务的完成时间不应该超过 1 ms。如果 1 个任务超过 1 ms,则必须进行任务的分割,具体办法在第四部分“RTOS 的定时处理”中有详细的介绍。

三、消息的获取和发送

消息获取函数仅仅由 RTOS 使用。主要功能是检查消息队列:如果消息队列为空,则置 CY = 1 后立即返回;若消息队列非空,则取出最“旧”的那一条消息(消息队列的排队规则遵循先入先出的规则 FIFO)。TASKID 放在寄存器 A 中;COMMAND、PARA1、PARA2 放在“当前消息区”MSGARA 中。

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;名称: GET_MSG
;功能:操作系统从消息队列中取消息(如果消息队列不为空),然后把取出的消息放置当前消息区中
;入口参数:无
;出口参数:  (TASKID), MSGARA(COMMAND, PARA1, PARA2),
;           CY:当 CY = 1,表示消息队列为空
;寄存器使用:A, B, PSW, R0, DPH, DPL
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
GET_MSG:
   PUSH    PSW
   MOV     A, MSGCNT
   CLR     C
   SUBB    A, #01H
   JC      GET_MSG99                  ;消息队列空
   MOV     A, RPTR                    ;测试消息队列的读指针是否越界
   ADD     A, # MSGSIZE
   CLR     C
   SUBB    A, # MSGSIZE * MSGTOTAL
   JC      GET_MSG10
```

```
    ;读指针越界
      MOV    A, #0
      MOV    RPTR, A                    ;重新调整读指针到队列头
      POP    PSW
    GET_MSG10:
      MOV    A, R0
      PUSH   ACC
      MOV    A, #MSGBUFFER
      ADD    A, RPTR
      MOV    R0, A
      MOV    A, @R0                     ;TASKID -> A
      INC    R0
      MOV    MSGARA, @R0                ;COMMAND -> (MSGARA)
      INC    R0
      MOV    MSGARA+1, @R0              ;PARA1 -> (MSGARA+1)
      INC    R0
      MOV    MSGARA+2, @R0              ;PARA2 -> (MSGARA+2)
      DEC    MSGCNT
      CLR    C
      POP    ACC
      MOV    R0, A
    GET_MSG99:
      RET
```

消息发送函数 SND_MSG 在 RTOS 中被声明成 1 个 PUBLIC 的函数,供外部模块调用。消息发送函数只是负责向消息队列发送消息,入口参数 TASKID、COMMAND、DPH、DPL 分别放在寄存器 A、B、DPH、DPL 中。需要特别注意的是,在消息发送过程中,必须禁止任何中断的发生,以免在发送消息到消息队列的过程被中断,同时在中断返回前,中断服务子程序又可能向消息队列发送新的消息。这样就会破坏消息的完整性而带来灾难性的后果。

```
;*********************************
;名称: SND_MSG
;功能:发送消息到操作系统的消息队列,该函数被外部模块调用
;入口参数:  (TASKID), B(COMMAND), DPH(PARA1), DPL(PARA2)
;出口参数:无
;寄存器使用: A, B, DPH, DPL, PSW, R0
;*********************************
SND_MSG:
  CLR    EA                             ;禁止中断
  PUSH   DPL
  MOV    DPL, R0
  PUSH   DPL
  PUSH   ACC
```

```
    PUSH   PSW
    MOV    A, WPTR                    ;测试消息队列的写指针是否越界
    ADD    A, #MSGSIZE
    CLR    C
    SUBB   A, #MSGSIZE*MSGTOTAL
    JC     SND_MSG10
;写指针已经越界
    MOV A, #0
    MOV WPTR, A                       ;重新调整写指针到队列头
    POP  PSW
SND_MSG10:
    MOV A, #MSGBUFFER                 ;消息队列的首地址
    ADD  A, WPTR
    MOV R0, A
    POP  ACC
    MOV @R0, A                        ;TASKID->消息队列
    INC  R0
    MOV @R0, B                        ;COMMAND ->消息队列
    INC  R0
    MOV @R0, DPH                      ; PARA1 ->消息队列
    INC  R0
    POP  B
    POP  DPL
    MOV @R0, DPL                      ; PARA2 ->消息队列
    INC  MSGCNT                       ;消息总数加1
    MOV R0, B                         ;恢复 R0 的原始值
    SETB EA                           ;开放中断
    RET
```

四、RTOS的定时处理

RTOS的定时处理是整个RTOS中相当重要的一个环节。我们选用T0作为RTOS的定时器,定时周期为1 ms。定时处理不仅提供系统时钟和各种不同的延时时间,而且对消息队列的操作、任务间的同步和长时任务(指执行时间超过1 ms的任务)的分割都必须使用到定时处理。

RTOS的定时处理提供的延时时间间隔都是以1 ms的定时周期作为基准,而提供给所需要的任务使用。举例来说,按键常常需要进行消除抖动的处理,延时时间通常在10 ms以上。在按键处理任务中,就有1个子函数KY_10 ms专门负责消除抖动的延时处理。

在RTOS中专门设置1个系统时钟CLOCK,由2个字节组成。高位字节为CLKH;低位字节为CLKL,用BCD码表示。每当1 ms定时时间到,CLOCK加1,可以提供1～9 999 ms的延时时间间隔。系统时钟CLOCK的结构如图3.7-2所示。

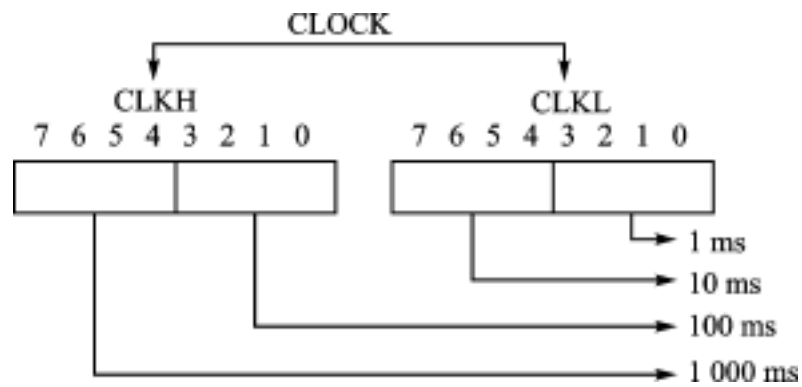


图 3.7-2 系统时钟 CLOCK 的结构

RTOS 定周期处理子程序:

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;名称: OS_CYCLE
;功能: OS 定周期处理子程序
;入口参数:无
;出口参数: A,DPTR
;寄存器使用: A, DPTR
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
OS_CYCLE:
   MOV    A, CLKL
   ADD    A, #1
   DA     A                          ;二至十进制调整
   MOV    CLKL, A
   MOV    A, CLKH
   ADDC   A, #0
   DA     A                          ;二至十进制调整
   MOV    CLKH, A
   MOV    A, CLKL
   ANL    A, #0FH
   MOV    DPTR, #CYCLE_TBL
   LCALL  TBL_JMP                    ;跳转到相应的延时处理
   RET
```

长时任务的分割也常常使用到定时处理。如果 RTOS 的定时基准设定为 1 ms,那么,1 个任务的完成时间(实际上精确地说应该是 RTOS 的 1 次循环加上可能发生中断处理的时间总和)不能超过 2 ms,否则会使系统定时周期少 1 ms。这样就必须把 1 个长时任务分割成一个个短时任务,每个短时任务执行时间不超过 1 ms。前一个短时任务返回之前向 RTOS 发送一条消息,TASKID 仍为该任务的 TASKID。但是 COMMAND 参数变为下一个短时任务的 COMMAND,PARA1 和 PARA2 则存放下一个短时任务执行时所需要的参数。当 RTOS 从消息队列中取出这条消息时,就可以唤醒下一个短时任务继续执行,这样直到该长时任务执行完毕。

五、核心管理模块

核心管理模块是基于RTOS上的一个重要任务模块。主要功能是记录当前RTOS的一些关键参数和状态;提供一些公共功能函数供外部任务调用;协调各个任务的运行。限于篇幅,这里不作详细的介绍。

本文介绍的基于非抢先式消息驱动机制的RTOS编译后的二进制代码只有360字节左右(注:不含核心管理模块),占用内部RAM大约60字节(消息队列为10个消息,位于内部RAM中)。该RTOS已经应用于实际系统,在提高软件的模块化、增强软件运行的可靠性等方面有很大的优越性。

参 考 文 献

1 屠祈,屠立德.操作系统基础.北京:清华大学出版社
2 马忠梅,籍顺心,张凯,马岩.单片机的C语言应用程序设计.北京:北京航空航天大学出版社
3 余永权.ATMEL 89系列(MCS-51兼容)Flash单片机原理及应用.北京:电子工业出版社
4 张国峰.Windows应用程序设计——原理、方法和技巧.北京:电子工业出版社
5 Franklin Software, Inc.RTX51 REFERENCE GUIDE

选自《单片机与嵌入式系统应用》月刊,2001年第5期

3.8 用结构化程序设计思想指导汇编语言开发

浙江师范大学计算机学院(321004) 吕振洪

一、引 言

单片机汇编语言是比较难学的语言,一方面由于它直接跟各种复杂的硬件环境打交道;另一方面汇编语言是一种低级语言,没有一套完整的编程理论来指导开发,源程序可读性差,初学者学好、学精有难度,协同开发较困难,更不用说调试程序了。正是由于此,单片机开发系统中出现了 C、Basic 系列的开发语言,它们帮助开发人员减轻了编程工作量的同时,却增加了开发成本,如增加 RAM 等。

但是,高级语言中的结构化、模块化程序设计思想是指导编程的比较完整的理论,它规范了编程,增加了程序的可读性,利于调试和开发人员协同开发,大大提高了开发效率。而把高级语言翻译成低级语言或汇编语言的编译程序,利用属性文法,用语法指导翻译法实现高级语言到低级语言的转化。如果我们把这思想加以消化,可用于指导我们的单片机语言的程序开发,可作为一种开发的规范或约束。有了该规范、约束,我们可从其他途径来提高汇编语言的开发效率和程序的可读性。

二、高级语言中的结构化语句和对应的汇编语言框架

高级语言中的结构化语句有三类:顺序语句、条件控制语句、循环语句。其中顺序语句主要包含于程序调用、赋值语句等,这一类跟汇编语言的类似,不是我们叙述的范围。下面我们以 C 语言为蓝本,列出结构化语句和对应的汇编语言框架,最后给出条件表达式部分的转化问题。为不失一般性我们用 jop 表示条件转移指令如 cjne、djnz 等,用 goto 表示无条件转移指令如 jmp、ajmp、ljmp、sjmp 等。

1. 条件控制语句

条件控制语句有两种:if 语句、多路选择语句。

(1) if 语句

① 无 else 部分的 if 语句

高级语言格式	汇编语言框架
if (条件表达式) 语句;	jop (第一操作数)[,(第二操作数)],endif 语句指令序列 endif:

② 带 else 部分的 if 语句

高级语言格式	汇编语言框架
if 条件表达式) 语句 1; else 语句 2;	jop 第一操作数)[,(第二操作数)],else 语句 1 指令序列 goto endif else: 语句 2 指令序列 endif:

(2) 多路选择语句

汇编语言中已经提供一种非常紧凑的多路选择语句,它可以通过跳转表和查表程序来共同实现,它的代码质量非常高,是一种常用的方法。这里我们提供另外一种通用形式。

高级语言格式	汇编语言框架
switch(选择子(E)) {case c_1: 语句 1; break; case c_2: 语句 2; break; … default: 语句 n; } 注:c_1、c_2…表常数	mov a,选择子 E 值 L_1: jne a,c_1,L_2 语句 1 指令序列 goto EndCase L_2: jne a,c_2,L_3 语句 1 指令序列 goto EndCase … L_{n-1}: jne a,c_{n-1},L_n 语句 n - 1 指令序列 goto EndCase L_n: 语句 n 指令序列 EndCase:

2.循环语句

循环语句有三种:while 语句、do…while 语句、for 语句。

(1) while 语句

高级语言格式	汇编语言框架
while(条件表达式) 语句;	Whilebegin: jop 第一操作数)[,(第二操作数)],endwhile 语句指令序列 goto whilebegin endwhile:

(2) do…while 语句

高级语言格式	汇编语言框架
Do 语句; while(条件表达式);	WhileBegin: 语句指令序列 jop(第一操作数)[,(第二操作数)],WhileBegin

这种形式的循环在汇编里是最常见的。

(3) for 语句

高级语言格式	汇编语言框架
for(i = n_0;i < n_1;i + +) 语句; 注:n_0、n_1 表常量或变量值,当循环结束条件改变时,可把右边的 cjne 条件转移指令用适当的指令替换	mow ,n_0 goto Over Again: inc a Over: cjne a,n_1,Continue Goto Endfor Continue: 语句指令序列 Goto Again EndFor:

(4) 循环转化的特殊性

上述列举的转化是一般的形式,考虑到汇编的指令系统时,这些转化不一定是最优的。有时只要根据具体的循环形式和条件,结合指令系统,稍做处理,可使程序更优。我们可以看下面一个例子。

高级语言格式	汇编语言框架
for(i = n_0;i < 0;i--) 语句; 注:n_0 表常量或变量值,n_0 > = 1	mov ,n_0 Continue: 语句指令序列 djnz a,Continue

另外,for、do…while 形式的循环可认为是 while 循环的特例,因为,只要能用 for、do…while 实现的,一定能用 while 循环来实现。

3.条件表达式转化成汇编指令序列

前面叙述中我们对高级语言中的条件表达式进行了简化处理,在实际应用中条件表达式可能是比较复杂的。由于 M51 汇编语言没有直接提供完备的关系比较的条件转移指令,像处理大于、小于等条件判断稍微有点麻烦。如判断 a1 > b1 可用下列指令流:

```
clr
mvo  A,a1
subb A,b1
jnc  标号
```

如判断 a1 > = b1,把上述的 jnc 改成 jc 即可。

由于把高级语言的条件表达式转化为汇编语言指令序列需要编译原理专业知识,我们在此就不叙述其原理,有兴趣的读者可查阅有关编译原理的书籍。这里我们只简要说明我们如何借鉴编译原理中把条件表达式翻译为条件控制指令流的方法,来优化的程序。优化的关键是转移语句应跳转到上下文的何处。我们用两个例子来说明:

高级语言格式	汇编语言框架

a .a1 > b1 & & c1 > d1(" & & "表" 与 ")
说明:① 当 a1≥b1 时,意味着" 与 "表达式为假,不必继续 c1 > d1 部分;② 当判断 c1 > d1 时,应该满足 a1 > b1
	lc
	mov ,a1
	subb a,b1
	jc _False;" 与 "表达式为假,转出
	clc
	mov a,c1
	subb a,d1
	jc EndJudge
	;下面是为真的情况
	goto…
	False:

b .a1 > b1 ‖ c1 > d1(" ‖ "表" 或 ")
说明:① 当 a1 > b1 时,意谓着" 或 "表达式为真,不必继续判断 c1 > d1 部分;② 当判断 c1 > d1 时,必须满足 a1≤b1
	lc
	mov ,a1
	subb a,b1
	jnc _True;" 或 "表达式为真,转出
	clc
	mov a,c1
	subb a,d1
	jnc _True
	;下面是为假的情况
	goto…
	True:

三、转化的意义

如果能把转化方法再做总结、归纳,我们可以利用编译技术,开发一些汇编语言的编程工具。一种方法是利用导向概念,开发出可生成各种条件控制语句、循环语句的汇编语言框架的逻辑编辑器;另一种方法是,定义一个类 C 的简单语言(包括上述的条件控制语句、循环语句等),开发时用类 C 的语言编程,利用预处理技术把类 C 的程序转化成汇编语言程序。笔者用后一种方法进行了尝试,效果比较理想。

参 考 文 献

1 吕能元,孙育才,杨峰编著 .MCS-51 单片微型计算机原理 接口技术 应用实例 北京:科学出版社,1993

2 吕映芝,张素琴,蒋维杜编著 编译原理 北京:清华大学出版社,1998

选自《自动化技术与应用》双月刊,2001 年第 1 期

3.9 单片机高级语言 C51 与汇编语言 ASM51 的通用接口

荆州江汉石油学院电子信息工程系(434102) 徐爱钧 彭秀华

随着计算机技术的发展和集成电路工艺的改进,广泛用于工业测控领域的单片机性能日益提高,对于单片机应用系统的开发要求已从单纯追求程序代码优化、硬件操作方便,发展到提高编程效率、实时多任务并行操作等现代化编程手段,采用高级语言编程已成为当前单片机应用系统的发展趋势[1]。单片机用于测控领域内某些场合汇编语言编程有其独特的优点,因此采用高级语言时要充分考虑与汇编语言的接口。本文基于在国内应用较为广泛的 Franklin C51 编译器讨论单片机高级语言与汇编语言之间通用接口程序设计原理和方法。

一、C51 编译器对目标程序的段管理

段是程序代码或数据对象的存储器单位,分为绝对段和再定位段。在 C51 编译器对 C 语言源程序编译所生成的目标程序中均为再定位段,段中代码或数据的存储器地址是浮动的,实际地址要由连接定位器在对目标程序进行连接时决定。再定位段可以保证在进行多模块程序连接时不会发生地址重叠现象。绝对段只能在汇编语言程序中指定,包括代码和数据的绝对地址说明,用于访问某个特定的存储器单元或 I/O 端口,或是提供某个中断向量的入口地址等。

C51 编译器为各模块中的每个函数生成一个以?PR?function_name?modul_name 为名的代码段。如果函数包含非寄存器传递的参数和无明确存储器类型声明的局部变量,C51 编译器还将生成一个字节类型的局部数据段和一个位类型的局部位段。局部数据段用于存放除位类型以外的所有其他无明确存储器类型声明的局部变量和参数,局部位段用于存放在函数内部定义的可再定位的位类型变量和参数。这些段的命名规则与函数的存储器模式有关[2],如表 3.9-1 所列。

表 3.9-1 在不同存储器模式下的段名

存储器模式	局部段类型	段描述	段 名
SMALL	CODE	函数代码	?PR?function_name modul_name
	DATA	局部数据段	?DT?function_name modul_name
	BIT	局部位段	?BI?function_name modul_name
COM-PACT	CODE	函数代码	?PR?function_name modul_name
	PDATA	局部数据段	?PD?function_name modul_name
	BIT	局部位段	?BI?function_name modul_name
LARGE	CODE	函数代码	?PR?function_name modul_name
	XDATA	局部数据段	?XD?function_name modul_name
	BIT	局部位段	?BI?function_name modul_name

C51 编译器对函数的局部变量采用覆盖技术,以提高 8051 单片机内部 RAM 的利用效率,为此对局部数据和局部位段建立一个可覆盖标志“OVERLAYABLE”,以便让连接定位器在对目标程序进行连接定位时作覆盖分析之用。每个局部段都有一个别名,别名的命名规则如下:

函数代码段　　function_name

局部数据段　　?function_name?BYTE

局部位段　　?function_name?BIT

每个段别名表示该段的起始地址。局部段的别名是全局共享的,因此它们的起始地址可被其他模块访问。为了实现与高级语言的正确接口,汇编语言程序应按 C51 编译器的规定建立相同的代码段和数据段。

二、C51 程序与汇编语言程序接口时的参数传递

C51 程序与汇编语言程序的接口可视为函数的调用,将汇编语言子程序作为一种外部函数。函数调用要考虑参数的传递及函数的返回值,C51 程序函数和汇编语言函数的相互调用时,可利用 8051 单片机内部工作寄存器最多传递 3 个参数,如表 3.9-2 所列。调用时如果参数较少,利用工作寄存器可以很方便地完成参数传递,如果在调用时参数较多以至于工作寄存器不够用,或是在编译时强制不使用工作寄存器,则参数的传递将发生在固定的存储器区域,称为参数传递段,其地址空间取决于编译时所选择的存储器模式。如果传递的参数是 char、int、long 和 float 类型的数据,参数传递段以 PUBLIC 符号"?_functionname?BYTE"作为首地址,若传递的参数是 bit 类型的数据,参数传递段以 PUBLIC 符号:"?_functionname?BIT"作为首地址,所有被传递的参数都被放在以首地址开始递增的存储器区域内。

表 3.9-2　参数传递时工作寄存器选择

传递的参数	char、1 字节指针	int、2 字节指针	long、float	一般指针
第一个参数	R7	R6,R7	R4～R7	R1,R2,R3
第二个参数	R5	R4,R5	R4～R7	R1,R2,R3
第三个参数	R3	R2,R3	无	R1,R2,R3

与 C51 程序接口时,被调用的外部汇编语言函数所有段名都应以 C51 编译器所规定的方法来建立;汇编语言函数中的局部变量应指定自己的局部数据段作为调用时的参数传递段,在局部数据段中先按被传递参数的顺序定义若干字节,然后再定义其他局部变量。调用结束后函数的返回值被放入 8051 单片机内部工作寄存器内,如表 3.9-3 所列。

表 3.9-3　函数返回值所占用的工作寄存器

返回值类型	寄存器	说　明
bit	进位位 C	
(unsigned) char	R7	
(unsigned) int	R6,R7	高位在 R6 中,低位在 R7 中
(unsigned) long	R4～R7	高位在 R4 中,低位在 R7 中
float	R4～R7	32 位 IEEE 格式,指数和符号位在 R7 中
一般指针	R1,R2,R3	R3 放存储器类型,高位在 R2 中,低位在 R1 中

三、编程实例

下面给出一个在 SMALL 编译模式下 C51 程序调用汇编语言函数的例子来说明具体调用过程。在 C51 程序文件“C_FUNC .C”中定义了一个被调用的外部汇编语言函数:

```
extern int afunc (int v_a, char v_b, long v_c, bit v_d)
```

汇编语言函数接受从 C51 程序传递过来的参数 v_a、v_b、v_c 和 v_d,并依次放入它自己的局部变量 a、b、c 和 d 中。函数调用时需要传递 4 个参数,按表 2 规定只有参数 v_a 在寄存器 R6、R7 中传递(高位在 R6,低位在 R7),参数 v_b 在 R5 中传递,而参数 v_c 和 v_d 则将在参数传递段中传递。在 SMALL 编译模式下,参数传递段位于 8051 单片机内部数据存储器 DATA 区,因此汇编语言函数应按 C51 编译器关于 SMALL 模式下段名的规定在 DATA 区建立相应的局部数据段。由于被调函数 afunc()具有 int 类型,按表 3 9－3 规定函数的返回值将在工作寄存器 R6(高位)、R7(低位)中。

高级语言程序 C_FUNC .C 文件如下:

```
 # pragma code small
/ * 说明被调用的外部汇编语言函数 */
extern int afunc (int v_a, char v_b,
   long v_c, bit v_d);
void main(){
/ * 说明函数调用参数 */
int v_a; char v_a;
   long v_c;bit v_d;
int A_ret;
/ * 带参数调用 */
A_ret = afunc (v_a, v_b, v_c, v_d);
}
```

汇编语言函数 AFUNC .A51 程序文件如下:

```
NAME AFUNC;定义函数模块名
;定义程序代码段
? PR ? _afunc ? AFUNC SEGMENT CODE
;定义可覆盖局部数据段和局部位段
? DT ? _afunc ? AFUNC SEGMENT DATA OVERLAYABLE
? BI ? _afunc ? AFUNC SEGMENT BIT OVERLAYABLE
;定义公共符号
PUBLIC ? _afunc ? BIT
PUBLIC ? _afunc ? BYTE
PUBLIC_afunc
   RSEG ? DT ? _afunc ? AFUNC      可覆盖局部数据段
? _afunc ? BYTE:                  ;起始地址
   v_a:       S                   定义参数传递字节
```

```
  v_b:     DS   1
  v_c      DS   4
  a:       DS   2                ;定义其他局部变量
  b:       DS   1
  c:       DS   4
  retval:  DS   2
 RSEG ? BI ? afunc ? AFUNC      ;可覆盖局部位段
? afunc ? BIT:                  ;起始地址
    v_d:  DBIT 1                ;定义参数传递位
      d:  DBIT 1                ;定义其他局部位变量
  RSEG  ?PR ? afunc ? A FUNC    ;程序代码段
USING    0                      ;定义工作寄存器组
afunc:                          ;起始地址
  MOV    a,R6                   ;a = v_a
  MVO    a + 01H,R7
  MOV    b,R5                   ;b = v_b
  MOV    c + 03H,v_c + 03H      ;c = v_c
  MOV    c + 02H,v_c + 02H
  MOV    c + 01H,v_c + 01H
  MOV    c,v_c
  MOV    C,v_d                  ;d = v_d
  MOV    d,C
  MOV    R6,retval              ;返回值高位
  MOV    R7,retval + 01H        ;返回值低位
  RET
END
```

对以上两个模块文件 C_FUNC .C 和 AFUNC .A51 分别用 C51 编译器和 A51 汇编器进行编译和汇编,再对所生成的目标文件用连接定位器 L51 进行连接,即可生成一个完整的应用程序。上述编程原理可推广到多模块程序设计,需要时可调用多个外部汇编语言函数。按以上规则实现高级语言和汇编语言之间的接口,可充分发挥两者的优点,将复杂的数学计算、多任务管理等交给高级语言完成,而系统底层的硬件操作则由汇编语言完成,两者有机结合必将使单片机应用系统的开发和研制上升到一个新的台阶。

四、编译控制命令 SRC 及嵌入式在线行汇编的应用

C51 编译器提供了一个十分有用的编译控制命令“SRC”,在编写汇编语言函数时可先按需要用 C 语言写一个相应的函数模块文件,并采用控制命令“SRC”进行编译,编译完成后将产生一个汇编语言源程序。然后再对这样产生的汇编语言程序作一些必要调整和修改,即可很方便地完成汇编语言程序的编写,而编写过程中各种段的安排全部由 C51 编译器自动完成,从而大大提高汇编语言程序的编写效率。另外 V3 .96 以上版本的 C51 编译器还支持嵌入式在线行汇编,可以直接在 C51 程序中插入汇编指令,这种方法对于编写某些具有特殊要求的汇编语言程序特别有用。

对于上面例子中的汇编语言函数可用如下 C51 函数模块文件 AFUNC .C 来实现:

```
#pragma SRC (AFUNC .A51) small                    * 使用 SRC 控制 */
int afunc (v_a, v_b, v_c, v_d)                   /* 函数定义 */
  int v_a, char v_b, long v_c, bit v_d
  {int a; char b; long c; bit d;
   int retval;
   a = v_a; b = v_b; c = v_c; d = v_d;            /* 参数传递 */
    #pragma ASM                                   /* 插入在线行汇编 */
    MOV P1,R5
    NOP
    NOP
    MOV P1, #0
    #pragma ENDASM
    return (retval);                              /* 函数返回 */
  }
```

对模块文件 AFUNC .C 用 C51 进行编译即可得到一个名为 AFUNC .A51 的汇编语言程序,需要注意的是,编译时采用的编译模式必须与编译 C51 程序文件 C_FUNC .C 的编译模式相同,在调用有参函数时这一点是十分重要的。如果两个文件采用不同的编译模式,将导致它们采用不同的存储器区域作为参数传递段空间而不能完成正确的参数传递。另外对于这样产生的汇编语言程序作适当调整也是十分必要的,例如为实现精确定时功能的外部汇编语言函数必须仔细计算各条汇编指令的机器周期,消除多余指令等[3]。

五、结束语

本文介绍了单片机高级语言 C51 与汇编语言 ASM51 之间通用接口的基本原理和实现方法,在研制单片机应用系统时,利用该接口可很方便地实现 C51 和 ASM51 程序之间的相互调用,对于更为有效地进行单片机应用系统软件设计十分有益。我们最近在研究项目"单片机电能管理远程通信系统"的软件设计中,采用 C51 编写前台管理程序,采用 ASM51 编写后台采样和实时时钟程序,利用本文介绍的通用接口实现 C51 程序与 ASM51 程序之间的相互调用及参数交换,极大地提高了编程效率,同时可十分方便地操作系统硬件,而且程序具有通用性,便于移植。实践证明,本文介绍的通用接口简单实用,具有推广价值。

参 考 文 献

1 徐爱钧,彭秀华 .单片机高级语言 C51 应用程序设计 .北京:电子工业出版社,1998

2 Keil Software . The Final World On The 8051 . Germany; Keil Elektronik Gmbh and Keil software, 1997, 88～104

3 肖培林 .用 C 语言开发 51 系列单片机高效代码 .电子技术应用,1996,22(3):13～15

选自《微计算机应用》双月刊,2000 年第 1 期

3 .10　ASM51 无参数化调用 C51 函数的实现

成都西南交通大学计算机与通信工程学院(610031)　苟　帅

MCS－51 系列单片机在目前和今后的相当一段时间内都将是我国的单片机主流机种。但在早期的开发过程中,程序员不得不从深奥的汇编语言开始搜索,同时要求开发人员对硬件亦有相当的了解。相比而言,专为 8051 系列单片机设计的 Franklin C51 语言是一种通用的高级结构化的程序设计语言。入门容易,程序可读性强,调试、移植都很方便,故开发效率高,尤其在数值运算处理方面具有很大的优势(这正是 ASM51 汇编语言的薄弱环节)。不过,C 语言虽然也可对计算机的硬件系统进行操作,但在处理特殊 I/O 口和中断向量方面,不如汇编那样直接、有效。因而,在效率为重的今天,将 ASM51 汇编与 C51 语言结合起来,充分发挥各自的优势,无疑是单片机开发人员的最佳选择。

一、汇编与 C51 的混合编程

一般的做法都是利用 C51 上手容易、便于理解的优势来编写主程序,在 C51 语言不便处理或者效率比较低时调用汇编函数。考虑到 MCS－51(尤其是 8031)内部的资源配置情况:可用的 RAM 不到 256 B,5 个固定地址的有限中断源,4 个 8 位并口中实际可作 I/O 口的只有 P1 口。因而要求开发者对单片机的内部结构有清楚的了解,并尽可能地统筹安排这些资源。事实也证明,不理解汇编语言是很难写出高效程序的。故笔者的观点是利用汇编语言对 I/O 接口、中断向量及程序空间分配的巨大优势,让程序员对 MCS－51 内的每一个字节甚至是每一比特(可位寻址的空间)全部进行统筹安排,设计好各个程序模块,包括 I/O 口地址和中断向量地址的处理;同时在具体的数据处理、通信等不需要过多与硬件直接打交道的程序模块中,充分利用 C51 语言强大高效的编程能力。

最后的关键是如何让汇编模块能够正确识别 C51 函数并调用它来完成相应的功能。ASM51 汇编与 C51 语言之间的调用约定并不简单,而且各种编译器使用的约定不尽相同,甚至还依赖于程序所选择的大、中、小存储模式。通常每个需传递的参数按调用顺序和类型分别由约定的寄存器来传递。如果参数过多或者无足够寄存器可用时,参数的传递将在固定的存储器区域内进行,相同类型的参数共享一个参数传递段,按参数调用顺序递增其存放地址,返回值也由约定的寄存器或地址段返回。由此可见程序调用的效率必将受到接口复杂度的影响。尽管目前的单片机仿真器已经提供了标准接口的全自动转换功能,减少了接口工作量,但在程序的调试及移植中,如果程序员不了解这些接口的各种约定,将对出现的错误不知所措。比如返回值不止一个时,编译器自己就无法正确完成接口配置。这里力荐一种简洁有效的调用方法——无参数化调用。

二、ASM51 无参数化调用 C51 函数的实现原理

所谓的无参数化调用是指让 C51 子函数不带任何参数,这样就可以从根本上避开调用参

数的传递和返回值的安排等繁琐易出错的问题,只需要简单地在汇编语言开头说明一下外部C51子函数(“EXTRN code(<C51模块名称>)”)。至于C51函数中需要使用的外部参数值及其返回值,完全可以通过加入C51的<absacc.h>头文件来解决。

<absacc.h>头文件中的函数原型为:

```
#  define CBYTE((unsigned char *)0x50000L)
#  define DBYTE((unsigned char *)0x50000L)
#  define PBYTE((unsigned char *)0x50000L)
#  define XBYTE((unsigned char *)0x50000L)
```

其中CBYTE定义为寻址CODE程序区;DBYTE定义为寻址DATA数据区;PBYTE定义为寻址相对于MOVX @R0"指令的分页数据XDATA区;XBYTE定义为寻址相对于MOVX @DPTR"指令的分页数据XDATA区。它们的类型决定了绝对地址空间的位置。

引进该头文件后,程序员就可以对8051系列单片机的存储器进行绝对地址的访问,把对参数值和返回值的操作转化为对存储器绝对地址的操作,像纯汇编操作一样,根本不用定义C51函数与汇编接口的参数和返回值的配置,从而提高了调用效率。具体做法是:先在C51函数中定义好传递参数和返回值所需要的各个绝对地址(视程序员自己的空间配置而定),在其他汇编模块中将C51函数中将要使用的参数值放入这些绝对地址中,把被调用C51模块中将输出的计算值(可以不止一个)也放入类似的绝对地址中。于是,当C51函数中需要使用某个参数值时,就直接从相应的绝对地址中读取该值;当别的汇编模块中需要使用C51函数的返回值时,也直接对存放返回值的绝对地址进行读操作即可。下面以一个调试通过的汇编调用C51函数的简单程序为例进行具体说明。

三、ASM51无参数化调用C51函数的实现示例

该系统要求用单片机根据实时采样输入的转速实现机车速度的测量,并可随键盘输入的车轮直径变化实时调整车速,最后将车速和轮径值都显示出来。设计任务很简单,编程中的最大难度就在于车速的计算程序。由于轮径值要求精确到mm(最大值超过了1 000),车速的计算结果要保留到小数点后一位,因此需要进行浮点数运算,期间还要完成数的各种进制间的换算。虽然算法简单,但实际用汇编语言实现起来经常考虑不周,调试起来费时费力(笔者调试通过的这段汇编代码长达近400行)。这样,自然就想到调用C51函数了,充分发挥两种语言的优势。先用汇编语言设计好各个模块,包括循环显示车速和轮径值的主程序模块,响应采样转速值和键盘输入两个中断模块,代码如下所示:

```
EXTRN   CO  E(CALL1)                    声明外部C51函数
    ORG     0000H
    LJMP    MAIN
    ORG     0003H
    AJMP    KEYINPUT                    ;键盘输入中断
    ORG     000BH
    AJMP    SETTIME                     ;采样时间到,采样转速值中断
    ORG     0100H
KEYINPUT:……                            ;键盘输入中断
```

```
            ……                    ;将键盘输入信号保存在 70H～73H 的地址空间中
    RETI
    ORG     0600H
SETTIME:    ……                    ;采样时间到,采样转速值中断
            ……                    ;将转速值放置在地址为 3AH 的空间中紧接着调用外
                                  ;部 C51 函数 CALL1()进行车速的计算
    LCALL   CALL1
    RETI
    ORG     2000H                 ;主程序模块
MAIN:       ……                    ;首先进行初始化操作
            ……                    ;直接从地址空间 70H～77H 中读取显示数据,循环显
                                  ;示车速和轮径值
END
```

这些小模块用汇编实现起来不仅容易,而且程序员可以清楚地了解到各个模块的出入口及其相应的功能,实现对程序空间的充分配置。最后用 C51 语言来实现车速的计算模块 CALL1()。以前用汇编编写的近 400 行代码,一下子被压缩到 20～30 行(真正的计算代码仅 9 行),不仅简短易懂,而且几乎就不需要调试了。

下面的代码是计算模块 CALL1()及其需要的绝对地址定义。

```
#pragma     code  small
#include    <absacc.h>
#include    <math.h>
#define     PI    3.1415926
#define     NCIRCLE   DBYTE[0x3A]       ;定义放置转速的绝对地址
#define     DIRECT1   DBYTE[0x70]       ;定义放置轮径千位的绝对地址
#define     DIRECT2   DBYTE[0x71]       ;定义放置轮径百位的绝对地址
#define     DIRECT3   DBYTE[0x72]       ;定义放置轮径十位的绝对地址
#define     DIRECT4   DBYTE[0x73]       ;定义放置轮径个位的绝对地址
#define     VELOCITY1   DBYTE[0x74]     ;定义返回车速的千位绝对地址
#define     VELOCITY2   DBYTE[0x75]     ;定义返回车速的百位绝对地址
#define     VELOCITY3   DBYTE[0x76]     ;定义返回车速的十位绝对地址
#define     VELOCITY4   DBYTE[0x77]     ;定义返回车速的个位绝对地址
void  call1()
{
    float data result;
    int data DIRECT;
DIRECT = DIRECT1 * 1000 + DIRECT2 * 100 + DIRECT3 * 10 + DIRECT4;
result = (DIRECT/1000.0) * PI * NCIRCLE * 3.6;
VELOCITY1 = result/100;
result = result - VELOCITY1 * 100;
VELOCITY2 = result/10;
result = result - VELOCITY2 * 10;
```

```
VELOCITY3 = result;
result = result - VELOCITY3;
VELOCITY4 = result * 10;
}
```

在本例中定义了绝对地址空间 70H～77H 和 3AH。其中 3AH 存放采样转速值输入模块输入的转速;70H～73H 的地址空间中存放键盘输入中断模块中键盘输入的轮径值;而地址为 74H～77H 的空间中则存放计算模块中的车速计算返回值。尽管需要传递和返回的参数比较多,但通过这些绝对地址的定义,完全解决了原来复杂的汇编与 C51 之间的调用接口配置。计算模块中需要使用转速和轮径值时,将自动从绝对地址 3AH 和 70H～73H 中取值;在循环显示车速和轮径值的主程序模块中则直接读取绝对地址空间 70H～77H 的各个数据进行循环显示。当然,程序员可以根据自己的空间配置另外定义这些绝对地址。

以上程序代码均已在 Dais－52 .196P 仿真器上顺利调试通过。

由上面的简单程序可以看出这种无参数化调用方法的优越性和有效性:从程序代码看,无论是编写 C51 子程序还是汇编主程序,都与编写纯 C51 函数或者纯汇编主程序的格式完全一样,从根本上简化了 C51 与汇编函数之间的接口编程,提高了程序调用的效率;充分利用了汇编与高级 C51 语言各自的优点,开发、调试快速方便,通用性强,尤其适合于初学者。对于复杂程序,同样可以利用无参数化方法来帮助实现。这对于提高单片机应用程序的开发效率很有意义。

无参数化调用实质上在 C51 函数中定义了几个全局变量(绝对地址),依靠它们直接完成参数值的传递和返回值的调用,相当于一种程序员自定义的传递方式,抛弃了传统 C 与汇编之间的接口约定。只要程序员安排得当,还可以进一步人工实现 C51 中的动态覆盖重用,提高 RAM 区的利用效率。由上也可看出:无参数化调用方法要在 ASM 汇编调用 C51 函数时才能充分发挥其巨大优势;如果全部采用 C51 编程,过分依赖无参数化思想,就违背了利用汇编优势的初衷,得不偿失。当然,如果开发人员已经对 C51 与汇编函数之间的参数传递接口很熟悉,完全可以按接口约定或者由编译器自动完成参数的传递。

参 考 文 献

1 徐爱钧,彭秀华 .单片机高级语言 C51 应用程序设计 .北京:电子工业出版社,1998

2 Dias 系列组合式仿真编程器使用手册 .1999

选自《电子技术应用》月刊,2001 年第 7 期

3 .11　TMS320C3X 的汇编语言和 C 语言及混合编程技术

北京航空航天大学工程系统工程系(100083)
袁宏杰　李传日　殷雪岩

TMS320 系列处理器因其强大的指令系统、高速的数据处理能力,已成为理想的 DSP 应用器件。TMS320C3X 系列是 TI 公司推出的第三代数字信号处理芯片。TMS320C3X 采用了 1 μm 的 CMOS 工艺,每秒可执行 16 .7 M 条指令,其全部存储器空间为 16 M(32 位),数据和指令字的长度为 32 位。其多处理器接口,内部或外部产生的等待状态、2 个定时器和串行口及多重中断结构,使 TMS320C3X 处理器能够作为主处理器或专用的协处理器。

TMS320C3X 基于寄存器的结构、庞大的地址空间、功能强大的寻址方式、灵活的指令系统,使该处理器能够实现高级语言。TI 公司提供了 C 语言开发工具,由于 C 编译器的支持,使得在 PC 机上调好的应用程序增加了转换为 TMS320C3X 程序的能力,使开发者能够把注意力集中到算法实现上。对代码执行效率要求高的部分可采用汇编程序实现,采用 C 语言和汇编语言混合编程,可降低编程的工作量,减少 TMS320C3X 程序的调试时间,提高编程效率,便于维护和调试。

一、TMS320C3X 的汇编语言

TMS320C3X 系列用硬件实现其他微处理器用软件实现的功能,从而优化了自身的速度。其支持浮点运算,浮点运算能力高达 3 300 万次/ s。TMS320C3X 系列汇编语言指令丰富,有指令 110 多条。其并行指令、位翻转指令、延迟转移指令、块重复指令等特别设计的指令和功能强大的寻址方式非常适合傅立叶变换、数字滤波等数字信号处理的各种算法。

体系结构上,TMS320C3X 采用流水线工作方式,利用 5 个主要的功能部件(F、D、R、E、DMA)控制取指、译码、读取、执行和 DMA 操作。为了得到最大的处理能力,这些部件可以并行地进行取指、译码、读取、执行指令和 DMA 操作,这种流水线工作方式是 TMS320C3X 成为高性能器件的关键。DMA 控制器拥有自己内部的数据和地址总线,能更有效地利用 CPU,因而得到更高的性能。

TMS320C3X 的并行操作指令可以在一个单周期内同时执行两条指令,即可以实现一个乘法操作和加法操作并行执行,也可以实现一个存储操作和一个乘法或逻辑操作并行执行等。并行操作指令增加了单调期内执行的操作次数,提高了效率。

TMS320C3X 的 ALU 单元可完成 32 位整数运算、32 位逻辑运算、40 位浮点运算。其内部带有浮点/ 整数乘法器,当运行浮点乘法时,输入为 32 位浮点数,结果为 40 位的浮点数。TMS320C3X 以整数的运算速度实现浮点运算,浮点运算可以防止整数运算的溢出、减少操作数的调整等。

TMS320C3X 微处理器内部 64 个 32 位的指令高速缓冲存储器可极大减少片外存储器存取的次数,并允许把代码存储到片外低速、低价的存储器中,外部总线也因此能从程序取指中

解脱出来数。块重复指令使循环不需化费附加的跳转时间。一般的微处理器,执行正常转移指令要 4 个指令周期,而 TMS320C3X 的延迟转移指令可单周期执行。

数字信号处理中如卷积和数字滤波算法要求在存储器内进行循环缓冲,TMS320C3X 的循环寻址方式专门为此设计。TMS320C3X 的块大小寄存器 BK 确定了循环缓冲区的大小,用户选择的辅助寄存器 ARn 的内容加偏移量 DISP 确定了数据的位置。循环寻址的算法为

如果 0 < ARn + DISP < BK,则 ARn = ARn + DISP

如果 ARn + DISP > BK,则 ARn = ARn + DISP - BK

如果 ARn + DISP < 0,则 ARn = ARn + DISP + BK

数字滤波器是数字信号处理系统中一种最普遍的应用。TMS320C3X 可以有效实现数字滤波器的特点:一是并行乘法和加法运算,使之可以在单周期内完成一次乘法和加法运算;二是循环寻址,使长度为 N 的有限缓冲区足以满足数据的需要。

假设 FIR 滤波器的有限脉冲响应为

$$H(Z) = \sum_{i=0}^{N-1} H(N)Z^{-i} \tag{1}$$

输入为 X(i),输出为 Y(i),可得

$$Y(i) = \sum_{i=1}^{N} X(i)H(N - i) \tag{2}$$

式(2)为一卷积形式,在 TMS320C3X 中使用循环寻址实现式(2)是极其简单的。下面是实现 FIR 滤波器的汇编子程序,AR0 的初值指向 H(N - 1),AR1 的初值指向 X(0)。

```
LDI N,BK                                  / 装入数据块大小到 BK
LDI H,AR0                                 / / AR0 的初值指向 H(N - 1)
LDI X AR1                                 / / AR1 的初值指向 X(0)
TOP:LDF SAMPLEVALUE R3                    / / 采集的数据送到寄存器
STF R3, * AR1 + + %                       / / 采集的数据放入存储器
LDF 0,R3                                  / / 寄存器清零
LDF 0,R2
RPTS N - 1                                / / 重复 N 次
MPYF3 AR0 + 1%, AR1 + 1%, R0 ‖ ADF3 RO R2,R2   / / 执行 ∑(i=1..N) X(i)H(N - i)
STF R2,Y                                  / / 存储结果
B TOP                                     / / 跳转到 TOP 处,计算下一点
```

TMS320C3X 的许多特点特别适用 FFT 运算。其浮点运算能力防止整数运算的溢出问题、操作数的调整问题等。当 FFT 的数据以正常的次序传递,FFT 的最后结果次序是二进制位翻转次序,为恢复变换的数据存储器的数据需要交换,对这种重新排列,TMSC320C3X 不需要花费附加的周期,应用位翻转寻址方式就可以以正确的次序存取 FFT 的变换结果。TMSC3203X 强大的间接变址寻址方式可以方便地存取 FFT 蝶式迭代中的数据。应用块重复指令可大大缩短循环所花费的时间。限于篇幅不给出 FFT 变换的汇编源程序。表 3 .11 - 1 给出了 TMSC320C31 执行 FFT 所需要的时间。

表 3 .11 - 1　TMSC320C31 FFT 执行时间

点　数	运算时间/ ms	
	基 2 的复数	基 4 的复数
64	0 .167	0 .123
512	0 .801	0 .624
1 024	3 .75	3 .040

二、TMS320C3X 的 C 语言和汇编语言接口

1 . C 语言开发 TMS320C3X 程序的过程

TI 公司提供的 C 语言开发工具包括 AC30 语法器、OPT30 优化器、CG30 翻译器、ASM30 汇编器、CL30 编译器、RTS30 .LIB 运行库(大模式为 RTS30L .LIB)、LNK30 连接器、DB30 源代码调试器和大量的 DSP 运行库。

采用 C 语言开发程序过程如图 3 .11 - 1 所示。AC30 语法器对输入的 C 语言源程序进行语法检查,OPT30 优化器对输入的程序进行优化生成 .IF 文件,CG30 翻译器对生成的 .IF 文件翻译成汇编代码,生成 .ASM 文件,ASM30 汇编器对输入的汇编文件生成 .OBJ 文件。以上步骤也可用 CL30 编译器一步完成。LNK30 对 .OBJ 文件进行连接,对内存重新定位生成可执行的 .OUT 文件,ROM30 把 .OUT 文件转换为十六进制文件,此文件可加载到存储器中。

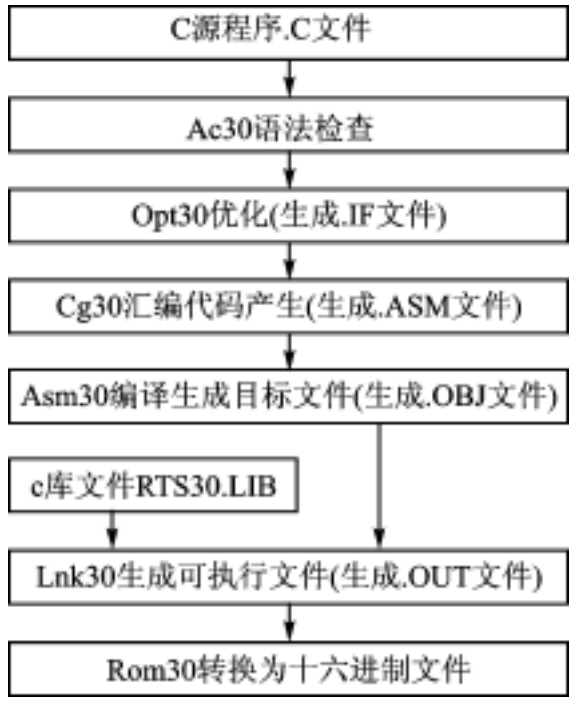


图 3 .11 - 1　C 语言开发 TMSC320C31 程序的过程

2 . 汇编语言调用 C 语言定义的变量

汇编语言和 C 语言相互调用时,调用的变量或函数必须声明为全局的,C 编译时自动在 C 变量和函数名前加一下画线前缀。

TMSC320C3X 浮点 C 语言支持小模式和大模式两种内存模式。小模式是系统隐含模式,其静态数据段 .BSS 最大为 64 K,也就是 TMSC320C31 的一页(一页的容量为 64 K),编译器把页指针设置为 .BSS 段处,在运算时不需调整页面指针 DP,在连接时应用 RTS30 .LIB 库。

如果在 C 语言定义了全局浮点变量 X;在汇编语言中可按如下调用

```
LDF @_X,R0                    / 把变量 X 的值送入寄存器 R0
```

大模式不限制静态数据段 .BSS 的大小,可以超过 64 K。在编译时对每一变量的操作调整页面指针 DP,在编译时应用 CL30 编译器的- MB 可选项,在连接时应用 RTS30L .LIB 库。

如果在 C 语言中定义了全局浮点变量 X;在汇编语言中可按如下调用

```
LDP_X                         //调整页面指针到变量所在的页面
LDF @_X,RO                    //把变量 X 的值送入寄存器 R0
```

3 .C 语言调用汇编语言定义的变量和函数

(1) C 语言调用定义在静态数据段 .BSS 的变量或函数

在汇编语言应在 .BSS 段定义变量(函数)为一全局变量(函数),其变量和函数名前加一下画线前缀(_)。

在 C 语言中,声明为 EXTERN 型变量(函数)即可。

下面给出了 C 语言调用定义在静态数据段 .BSS 变量的例子。

汇编语言程序:

```
.BSS_VAR,1                    //定义变量
.GLOBAL_VAR                   //声明变量为全局变量
```

C 语言程序:

```
EXTERN INT VAR                //声明外部变量
VAR = 1                       //应用此变量
```

(2) C 语言调用不在静态数据段 .BSS 定义的变量或函数

首先在汇编语言定义中应给定义的变量设置一个段,定义此变量为全局变量,在 C 语言中声明此变量为 EXTERN 型变量,用指针间接调用。下面给出了一个例子。

汇编语言程序:

```
.GLOBAL_SINE                  //为全局变量
.SECT“ SINE - TAB ”           //设置一个段_SINE
.FLOAT 1 .0                   //定义一数值表
.FLOAT 1 .2
.FLOAT 1 .4
```

C 语言程序:

```
EXTERN FLOAT SINE[]           //外部变量
FLOAT  * PSIN = SINE;         //设置一指向此变量的指针
FILOAT F;                     //定义一浮点变量 F
F = PSIN[1];                  //F 的值为 1 .2
```

4 .在线汇编

在 C 语言中可采用 ASM()语句在线插入一行汇编语言。

在线汇编提供了能直接读写硬件的能力,如读写中断控制允许寄存器等,但 C 编译器并不检查和分析在线汇编语言,插入在线汇编语言改变汇编环境或可能改变 C 变量的值可能导

致严重的错误。

5 . 寄存器的使用

编译器使用了部分寄存器作为保留寄存器,汇编语言程序在使用这些寄存器之前应保留这些寄存器的值,返回时应恢复这些寄存器的值。寄存器的用途如表 3 .11 - 2 所列。

表 3 .11 - 2　寄存器的用途

寄存器	用　途
R4～R5	整型变量
R6、R7	浮点变量
AR4～AR7	指针变量
DP	页面指针(大模式时)
SP	堆栈指针
AR3	程序指针

选自《测控技术》月刊,2000 年第 6 期

3 .12 TMS320C6000 嵌入式系统优化编程的研究

北京邮电大学电信工程学院无线技术研究室(100876)
赵训威 王东昱 张 平

一、TMS320C6000 的硬件设计和指令系统

TMS320C6000 系列 DSP(数字信号处理器)是 TI 公司最新推出的一种并行处理的数字信号处理器。它是基于 TI 的 VLIW 技术的,其中 TMS320C62XX 是定点处理器,TMS320C67XX 是浮点处理器。本文主要讨论 TMS320C6201。该处理器的最高工作频率可以采用 50 MHz,经内部 4 倍频后升至 200 MHz,每个时钟周期最多可以并行执行 8 条指令,从而可以实现 1 600 MIPS 的定点运算能力,而且完成 1024 定点 FFT 的时间只需 70 μs。

1 . TMS320C6000 的硬件结构

图 3 .12 - 1 是 TMS320C6000 CPU 的结构图。

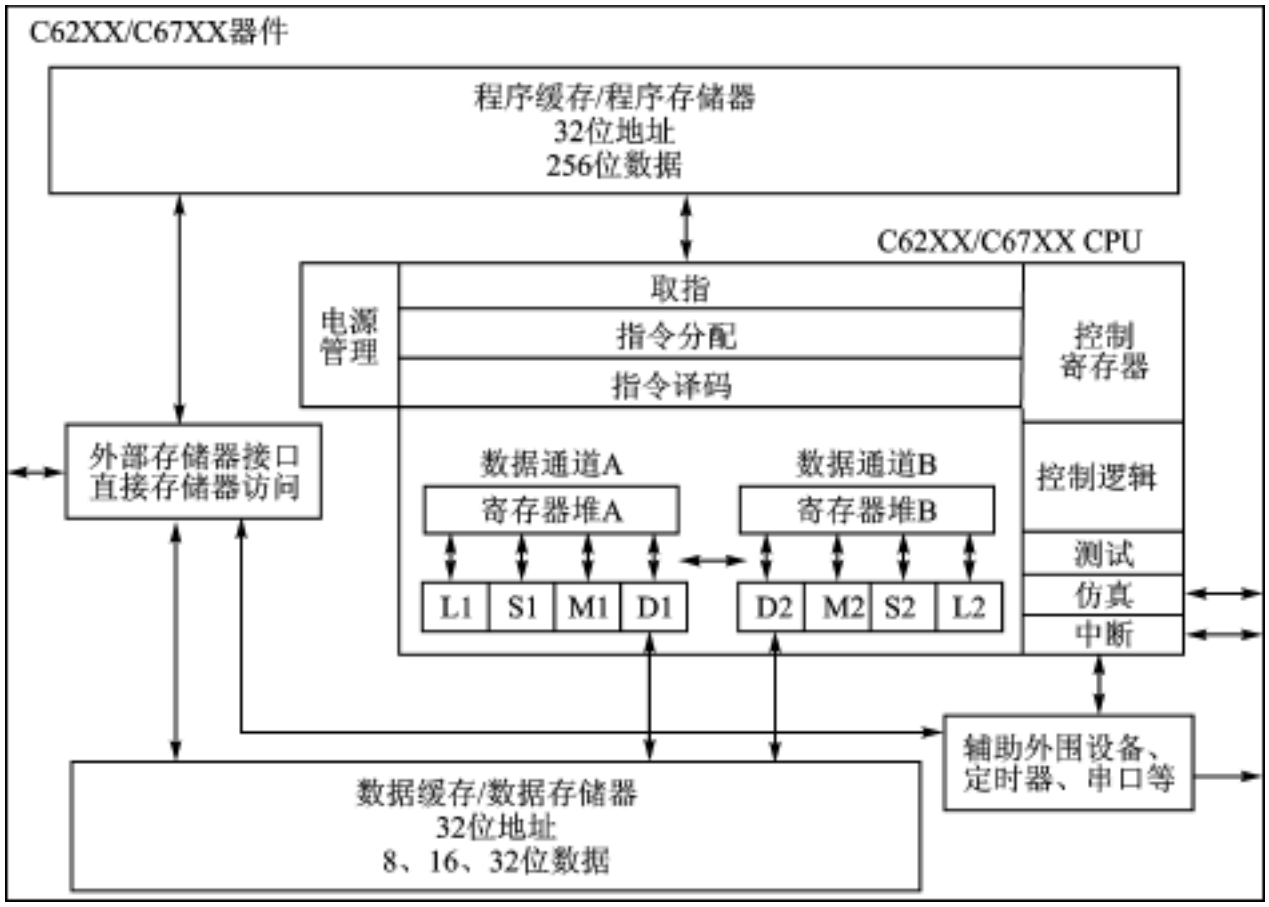


图 3 .12 - 1 TMS320C6000 CPU 结构图

TMS320C6000 的 CPU 有两个数据通道 A 和 B,每个通道有 16 个 32 位字长的寄存器(A0～A15,B0～B15),四个功能单元(L,S,M,D),每个功能单元负责完成一定的算术或者逻辑运算。A、B 两通道的寄存器并不是完全共享,只能通过 TMS320C6000 提供的两个交换数据通道 1X、2X,才能实现处理单元从不同通道的寄存器堆那里获取 32 位字长的操作数。

TMS320C6000 的地址线为 32 位,存储器寻址空间是 4G。C6201 片内集成有 1 M 位 SRAM——512 K 位的程序存储器(根据需要可全部配置成 Cache)和 512 K 位的数据存储器。通过片内的程序存储空间控制器,CPU 一次可以取出 256 位。即一次最多可以取出 8 条 32 位指令。

C6201 有 32 位的外部存储接口 EMIF 为 CPU 访问外围设备提供了无缝接口。外围设备可以是同步动态存储器(SDRAM)、同步突发静态存储器(SBSRAM)、静态存储器(SRAM)、只读存储器(ROM),也可以是 FIFO 寄存器。

为了便于进行多信道数字信号处理,TMS320C6000 配备了多信道带缓冲能力的串口 McBSP。McBSP 的功能非常强大,除具有一般 DSP 串口功能之外,还可以支持 T1/E1、ST-BUS、IOM2、SPI、IIS 等不同标准。McBSP 最多支持 128 个信道;支持多种数据格式(8/12/16/20/24/32 位)的传输;可自动进行 u 律、A 律压扩。其工作速率可达到 1/2 时钟速率。

TMS320C6000 提供的 16 位主机接口(HPI)使得主机设备可以直接访问 DSP 的存储空间。通过内部或外部存储空间,主机和 DSP 可以交换信息。主机也可以利用 HPI 直接访问映射进存储空间的外围设备。

DSP 器件一般都带有 DMA 控制器,可以在 CPU 操作的后台进行数据传输。TMS320C6201 的 DMA 控制器有 4 个独立的可编程通道,可以同时进行四个不同的 DMA 操作,每个通道的优先级可以通过编程设定。每个通道可以根据需要传输 8/16/32 位的数据,并且 DMA 控制器可以访问全部 32 位的地址空间。此外,还有一个辅助通道允许 DMA 控制器响应主机通过 HPI 口发来的请求。

2.指令系统

C62XX 和 C67XX 共享同一个指令集。C67XX 可以使用所有的 C62XX 指令,但因为 C67XX 是浮点芯片,所以 C67XX 的指令集中有一些指令只能用于浮点运算。TMS320C6201 CPU 的设计采用了类似于 RISC 的结构,指令集简单、运算速度快。8 个功能单元负责不同功能的运算,指令和功能单元之间存在一个映射关系。其中,L 单元有 23 条指令,M 单元有 20 条指令,S 单元有 29 条指令,D 单元有 26 条指令。

TMS320C6201 的大部分指令都可在单周期内完成,都可以直接对 8/16/32 位数据进行操作。同时,TMS320C6201 指令集针对数字信号处理算法提供了一些特殊指令:为复杂计算提供的 40 位的特殊操作的加法运算;有效的溢出处理和归一化处理;简洁的位操作功能等。TMS320C6201 中最多可以有 8 条指令同时并行执行;所有指令均可条件执行。以上所有特点提高了指令的执行效率,缩短了代码长度,大大减少了因跳转引起的开销,提高了编码效率。

流水线操作是 DSP 实现高速度、高效率的关键技术之一。TMS320C6000 只有在流水线充分发挥作用的情况下,才能达到 1 600 MIPS 的速度。C6000 的流水线分为三个阶段:取指、解码、执行,总共 11 级。和以前的 C3X、C54X 相比,有非常大的优势。主要表现在:简化了流水线的控制以消除流水线互锁;增加流水线的深度以消除传统流水线结构在取指、数据访问和乘法操作上的瓶颈。其中取指、数据访问分为多个阶段,使得 C6000 可以高速地访问存储空间。

二、优化编程的几个方法

使用 TMS320C6000 进行程序设计时,首先的感觉是汇编指令集太小了。C6000 在设计时采用了一种类 RISC 机的结构,运算速度特别快,但是指令集却非常简单。像 DSP 算法中

常用的乘加指令、循环操作指令等,在 C54X 和 C3X 中两条指令就可以完成的功能,而在 C6000 中却需要一个循环体,所以它的程序设计一般比较复杂。要想充分发挥 C6000 的运算能力,必须从它的硬件结构出发,最大限度地利用八个功能单元,使用软件流水线,尽量让程序无冲突的并行执行。

并行处理的长处在于,在处理彼此之间没有承接关系的运算时,在 CPU 资源允许的情况下可以并行完成。但对于前后有承接关系或者判断、跳转频繁的情况,就无法发挥并行的优势。一般循环体都满足并行处理的条件,并且循环体往往是程序中耗时最长的地方。因此进行 C6000 应用开发时应将优化重点放在循环体上。为了降低开发难度,C6000 提供了很多在高级语言(如 ANSI C)一级对程序进行优化的方法。在应用满足实时性处理要求时,应尽量采用这种方法。但是这种方法的效率比较低,C 语言优化最好的例子是点乘,这种循环使用 C 语言进行优化可以百分之百地利用 CPU 资源,程序的并行性达到最好。但是我们在做 20 点的点乘时发现它的耗时是汇编语言程序的 3 倍。所以如果系统的实时性要求比较高,就不能使用这种优化方法了。

这时可以考虑使用线性汇编语言进行开发。线性汇编语言是 TMS320C6000 中独有的一种编程语言,介于高级语言和低级语言之间。因为在用手写汇编语言进行应用开发时,开发者除了要精通 C6000 的指令系统之外,还必须为指令分配功能单元、考虑指令的延迟和功能单元之间的配合以及合理分配使用 32 个寄存器,才能写出高效的并行指令,发挥 C6000 的威力。上面任何一个方面出现问题,都会严重影响算法的效率。

线性汇编语言的指令系统和汇编语言的指令系统完全相同,但是它有自己的汇编优化器指令系统,用于和汇编优化器配合使用。与汇编语言的最大区别在于,编写线性汇编语言时不需要考虑指令的延时、寄存器的使用和功能单元的分配,完全可以按照高级语言的方式进行编写。当然由于它不是高级语言,有许多编程的限制。例如,在优化循环体时,不能使用跳转到循环体之外的跳转指令;另外计数器只能使用减计数,如果使用加计数,优化器将不能工作等等。但总的说来,它的代码效率远远高于高级语言,而且开发难度和开发周期比汇编语言要小得多。

在实际开发过程中需要具体情况具体分析,选择一种高效、快捷的开发方法。以下结合应用开发中的几个模块来简述我们使用的优化方法。

1 . 使用汇编语言

使用汇编语言进行并行编程难度比较大。但在有些情况下,程序中数据有非常强的承接关系,并且该程序体逻辑关系清楚,使用的寄存器不超过 32 个,这时直接使用汇编语言实现,效率会更高。另外,有些使用 C 语言比较难实现的运算函数,在 C6000 的汇编指令集中可能有专用 DSP 指令,这时就可以直接使用汇编语言实现。

使用汇编语言进行编程时特别需要注意的是 C6000 指令的延迟情况,有些指令并不是立刻就能得到结果。C6000 指令集中有延迟的指令如表 3 .12 - 1 所列。

表 3 .12 - 1 C6000 的有延迟指令

指令类型	延时长度(时钟周期)
Branch(跳转)	5
Load(从存储器中取数据)	4
Multiply(乘法)	1

例 1 32 位归一化函数 norm_1()

```
short  norm_1(long L_var1)
 {short  var_out;
 if (L_var1 = = 0L) {
 var_out = (short)0;
 }
else {
  if (L_var1 = = (long)0xffffffffL) {
       var_out = (short)31;
       }
  else {
       if ( _var1 < 0L) {
          L_var1 = ~ L_var1;
          }
         for(var_out = (short)0;L_var1 < (long)0x40000000L;
              var_out + + ) {
          L_var1 < < = 1L;
       }}}
       return(var_out);
       }
```

使用汇编语言进行优化:

```
              .global _norm_1
_norm_1:
            B          3
            CMPEQ    0,A4,B0
[!B0] NORM     A4,A4
            NOP      3
```

消耗时间(时钟周期):C 语言 norm_1()为 723;汇编语言为 11。

2.使用线性汇编语言重写整个函数

对于某些以循环体为主的函数可以使用线性汇编语言重写整个函数。使用汇编优化器进行优化之后,效率是非常高的。

下面例子是算法中计算帧能量的函数,其中包含两个单循环体。进行优化时,首先要确定循环的次数。对于循环次数是变量的情况,优化器不进行并行优化;其次尽量减少数据存取次数,例如以 32 位存取指令对 16 位数据进行存取,可以节省一半的存取周期。仔细观察 C 代码,会发现两次循环次数相同。第二个循环要用到第一个循环的结果,因此可以将两个循环合并在一起,这样就避免了在第二个循环中再从存储器中取结果,减少了一半的 Load 操作。

```
long  Comp_En(short * Dpnt)
{   nt     ;
```

```
    long    Rez;
    short   Temp[60];
    for (i=0;i<60;i++)
        Temp[i] = shr(Dpnt[i],(short)2);
    Rez = (long) 0;
    for (i=0;i<60;i++)
      Rez = L_mac(Rez, Temp[i], Temp[i]);
    return Rez;
}
```

相应的线性汇编程序如下:

```
      .global  Comp_En                函数名定义,对 c 变量前加_
_Comp_En  .cproc Dpnt                 ;函数头定义,Dpnt 是参数
      .reg    Rez, Rez1, Rez2,I       ;寄存器定义,不必考虑实际的寄存器分配
      .reg    t1,t2,x1,c1,m1,m2
     zero     Rez
     zero     Rez1
     zero     Rez2
     mv       Dpnt,c1
     mvk      30,i                    ;确定循环次数。因为用 LDW 代替 LDH,循环次数减少一半
loop1  .trip 30
       ldw    * c1++,x1
       shl    x1,16,t1
       shr    t1,2,t1
       shr    x1,2,t2                 ;将两个循环合在一起,又减少了一半的从内存取数据的时间

       smpyh  t1,t1,m1
       smpyh  t2,t2,m2
       sadd   Rez1,m1,Rez1
       sadd   Rez2,m2,Rez2
       [i] sub i,1,i                  ;循环计数器从 30 递减
       [i] b loop1
       sadd Rez1,Rez2,Rez
    .return Rez
    .endproc
```

消耗时间(时钟周期):C 语言为 32971;**线性汇编语言为** 93。

3.使用线性汇编改写复杂函数中的循环体

当函数的逻辑关系复杂,判断、跳转、函数调用情况特别多时,上面方法的效果就会大打折扣。这时可以使用线性汇编将其中的循环部分改写成一个函数,以优化后的函数调用代替循环部分,而不是优化整个复杂函数。

高速数字信号处理器件的应用范围越来越广,特别是在移动通信领域中,软件无线电、智能天线等新技术的实现都需要强大的实时数字信号处理的支持。 TMS320C6000 **系列** DSP **完**

全可以满足此类要求。但目前对于并行 DSP 技术的软硬件开发还处在摸索阶段,如何充分利用高速 DSP 的资源,是这方面的研究重点。本文研究了最新推出的 TMS320C6000 的优化策略,从工程和系统的角度总结出一套既能满足实时性又能保证开发时效性的实用的优化编程方法,以供分飨。

参 考 文 献

1　I 公司 .TMS320C6000 DSP User s Guide .1999

2　TI 公司 .TMS320C6000 DSP Reference Set .1999

选自《电子技术应用》月刊,2001 年第 4 期

3.13 TMS320C54X 软件模拟实现 UART 技术

长沙国防科技大学电子科学与工程学院(410073) 胡延平 李广森

一、引 言

DSP 与其他外设之间进行串行异步通信是系统经常遇到的要求。美国 TI 公司生产的高速 DSP 芯片以其优良的性能获得了广泛的应用,但大都不具有标准的 UART 串行通信接口。以 TMS320C54X 系列 DSP 为例,虽然它们分别带有多个不同的串口(见表 3.13-1),但都不是 UART 串口,因而均不便与带 UART 接口的外设(如 8031)直接进行异步通信。

表 3.13-1 TMS320C54X 通信串口

器件名称	标准同步串口	带缓存串口(BSP)	时分复用串口(TDM)
TMS320C541	2	0	0
TMS320C542	0	1	1
TMS320C543	0	1	1
TMS320C545	1	1	0
TMS320C546	1	1	0
TMS320C548	0	2	1
TMS320C549	0	2	1

TMS320C54X 实现串行异步通信,有两条途径。一是通过硬件办法,即增加专用 UART 接口(如 INS8250);二是通过软件办法,即采用软件模拟 UART 方式同样可以较好地解决这个问题。

二、TMS320C54X 软件模拟 UART 的实现原理

TMS320C54X 系列 DSP 软件模拟实现 UART,无需额外硬件开销,仅仅需要两个通用 I/O 脚($\overline{\text{BIO}}$和 XF)、外部中断 INT0 以及一个定时器就可实现,在硬件连接上是非常简单的(见图 3.13-1)。

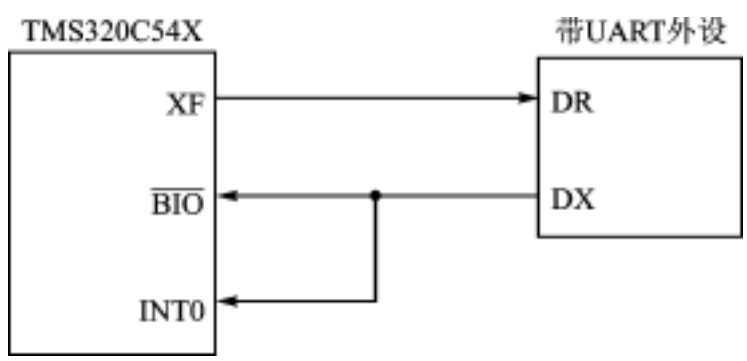


图 3.13-1 硬件连接图

图 3.13-1 中,XF 脚用于发送数据,$\overline{\text{BIO}}$脚和 INT0 脚用于接收数据。此方法可以模拟半双工通信(参见 2.1 和 2.2 节)与全双工通信(参见 2.3 节)。

软件模拟 UART 技术允许用户设置:

(1) 数据位数(1～16);

(2) 奇偶校验(奇校验或偶校验)或无奇偶校验;

(3) 停止位(1～2);

(4) 用户指定每秒传输比特数 BPS(Bits per Second),即波特率:

每秒传输比特数计算公式: $\text{bps} = \dfrac{1}{\text{clkout} \times (\text{tddr} + 1) \times (\text{prd} + 1)}$ 改变寄存器 TDDR 和 PRD 的设置值,可以获得不同的波特率。bps 最大值是由下式决定的: $\text{bps}_{\max} = \dfrac{\text{clkfreq}}{\#\text{cycles}} \times \text{bit}$。其中,clkfreq 是 DSP 系统时钟,#cycles 是子程序执行周期,bit 是完成 UART 功能限定条件(1/2 或 1)。

另外,需要两个存储单元分别用作 UART 状态寄存器(见表 3.13-2)和计数器,以便控制软件完成接收与发送任务。

表 3.13-2 状态寄存器

15～5	4	3	2	1	0
保留	校验错误位	Delay	UART	TX	RCV
比特位	数值	代表含义			
RCV	1	UART 正在接收数据			
	0	没有数据正在被接收			
TX	1	有数据正在发送			
	0	无数据正在发送			
UART	1	UART 正在忙于发送数据			
	0	UART 不在发送数据			
Delay	1	延迟发送准备好			
	0	无延迟发送			
校验错误位	1	最后一次接收奇偶校验错误			
	0	奇偶校验正确或无奇偶校验			

需要特别指出的是,该方法对时间是敏感的。因为数据必须在规定时间间隔内发送或接收,所以如果定时器要求 CPU 中断时,DSP 正在执行不可打断的程序代码,或执行中断接收发送服务子程序时被更高级别中断所打断,就会破坏丢失数据。

1. 接收功能

要正确接收数据流,首先要检测起始位。由于$\overline{\text{BIO}}$和 INT0 连在一起,一旦低电平有效则启动 INT0 中断。首次响应中断,定时器设置定时周期为 1/2 码元宽度,以便保证在起始位中间位置定时中断采样$\overline{\text{BIO}}$脚。如果采样电平为高,起始位就是虚假信号,定时器立即停止定时并退出子程序。如果为低,则认为检测到起始位,关闭外部中断 INT0 以免数据流中的 0 再被误认为起始位。接收后续数据时,每次定时器设置定时周期为一个码元宽度,同样为了保证在码元中间采样,从而减少误码。当收到若干个(数据位数可根据需要设定为 1～16)数据后,再

检测停止位和奇偶校验位。若数据接收正确则格式化输出数据(去掉起始位、停止位和奇偶校验位)。同时计数器清零,复位 RCV 标志位并开中断为下一次检测数据流的起始位做好准备。参见流程框图 3 .13－2 和图 3 .13－3。

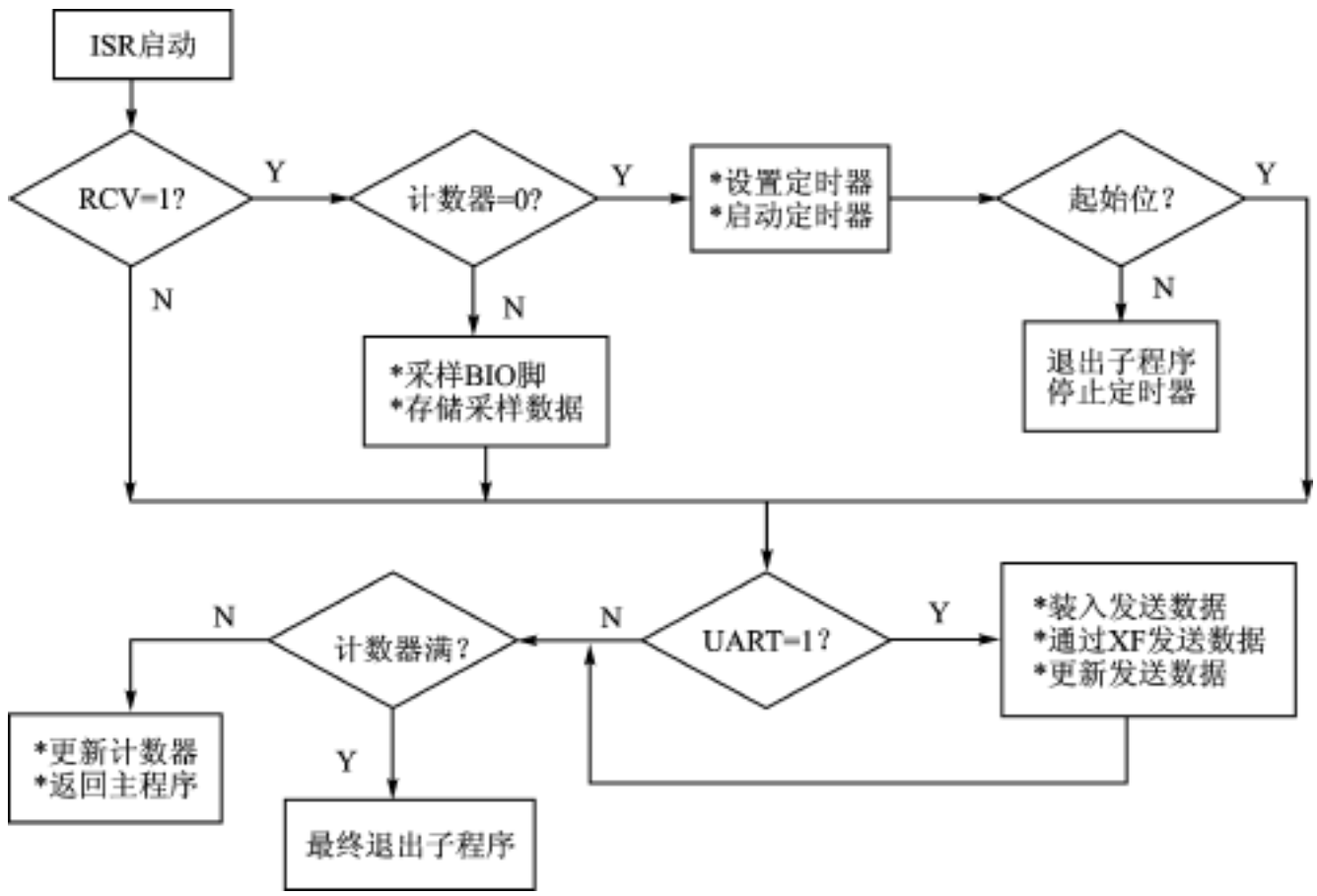


图 3 .13－2 定时器接收与发送中断服务子程序流程

2 . 发送功能

一旦有发送数据并且发送子程序被调用,必须先将发送数据格式化(加上起始位、停止位和奇偶校验位)存在某个存储单元以备发送。然后,检测 TX 位是否被置位(即有无数据正在发送)。若是则循环等待直到 TX 被复位为止。要知道何时开始发送下一个数据,需要定时器定时中断来控制,同时还需要一个发送计数器控制程序流程(见图 3 .13－2 与图 3 .13－3)。由于发送和接收是独立进行的,它们共用惟一的一个定时器,如果 UART 在接收数据过程中定时器突然被发送功能占用,则必然会破坏数据。因此,发送数据前必须检测 RCV 标志位,确认没有正在接收数据后才能开始发送数据,否则发送程序等待。因此,准确来讲,这种发送接收方式只是一种半双工方式,发送和接收不能同时进行。下面,我们介绍一种改进的方法,可以实现全双工通信。

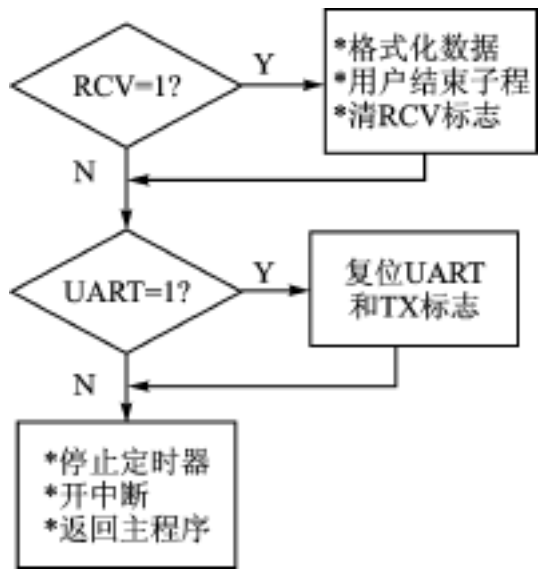


图 3 .13－3 接收与发送结束子程序流程

三、全双工通信

虽然发送和接收是各自独立进行的,但从二.1 节和二.2 节的描述来看,要同时进行接收和发送的惟一制约因素就是被迫共用一个定时器。由于异步接收数据,起始位随时可能到来,而发送数据则可以受程序控制。因此模拟全双工通信的关键就是引入延迟发送功能。具体来讲,就是发送数据被格式化存入相应存储器单元后,并不立即发送,而是将 UART 状态寄存器中的发送延迟标志位置位,执行代码返回主程序等待。当下一次接收数据起始位检测到后,程序检测发送延迟标准位,若置位则发送和接收同步进行,程序流程如图 3.13-4 和图 3.13-5 所示。显然,模拟半双工通信与模拟全双工通信的差别在于:前者是发送前首先检测有无数据正在发送,若无则立即发送数据;而后者则是无论是否正在接收数据,发送都延迟一段时间等待下一次接收开始后与其同步进行。

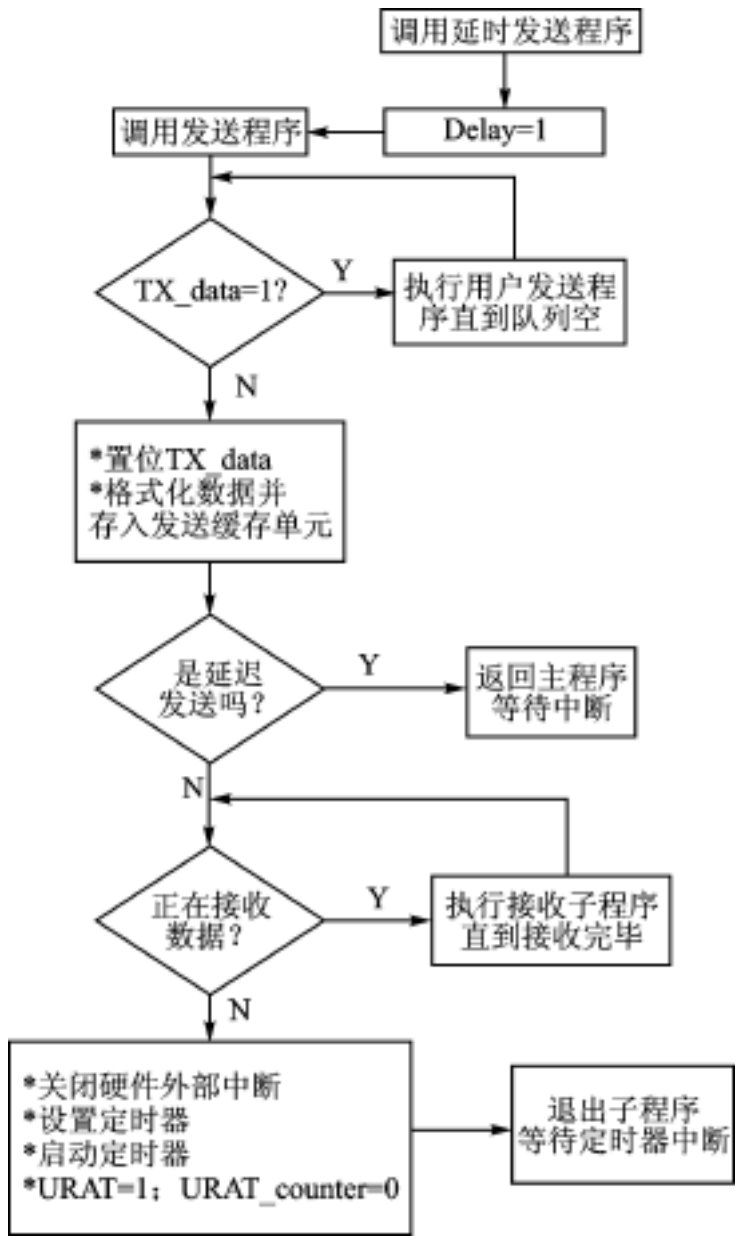


图 3.13-4 发送程序流程

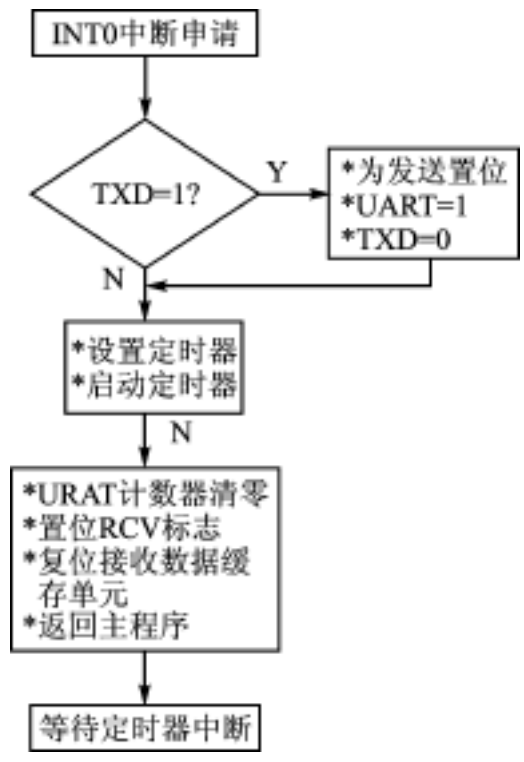


图 3.13-5 接收程序流程

四、结 论

在允许发送有一定延迟并且接收数据源源不断的情况下,比较适合采用延迟发送方式来实现全双工通信。如果对方不定期地偶尔发送一组数据,则发送延迟等待方式就不太合适,可以通过限定延迟时间,超时立刻就发送的办法来改进。TI 公司给出了软件模拟 UART 的源

程序,用户可以参考其源程序并根据自己的需要做必要的改动。假定全双工通信格式为:8Bits 数据、2 个停止位、带奇偶校验,在连续全双工通信情况下,平均每比特大约需要占用 78 个指令周期。当波特率为 9 600 bps 时,大约占用 0 .75 MIPS,这点时间开销在多数情况下是可以接受的。利用必要的电平转换器件,TMS320C54X 可以和符合 RS-232 规范的串口直接相连。总之,软件模拟实现 UART 技术是解决 TMS320C54X 系列 DSP 芯片与其他外设之间异步串行通信问题一种比较理想的方法。

参 考 文 献

1 mplementation of a Software UART on TMS320C54X Using General-Purpose I/O Pins . TEXAS INSTRUMENTS Application Report spra555

2 CPU and Peripherals . TMS320C54X DSP Reference Set Volume 1 SPRU131D

3 李华 .MCS-51 系列单片机实用接口技术 .北京:北京航空航天大学出版社,1993

选自《微处理机》季刊,2000 年第 3 期

3 .14　W78E516 及其在系统编程的实现

北方交通大学电气电力工程学院　崔冬建　王立德　史大北

使用传统方法对 CPU 重新编程存在诸多不便,在系统编程技术(以下简称 ISP 技术)的出现是对传统编程方法的突破。ISP(In System Programming)技术是指,在用户设计的目标系统中或印刷电路板上,为重新配置逻辑或实现新的功能,而对器件进行编程或反复编程的能力。ISP 技术的出现和发展开创了数字电子系统设计技术新的一页。ISP 技术无需编程器和较高的编程电压,打破了先编程后装配的惯例,形成产品后还可以在系统内反复编程,使具有 MTP－ROM 可多次编程或反复编程的微控制器的优越性得以更充分的发挥。尤其是在互联网时代,依据系统环境和需要,经调制解调装置(MODEM)、串行口或专用的编程接口就能够通过软件控制,实现系统远程升级和调试,提高产品的适应性,延长产品生存周期,经济效益显著。真正的可编程系统的时代即将到来。

一、W78E516 的结构

W78E516 是一种 8 位微控制器,内部含有在系统可编程的 MTP-ROM,用于系统更新。W78E516 与标准的 8052 完全兼容。

1 .特　征

(1) 全静态设计,最高工作频率为 40 MHz。

(2) 64KB APROM 存储应用程序和 4 KB LDROM 存储控制 ISP 操作的程序。2 块存储器均为 MTP-ROM。

(3) 512 字节的片内 RAM(包括 256 字节 AUX-RAM,可由软件选择)内部数据 RAM 有 512 字节。它分成 2 个存储单元:256 字节高速暂存和 256 字节辅存。这些地址有不同的确定方式:

① RAM 0H～127H:同 8052 一样直接或间接寻址,地址指针是被选中的寄存器单元中的 R0 和 R1。

② RAM 128H～255H:同 8052 一样只能间接寻址,地址指针是被选中的寄存器单元中的 R0 和 R1。

③ AUX-RAM 0H～255H:采用外部数据存储器的方式间接寻址,用 MOVX 指令,地址指针是选中寄存器单元的 R0 和 R1 以及 DPTR 寄存器。在 CHCON 寄存器中的第 4 位置位后,AUX-RAM 有效,访问 AUX-RAM 使用“MOVX@Ri”指令。当执行内部程序存储器的指令时,访问 AUX-RAM 不会影响 P0,P2,WR 和 RD。AUX-RAM 在复位后失效。

(4) 程序存储器和数据存储器可寻址空间范围为 64 KB。

(5) 4 个 8 位双向口:P0～P3,1 个 4 位双向多用途编程口 P4。

(6) 3 个 16 位的定时器/计数器:T0、T1、T2。T0 和 T1 功能与 8051 相同。T2 是一个 16 位定时器/计数器,它由 T2CON 配置和控制。T2 能作为外部时钟计数器,也能作为内部定时

器,这取决于 T2CON 的 C/ T2 位的配置情况。T2 有 3 种操作方式:清零、自动重写、波特率发生器。在清零和自动重写方式时,时钟频率与 T0 和 T1 相同。

(7) 具有一个全双工串行口。

(8) 具有 6 个中断源和 2 级中断能力。

(9) 内部电源管理:空闲方式和掉电方式,这两种方式可由软件选择。

(10) 具有编程后的编码保护功能。

2 .与 ISP 操作相关的特殊功能寄存器

(1) 在系统编程控制寄存器 CHPCON(BFH)功能如表 3 .14 -1 所列。

表 3 .14 -1

BIT	NAME	功 能
7	软件复位 F04KMODE	该位置 1 且 FBOOTSL 和 FPROGEN 都置为 1 时,微控制器复位,重新开始正常操作。读该位结果为逻辑 1 时,可以确认 CPU 处于 F04KBOOT 模式
6	—	保留
5	—	保留
4	ENAUXRAM	0:使 AUX-RAM 无效 1:使 AUX-RAM 有效
3	0	必须置为 0
2	0	必须置为 0
1	FBPPTSL	程序地址选择 1:装载程序位于 64 KB 的 APROM。4 KB LDROM 是重新编程的目标地址 0:装载程序位于 4 KB 的存储器。64 KB 的 APROM 是重新编程的目标地址
0	FPROGEN	MTP-ROM 编程使能 1:使编程功能有效。微控制器进入在系统编程状态。在这种编程模式下,清除、编程、读操作在设备进入空闲模式后可以实现 0:不能对 ROM 执行写操作

(2) 编程状态下 MTP-ROM 的控制字节寄存器 SFRCN(C7H)功能如表 3 .14 -2 所列。

表 3 .14 -2

BIT	NAME	功 能
7	—	保留
6	WFWIN	选择 ISP 操作目标存储器 0:对 LDROM 重新编程 1:对 APROM 重新编程
5	OEN	MTP-ROM 输出使能
4	CEN	MTP-ROM 使能
3,2,1,0	CTRL[3:0]	ROM 控制信号

(3) SFRAH, SFRAL: 在系统编程状态下的目标地址。SFRAH 包含地址的高位字节; SFRAL 包含地址的低位字节。

(4) SFRFD: 编程状态下 MTP-ROM 的编程数据。

二、W78E516 的在系统编程方法

1 . ISP 操作实现过程

微控制器通常执行 APROM 中的程序。如果 APROM 中的程序需要修改,用户需要通过设置 CHPCON 寄存器来激活在系统编程模式。在默认情况下, CHPCON 是只读的, 必须依次向寄存器中写入 # 87H 和 # 59H, 才能使 CHPCON 的写特性有效。激活 CHPCON 的写特性后,在其 0 位置位, 进入在系统编程模式。ISP 操作包括进入/ 退出在系统编程模式、编程、擦除、读等, 它们是在 CPU 处于空闲模式时完成的, 因此, 设置 CHPCON 寄存器后使 CPU 进入空闲模式,并由定时器中断的发生来控制执行每一种 ISP 操作的时间。定时器中断到来时, 转入 LDROM 中执行相关的中断服务程序。第一次执行 RETI 指令后, PC 指针清零, 指向 LDROM 中的 00H。当 APROM 中的内容被完全更新后, 将 CHPCON 的第 0, 1, 7 位设置为逻辑 1, 通过软件复位的方式返回 APROM 执行其中的新程序。在应用程序需要频繁更新的情况下, 这种在系统编程方式使工作简单而高效。

在默认情况下, 上电复位后 W78E516 从程序中启动。在某些情况下, 可以使 W78E516 从 LDROM 中启动。当 APROM 中的程序不能正常运行, W78E516 无法跳到 LDROM 中执行 ISP 操作时, CPU 进入 F04KBOOT 模式。在应用系统设计中一定要注意 P2, P3, ALE, EA 和 PSEN 引脚在复位时的值, 以避免意外激活编程模式或 F04KBOOT 模式。复位时进入 F04KBOOT MODE 时 P4. 3, P2 .7, P2 .6 引脚电平及时序如图 3 .14 - 1 及图 3 .14 - 2 所示。

F04KBOOT MODE

P4 .3	P2 .7	P2 .6	MODE
X	L	L	F04KBOOT
L	X	X	F04KBOOT

图 3 .14 - 1　F04KBOOT 模式下引脚电平

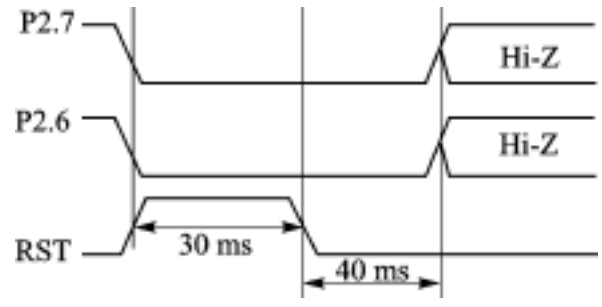


图 3 .14 - 2　进入 F04KBOOT 模式时序图

W78E516 处于在系统编程模式时, MTP-ROM 可以被反复编程和检验。向 MTP-ROM 中完整、正确地写入新代码后, 新代码即被保护起来。W78E516 有专用设置寄存器组(special setting registers), 其中包括安全性寄存器(security register)和公司/ 器件识别寄存器(company/ device ID registers), 处于编程模式时不能访问这些寄存器。安全性寄存器在 LDROM 空间的地址是 0FFFFH, 当它的各个位被从 1 编程为 0 后就不能再被改变, 将它们重新置位的惟一方式是执行全部擦除操作, 这样就能保证其安全性。

一般情况下, 具有 ISP 功能的微控制器一般都具备 2 块程序存储区(暂时称为 A-ROM 和 B-ROM), 其中 A-ROM 用于存储通常状况下的应用程序, B-ROM 用于存储控制 ISP 操作的程序, 向 A-ROM 中写入新代码。有些微控制器, A-ROM 和 B-ROM 中的程序代码均能控制 ISP 操作, 由特殊功能寄存器来选择其一, 为设计人员提供了灵活的设计应用空间。针对不同

类型的 ISP 器件,对 CPU 进行在系统编程的方法具有共同之处。

执行 ISP 操作时,2 块程序存储器中的程序流程图分别如图 3 .14 - 3、图 3 .14 - 4 所示。

图 3 .14 - 3 64 KB APROM 中的 ISP 流程

图 3 .14 - 4 ISP 操作流程

2. W78E516 的 ISP 功能特点

在 MTP 产品中,W78E516 颇具特色。它在 ISP 功能方面具有突出的优点:

(1) 开发灵活性。可由设计者自定任何编程通信协议,经计算机或简单工具,将要修改的程序通过任何 I/ O 口或 UART 口送入单片机内,不能像其他具有 ISP 功能的芯片那样,而必须针对其特定引脚及特殊的 TIMMING 协议来实现。

(2) 操作连续性。市场上目前具有 ISP 功能的单片机在执行 ISP 操作时(在未带配件的情况下)必须停止其他操作;而有些应用希望此时 UART 或 TIMER/ COUNTER 等功能仍然能够运作。W78E516 可以满足这种要求。因为在执行 ISP 操作时只是控制权从 64 KB APROM 变换到 4 KB LDROM,故仍可由 4 KB 中的程序来继续操作控制。

(3) 断电时具有存储数据能力。因 W78E516 拥有 2 块大小不同的闪速存储器,其中 1 块可用于存储断电后仍必须被单片机保留的数据,因此,设计者可减少外接 EE2PROM 芯片的线路与成本。

除具有上述特点外, W78E516 在执行 ISP 操作时不需辅以任何配件,受到用户的欢迎。

三、应用实例

此实例是在机车故障检测记录仪系统内对 W78E516 进行 ISP 操作的实验。这是一个由 PC 机和微控制器组成的主从式系统。PC 机经串行通信将新程序的二进制代码以数据形式下载,微控制器接收数据,由软件控制更新 64KB APROM 中的程序代码。实验中微控制器经 RS-232 接口接收数据并暂存于内部 AUX-RAM 中,不需扩展外部数据存储器,节省了板上空间。检测记录仪与 PC 机的通信采用 RS-232 标准,为简化硬件,只使用了该标准中的 TXD、RXD 以及地线 3 根连线,电平转换由 MAXIM232 专用芯片完成。实验电路原理图如图 3 .14 - 5所示。

实现 ISP 操作的软件由两部分组成:一是微控制器部分(包括 APROM 和 LDROM 中的

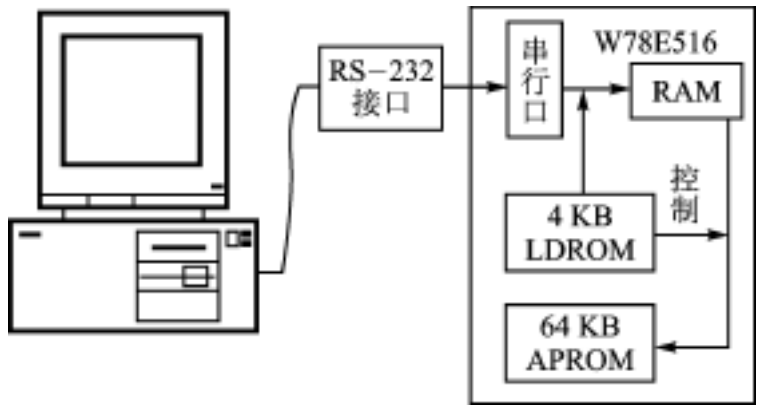


图 3 .14 − 5　ISP 操作实现原理

程序),用 MCS-51 汇编语言编写;二是 PC 机部分,由 Microsoft Visual Basic 开发而来,主要应用 MSCOMM 控件与 W78E516 通信,完成数据下载。

微控制器上电后一般从 64 KB APROM 中启动。64 KB APROM 中,包括了在系统编程子程序,还有其他检测记录系统的子程序。微控制器必须读取拨码盘的输入,决定进入哪一种工作状态,是否进行在系统编程。值得注意的是,在写 CHPCON 寄存器时,应将其第 4 位置 1,使 AUX-RAM 有效;64KB APROM 中的程序应该始终包含图 3 .14 − 3 流程所示的程序段,以使系统具有进入下一次在系统编程的能力。4KB LDROM 中的程序主要作用是接收来自 PC 机的下载数据,并控制各项 ISP 操作。执行在系统编程时,利用 SFRAL、SFRAH、SFRFD、SFRCN 这几个特殊功能寄存器,选择在系统编程的地址单元,准备待写入的数据,选择待执行的操作类型。当从 F04BOOT 模式启动时,软件复位失效,必须硬件复位。在系统编程的数据由在此期间仍能正常工作的串行通信口进入。这部分流程如图 3 .14 − 6 所示。

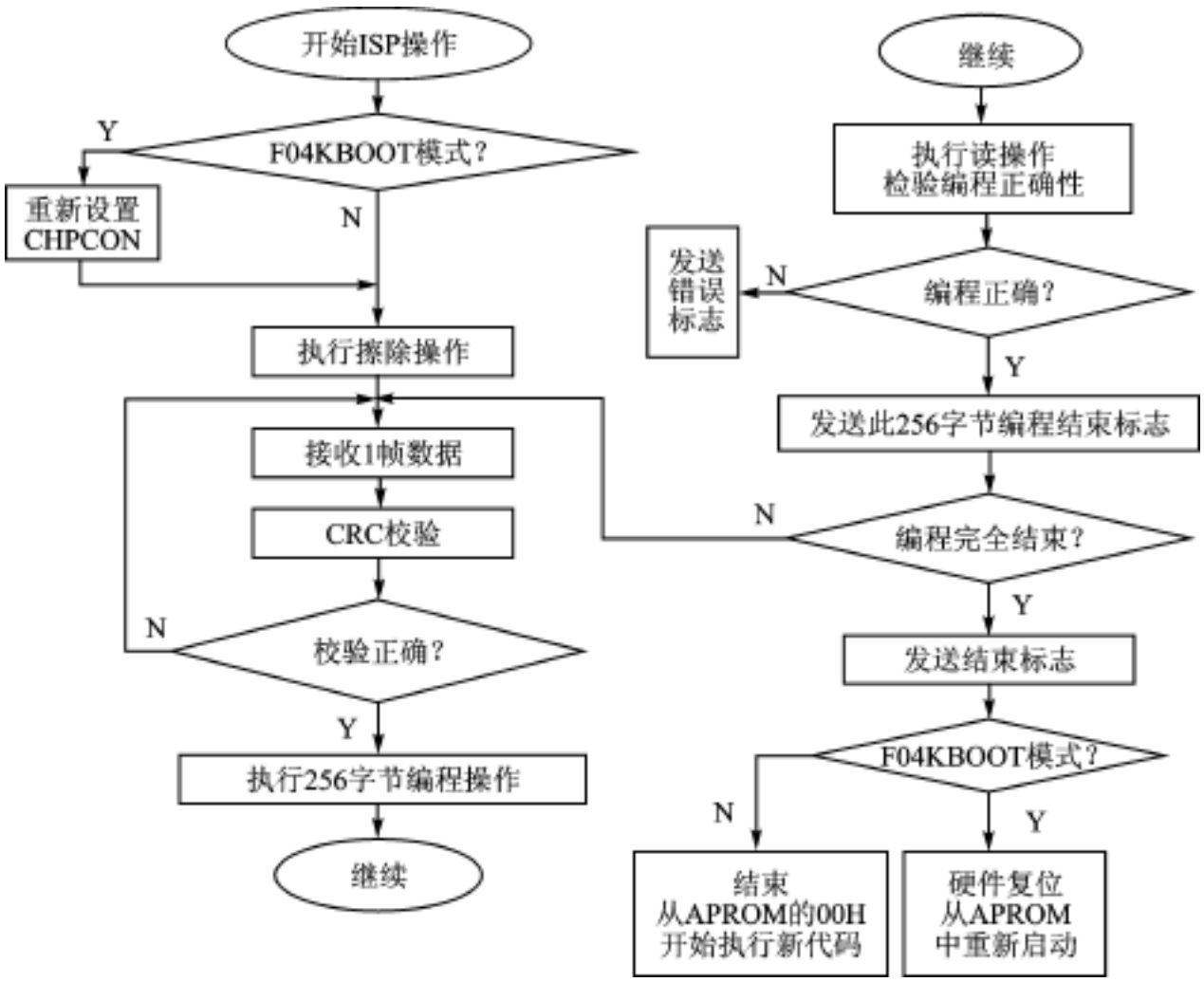


图 3 .14 − 6　4 KB LDROM 控制 ISP 操作流程图

PC 机通过 RS-232 串口发送数据。每一帧的前 3 字节都为 7EH,作为帧同步信号。随后 2 字节内容均为帧指针,表明本帧数据的帧号。PC 机在发送 1 帧数据后,等待单片机发回确认数据帧,该帧数据应包括数据接收正确与否的标志及接收到的数据的帧号。数据帧格式及 PC 机通信软件流程分别如图 3 .14 - 7、图 3 .14 - 8 所示。

7EH	7EH	7EH	帧指针	帧指针	256 字节数据	CRC 校验和

图 3 .14 - 7 串口通信帧格式

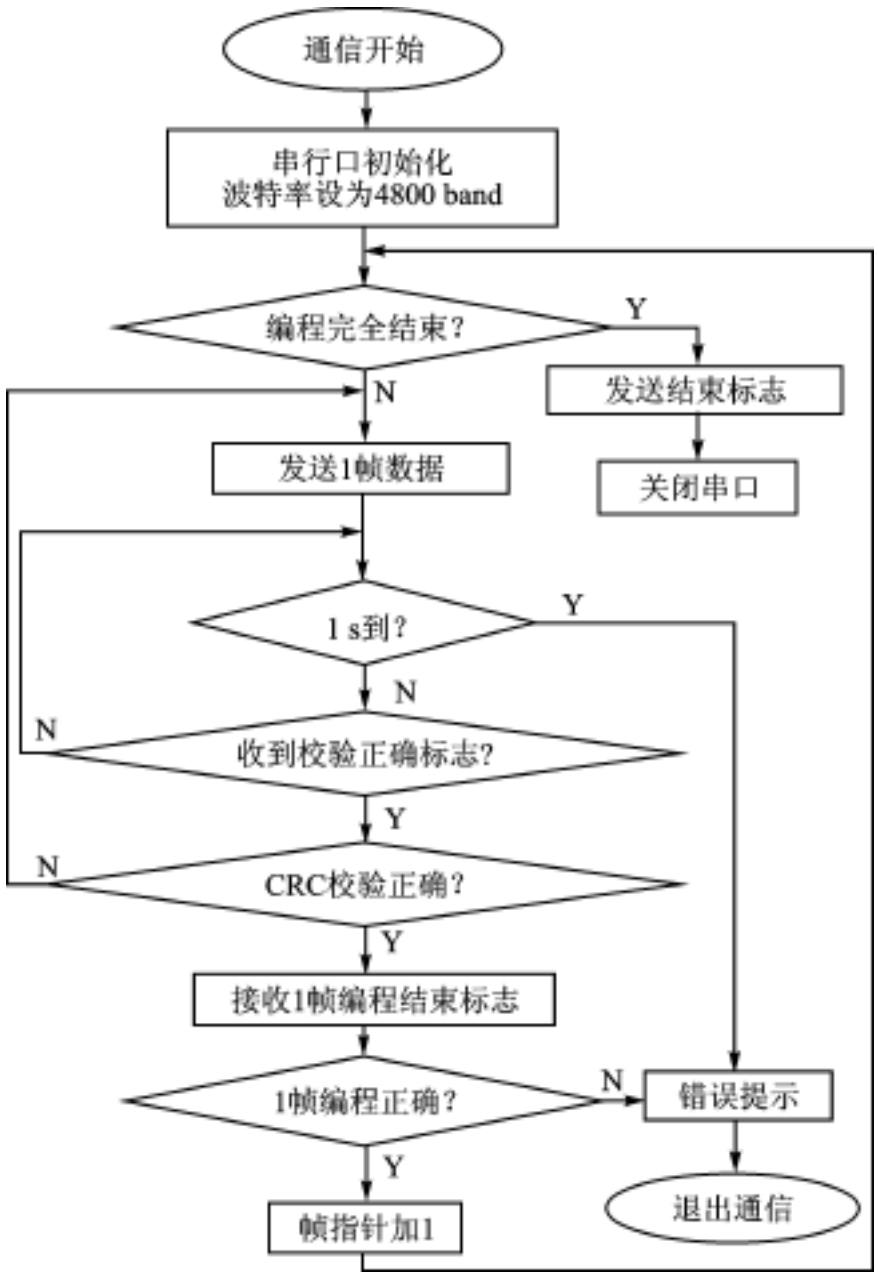


图 3 .14 - 8 PC 机软件通信流程

四、结束语

根据本文介绍的方法,在机车故障检测记录仪系统内实现了对 W78E516 的在系统编程。

ISP 技术高度的灵活性使同一种硬件结构能够实现多种系统功能,成为多功能硬件,减少了系统所需电路板的品种,简化了生产流程;多功能硬件还能够减少板上元件数目和系统电路板数目,使系统成本显著降低。在机车系统中要对各部分进行多种不同的测试,比如轴温、轴速以及车门的开关状态等等,以便了解机车的运行状况。在现阶段,处理这些模拟量或数字量需要设计生产多种不同的模块。应用 ISP 技术以后这一现状会得以改变:设计人员设计出包

括微控制器、A/ D 和 D/ A 转换芯片、I/ O 口等在内的通用模块,将其安装到需要进行检测的各个部分,然后利用 ISP 技术把不同的应用程序下载到微控制器中,就可以完成各种不同的测试功能,其综合经济效益不可低估。此外,ISP 技术也为其他许多领域带来了变革。总之,在系统编程技术具有广阔的开发应用前景。

参 考 文 献

1　华邦公司 .在系统可编程系列(W78E 系列) .世界电子元器件,1999(2)

2　黄正瑾 .在系统编程技术及其应用 .南京:东南大学出版社,1997

选自《单片机与嵌入式系统应用》月刊,2001 年第 6 期

3.15 键盘键入信号软件处理方法探讨

河南安阳大学计算机系(455000) 赵重明

计算机控制系统中,人控信号及一些数据的输入主要使用键盘。键盘接口,尤其是键入信号的软件处理方法是影响系统使用和操作性能的重要因素。键盘接口及其软件的任务主要包括以下几个方面:

(1) 检测并判断是否有键按下;

(2) 按键开关的延时去抖;

(3) 计算并确定按键的键值;

(4) 程序根据计算出的键值进行一系列的针对性处理,即利用有效键值执行具体动作或处理。

一、键盘接口

计算机键盘按结构的不同分为独立式按键键盘和行列式键盘两类,每类按译码方法的不同又分为编码式及非编码式两种。

独立式非编码键盘占用较多的 I/O 位口(每键占 1 个),行列式非编码键盘用较少的 I/O 线实现较多的按键,这两种键盘的检测、去抖、确定键值等工作都由软件完成。另外,由于这两种键盘的各按键间相互无制约关系,可实现多键同时按下的处理。

独立式编码键盘接口的主要部件是优先编码器,这种键盘不能实现多键同时按下的处理。行列式编码键盘的编码方法有静态和动态两种。静态接口主要由一个行编码器和一个列编码器构成;动态接口可采用计数器、译码器和数据选择器来构成。这两种键盘由硬件完成键的编码任务;如果接口中加入去抖电路,则软件也不用考虑按键去抖;如果接口中加入中断请求器件,则键盘的检测可以在中断服务程序中进行。

一般在小型仪器仪表和控制系统中,使用较多的是独立式和行列式非编码键盘;如果系统要求实现多键同时按下的处理,则用非编码独立方式较为合适。

二、键信号处理软件

在键盘及其软件的 4 项任务中,键的检测和根据键值进行的针对性处理必须由程序来完成,当然键盘接口是检测按键动作的物质基础,没有它的作用根本谈不上键盘的检测;而去抖和计算键值这两项功能,软件和硬件都能实现。很明显,计算键值比较简单,而根据键值进行的针对性处理则每个系统的要求都不一样。实际上键信号检测时机的选择和键去抖是键处理软件的关键,而这两个问题又是相互联系的,下边就详细讨论这两个问题。

根据检测键盘接口时机的不同,键信号处理软件有 3 种工作方式。

第 1 种,利用空闲时间扫描方式。该法不管有无键按下软件都去扫描键盘接口,占用处理器时间较多。如果系统平时大部分时间空闲,可一直扫描、查寻、等待键按下,一旦发现有键按

下,调用延时子程序去抖确认,然后处理的办法;如果系统较忙,则必须合理分配系统的时间资源,使得系统能以每 20 ms 一次的扫描频度进行键盘扫描,这样才能做到不漏检。

第 2 种,中断扫描方式。该法无键按下时不扫描,有键按下时,键盘接口产生中断请求,在中断服务程序中进行键盘扫描、去抖,进而完成处理工作。显然,这种方式以占用一个中断资源为代价节省了处理器时间。该法适合处理器比较繁忙或是要求响应速度快的系统。

第 3 种,定时中断扫描方式。该法一般利用定时器每 20 ms 产生一次中断,在定时中断服务程序中完成扫描、计算和处理,本质上讲,这也是一种中断性质的方式。由于键入信号是人手触击所至,键入速度不可能太快,常用按键的速度一般在 20 次/ s 以内,自然点按的按键持续时间一般在 100 ms 左右,因此 20 ms 一次的采样速度一般不会漏检。按键的机械抖动时间通常不超过 10 ms。为了去抖,一般的处理方法是:中断程序连续两次以上采样,直到有键按下后确认是有效按键。

比较 3 种方式,在处理器不是太忙的情况下,采用定时中断扫描方式比中断扫描方式在保证快速响应的前提下能节省中断请求器件和计算机中断资源。下面以定时中断扫描方式为例,按系统的不同使用和处理要求,分 3 种情况详细讨论键入信号的软件处理方法。

(1) 系统只要求处理单按键,并且每按一次键只处理一次

定时每 20 ms 一次采样的一个完整的按键过程如图 3.15-1 所示。为了便于说明,先定义几个状态:第 N 次采样时无键按下,此时键处于初态;N + 1 次采样时,第一次采到,有键按下,键进入状态,此时按键尚不稳定;N + 2 次采样时,为连续第二次采到,有键按下,键进入初稳态,次态之后初稳态之前的这段 20 ms 的时间称为去抖周期;N + 3 次采样时,为连续第三次采到,键进入持续稳定态(续稳态);第 N + I + 1 次采样时,为抬键后第一次被采到,此时键又回到初态。由于计算机的采样略滞后于实际,可以把 N + 2—N + I + 1 次采样之间的时间当作稳定按键期。经过分析不难发现,计算机采样的稳态有 4 种情况:初态、次态、初稳态和续稳态,我们依次分别编号为 0、1、2、3 态,当然也可以把 2、3 态合并为一个状态称为稳态。这里定义的键态(号)反映了键盘处于一次按键的哪一个阶段。

为了使程序根据键态进行相应的处理,键盘扫描程序应为按键设置状态单元,根据每 20 ms一次采样到的键的即时状态是开或是关,再结合原来的键态得到现在的键态。图 3.15-2所示的一个键态转换图表示了 4 种状态的转换关系。图中带 Y 的箭头表示采到键按下时的状态变化,带 N 的箭头表示采到键没有按下时的状态变化。例如,原来键处于 0 态(无键按下),之后采到一次,按键后,键态变为 1 态(次态);如果紧接着一次采样又采到,该键处于按下态,则键态由 1 态变为 2 态(初稳态);如果紧接着一次采样发现该键处于抬起态,则系统认为是抖动,键态由 1 态又回到 0 态。

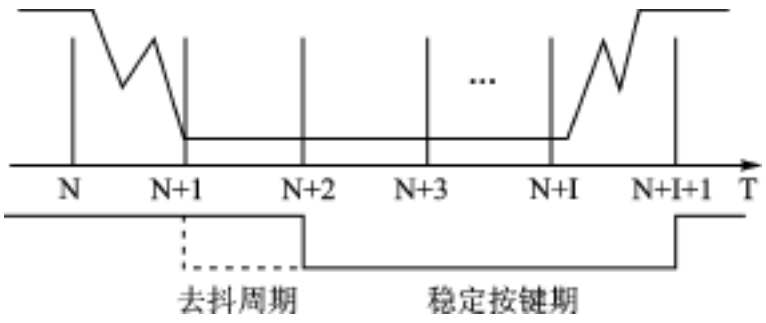


图 3.15-1 一个完整的按键过程

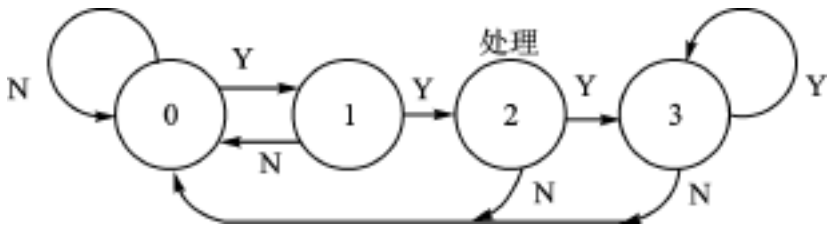


图 3.15-2 键态转换图

当系统要求对按键快速响应时,可采用按键稳定立即处理的键处理方法,即确认可靠按键后(进入 2 态后)马上处理一次,对于此后的持续按键,系统不予处理,抬键后本次按键处理过程结束。键处理程序流程图可以根据状态转换图获得,如图 3 .15 - 3 所示。

有的系统工作环境比较复杂,由于电磁干扰和电路噪声的原因,当采样键盘时即使无键按下也可能采到虚假的按键信号,我们暂且称之为键盘接口的电抖动,电抖动持续时间一般不长于 1 ms,并有随机性。这时可以把扫描改为每 5 ms 进行一次,连续 4 次以上采到有键按下才确认按键有效,一般连续 4 次采到的都是电抖动的可能性很小,因此这种方法可有效滤除电干扰,同时又不降低对键盘的响应速度。

如果系统对键盘响应速度要求不高,也可采用抬键处理的方法,即确认按键后,并不急于处理,一直等到抬键后(N + I + 1 次采样后)方才处理,之后返回初态。我们可以把初稳态和续稳态合并为一个状态——稳态(2 态),再引入一个暂时状态(3 态)表示第一次采样的抬键,有待于进行处理,等处理完成后由软件把状态单元中的 3 态置为 0 态,表示已经抬键,而且已经处理过了。其状态转换如图 3 .15 - 4 所示。读者可以仿照前图容易地得到键处理软件流程图。

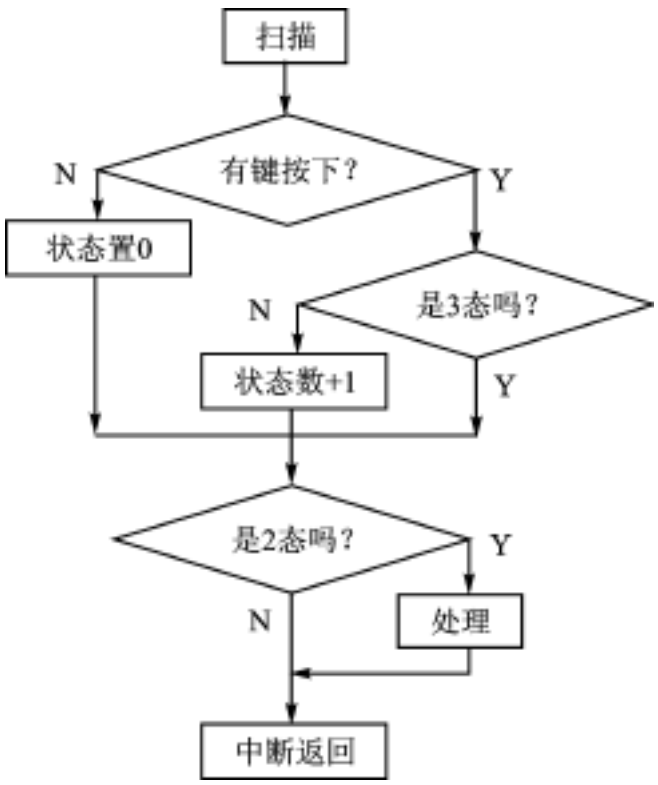


图 3 .15 - 3　键处理程序流程图

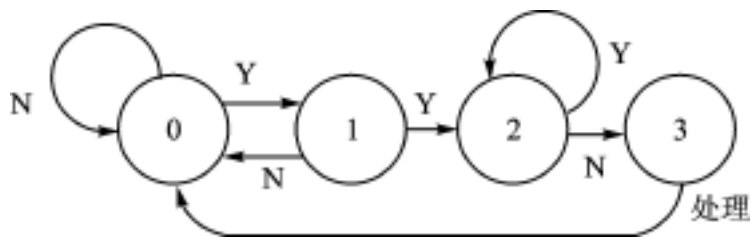


图 3 .15 - 4　键处理程序转换流程图

(2) 系统要求对持续性按键多次响应

对于有这种要求的键入信号,可在按键去抖后马上处理一次,以后如果按键持续一定时间(比如 500 ms)还不抬起则处理第二次,接下来每持续一定时间(比如 100 ms)即再处理一次,直到抬键后,结束对一次按键的完整处理过程。其键态转换如图 3 .15 - 5 所示。其中前 4 个状态与要求按键处理一次时相同,状态 3 以后实际是计数态(计时态),头一次计数到 27 态(需 500 ms)处理第二次,同时软件将状态单元置回到 22 态,以后每计数够 5 次(需 100 ms)即再进行一次处理,直到抬键时,软件使键状态单元返回 0 态。仿照前面的方法,读者也可画出软件处理流程图。当然其中的时间参数可以方便地修改,使键盘滞后时间和连发速度发生变化。

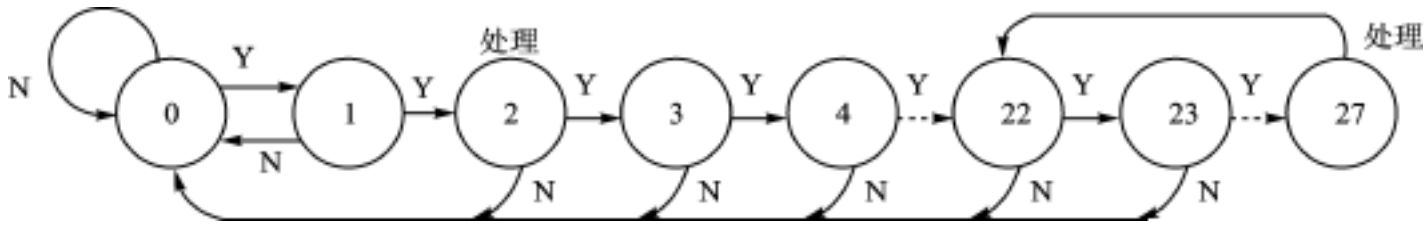


图 3.15-5 按键处理的键态转换图

(3) 系统要求对同时按下多键进行处理

在大多数系统中一般只配备一个键盘,在键资源够用的情况下,总是每个键实现单一功能,此时对于多键同时按下,要坚持最多只处理一个按键的原则。根据具体情况,可只处理先按下的键或最后松开的键,当然也可以认为多键按下是错误按键,对所有按键都不处理。如果键数较少,不够使用时,可采用一键多能的办法,常用的方法有两种,一种是状态转换法,即在键盘上设定专门的键盘状态转换键,通过状态转换键转换键盘状态,系统在不同状态下收到同一物理键并不实现同一逻辑功能;另一种是用多键组合实现多功能,组合键功能可由键盘接口硬件实现,也可用软件实现。

有的系统要求配两个以上的键盘,各键盘可同时操作。考虑比较简单的只有两个键盘的情况,如果两个键盘的优先权级相同且彼此无制约关系,软件相对较易处理;如果两个键盘权级不同或彼此间有制约关系,则键处理软件的结构将比较复杂。

一般解决同时多键按下的办法是:第一步,在键盘扫描程序中先采集所有键的即时开关状态,得到当前时刻整个键盘所有按键的即时状态表;第二步,再根据即时键状态表,依照键态转换原则,由原来的键态表得到新的键态表;第三步,根据键态表,结合当前系统状态和键之间的相互制约与组合关系,以及设定的优先级次序,执行键处理程序。

以上只是讨论了定时中断扫描(采样)键盘的键处理方法,这种处理方法的关键是首先定时扫描键盘即时状态,而后得到键状态,再根据键状态按具体要求进行键处理,实际上这种方法可以方便地移植到中断及空闲扫描处理方式中。

参 考 文 献

1 赵依军等.单片微机接口技术[M].北京:人民邮电出版社,1989
2 张积东等.51/98 单片机原理及开发[M].北京:电子工业出版社,1994
3 戴梅萼等.微型计算机技术及应用[M].北京:清华大学出版社,1996

选自《计算机自动测量与控制》双月刊,2000 年第 4 期

3 .16　单片机系统中数字滤波的算法

杭州浙江大学光电信息工程学系(310027)　赵　毅　牟同升
杭州中国计量学院(310034)　沈小丽

一、前　言

随机误差是由随机干扰引起的,其特点是在相同条件下测量同一量时,其大小和符号作无规则变化而无法预测,但多次测量结果符合统计规律。为克服随机干扰引入的误差,硬件上可采用滤波技术;软件上可采用软件算法实现数字滤波,其算法往往是系统测控算法的一个重要组成部分,实时性很强,常采用汇编语言来编写。

采用数字滤波算法克服随机干扰引入的误差具有以下优点:

(1) 数字滤波无需硬件,只用一个计算过程,可靠性高,不存在阻抗匹配问题。尤其是数字滤波可以对频率很高或很低的信号进行滤波,这是模拟滤波器做不到的。

(2) 数字滤波是用软件算法实现的,多输入通道可共用一个软件“滤波器”从而降低系统开支。

(3) 只要适当改变软件滤波器的滤波程序或运算参数,就能方便地改变其滤波特性,这对于低频、脉冲干扰、随机噪声等特别有效。

二、数字滤波常用算法

常用的数字滤波器算法有程序判断法、中值判断法、算术平均值滤波法、加权滤波法、滑动滤波法、低通滤波法和复合滤波法等。

1 .程序判断法

程序判断法又称限幅滤波法,其方法是把两次相邻的采样值相减,求出其增量(以绝对值表示),然后与两次采样允许的最大差值 ΔY 进行比较,ΔY 的大小由被测对象的具体情况而定,若小于或等于 ΔY,则取本次采样值;若大于 ΔY,则取上次采样值作为本次采样值,即

$|y_n - y_{n-1}| \leq \Delta Y$,则 y_n 有效,

$|y_n - y_{n-1}| > \Delta Y$,则 y_{n-1} 有效。

式中　y_n——第 n 次采样值;

y_{n-1}——第(n - 1)次采样值;

ΔY——相邻两次采样值允许的最大偏差。

设 R1 和 R2 为内部 RAM 单元,分别存放 y_{n-1} 和 y_n,滤波值也存放在 R2 单元。采用 MCS - 51 指令编写的程序判断法子程序如下:

```
FILT1:      OV     ,R2                  |yn - yn-1|
            CLR    C
```

```
        SUB   A,R1
        JNC   FILT11
        CPL   A                    ;以下两句求 A 的补码
        INC   A
FILT11: CJNE  A,#ΔY,FILT12
        AJMP  DONE
FILT12: JC    DONE
        MOV   R2,R1
DONE:   RET
```

程序判断法主要用于处理变化比较缓慢的数据,如温度、物体的位置等。使用时,关键是要选取合适的门限值 ΔY。通常可根据经验数据获得,必要时也可由实验得到。

2.中值滤波法

中值滤波法即对某一参数连续采样 N 次(一般 N 为奇数),然后把 N 次采样值按从小到大排队,再取中间值作为本次采样值。

设 DATA 为存放采样值的内存单元首地址,SAMP 为存放滤波值的内存单元地址,N 为采样值个数。用 MCS-51 指令编写的中值滤波子程序如下:

```
FILT2:  MOV   R7,#N-1          ;置循环初值
SORT:   MOV   A,R7
        MOV   R6,A
        MOV   R2,A
        MOV   R0,#DATA         ;采样值首地址送 R0
LOOP:   MOV   A,@R0
        INC   R0
        CLR   C
        SUBB  A,@R0            ;yn - yn-1 →A
        JC    DONE             ;yn < yn-1 转 DONE
        ADD   A,@R0            ;恢复 A
        XCH   A,@R0            ;yn ≥ yn-1 ,交换数据
        DEC   R0
        MOV   @R0,A
        INC   R0
DONE    DJNZ  R6,LOOP          ;R6≠0,继续比较
        DJNZ  R7,SORT;         ;R7≠0,继续循环
        MOV   A,R0
        ADD   A,#DATA          ;计算中值地址
        CLR   C
        RRC   A
        MOV   R0,A
        MOV   SAMP,@R0         ;存滤波值
        RET
```

中值滤波法比较适于去掉由偶然因素引起的波动和采样器不稳定而引起的脉动干扰。若

被测量变化比较缓慢,采用中值滤波法效果比较好;但对快速变化的数据,则不宜采用中值滤波。

3.算术平均滤波法

算术平均滤波法就是连续取 N 次采样值进行算术平均,其数学表达式是:

$$y = \sum y_i$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

式中 $\bar{y}$——N 个采样值的算术平均值;

y_i——第 i 个采样值。

设 8 次采样值依次存放在地址 DATA 开始的连续单元中,滤波结果保留在累加器 A 中,程序如下:

```
FILT3:   CLR   A              ;清累加和
         MOV   R2,A
         MOV   R3,A
         MOV   R0,#DATA       ;指向第一个采样值
         MOV   R7,#08H
FILT31:  MOV   A,@R0          ;取一个采样值
         ADD   A,R3           ;累加到 R2、R3 中
         MOV   R3,A
         CLR   A
         ADDC  A,R2
         MOV   R2,A
         INC   R0
         DJNZ  R7,FILT31      ;累加 8 次
FILT32:  SWAPA                ;(R2、R3)/ 8
         RL    A
         XCH   A,R3
         SWAPA
         RL    A
         ADD   A,#80H         ;四舍五入
         ANL   A,#1FH
         ADDC  A,R3
         RET
```

算术平均滤波法适用于对一般具有随机干扰的信号进行滤波。这种信号的特点是有一个平均值,信号在某一数值附近上下波动。算术平均滤波法对信号的平滑程度完全取决于 N。当 N 较大时,平滑度高,但灵敏度低;当 N 较小时,平滑度低,但灵敏度高。为方便求平均值,N 一般取 4、8、16 之类的 2 的整数幂,以便于用移位来代替除法。

4.加权平均滤波法

算术平均滤波法存在前面所说的平滑性和灵敏度之间的矛盾。采样次数太少,平滑效果差;次数太多,灵敏度下降,对参数的变化趋势不敏感。为协调两者关系,可采用加权平均滤

波。对连续 N 次采样值,分别乘上不同的加权系数之后再求累加和,加权系数一般先小后大,以突出后面若干采样的效果,加强系统对参数变化趋势的辨识。各个加权系数均为小于 1 的小数,且满足总和等于 1 的约束条件。这样,加权运算之后的累加和即为有效采样值。为方便计算,可取各加权系数均为整数,且总和为 256,加权运算后的累加和除以 256(即舍去低字节)后便是有效采样值。

设每批采样 8 个数据,依次存放在地址 DATA 开始的单元中。各加权系数用一个表格存放在 ROM 中。MCS-51 指令编写的算术平均滤波程序如下:

```
FILT4:   MOV   R0,#DATA       ;指向采样数据首地址
         MOV   DPTR,# TAB     ;指向加权系数表格首地址
         MOV   R2,#0          ;累加和清零
         MOV   R3,#0
         MOV   R7,#08H
FILT41:  MOV   B,@R0          ;取采样数据
         CLR   A
         MOVC  A,@A+DPTR      ;取加权系数
         MUL   AB             ;加权运算
         ADD   A,R3           ;求累加和
         MOV   R3,A
         MOV   A,B
         ADDC  A,R2
         MOV   R2,A
         INC   DPTR
         INC   R0
         DJNZ  R7,FILT41      ;未完继续
         MOV   A,R3           ;四舍五入
         RLC   A
         CLR   A
         ADDCA,R2
         RET
TAB:     DB    18,22,26
         DB    30,34,38
         DB    42,46
```

上述加权系数表格中 8 个系数按线性递增排列,总和为 256。也可按实际情况自行调整。

5.滑动平均滤波法

以上介绍的各种平均滤波算法有一个共同点,即每取得一个有效采样值必须连续进行若干次采样,当采样速度较慢(如双积分型 A/D 转换)或目标参数变化较快时,系统的实时性不能得到保证。滑动平均滤波算法只采样一次,将这一次采样值和过去的若干次采样值一起求平均,得到的有效采样值即可投入使用。如果取 N 个采样值求平均,RAM 中必须开辟 N 个数据的暂存区。每新采集一个数据便存入暂存区,同时去掉一个最老的数据,保持这 N 个数据始终是最近的数据。这种数据存放方式可以用环行队列结构方便地实现。设环行队列为

40H～4FH 连续 16 个单元,R0 作为队尾指针。滤波程序如下:

```
FILT5:  ACALL INPUT        ;采样新值
        MOV   @R0,A        ;排入队尾
        INC   R0
        MOV   A,R0
        ANL   A,#4FH
        MOV   R0,A
        MOV   R1,#40H      ;初始化
        MOV   R2,#0
        MOV   R3,#0
        MOV   R7,#16
FILT51: MOV   A,@R1        ;取一队列元素
        ADD   A,R3         ;求累加和
        MOV   R3,A
        CLR   A
        ADDCA,R2
        MOV   R2,A
        INC   R1
        DJNZ  R7,FILT51    ;累加 16 个元素
        SWAPA
        XCH   A,R3
        SWAPA
        ADD   A,#80H
        ANL   A,#0FH
        ADDCA,R3
        RET
```

6.低通滤波法

将普通硬件 RC 低通滤波器的微分方程用差分方程来表求,便可以用软件算法来模拟硬件滤波的功能。经推导,低通滤波算法如下:

$$Y_n = a \cdot X_n + (1 - a) \cdot Y_{n-1} \tag{1}$$

式中 X_n——本次采样值;

Y_{n-1}——上次的滤波输出值;

a——滤波系数,其值通常远小于 1;

Y_n——本次滤波的输出值。

由上式可以看出,本次滤波的输出值主要取决于上次滤波的输出值(注意不是上次的采样值,这和加权平均滤波是有本质区别的),本次采样值对滤波输出的贡献是比较小的,但多少有些修正作用。这种算法便模拟了具有教大惯性的低通滤波功能。滤波算法的截止频率可用下式计算

$$f_L = \frac{a}{2\pi t} \tag{2}$$

式中 a——滤波系数;

t——采样间隔时间。

例如当 t = 0.5 s(即每秒 2 次),a = 1/32 时

$$f_L = \left(\frac{1/32}{2 \times 3.14 \times 0.5} \right) \approx 0.01(\mathrm{Hz})$$

当目标参数为变化很慢的物理量时,这是很有效的。另一方面,它不能滤除高于 1/2 采样频率的干扰信号,本例中采样频率为 2 Hz,故对 1 Hz 以上的干扰信号应采用其他方式滤除。

低通滤波算法程序与加权平均滤波相似,但加权系数只有两个:a 和 1 - a。为计算方便,a 取一整数,1 - a 用 256 - a 来代替,计算结果舍去最低字节即可。因为只有两项,a 和 1 - a 均以立即数的形式编入程序中,不另设表格。虽然采样值为单元字节(8 位 A/D),为保证运算精度,滤波输出值用双字节表示,其中一字节整数,一字节小数,否则有可能因为每次舍去尾数而使输出不会变化。

设 Y_{n-1} 存放在 30H(整数)和 31H(小数)两单元中,Y_n 存放 32H(整数)和 33H(小数)中。滤波程序如下:

```
FILT6:    OV    0H,32H          更新 Yn-1
          MOV   31H,33H
          ACALL INPUT           ;采样 Xn
          MOV   B,#8            ;计算 αXn
          MOV   AB
          MOV   32H,B           ;临时存入 Yn 中
          MOV   33H,A
          MOV   B,#248          ;计算(1-α)Yn-1
          MOV   A,31H
          MUL   AB
          RLC   A
          MOV   A,B
          ADDC  A,33H           ;累加到 Yn 中
          MOV   33H,A
          JNC   FILT61
          INC   32H
FILT61:   MOV   B,#248
          MOV   A,30H
          MUL   AB
          ADD   A,33H
          MOV   33H,A
          MOV   A,B
          ADDCA,32H
          MOV   32H,A
          RET
```

除了低通滤波外,同样可以用软件算法来模拟高通滤波和带通滤波。

7. 复合滤波法

为了进一步提高滤波效果,有时可以把两种或两种以上不同滤波功能的数字滤波器组合

起来,组成复合数字滤波器,或称多级数字滤波器。例如防脉冲干扰平均值滤波算法就是一种应用实例。这种算法的特点是,用中值滤波算法滤掉采样值中的脉冲干扰,然后把剩余的各采样值进行递推平均滤波。其基本算法如下

若 $y_1 \leq y_2 \cdots \leq y_N$,其中 $3 \leq N \leq 14$

则 $\bar{y} = (y_2 + y_3 + \cdots + y_{n-1})/(N-2)$ (3)

由于这种滤波方法兼容了递推平均滤波算法和中值滤波算法的优点,所以无论对缓慢变化的信号,这是对快速变化的信号,都能取得较好的滤波效果。

三、结束语

微型计算机在仪器仪表系统中的成功应用,使传统电子仪器以及复杂的跟踪测量装置发生了许多革命性变化,其中一个突出表现就是系统中包含了智能性运作。微机具有很强的分析和运算能力,智能系统可完成复杂的收据处理;智能系统采用软硬件相结合的方法进行随机误差的数字滤波和系统误差的修正,可以实时地修正、校准测量数据,这些都是传统仪器难以比拟的。

参 考 文 献

1 周航慈著 .单片机应用程序设计技术[M] .北京:北京航空航天大学出版社,1991

2 陈粤初,窦振中,吴悌远,刘伟民等编著 .单片机应用系统设计与实践[M] .北京:北京航空航天大学出版社,1991

3 何立民 .MCS-51 系列单片机硬件配置与接口设计[M] .北京:北京航空航天大学出版社,1990

4 吴勤勤,都志杰等编著 .微机化仪表原理设计[M] .华东化工学院出版社,1991

选自《电测与仪表》月刊,2001 年第 6 期

第四章

网络、通信与数据传送

4.1 实时单片机通信网络中的内存管理

合肥市中国科技大学自动化系(230027) 陈小龙 庞国仲 王祥忠

在 AUTO 2000 DCS 系统中,我们采用了单片机实时通信网络。这是一种具有不确定控制站监控的主从式总线网络。大致的工作过程是:网络系统上电,各站点进行控制权争夺确立控制站和非控制站;控制站轮询各站点,被轮询的站点若有数据发送即成为主站,执行主机/ 从机的通信;通信结束后(无论成功与否)主站将控制权归还给控制站;控制站然后轮询下一个站点,周而复始。

为了提高通信效率,采取了一系列相应的措施。例如,我们把站点集分为活动站点集和非活动站点集。这样,单片机网中如有某站点出现故障,该站点被控制站询问时响应将超时,则该站点将从活动站点集转到非活动站点集中。这样,系统在下一次轮询时就不再轮询该站点。故障站点恢复正常后或有新的站点开机上线,控制站将在轮询周期结束后的测试周期中将它们加入到活动站点集中。事实上,控制站轮询的是活动站点,测试的是非活动站点。这无疑是一种提高网络效率的方法。

另外还有一种更为重要的途径即为本文将要介绍的动态内存管理方法。DCS 网络系统所要传输的数据一般有下述几种:较长的周期性数据、较短的随机上报数据及较短的命令/ 响应数据。对周期性数据,稍微的滞后甚至一两次丢失都是允许的;而对随机上报数据和命令/响应数据,则要求尽可能快地完成传输。在传统的静态内存分配方式下,甲网卡接收到乙网卡的数据后在上传至宿主机前宣布内存缓冲区满而不能接收其他站点的数据。这显然是对内存资源的一种浪费,同时也严重影响了网络的通信效率。采用动态内存分配方式后,甲网卡每收到一批数据,只从自己有限的空闲内存中分配出合适的一块来存放该批数据,剩下的空闲内存仍可接收其他网卡的数据,并当宿主机有空时,将所有接收的数据一并上传至宿主机并清空内存。因此通信效率大为提高。

一、内存表结构

为了对网卡上单片机内存进行管理,设置了一张内存状况表(简称内存表)来记录当前内存的使用情况。所谓内存管理,实际上即为内存的分配和回收,主要解决两个问题。

(1) 对需要申请的内存长度,分配程序需从内存表中寻找出合适的空闲区。分配给该批数据使用,并对内存表进行更新。

(2) 进程或作业释放内存资源时,和相邻的空闲区进行链接合并,更新可用表。

具体地,以某一基本容量(视系统通信的数据量而定,在此为 1 KB)对可用内存区域(如 0400H～3FFFH)进行划分和编号(1～15),每一个实际的已分配内存区和空闲内存区在内存表中占据一个表项位置,而每一表项结构为:

标志位	终止块号	块数

这样在某一时刻,可能有 1～15 个独立的内存区(空闲的或已分配的)。也就是说,可用内存表表项的最大数目为 15。但在某一时刻,却可能只有 1 个区(如初始化后只有 1 个空闲内存区)。为了查找某时刻实际内存的分配情况,我们设计了逆向搜索链,该链由内存表项的后两栏组成,如图 4.1-1 所示。

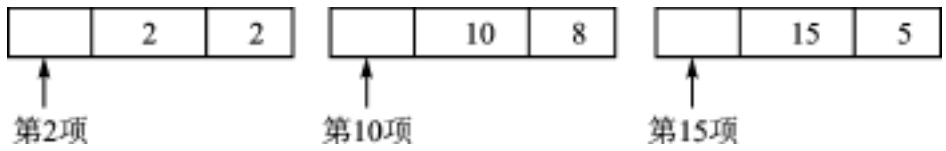


图 4.1-1

该图中,第 15 表项的结束块号肯定为 15,表示编号为 15 的内存区域(3C00H～3FFFH)肯定是某个已分配区域(或空闲区域)的一部分。但该区域究竟有多大,要看第 15 表项的第 3 栏(块数),设为 5。这就表明其相邻的上一区域的结束块号为 15 减去 5。然后查找内存表的第 10 表项,可知其大小为 8。接下来查找第 2 表项,得到其分配块大小为 2。从而可得该时刻内存中实际分配 3 个区,大小分别为 5,8,2。至于内存表中的其他表项在该时刻是无用的。运行初始化程序 init_table 后,内存表的结构如图 4.1-2 所示。

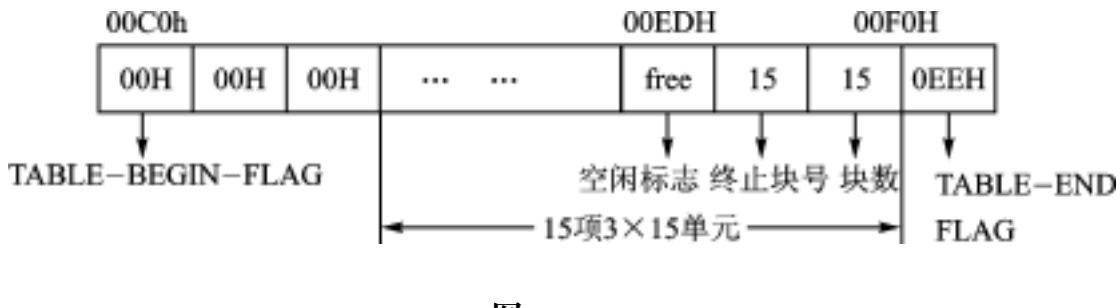


图 4.1-2

其中,00C0H～00C2H 和 00F0H 的地址用来存放表头和表尾的标志,内存表的主要内容有 15 项,每一项 3 个栏目,每一个栏目占用一个实际内存单元。初始化后的 0400H～3FFFH 的 15 KB 内存均为空闲,故从终止块号 15 逆推 15 块,1～15 块均为空闲块,即内存表中只有一个内存区。在内存表的结构图中反映为表中主要内容的第 15 项的标志位为 free,内存区的终止块号为 15,块数为 15,其他表项则是无用的。

二、内存的分配

那么,如何利用这张内存表进行内存分配呢?动态分区分配方法采用最先适应法。用一个例子来说明这个问题。例如经过若干次分配和空闲区回收后,甲站点内存表的主要情况如图 4.1-3 所示(卡上还有一批数据未传入宿主机)。

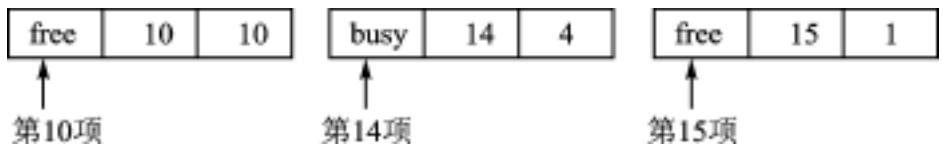


图 4.1-3

在这种情况下,乙站点申请分配 4 块内存,则先把地址指针定位 00EDH 处(第 15 项的标志栏处),由于该项的标志为 free 但大小不够,故需向前查找。用该项的终止块号(15)减去该项的块数(1)得到数字 14,故将地址指针定位到第 14 项处的标志栏处,虽块数大小够但该内

存区为 busy,故仍需往前找。此时用第 14 项处的终止块号(14)减去该项的块数(4)得到数字 10,所以将地址指针定位到第 10 项的标志栏处。第 10 项的块数大于需分配的内存块数,故可以在此处分配。分配后的内存表的主要情况如图 4 .1 - 4 所示。

图 4 .1 - 4

在单片机的汇编程序中,动态分配内存是调用 allocate 子程序来完成的。具体步骤是这样的:

(1) 先由入口参数寄存器 R3 和 R2(存放需要申请的内存长度)折合成需分配的内存块数,并将其值传递给变量 required_size。

(2) 从内存表末(00ED H 处)自后向前找。即先把地址指针定位在 00EDH 处(第 15 项的标志栏处),如果该项的标志不为 free 或该项的标志为 free 但该项的内存块数小于需分配的内存块数,则用该项第二栏的数字(终止块号)减去该项第三栏的数字(块数)作为下一次向前查找的表项号,并由此定位下一次向前查找的地址指针。就这样一直向前查找,直到找到不小于需分配的内存块数或已到内存表头。

(3) 如找到不少于申请块数的空闲区,将空闲区的低端分配给该批数据使用,相应地修改内存表中的相关项,子程序返回分配成功标志;否则,子程序返回分配不成功标志。

三、内存的释放和合并

虽然我们成功地制定了内存分配的算法和子程序,但是在内存管理中还要处理空闲区的释放和相邻空闲区的合并。free_and_join 子程序就是用来完成这个任务的。当接收方(从机)接收完发送方(主机)发来的报文数据并将数据上传给宿主机后,就需要在 free_and_join 子程序中释放该报文数据在本机中所占的内存(free),并且需要检测该被释放的内存与其相邻的空闲内存是否可以合并。如能,合并之。这样,该机如接收其他主机发来的长的报文数据时,可以有足够大的连续的空闲内存来分配;同时,也保证了内存操作的完备性。算法是这样实现的:

(1) 内存表中被释放项的标志改为 free。

(2) 由于表项中的第二栏为终止块号而非起始块号,且在内存表中向前查找是沿着内存表的地址减小的方向进行的,故需先寻找上相邻,再寻找下相邻。

(3) 判断上相邻表项是否 free;如果是,合并之。

用被释放项的终止块号 r_block_end (R2)减去该项的块数 r_block_num (R3),并将其值传给 R4,再调用 block_to_address 子程序来向前找上相邻。如果该项处的标志为 free,则是上相邻。取出上相邻表项处的块数,加到本表项(被释放项)的块数 R3 上,并写入本表项;并将上相邻表项处的标志置为 03H(异于 busy 和 free 即可)。

(4) 判断下相邻表项是否 free;如果是,合并之。

把地址指针定位 00ED H 处(内存表尾),从内存表尾向前找。取出地址指针处的表项的终止块号并存入 R5,再取出该表项的块数并存入 R4。直到 R5 减去 R4 的结果等于被释放项的终止块号为止。若此时地址指针处的表项的标志为 free,则该表项是被释放项的下相邻。

在下相邻表项处,将该项的块数加上被释放项的块数并存入;在被释放项处将标志置为 03H (异于 busy 和 free 即可)。

举一个例子能更好地表明这种动态分区的释放和合并算法。例如将被释放的内存区在内存表中是第 6 块和第 7 块。其上相邻处内存区为第 3,4,5 块,且其标志为 free;其下相邻处内存区为第 8,9,10 块,且其标志为 free;合并的过程如图 4.1-5 所示。

free	5	3	busy	7	2	free	10	3
free	5	3	free	7	2	free	10	3
03H	5	3	free	7	5	free	10	3
03H	5	3	03H	7	5	free	10	8

图 4.1-5

总之,与固定分区法相比,动态分区法在报文接收前不建立分区。分区的建立是在报文接收的过程中进行的,且其大小可随报文长度动态改变,这就改变了静态分区法中的即使是小数据量也要占据大分区的浪费堵塞现象,从而提高了内存的利用率。另外,也提高了 DCS 系统网络通信的短数据的平均传输速率,更适应 DCS 系统的实时性要求。

参 考 文 献

1 王祥忠,庞国仲.具有不确定控制站的主从式总线网络设计.微计算机信息,1998(4)

2 张尧学,史美林.计算机操作系统教程.北京:清华大学出版社

选自《电子技术应用》月刊,2000 年第 1 期

4.2 CRC-16编码在单片机数据传输系统中的实现

成都西南交通大学计算机通信学院(610031) 隗永安

一、引 言

CRC码是常用的检错码,在一些工业控制系统中,如由IBM开发的SDLC通信协议中主要采用CRC-16算法。其生成多项式为

$$G(X) = X^{16} + X^{12} + X^{5} + 1 \qquad (1)$$

串行编码器如图4.2-1。为了节约硬件开销,以计算机为数据终端的传输系统通常由软件完成编码及解码运算。

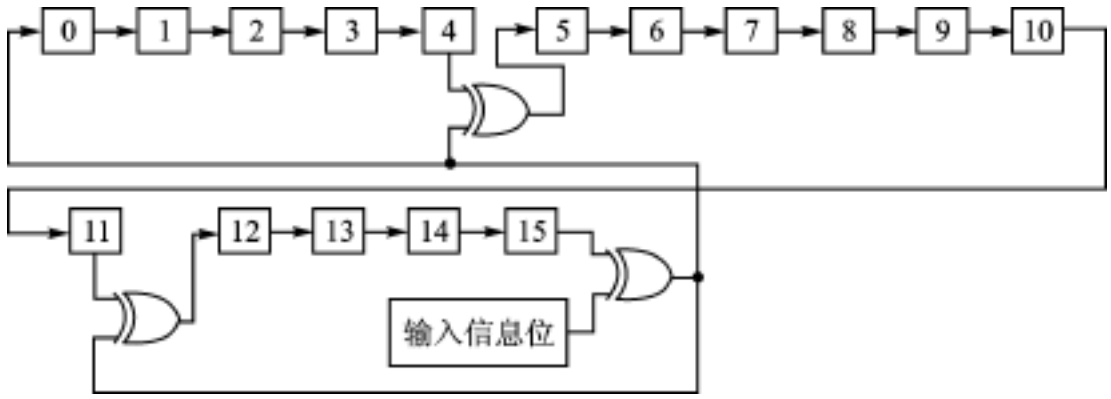


图4.2-1 串行编码器结构图

SDLC通信协议中,标准帧结构由以下部分组成:(1) 标志段(1字节);(2) 地址段(1字节);(3) 控制段(1字节);(4) 信息段(整数个字节),前四段称为信息码;(5) 校验码(2字节)。

循环码编码计算由下式来完成。

$$\frac{D^{n-k}M(D)}{G(D)} = q(D) + \frac{r(D)}{G(D)} \qquad (2)$$

式中 M(D)为上述除校验码之外的四段信息码;

r(D)为余式,也就是校验码;q(D)为商式,弃之不用。

上述5部分由码多项式表示为

$$A(D) = D^{n-k}M(D) + r(D) \qquad (3)$$

若信息段为1个字节,则由五段组成1个(n,k)系统循环码,其中n=48,k=32,可以软件完成上述码的编码运算。当信息码为7F010101H(7F为SDLC帧结构标志段),其码多项式由

$$M(D) = D^{30} + D^{29} + D^{28} + D^{27} + D^{26} + D^{25} + D^{16} + D^{8} + 1 \qquad (4)$$

表示时,则必须由软件完成如图4.2-2所示的长除式运算。

为算出16位余式r(D)需进行长除直式的15层运算。这对于由单片机为数据终端的传输系统,要完成这样的运算,不论是从运算速度还是从存储容量来说都是力不从心的;而完成CRC-16并行编码的运算则是绰绰有余的。以微机为数据终端的系统,虽然以其存储容量完

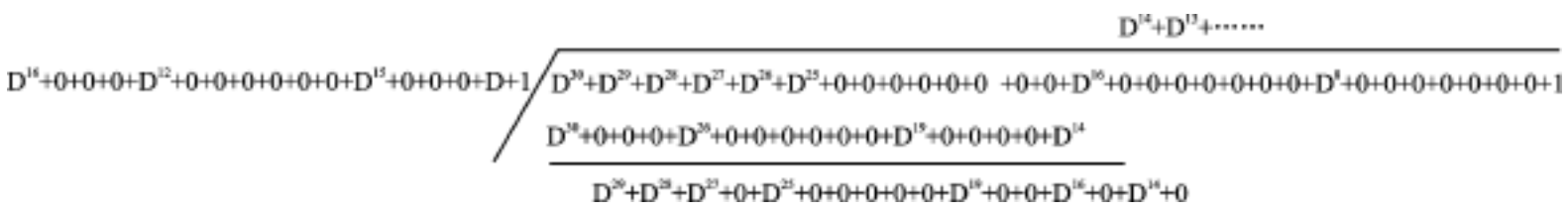


图 4.2-2 长除直式

成复杂运算是不成问题的,但采用 CRC-16 并行编码运算可大大提高运算速度。

二、CRC-16 并行编码器及算法

通过对图 4.2-1 按位输入串行移位的分析,找出编码器的状态与输入数据的关系[1],我们得出 CRC-16 并行编码器的逻辑框图如图 4.2-3 所示。其中 M_0、M_1 为联接矩阵;D 为并行 8 位输入数据;P_0、P_1 为移位寄存器现态;F_0、F_1 为移位寄存器次态。

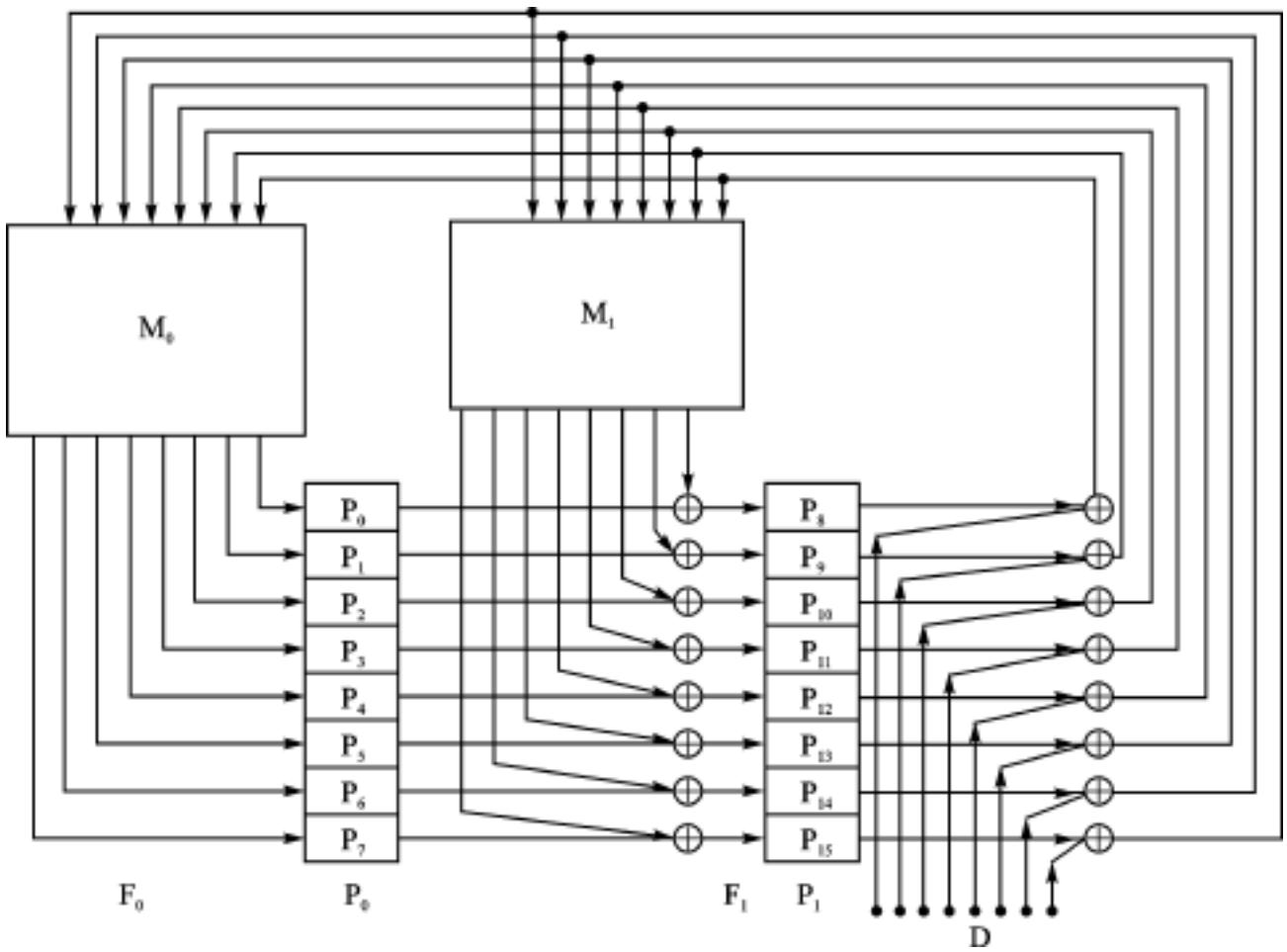


图 4.2-3 并行编码器结构图

$$M_0 = \begin{bmatrix} 1&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&1&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0 \\ 1&0&0&1&1&0&0&0 \\ 0&1&0&0&0&1&1&0 \\ 0&0&1&0&0&0&1&1 \\ 0&0&0&1&0&0&0&1 \end{bmatrix} \quad (5) \qquad M_1 = \begin{bmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 1&0&0&0&0&0&0&1 \\ 0&1&0&0&1&0&0&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&1&0&0&1&0 \\ 1&0&0&0&1&0&0&1 \end{bmatrix} \quad (6)$$

式(5)和式(6)是两个固定矩阵。不管寄存器 P_0、P_1 的现态和并行输入数据 D 为何值，P_1 + D都有 256 种取值，它分别与 M_0 与 M_1 相乘仍各自有 256 种取值。这样我们就可以把

$(P_1 + D)M_0$ 和 $(P_1 + D)M_1$ 制成两个表，分别以 P_1 + D 为偏移量对应放置。

由图 4.2-3 可知，每一帧数据计算之前寄存器 P_0、P_1 先清零。输入数据 D 每输入 8 位可计算出偏移量 P_0 + D 然后查表 4.2-1 得到 $P_0 = (P_1 + D)M_0$，查表 4.2-2 再与 P_0 相异或则为 P_1。一帧数据的每 8 位输入运算完毕，P_0 及 P_1 中则为监督码，放于一帧数据之后，则编码运算完毕。对于具有整数个 8 位的 SDLC 帧结构，这种运算方法是非常合适的。存储两个表，计 512 个字节，完成简单运算，对于任何单片机系统都是力所能及的。

表 4.2-1　并行寄存器 P_0

P_0 + D	$(P_1 + D)M_0$
00	00
01	21
02	42
03	63
04	84
05	95
06	C5
07	E7
08	08
09	29
…	…
FF	F0

表 4.2-2　并行寄存器 P_1

P_1 + D	$(P_1 + D)M_1$
00	00
01	10
02	20
03	30
04	40
05	50
06	60
07	70
08	81
09	91
…	…
FF	17

三、MCS-51 系列单片机为数据终端的传输系统中 CRC-16 编解码运算及源程序清单

根据前述 CRC-16 并行编码器编码算法，编码流程图如图 4.2-4 所示。

根据流程图，编码算法源程序清单如下所示。

以 MCS-51 单片机为终端的数据传输系统中，发送和接收方的数据终端都存储有表 4.2-1 和表 4.2-2，都作相同查表和简单运算，发送端把包括计算出的监督码的数据帧一起发往接收端，接收端以接收到信息码 M(D)(包括标志段、地址段、控制段、信息段)为输入数据进行与发送端相同的查表运算。

得出的监督码与发送端发来的监督码进行比较，若相同则传输无误，若不同则传输错误。

源程序清单：

```
Main program      (发送端主程序)
    OV      MOD,        20              定时器 1 为模式 2
   MOV      TL1,        #0E8
   MOV      TH1,        #0E8
   SETB     TR1,
   MOV      SCON,       #40             ;串行口为模式 1
```

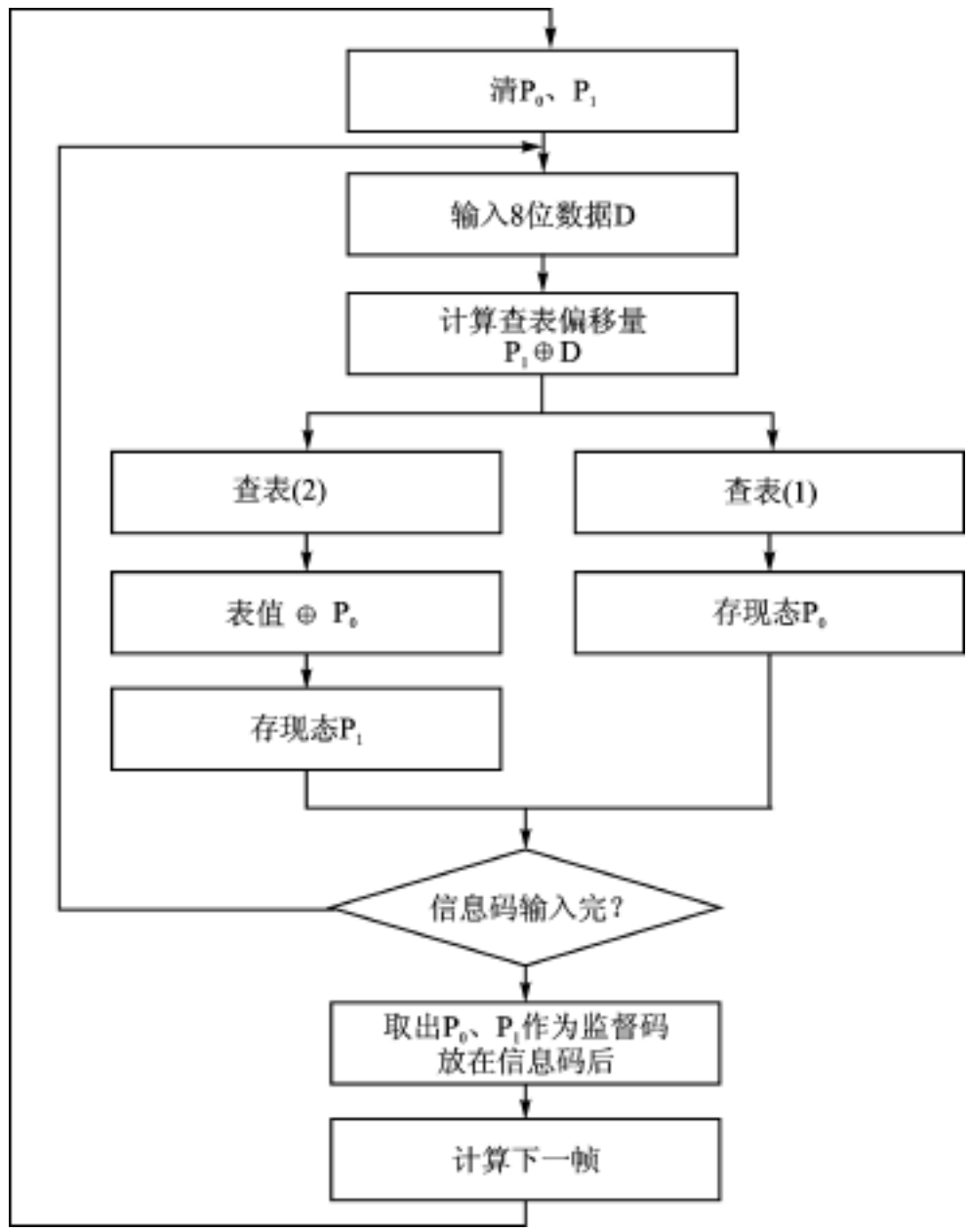


图 4.2－4 编码流程图

```
    MOV   R0,      #20          ;数据区
    MOV   R1       #32          ;字节数
L0  MOV   A,       #00          ;清 P0、P1
    MOV   10H      A
    MOV   11H      A
    MOV   R2       #04          ;每帧 4 字节信息码
L1  MOV   R3,      @R0          ;取数
    LCALL SP－OUT               ;发信息码
    LCALL CODE                  ;编码
    DJNZ  R2,      L2
    MOV   A,       10H          ;一帧编码完,发送监督码
    LCAL  SP－OUT               ;发监督码
    MOV   A,       #11H
    LCALL SP－OUT               ;发监督码
    INC   R0
    SJMP  L0;                   ;发下一帧
L2  INC   R0,                   ;一帧编码未完
```

```
        DJNE   R1,L1
        ……

        SP-OUT      (发送子程序)
        MOV    A,          R3
        MOV    SBUF,       A              ;启动发送
$       JNB    T1          $              ;等待发送完
        CLR    T1
        RET

        CODE        (编码运算子程序)
        MOV    A,          R3             ;取数
        XRL    A,          11H            ;计算偏移量
        MOV    13H,        A              ;存偏移量 P1 + D
        MOV    DPTR,       # M1           ;查表 1
        MOVC   A,          @A + DPTR
        XPL    A,          10H            ;与 P0 异或
        MOV    11H,        A              ;写 P1
        MOV    DPTR,       # M0           ;查表 0
        MOV    A,          13H
        MOVC   A,          @A + DPTR
        MOV    10H,        A              ;写 P0
        RET
```

本文以实际科研开发合作项目为基础成文,当数据传输速率为 1 200 bps,信道误码率为 10^{-3} 时,经长时间运行,尚未发现接收方数据终端所控制的外部设备有误动作。

参 考 文 献

1 陲永安.CRC-16 的一种快速编码器的实现[J] 通信保密,1999(3):61～66

2 孙涵芳,徐爱卿.MCS-51/96 系列单片机原理及其应用[M] 北京:北京航空航天大学出版社,1996

选自《探测与控制学报》季刊,2000 年第 2 期

4.3 在 VC++中用 ActiveX 控件实现与单片机的串行通信

石油大学 李志刚 王以法

在单片机应用系统中,经常需要通过 RS-232 串行口与微机进行通信。目前在各种操作系统中,Microsoft 的 Windows 较为常见,而且大多为 Windows95/98 等 32 位平台。以往在 Windows 平台上的串行通信多使用其提供的 API 函数来实现,这种方法使用起来需要许多底层设置,因而较为繁琐,并且难以理解。Microsoft 推出的 ActiveX 技术提供了另外一种实现串行通信的方法。这种方法不仅相对较为简单,而且非常实用。尤其是在 Visual C++这种可视化面向对象的编程环境中,可以真正把串口看作一个对象,编程时只需简单的设置,理解起来也很容易。下面详细讨论 Microsoft 提供的串行通信 ActiveX 控件的使用方法。该控件的相应文件是 MSCOMM32.OCX,以下简称为 MSCOMM 控件。

一、MSCOMM 控件

MSCOMM 控件,即 Microsoft Communication Control,是 Microsoft 为简化 Windows 下串行通信编程而提供的 ActiveX 控件。它提供了一系列标准通信命令的使用接口,利用它可以建立与串口的连接,并可以通过串口连接到其他通信设备(如调制解调器),发出命令,交换数据以及监视和响应串行连接中发生的事件和错误。MSCOMM 控件可用于创建电话拨号程序、串口通信程序和功能完备的终端程序。

MSCOMM 控件提供了两种处理通信的方式。

(1) 事件驱动方式。当通信事件发生时,MSCOMM 控件会触发 OnComm 事件,调用者可以捕获该事件,通过检查其 CommEvent 属性便可确认发生的是哪种事件或错误,从而进行相应的处理。这种方法的优点是响应及时、可靠性高。

(2) 查询方式。在程序的每个关键功能之后,可以通过检查 CommEvent 属性的值来查询事件和错误。如果应用程序较小,这种方法可能更可取。例如,如果写一个简单的电话拨号程序,则没有必要每接收 1 个字符都产生事件,因为惟一等待接收的字符是调制解调器的“确定”响应。

在使用 MSCOMM 控件时,1 个 MSCOMM 控件只能同时对应 1 个串口。如果应用程序需要访问和控制多个串口,那么必须使用多个 MSCOMM 控件。

在 VC++中,MSCOMM 控件只对应着 1 个 C++类——CMSComm。由于 MSCOMM 控件本身没有提供方法,所以 CMSComm 类除了 Create()成员函数外,其他的函数都是 Get/Set 函数对,用来获取或设置控件的属性。MSCOMM 控件也只有 1 个 OnComm 事件,用来向调用者通知有通信事件发生。

MSCOMM 控件有许多很重要的属性,限于篇幅只给出几个较为重要和常用的属性,如表 4.3-1 所列。

表 4.3-1 MSCOMM 控件的重要属性

属 性	说 明
CommPort	通信端口号
Settings	以字符串形式表示的波特率、奇偶校验、数据位、停止位
PortOpen	通信端口的状态,打开或是关闭
Input	接收数据
Output	发送数据
InputMode	接收数据的类型:0 为文本;1 为二进制

二、编程实现

从表 4.3-1 可以看到,MSCOMM 可以两种不同的形式接收数据,即以文本形式和以二进制形式。用 MSCOMM 控件进行字符数据传输的文献和资料可以找到很多,在 Microsoft 的 MSDN(Microsoft Developer Network)中就可以找到这样的例子,即 VCTERM。可是几乎所有以单片机为核心的测量系统所得到的原始数据都是二进制形式的,所以,以二进制形式传输数据将是最为直接而又简洁的办法。不仅如此,由于 MSCOMM 控件在文本形式下,其传输的是宽字符格式的字符,要想得到有用信息,还要额外处理。因此本文主要讨论在二进制形式下的使用方法。

在 VC++6.0 中,用 APPWizard 可以生成三种应用程序:单文档(SDI)、多文档(MDI)和基于对话框的应用程序。为了说明问题和省去不必要的细节,下面以基于对话框的应用程序为例。

1.创建一个基于对话框的应用程序

打开 VC++6.0 集成开发环境,选择菜单项 File/New,在出现的对话框中选中 Projects 标签中的 MFC AppWizard(exe),然后在 Project Name 框中填入 MyCOMM(可根据需要命名),之后点 OK 按钮。在接着出现的对话框中选中 Dialog Based 项,然后点 NEXT 按钮。以下的各对话框都按照缺省设置,这样即可生成一个基于对话框的应用程序。在资源编辑器中会出现其对话框模板。

2.插入 MSCOMM 控件

选择菜单项 Project/Add to project/Components and Controls…,在弹出的对话框中选择 Registered ActiveX Controls 文件夹下的 Microsoft Communications Control, version6.0,然后按下 Insert 按钮,接着会弹出一个对话框,提示生成的类名及文件名,按 OK 按钮即可实现控件的插入。这时在对话框的控件工具栏上会多出一个电话机模样的控件图标,Workspace 的 Classview 中也多了一个类 CMSComm。

此时即可将 MSCOMM 控件加入到对话框模板中,加入方法与其他控件一样。然后还要在对话框类中相应加入一个成员变量,此处我们将其命名为 m_comm。加入方法为:首先,在对话框模板中,右击该控件,选择 ClassWizard,在出现的对话框的 Member Variables 标签的

Control Ids 项下,选中 IDC_MSCOMM1。然后,按 Add Variable…按钮,在出现的对话框的 Member Variable Name 项中输入 m_comm。最后,按 OK 按钮即可。

3.设置属性

可以在两个地方对控件的属性进行设置:

(1) 对话框资源编辑器中。在对话框模板上,右击 MSCOMM 控件,然后选择 Properties…菜单项,最后便可设置各项属性。此处只对以下几处进行改动,其他接受缺省设置:Rthershold:1,InputLen:1,DTREnable:不选,InputMode:1-Binary。

(2) 对话框类的 OnInitDialog()函数中。下面是以上设置的函数实现:

```
BOOL CMyCOMMDlg::OnInitDialog()
{
  CDialog::OnInitDialog();
  //此处为应用框架自动生成代码,不予列出
  // TODO:Add extra initialization here
  m_comm.SetCommPort(1);                    //使用串口 1
  m_comm.SetSettings("9600,N,8,1");
    //波特率为 9 600,无奇偶校验,8 位数据位,1 位停止位
  m_comm.SetRThreshold(10);                 // 每接收 10 个字符就触发 1 次接收事件
  m_comm.SetSThreshold(0);                  // 不触发发送事件
  m_comm.SetInputLen(10);                   // 每次读操作从缓冲区中取 10 个字符
  m_comm.SetInputMode(1);                   // 二进制数据传输形式
  m_comm.SetPortOpen(TRUE);                 // 打开串口
  return TRUE;                              //  return TRUE unless you set the focus to a control
}
```

4.发送二进制数据

如果需要发送二进制数据,可将数据作如下处理。具体代码如下:

```
CByteArray bytOutArr;
bytOutArr.Add(0×0);                         // 给数组赋值
bytOutArr.Add(0×1);
bytOutArr.Add(0×2);
bytOutArr.Add(0×3);
bytOutArr.Add(0×4);
COleVariant varOut;
varOut = COleVariant(bytOutArr);            // 将数据转换为变体数据类型
m_comm.SetOutput(varOut);                   // 发送数据
```

5.接收二进制数据

当需要接收大量数据时,最好采用事件驱动方式进行编程。具体步骤如下:

(1) 响应 OnComm 事件。在对话框资源编辑器中,双击对话框模板上的 MSCOMM 控件,在弹出的对话框中填入您所希望的事件响应函数名,此处将其命名为 OnCommMscomm1()。

(2) 在事件响应函数中接收和处理数据。接收来的数据为变体数据,所以需要做一些处理,具体代码如下:

```
void CMyCOMMDlg::OnCommMscomm1()
{
   COleVariant varRcv;
   CByteArray byt;
   int i;
   long num;
   switch(m_comm.GetCommEvent())
   {
      case 1://数据发送事件
         break;
      case 2://数据接收事件
         varRcv = m_comm.GetInput();
         varRcv.ChangeType(VT_ARRAY | VT_UI1);
         BYTE HUGEP * pbstr;
         HRESULT hr;
         hr = SafeArrayAccessData(varRcv.parray,(void HUGEP* FAR* )&pbstr);//获取安全数组指针
         if(FAILED(hr)){
            AfxMessageBox("获取数组指针失败!");
            break;}
         num = 0;
         hr = SafeArrayGetUBound(varRcv.parray,1,&num);//获取数组上界
         if(FAILED(hr)){
            AfxMessageBox("获取数组上界失败!");
            break;}
         for(i = 0;i <= num;i++)
            byt.Add(pbstr[i]);
         SafeArrayUnaccessData(varRcv.parray);
      //此时数据已保存在二进制数组 byt 中,可根据需要进行相关处理
         break;
            default:
         break;
      }
}
```

以上代码中的处理部分可以做成一个单独的函数,在此处调用即可。经过以上代码的处理,接收来的数据已存放在二进制数组 byt 中,可以根据自己的需要对其进行相关处理,如保存和显示等。

三、硬件接口

单片机与微机之间的硬件接口可以用 1 片 MAX232 或 ICL232 与几个电容即可实现,有许多文献讨论过,此处不再赘述。

以上方法经过笔者在实践中的应用,感到非常简洁、方便,具有很强的实用意义。

参 考 文 献

1 icrosoft Develeper Network

2 Kruglinski J David .Visual C++技术内幕 .潘爱民,王国印译 .北京:清华大学出版社

选自《单片机与嵌入式系统应用》月刊,2001 年第 6 期

4.4 利用 Windows API 函数构造 C++类实现串行通信

昆明理工大学电力工程学院(650051) 项举伟 高 峰 束洪春

在工业检测和数据采集系统中,计算机和计算机、计算机和单片机之间经常需要进行数据交换,串行通信是主要的通信手段。随着 Windows 操作系统逐步取代 DOS 而成为微机上的主流操作系统,程序员必须掌握基于 Windows 的串行通信编程。

由于 Windows 是一个与设备无关的、基于消息驱动的操作系统,它不提倡用户直接对硬件进行操作,而是由系统自动进行处理。Windows API 提供了一系列标准的串行通信函数,用户可以利用这些标准函数来完成 Windows 系统下的串行通信编程。由于这类函数较多,且其所带的参数也比较复杂,因而使用时很不方便。但程序员可以根据自己应用程序的特点,利用面向对象的程序设计思想,通过将 Win95 API 中有关的数据结构和函数封装起来,构成自己的 C++类。在具体编程时,就不必再考虑 Win95 API 中复杂的通信函数和其他具体的实现细节,只要声明串行通信类 CComm 的一个对象,调用该类的成员函数,就可以完成对串口的初始化和读写操作,从而可以简单、方便、安全地实现基于 Windows 的串行通信编程。

本文提供了用 Visual C++ 6.0 实现的串行通信类 CComm,它封装了部分 Windows API 有关串行通信函数,实现了对串行口的选择、初始化、读端口和写端口等功能,因而能很好地满足用户进行串行通信的基本要求。

一、Windows API 函数和串行通信类 CComm

在 Windows 95 以上版本中,系统把串行通信设备当作一般的普通文件,因而对串行通信设备操作(如串行通信设备的打开、读写和关闭等)均采用与文件操作相同的函数。Win95 API 提供了一系列支持基于 Windows 95 的串行通信标准函数,主要包括:CreateFile()、CloseHandle()、GetCommState()、SetCommState()、SetupComm()、PurgeComm()、SetCommMask()、WaitCommEvent()、ClearCommError、ReadFile()、WriteFile()。

由于 Windows 95 系统中取消了 Windows 3.X 中的 WM_COMMNOTIFY 消息,因此当应用程序工作于“事件驱动”方式时,应创建独立的线程以监视有关的串行通信设备,当串行通信设备上发生指定事件时,就给应用程序发出消息,通知应用程序进行相应的操作。这里也给出一些支持多线程编程的 API 函数:CreateEvent()、SetEvent()、ResetEvent()、WaitForSingleObject()、CreateThread()、ExitThread()。

根据面向对象的编程原理,对以上的函数进行封装,构成串行通信类 CComm。CComm 类的程序代码如下:

串行通信类 CComm 的头文件:

```
class CComm
{
protected:
```

```
    HANDLE hComm;                    //通信设备句柄
    DCB dcb;                         //通信设备的 DCB 结构
public:
    COMSTAT ComState;                //通用设备状态
    OVERLAPPED m_ReadOverlapped, m_WriteOverlapped;
    HANDLE hEvent;                   //串行通信中用于线程间同步的事件句柄
    BOOL m_ExitFlag;                 //退出串行通信标志
    CComm();
    virtual ~CComm();
    HANDLE GetHandle();              //取得通信设备句柄
    BOOL Initialize (LPCTSTRlpCommPortName="COM1", int m_SendBuffSize=4096,
    int m_RecieveBuffSize=4096, DWORD m_RaudRate=9600, BYTE m_ByteSize=8,
    BYTE m_Parity=EVENPARITY, BYTE m_StopBits=0);     //创建及初始化串行口
    void CloseCommPort();            //关闭串行口
    BOOL SetCommDCB (DWORD m_BaudRate, BYTE m_ByteSize, BYTE m_Parity, BYTE m_Stop-
Bits);                               //设置串行通信设备的 DCB 结构
    BOOL Write (char * WriteBuffer, DWORD dwMaxNumber);     //发送 WriteBuffer 中的 dwMax-
                                                            Number 个字节数据
    DWORD Read (char * ReadBuffer, DWORD dwWriteLength);    //接收 dwWriteLength 个字节数
                                                            据放入 ReadBuffer 中
};
```

串行通信类 CComm 的实现文件:

```
CComm::CComm()
{
    hComm=0;
    hEvent=CreateEvent (NULL,TRUE,TRUE,NULL);      //创建同步事件句柄
    m_ExitFlag=FALSE;                              //置退出标志为 FALSE
}
CComm::~CComm()
{
    if (hComm) CloseHandle (hComm);                //关闭通信设备
    if (hEvent) CloseHandle (hEvent);              //撤销同步事件
}
HANDLE CComm::CetHandle()
{
    return hComm;
}
BOOL CComm::Initialize (LPCTSTR lpCommPortName, int m_SendBuffSize, int m_RecieveBuffSize,
DWORD m_BaudRate, BYTE m_ByteSize, BYTE m_Parity, BYTE m_StopBits)
{
    hComm=CreateFile (lpCommPortName, GENERIC_READ|GENERIC_WRITE, 0, NULL, OPEN_
EXISTING, FILE_FLAG_OVERLAPPED, NULL);         //打开串口,并取得通信设备句柄
```

```
    if ( !hComm) return 0;
    if ( !SetupComm (hComm, m_SendBuffSize, m_RecieveBuffSize)) return 0;      //设置通信缓冲区大小
    PurgeComm (hComm, PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURGE_RXCLEAR);      //清空通信缓冲区
    if ( !GetCommState (hComm, &dcb)) return 0;
    if ( !SetCommDCB (m_BaudRate, m_ByteSize, m_Parity, m_StopBits)) return 0; //设置串口波特率、数据位数、奇偶校验、停止位数
    return 1;
}
void CComm::CloseCommPort()
{
    if (hComm) CloseHandle (hComm);      //关闭串口
}
BOOL CComm::SetCommDCB (DWORD m_BaudRate, BYTE m_ByteSize, BYTE m_Parity, BYTE m_StopBits)      //设置通信设备的 DCB 结构
{
    if ( !hComm) return 0;
    dcb.BaudRate = m_BaudRate;
    dcb.ByteSize = m_ByteSize;
    dcb.Parity = m_Parity;
    dcb.StopBits = m_StopBits;
    if ( !SetCommState (hComm, &dcb)) return 0;
    return 1;
}
DWORD CComm::Read (char * ReadBuffer, DWORD dwMaxNumber)
{
    if ( !hComm) return 0;
    DWORD dwBytesRead, dwErrorFlag;
    ClearCommError (hComm, &dwErrorFlag, &ComState);      //获取通信设备当前状态
    if (ComState.cbInQue > dwMaxNumber) dwBytesRead = dwMaxNumber;
    else dwBytesRead = ComState.cbInQue;
    if (dwBytesRead > 0)
    {
        if ( !ReadFile (hComm, ReadBuffer, dwBytesRead, &dwBytesRead, &m_ReadOverlapped)) return 0;      //读 dwBytesRead 个字节到 ReadBuffer 中
    }
    return dwBytesRead;
}
BOOL CComm::Write (char * WriteBuffer, DWORD dwWriteLength)
{
```

```
  if ( !hComm) return 0;
  DWORD dwErrorFlag;
  ClearCommError (hComm, &dwErrorFlag, &ComState);      //获取通信设备当前状态
  if !WriteFile (hComm, WriteBuffer, dwWriteLength, &dwWriteLength, &m_WriteOverlapped))
return FALSE;      //把 WriteBuffer 中的 dwWriteLength 个字节写到输出缓冲区中
  return 1;
}
```

二、应用实例

在串行通信的应用开发中,通常有两种方法:查询法和事件驱动法。查询法一般只适于较为简单的应用,对于复杂的串行通信,应采用事件驱动法。作者在开发计算机和单片机的串行通信应用程序中,采用的是事件驱动法。由于 Windows 95 以上版本中,取消了原来 Windows 3.x 下的 WM_COMM_NOTIFY 消息,所以程序员在 Visual C++6.0 编程环境下必须定义消息如下:

```
# define WM_COMM_NOTIFY WM_USER +10
```

消息映射入口:

```
ON_MESSAGE (WM_COMM_NOTIFY, OnCommEvent)
```

消息响应函数说明:

```
afx_msg DWORD OnCommEvent (UINT wParam, UINT lParam)
```

在本文的串行通信应用程序中,利用计算机收集单片机中存储的数据。计算机和单片机之间采用"软握手"方式,即:计算机发联络信号 0AAH 给单片机,单片机发应答信号 0BBH 予以应答,接着计算机发信号 0A1H 确认接收数据准备好,单片机可以发送数据给计算机。在单片机发送数据,计算机接收数据过程中,采用事件驱动法。为了实时响应事件,作者创建 WatchCom 线程以监视串口有无数据,若串口有数据,则发送消息 WM_COMM_NOTIFY 给应用程序,应用程序运行相应的接收数据处理程序,以完成相应的工作。应用实例程序清单如下:

```
CComm Com;                                  //定义串行通信类的一个全局变量
HWND m_pView;                               //发送消息 WM_COMM_NOTITY 到该窗口的窗口句柄
UINT WatchComPort (LPVOID pView)            //辅助监视线程
{
  DWORD dwEvtMask;
  memset (&Com.m_ReadOverlapped, 0, sizeof (OVERLAPPED));
  SetCommMask (Com.GetHandle(), EV_RXCHAR);      //设置缓冲区收到数据事件
  while(1)
  {
    if (Com.m_ExitFlag==TRUE)::AfxEndThread(1);      //退出辅助线程
    dwEvtMask=0;
```

```
    WaitCommEvent (Com . GetHandle(), &dwEvtMask, &Com . m_ReadOverlapped); // 等待指定
                                                                              事件发生
    if ((dwEvtMask&EV_RXCHAR) = = EV_RXCHAR)
      WaitForSingleObject (Com . hEvent, INFINITE);     // 等待同步事件有信号
      ResetEvent (&Com . hEvent);     // 复位同步事件,使同步事件不发信号
      ::PostMessage (m_pView, WM_COMM_NOTIFY, NULL, NULL);   // 发送消息
}
  return 0;
}
DWORD CComm VCView::OnCommEvent (UINT wParam, UINT IParam)   // 消息响应处理函数
{
  char readBuff [400];
  DWORD m_dwBytesRead = Com . Read (readBuff, 400);                  // 一次最多读 400 字节
  if (m_dwBytesRead > 0)
  {
    // 对接收到的数据进行处理
  }
  SetEvent (&Com . hEvent);                                          // 置位同步事件,使同
                                                                        步事件处于发信号
                                                                        状态
  return m_dwBytesRead;
}
void CCommVCView::OnCommRecieve()                                    // 初始化串口、建立握
                                                                        手信号、启动辅助
                                                                        线程
{
  Com . Initialize();                                                // 初始化串口
  char m_readbuff, m_writebuff;
  itoa (0xAA, &m_writebuff, 16);
  Com . Write(&m_writebuff,1);                                       // 发送 0AAH
    // 进行延时及其他处理
      ……
  Com . Read (&m_readbuff, 1);                                       // 接收应答信号
  if (atoi (&m_readbuff) = = 0xBB)                                   // 若应答信号为 0BBH,
                                                                        则发送 0A1H
  {
    itoa (0xa1, &m_writebuff, 16);
    Com . Write (&m_writebuff, 1);
  }
  else{
    出错处理;
    return;
    };
```

```
    m_pView = m_hWnd;
    AfxBeginThread (WatchComPort, &m_pView, THREAD_PRIORITY_ABOVE_NORMAL);
// 启动辅助监视线程
}
void CCommVCView::OnStoprecieve()                                    // 终止接收程序
{
    Com.m_ExitFlag = TRUE;
    Com.Close();
}
```

三、结束语

在 DOS 操作系统下编制串行通信应用程序,程序员可以对 UART(串行通信芯片)直接操作。但由于 Windows 操作系统与设备的无关性,在开发基于 Windows 串行通信应用程序时,程序员应利用 Windows API 中有关串行通信的函数。进一步,程序员可以根据自己的应用程序特点,把 Windows API 中相关的函数封装起来,构成自己的 C++ 类,这样就可以利用面向对象编程的优点,大大方便应用程序的开发,且程序维护简单,不易出错。

参 考 文 献

1 美]Kruglinski D J. Visual C++ 6.0 技术内幕 北京:北京希望电子出版社,1999

2 沈民,张燕译.Microsoft Windows API 大全 北京:北京科海培训中心,1995

选自《测控技术》月刊,2000 年第 11 期

4.5 用 Win32 API 实现 PC 机与多单片机的串行通信

东南大学 黄 波 张晓晨

用 1 台 IBM-PC 或其兼容机作为主机,多台 MCS-51 单片机作为从机,通过 RS-232C 总线互连而成的主从式多微机串行通信系统,为开发小型分布式控制系统创造了良好的硬件环境。本文结合笔者在工作中的实际经验,介绍在 32 位 Windows 操作系统下,用 VC++6.0 和 MCS-51 汇编语言来开发 PC 机和多 MCS-51 单片机通信软件的一些技巧和方法。

一、系统硬件结构

如图 4.5-1 所示,以 1 台 IBM-PC 为主机,2 台 MCS-51 单片机为从机所组成的总线型多微机串行通信系统为例。IBM-PC 通过机内的异步通信适配器挂在总线上,异步通信适配器的核心是 8250 芯片。但在 Win32 编程下,程序员通过 Win32 API 应用程序接口函数来和通信硬件接口打交道,因此,本文对 8250 芯片的工作原理不予介绍。

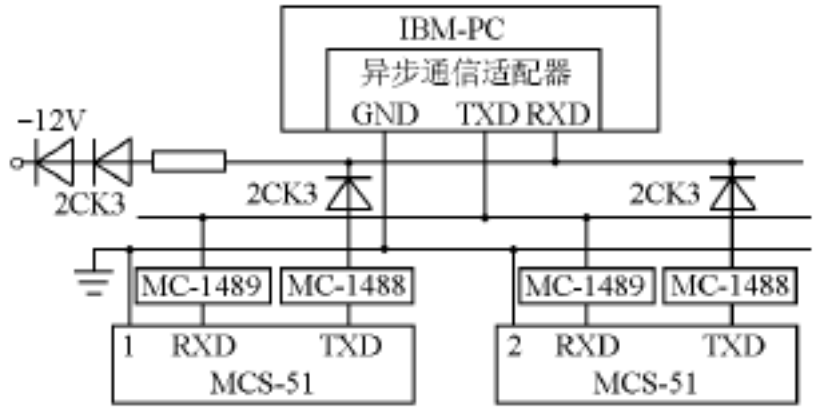


图 4.5-1 用 1 台 PC 与 2 台 MCS-51 组成串行通信系统

每台 MCS-51 都是通过片内的串行接口再经过电平转换器后挂在总线上。IBM-PC 的异步通信适配器符合 RS-232C 标准。MCS-51 串行口提供的信号在功能上支持 RS-232 总线标准,但在电平上不符合其规定,必须经过 MC-1488、MC-1489 变换。但由于 MC-1488 的输出不能连在一起,所以 2 台 MCS-51 发送端的 MC-1488 都须经过二极管隔离后再互连。

二、通信协议

主 PC 机承担主控任务,负责控制参数的设定,程序由 VC++6.0 编写。2 台 MCS-51 单片机接收 PC 机指令,并根据指令信息来控制被控对象或上传数据。通信协议如下:

采用 RS-232C 串口异步通信,1 位起始位;8 位字长;把数据第 9 位设定为“固定奇偶位”,用来支持 PC 和 MCS-51 的相互通信,当第 9 位为 0,表示主机发送的是“数据/命令帧”;当第 9 位为 1,表示主机发送的是“呼叫帧”;1 位停止位。波特率为 9 600 bps。

三、主机通信软件

在 32 位 Windows 环境下,Win32 API 提供了接口函数来与通信硬件接口。通信函数是中断驱动的:发送数据时先把数据放入缓冲区,串口准备好以后将其发出;传来的数据申请中断,使 Windows 接收并将其存入缓冲区,以供读取和解析。在编写通信程序时,用 CreateFile 打开通信资源,SetupComm()设置输入输出队列的大小,GetCommState()获得端口参数当前配置,SetCommState()设置端口参数,ReadFile()及 WriteFile()读、写端口,最后用 CloseFile()关闭端口。函数的使用请参阅 MSDN。为了便于读者熟悉 VC++6.0 的通信编程,在此简要地给出部分用 VC++6.0 编写的主机通信软件:

```
// 创建一工程—Comm,在 CommView.h 中定义变量
HANDLE hComm,hEvent;
HWND hwCommWnd;
BOOL Error,Result,Success,threadFlag;
DCB dcb; // 定义数据块结构参数
DWORD dwEvtMask,dwLength;
OVERLAPPED CommRead,CommRead;
COMSTAT ComState;
char ReadBuffer[num]="";// 根据需要设置 num
char * send,* Receive;
// 在文件头中加入
#define WM_COMM_READ WM_USER+5
// Generated message map functions
// {{AFX_MSG(CCommView)
afx_msg LONG OnCommRead(UNIT wParam,LONG lParam);
// }}AFX_MSG
DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CCommView,CFormView)
// {{AFX_MSG_MAP(CCommView)
   ON_MESSAGE(WM_COMM_READ,OnCommRead)
// }}AFX_MESSAGE_MAP
END_MESSAGE_MAP()
#include"stdafx.h"
#include"CommView.h"
#include"math.h"
#include"stdlib.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[]=_FILE_;
#endif
```

```
IMPLEMENT_DYNCREATE(CCommView,CFormView)
CCommView::CCommView():CFormView(CCommView::IDD)
{
  //{{AFX_DATA_INIT(CCommView)
  //}}AFX_DATA_INIT
}
CCommView::～CCommView()
{
}
void CCommView::OnInitialUpdate()
{
   CFormView::OnInitialUpdate();
// TODO:Add your specialized code here and/or call the base
// class
hComm = CreateFile( "COM2",
                    GENERIC_READ|GENERIC_WRITE,
                    // 允许读写
                    0, // 此行必须为零
                    NULL, // 无安全参数
                    OPEN_EXISTING, // 设置产生方式
                    FILE_FLAG_OVERLAPPED, // 异步方式
                    NULL);
  if(hComm = = INVALID_HANDLE_VALUE)
     {
       MessageBox("端口设置错误","警告",MB_OK);
     }
  Success = SetCommMask(hComm, EV_RXFLAG);
If( ! Success)
{ MessageBox("SetCommMask Error !","警告",MB_OK);
}
Error = SetupComm(hComm,1024,1024);// 设定输入/出缓冲大小
If( ! Error)
{ MessageBox("SetupComm Error !","警告",MB_OK);
}
  Error = GetCommState(hComm, &dcb); // 获得串口设置
If( ! Error)
{
  MessageBox("GetCommState Error !","警告",MB_OK);
}
dcb.BaudRate = 9 600;
dcb.ByteSize = 8;
dcb.Parity = NOPARITY;
dcb.StopBits = ONESTOPBIT;
```

```
Error = SetCommState(hComm, &dcb);
If( ! Error)
{
   MessageBox("SetupCommState Error !","警告", MB_OK);
}
hEvent = CreateEvent(   ULL, // 无安全属性
                        TRUE, // 手工重置事件
                        TRUE, // 初态无信号
                        NULL,// 初态无名字
                        );
if(m_Comm)
{
   threadFlag = TRUE;
   m_Comm - > ResumeThread();
   ::WaiteForSingleObject(m_Comm - > m_hThread,INFINITE);
   delete m_Comm;
}
m_Comm = AfxBegineThread(ComReceive,&m_hWnd,THREAD_PRIORITY_NORMAL,0,
CREATE_SUSPENDED,NULL);
m_Comm - > m_bAutoDelete = FALSE;
threadFlag = FALSE;
m_Comm - > ResumeThread();
hwComWnd = m_hWnd;
}
UINT CommReceive(LPVOID m_View)
{
   DWORD dwRead;
   MemSet(&CommRead,0,sizeof(OVERLAPPED));
   If( ! SetCommMask(hCom,EV_RXCHAR))
   {
      DWORD error = GetLastError();
      Return(FALSE);
   }
   while(1)
   {
      dwRead = 0;
      WaitCommEvent(hComm,&dwRead,&CoRead);
      if(dwRead&EV_RXCHAR) = = EV_EVENT)
         WaitForSingleObject(hEvent,OxFFFFFFFF);
      ResetEvent(hEvent);
      PostMessage(hEvent,WM_COMM_READ,NULL,NULL);
   }
   return 0;
```

```
}
LONG CCommView::OnCommEvent(UNIT wParam,LONG lParam)
{
  ClearCommError(hComm, &dwEvtMask, &comState;
  DwLength = comState .cbInQue;
  If(dwLength > = 7)
    Error = ReadFile(hComm, ReadBuffer, dwLength, &dwLength, &CommRead);
  If( ! Error)
  {
    MessageBox(" 读串口出错 !"," 警告 ",MB_OK);
  }
  SetEvent(hEvent);
  Return 0;
}
void CCommView::Sent()
{
  char SendBuffer =" &ASKQ ? * ";
  Send = SendBuffer;
  Error = WriteFile(hComm,Send,7, &dwLength, &CommWrite);
  If( ! Error)
  {
    MessageBox(" 写串口出错 !"," 警告 ",MB_OK);
  }
}
void( CCommView::OnDestroy()
{
  CFormView::OnDestroy();
  CloseHandle(hComm);
}
```

四、从机通信软件

MCS-51 单片机串口首先进行初始化,把定时器 T1 设置为 9 600baud 的波特率发生器、数据格式为 1 位起始位,8 位字长,第 9 位为呼叫帧与数据/ 命令帧区别位,1 位停止位的 11 位异步方式,与 IBM-PC 一致。采用中断方式控制 CPU 与串行口的数据交换,串行口方式设置为 3。MCS-51 程序清单如下:

主程序清单:

```
COMM1: MOV   TMOD, #20H
       MOV   TH1, #0FDH
       MOV   TF1, #0FDH        ;T1 设置为 9 600baud 的波特率发生器
       SETB  TR1               ;启动 T1
       SETB  EA                ;开中断
```

```
RPT:    SETB   ES              ;允许串口中断
        MOV    SCON, #0F8H     ;串口设位方式 3,第 9 位为 1
        MOV    PCON, #80H      ;波特率系数选择
        MOV    23H, #0CH       ;设置接收数据指针
        MOV    22H, #00H
        MOV    21H, #08H       ;设置发送数据指针
        MOV    20H, #00H
        MOV    R5, #00H        ;累加和 SUM 清零
        MOV    R6, 25H         ;取传送字节数
        MOV    R7, 26H
        INC    R6
        INC    R7
PRTT:   SJMP   PRTT            ;循环等待中断
PRTR:   CLR    ES              ;关中断
PRTR1:  SJMP   PRTR1           ;结束
        …
        ORG    0023H
        LJMP   INPU
        …
```

中断服务程序

```
INPU:   JBC    RI, RI1         ;若 RI = 1,转 RI1 接收,并 RI = 0
INTUR:  JBC    TI, INTUR1      ;若 TI = 1,转 INTUR,并 TI = 0
INTUR1: RETI                   ;返回
RI1:    JNB    SM2, RI3        ;SM2 = 1,呼叫帧, M2 = 0,数据/ 命令帧
        MOV    A, SBUF         ;读入呼叫帧
        CLR    C
        SUBB   A, 27H          ;(27H)中为本机号,进行对号
        JNZ    RI2             ;若机号不等,转 RI2
        MOV    SBUF, #00H      ;对上号向主机发回 00 应答
        CLR    TB8             ;使第 9 位为零
        CLR    SM2             ;使 SM2 = 0
RI2:    RETI
RI3:    DJNZ   R6 ,RI4         ;传送数据字节数完成否
        DJNZ   R7,RI4          ;
        MOV    24H,SBUF        ;最后收到的是 SUM 累加和
        AJMP   TI1             ;转 TI1 进行累加和校验
RI4:    MOV    A,SBUF          ;接收数据
        MOV    DPH,23H         ;把数据放接收缓冲区
        MOVX   @DPTR,A
        ADD    A,R5            ;累加和
        MOV    R5,A            ;
```

```
        INC     DPTR            ;接收指针加 1
        MOV     23H,DPH
        MOV     22H,DPL
        AJMP    TI4
TI1:    MOV     A,24H           ;(24H)中为发送来的累加和
        XRL     A,R5            ;累加和校验
        JZ      TI3             ;正确转 TI3
TI2:    POP     ACC             ;错误,把原保存断点弹出
        POP     ACC
        MOV     DPTR,#RPT
        PUSH    DPL             ;把 #RPT 压栈作为新断点
        PUSH    DPH
        MOV     SBUF,#0FFH      ;用"0FFH"做出错应答
        RETI
TI3:    POP     ACC             ;检验正确,原断点弹出
        POP     ACC
        MOV     DPTR,#RPTR
        PUSH    DPL             ;新断点压栈
        PUSH    DPH
        MOV     SBUF            ;发 00H 作为主机应答
        RETI
TI4:    MOV     DPH,21H         ;发送指针放 DPTR
        MOV     DPL,20H
        MOVX    A,@DPTR
        INC     DPTR            ;发送指针加 1
        MOV     21H,DPH
        MOV     20H,DPL
        MOV     SBUF,A          ;发送完数据
TI5:    RETI
```

本文介绍了利用 PC 机作上位主机,利用多台单片机作下位从机,通过 RS-232C 或RS-422 构成主从式总线型多微机串行通信系统,在实时控制系统中广为应用。

参 考 文 献

1 邱公伟,赵祥元,巫淑萍等著. 实时控制与智能仪表多微机系统的通信技术. 北京:清华大学出版社,1995

2 吴华,岳晋生. Windows NT Win32 软件开发使用详解. 北京:电子工业出版社,1995

3 David J Kjuglinski, Scot Wingo, George Sheperd .Visual C + + 6 .0 技术内幕 第五版(修订版) .北京:北京希望出版社,2001

选自《单片机与嵌入式系统应用》月刊,2001 年第 5 期

4.6 GPS 接收机与 PC 机串行通信技术的开发与应用

(东营市)石油大学自动化系(257062) 郑金吾 朱兴昌
(东营市)胜利石油管理局(257000) 邢 宏

GPS(Global Positioning Sysetm 全球定位系统)是美国于 1993 年 8 月全面建成并运行的新一代卫星导航、定位和授时系统。它主要由 GPS 卫星、地面支撑系统、GPS 用户接收机三部分组成,通过分布在互成 60°的 6 个轨道面上(轨道夹角为 55°)的 24 颗 GPS 定位导航卫星,全天候、连续定时、覆盖全球地发送定位导航和时间信息。卫星的位置是由卫星星历计算出来的,地面支撑系统测量和计算每颗卫星的星历,编辑成电文发送给卫星,然后由卫星实时地播送给用户。GPS 用户接收机通过接收卫星信号解算出自身的位置、速度,以实现定位导航及定时功能。

虽然 GPS 接收机有着强大的功能,但是它也存在着显示面板小,液晶显示难以读取数据,操作菜单繁琐、重复,不易进行查询等缺点,给使用者带来一些不便。GPS 接收机对外提供了串行通信的功能,还提供了 3 个 Port 口用于采集水深仪、潮汐表等第三方数据信息。利用一台便携式计算机与 GPS 用户接收机来读取 GPS 定位信息和第三方数据信息,便可以方便地开发出各种 GPS 应用软件,广泛应用于导航定位、海洋测绘、电子地图、大地测量、姿态测定等领域中。

在 Windows 环境下开发 GPS 应用软件,可以利用 Windows 的丰富资源生成操作简单、美观大方的可视化图形界面。Visual Basic6 .0 是 Microsoft 公司推出的功能强大、易学易用的可视化 Windows 开发软件。VB 为用户提供了很多功能丰富的 Active 控件,Microsoft Comm Control(简称 MCComm)就是一个专门用于串行通信的 Active 控件。本文介绍了利用该控件实现 Windows 环境下 PC 机与 GPS 接收机串行通信,成功地完成了对 GPS 接收机的二次开发。

一、GPS 接收机的结构功能及通信协议

本文以美国 DEL NORTE Technology .-Inc .(DNTI) Global Positioning System Receiver, Model 1008 为例。1008 接收机按照固定的格式“即时”接收来自多达 12 颗卫星的信号。

用户接收机通过检测信号从卫星传到地球的时间,运用 C/A 码技术来测定卫星到地球的距离。用四颗卫星的伪码距就可推算出它的时钟偏差。通过计算,接收器还可以得到在三维地理中,经过真实时间而计算的实时范围。

为了提高位置精度,1008 接收器能获得来自卫星的载波——相位信号。整个载波波长(取整偏差)的准确数值的得到需要至少在 60 min 内有四颗卫星同时传播当前的星历信息。载波——相位仅用于后处理过程中的静态测量数据;而在即时进行过程中不被使用。

由测量的结果可得到 GPS 接收器相对于地球中心的位置。其矢量等式如下:

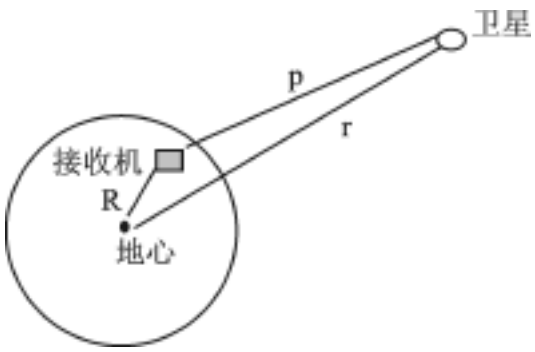


图 4.6-1

$$p = r - R$$

GPS 接收机提供了 2 个串口:可以设置一个串口传送 GPS 定位数据及其他原始的卫星信息数据,同时也接收用户的控制信号;另一个串口设置为接收差分 GPS 数据,一般与电台相连用于接收 GPS 基准站的差分校准数据。接收机提供 3 个 Port 口用于接收第三方数据信息。如测水深的水深仪输出信号可以通过一个 Port 口进入 GPS 接收机,然后与 GPS 卫星定位信号一起通过串口上传入 PC 机。

接收机串口通信协议为:接收差分数据时为 RTCM 差分 GPS 数据格式,波特率为 9 600 bps;无奇偶校验,8 个数据位,1 个起始位,1 个停止位。GPS 接收机传输的数据包括经度、纬度、高程、速度、日期时间、偏移、距离、卫星状态、GPS 状态信息、由 Port 口采集的第三方数据等许多信息。

设置接收机输出 GPS 格式的数据,则每帧数据内容为 D1,…,D11,各位数据的含义如下:

D1:日期(mm/ dd/ yy)9 位;
D2:时间(hh .mm .ss)9 位;
D3:　PS 卫星状态,3 位,
　　0—无效,
　　1—GPS 有效,
　　2—DGPS 有效;
D4:捕获卫星数目,3 位;
D5:位置 Y(经度),15 位;
D6:位置 X(纬度),15 位;
D7:高程,10 位;
D8:事件号,5 位;
D9:测线号,5 位;
D10:标识位 C1,3 位;
D11:水深,10 位。

二、VB 下对 GPS 接收机的数据通信

1 . Microsoft Comm Contrlo 6 .0 ActiveX 控件

在 Windows 环境下,操作系统完全接管了各种硬件资源,一般不允许用户直接控制串行口的中断管理。VB 为用户提供了一个 ActiveX 控件——MCComm 控件,进行计算机串口的通信管理,MCComm 控件有很多属性(Property),其中一些重要的属性如下:

CommPort:设置串口号;类型:short。
Settings:设置串口通信参数;类型:Cstring。
PortOpen:设置或返回通信口的状态;类型:BOOLean。
Input Mode:设置或返回 Input 属性取回的数据类型;类型:Integer。
InBufferSize:接收缓冲区的大小;类型:Integer。
InputLen:设置并返回从缓冲区读取的字节数;类型:Integer。
OutBufferSize:发送缓冲区的大小;类型:Integer。
HandShaking:设置握手信号;类型:Integer。
Rthreshold:设置并返回的要接收的字符数;类型:Integer。
Sthreshold:设置并返回传输缓冲区中允许的最小字符数;类型:Integer。
EOFEnable:确定在输入过程中 MCComm 控件是否寻找文件结尾(EOF)字符;类型:Boolean。

其中串口号(CommPort)设置为 1、2,表示采用串口 COMM1,COMM2 进行通信。参数设置(Settings)的格式为“B,P,D,S”,B 表示波特率;P 表示奇偶校验(N 为无校验,E 为偶校验,O 为奇校验);D 表示字节有效位数;S 表示停止位数。串口状态(PortOpen)为 Boolean 变量,TRUE 表示打开串口,FALSE 表示关闭串口。InputMode 用于选择从缓冲区读取数据的格式,设置为 0 时为字符串格式(Text),设置为 1 表示二进制格式(Binary)。HandShaking 用来设置握手信号,0 表示无握手信号;1 表示 comXOnXOff(XON/XOFF)方式握手;2 表示 RTS/CTS(request to send/clear to send)方式握手;3 表示 request to send 和 XON/XOFF 握手皆可。EOFEnable 若设置为 True 则当 EOF 字符找到时 OnComm 事件被激活,设置为 False(缺省)则当 EOF 字符找到时 OnComm 事件不被激活。

当串口有错误发生或有通信事件(如输入缓冲区满)发生时,系统会激活 OnComm()事件,用户可以在该事件中根据 CommEvent 的返回值添加处理代码,则可以实现串口处理程序。

2.通信软件编制

在 VB 环境下,在工程中放入一个窗体 Form(取名 Forml),在 Forml 中放置一个 MSComm 控件(取名 MSComml)。将 GPS 接收机设置为实时发送数据,发送的数据有日期、系统时间、GPS 卫星状态、接收卫星数目、经度、纬度、水深、高程。在 OnComm()消息处理函数中加入响应的处理代码,取出所需要的数据进行显示、存盘等处理,下面给出用事件驱动方式读出 GPS 接收机 GPS 数据的程序源代码。

在 Forml_load()事件中对串口初始化如下:

```
Private Sub Form_Load()
    MSComml .CommPort = 1                          设置串口 1 进行通信
    MSComml .Settings =“9 600, n, 8, 1”            '串口参数设置
    MSComml .InputMode = 0                         '设置 Text 缓冲输入方式
    MSComml .InBufferSize = 200                    '设置输入缓冲区大小为 200
    MSComml .RThreshold = 0                        '设置当输入缓冲区满时再触发 OnComm()事件
    If MSComml .PortOpen = False Then MSComml . PortOpen = True    '打开串口
End Sub
```

OnComm()事件处理程序代码:

```
Private Sub MSComml_OnComm()
    Dim myinput As String
    myinput = MSComml .Input
…   '相应的数据处理代码
End Sub
```

由于通信中所传的数据内容一帧为 95 位,所以将输入缓冲区的大小(InputSize)设为 200,这样就可以保证每次通信中都有一帧完整的数据,将这一帧完整的数据取出并分离出需要的各种数据进行相应的代码处理,便可实现定位、导航、数据保存的功能。

三、应 用

利用上述 PC 机与 GPS 接收机的串口通信技术,我们为某油田浅海公司开发了一套海洋

导航和水深测绘系统。油田浅海作业如建设海洋平台、铺设海底输油管道等都必须对海底资料十分清楚,尤其是水深数据,这就需要绘制海底等高线图。水深数据由水深仪测得,将水深仪的输出接到 GPS 接收机的 Port 口,同 GPS 定位数据一起通过串口送入 PC 机,这样就可以实现两种功能:一是为作业船只进行动态导航定位,并同时记录 GPS 定位数据和水深资料;二是导航结束后,由系统的管理部分综合保存的数据进行处理,形成打印报表,并实现按任意比例绘制出航迹图和水深图,从而为油田海洋作业提供可靠的海底水深资料。该系统 PC 机与 GPS 接收机串行通信程序如下:

```
Private Sub MSComm1_OnComm()
    Dim instring As String                          从串口得到的 200 位数据
    Dim substring (11)                              '保存各定位数据和水深数据
    Dim thewz1, thewz2 As Integer
    instring = Trim (MSComml .Input)                '从串口取得数据
    MSComml .PortOpen = False
    thewz1 = InStr (1, instring, " / ", 1)
    thewz2 = InStr (thewz1 + 1, instring, " / ", 1)
If (thewz2 < 6) Or ((thewz2 - thewz1) > 5) Then
    instring = Trim (Mid(instring, 5))
    thewz1 = InStr (1, instring, " / ", 1)
    thewz2 = InStr (thewz1 + 1, instring, " / ", 1)
End If
instring = Trim (Mid (instring, thewz1 - 2, thewz2 - 3))     '从串口送入的数据中取出一帧完整数据

'下面程序实现各定位数据和水深数据的分离
thewz1 = InStr (1, instring, " ", 1)
substring (0) = Trim (Mid (instring, 1, thewz1))
instring = Trim (Mid (instring, thewz1))
thewz1 = InStr (1, instring, " ", 1)
substring(1) = Trim (Mid (instring, 1, thewz1))
instring = Trim (Mid (instring,thewz1))
thewz1 = InStr (1, instring, " ", 1)
…

'如果该次数据需要保存并且数据信息正确,则将数据保存在数据库文件中
If issave = TRUE and val (substring (2)) = 1 and val (substring (3)) > 4 Then
  With Datal . Recordset
    If . EOF = False Then . MoveLast
     .AddNew
     ! starstate = Val ( substring (2))
     ! pstate  = Val (substring (3))
    …
  End With
  Datal . UpdateRecord
```

```
End If

  If isdw = True Then Call dingwei          进行定位操作
  If ispx = True Then
    Call pxdh                               '进行平行线动态导航操作
  Else
    Call zxdh                               '进行直线动态导航操作
  End If
End sub
```

该系统投入使用后,具有精度高、效率高和运行可靠等特点,大大提高了海洋测绘工作的效率,已取得良好的社会效益。该项技术也可以应用于其他类似的工业场合中。

选自《测控技术》月刊,2000年第8期

4.7 TCP/IP 协议问题透析

中国科学院计算所网络研发中心(100080) 杨 燊 王 钢

一、引 言

随着计算机和通信技术的发展,人们能够越来越方便地实现信息的共享,TCP/IP 协议使得世界上不同体系结构的计算机网络互联在一起形成一个全球性的广域网络 Internet,这为各种信息的共享提供了便捷的途径。在 Internet 中的每一台计算机可以访问 Internet 上的其他任意一台计算机,好像它们在一个局域网内用双绞线或同轴电缆直接连接起来一样(不同之处是速度比局域网的慢)。目前,TCP/IP 协议已成为网络通信协议的标准,这关键在于 TCP/IP 能够适应不同的网络体系结构和不同的传输链路,为客户端-服务器模式提供了很好的支持。TCP/IP 并不是完美无缺的,TCP/IP 在性能和安全方面都有很多缺点,尤其当它应用于无线通信和安全性要求很高的领域时,这些缺点就显得非常明显。

随着 TCP/IP 在无线通信中的应用日益增加,将会有大量的移动主机连接到 Internet 中,包括卫星链路在内的无线链路将会越来越多。这需要深入研究 TCP/IP 在无线通信中存在的问题和 TCP/IP 的安全问题。为此本文对最流行的 TCP/IP 实现作了认真的分析,指出了存在的问题,并对这些问题给出了解决对策。

二、TCP 的性能问题

1.TCP/IP 应用于无线链路时的缺陷

无线链路的特点是易受干扰、多径衰减的影响。信道通信行为会随时间和地理位置而变化,链路层差错控制对包一级的 QoS (Quality of Service)的影响也会随时间变化。因此为固定网络开发的 TCP 无法很好地应用于移动通信和卫星等无线链路中,这是因为 TCP 缺乏网络自适应性。

在有线网络中,流量控制和资源分配策略均基于底层的物理媒质是高度可靠的这一假定,但这对无线网不成立。在无线网上进行 TCP 传输,TCP 认为包的丢失是由拥塞引起的,而实际上包丢失可能是由于信道错误引起的包丢弃或网络延时而引发的,这将导致 TCP 超时并启动拥塞控制算法,这显然不必要,也减少了无线信道的吞吐率。

在无线信道上的传输性能可以通过使用链路层差错控制的方法 ARQ (Automatic Repeat Request)和前向错误纠正 FEC (Forward Error Correction)来改善。ARQ 用于传输要求高可靠性的数据,FEC 用于传输时延敏感的数据流。变种的 ARQ/FEC 方案更适合于宽带无线网的传输,特别是在传输的数据流表现出不同的特色和 QoS 要求时。

在无线通信中还有一个问题就是当主机不断移动时,它可能离开其 IP 地址标识的那个区域,从而无法连接到网络中。解决主机移动性问题的基本难题是:主机的 IP 地址有双重意义,它既是对主机的惟一标识,又指示它所在位置。而移动主机的位置要经常变化,这意味着 IP

地址也应改变,这是 TCP/IP 无法解决的问题。Mobile IP 作为一种移动主机协议解决了这个问题,其办法是:在主机标识到位置的最近映象可利用时,把 IP 地址的主机和位置意义分开。

移动主机协议的主要设计目标是不需要用户重新配置就可以改变访问点,这称为操作透明性。移动主机协议设计的另一个目标是支持 Handoff(在连接保持时改变访问点叫 Handoff),但不会严重扰乱正在进行的数据传输。

许多建议的移动主机协议都有"Two tier addressing"结构,即一个移动主机在逻辑上关联两个 IP 地址:一个是本地地址,用作不变的主机标识;另一个地址反映它当前连接到 Internet 的点。这种结构由以下三个基本成分组成:

(1) 本地目录　一个数据库,可以是分布式的,它包含最新的两个地址空间之间的映象。

(2) 地址翻译代理　完成主机标识到实际目标地址的翻译,这涉及查询本地目录或本地缓存。

(3) 转发代理　确保到达移动主机的包的目的地址字段有它恒定的本地地址。

Mobile IP 解决方案满足操作透明目标而且也支持 Handoff,但是它只是对慢速移动的主机是优化的。若主机经常移动,它的效率极其低下。Mobile IP 要求移动主机无论在什么时候移动到一个新的外部代理(Foreign Agent),都应通知本地代理。在修改信息阶段,包被转发到旧的位置,而不会扰乱正常的数据传输。类似,在路由优化阶段,尽管相应的主机已经获得一个新的绑定,数据传输也不会被打断。这些延迟的影响会随着 Handoff 出现的频率的增加而增加。修改信息是由 Internet 和本地代理负责的。这个负担与移动主机的数量成比例,而与产生的流量无关。随着移动主机越来越普遍,蜂窝越来越小,这可能是一个问题。

Cellular IP 提供了一个与蜂窝电话差不多的框架,该框架能够把操作从小的办公系统扩展到大的校园网和区域网。每一个都使用同样的 Cellular IP,只是有不同的本地管理设备环境。Cellular IP 协议有一个有效的本地管理和搜索方案,这可以避免对闲着的用户的跟踪,因而可以降低无线访问网络的负载。

2. TCP 慢速启动算法的延迟响应策略和初始窗口大小降低了 TCP 的性能

慢启动算法在传输开始时和发生拥塞时开始启动。许多 TCP 在连接空闲某一段时间(1个往返时间)之后也使用。慢速启动算法的主要目的是用来决定可利用的网络带宽,阻止发送方发送大量的数据淹没中间网关从而引起数据丢失。从这一点来讲,它是很好的,但是慢速启动算法使得 TCP 在慢速启动期间对于那些链路的带宽时延乘积较小的链路和 RTT (Round Trip Time)较大的链路(如卫星网)不能很好地利用网络带宽。TCP 的延迟响应算法一直被证明是有损于 TCP 性能的。慢速启动算法经常是以大量的报文段丢失而终止的,这显然有损 TCP 的性能。

降低慢启动对性能影响的方法是:采用一个合适的 SSTHRESH (Slow Start Thresh,慢启动阈值),使得慢启动结束不会出现大量报文段的丢失。这就需要估计合适的 SSTHRESH,同时采用不同于标准的慢速启动算法的 cwnd (Congestion Window,拥塞窗口)的增加策略。另外,使用大的初始 cwnd,增加初始的 cwnd 的大小对较短的数据流和低带宽网络最为有利。慢启动算法把初始 cwnd 设置为 1,这使得发送方必须等待接收方延迟响应计时器的时间才能收到接收方的响应来发送新的数据段。这显然严重降低了连接带宽的利用率。假如首先使用一个较大的窗口 IW(Initial Window),在发生超时重传时才设置为 1,这就能提高慢启动开始时的网路带宽利用率。

$$W_I = \min(4 * MSS,\ \max\ (2 * MSS,\ 4\ 380))$$

其中 W_I 为初始窗口大小,MSS(Maxium Segment Size)为网络能够传输的最大数据单元的字节数,这样做的好处是缩短了到达接收方报告的窗口尺寸所需要的时间。采用大的初始窗口后,慢启动所需时间将变为:

$$\text{慢启动时间} = R(1b\ W_A - 1b\ W_I)$$

(不采用延迟响应,每一个报文段都响应)

$$\text{慢启动时间} = 2R(1b\ W_A - 1b\ W_I)$$

(采用延迟响应,每两个报文段给出一个响应)

其中 W_A 为接收方报告的窗口大小。这对于大数据量的块传输没多大影响,因为随着传输长度的增加,慢启动对总体性能的影响会降低。

如果 MSS = 1 024,则 $W_I = 4$,将节省 3RTT 和一个延迟响应时间(因为不采用大的初始窗口时,cwnd = 1;另外,因为 cwnd = 1 时,接收端需要等待延迟响应超时才发送响应)。

增加 W_I 也有缺点,在高度拥塞的环境下,这可能增加 TCP 连接的报文段丢失,也会引起在共享的高度拥塞的瓶颈竞争 TCP 连接上的报文丢失。

在慢启动过程中,cwnd 一直增加到接收方报告的窗口大小或发送方检测到网络拥塞。慢速启动算法的窗口增加策略是,每收到一个 ACK(响应报文段),cwnd 增加一个报文段的长度,如果接收端对接收到的每一个报文段都发送 ACK,假设往返时间 RTT 为一不变的数,并且 cwnd 增加过程中没有发生拥塞,那么慢启动所需时间就是:

$$\text{慢启动所需时间} = R\ 1b\ W_A$$

其中 R 为往返时间 RTT,W_A 为接收方报告的窗口大小。

如果把响应的数量减少一半,那么 cwnd 到达需要的时间就会翻倍。即:

$$\text{慢启动所需时间} = 2R\ 1b\ W_A$$

因而 TCP 所采用的延迟响应策略也影响了慢启动,即减慢 cwnd 的增加速度,从而影响了 TCP 的性能。使用受限字节计数(Limited Byte Counting)算法可以减轻延迟响应对 TCP 性能的影响,使用 LBC,cwnd 增加的数量基于每一个 ACK 覆盖的新的数据段的数量不是一个常数。但 LBC 在基于丢失恢复的慢速启动阶段,对 cwnd 的增加有点冒进。由于在丢失恢复的慢启动间 LBC 传输额外的不必要的报文段,所以 ABC(Appropriate Byte Counting)比 LBC 更合适。ABC 增加了吞吐率,而 SABC(the Scaled version of ABC)对 cwnd 的增加提供了更好的粒度控制。

3 . TCP 协议的重传超时计时器存在的问题

超时时间值的确定对 TCP 协议的性能有很大的影响。RTO(Retransmisson Timeout)值是一个包发送出去到发送者认为该包已经丢失需要被重传所经历的时间,发生超时重传叫 Timeout。RTO 是对 RTT 的上限值的推测值,RTT 是从一个包离开发送者到发送者接收到接收者发送的对该包的响应所经历的时间。在发生包超时前一直由 REXMT(Retransmission Timer State)维持时间值。RTO 是 REXMT 的初始值。重传超时计时器就是指 REXMT 和 RTO。

RTO 较小的超时器常常过早超时,就会引起不必要的流量,降低一个连接的有效吞吐率,也会引发拥塞控制,导致端到端的吞吐率的进一步恶化。而 RTO 较大的重传计时器会导致丢失包重传前有一段很长的空闲时间,这也会降低性能,也会在发送者用完窗口时影响块数据

的传输。

目前,大多数 TCP 使用的估计 RTO 算法(如流行的 TCP 实现版本 TCP-Lite-4.4 BSD(-Lite)是:

$$DELTA_L = RTT_{Sample} - SRTT_L$$

$$SRTT_L = SRTT_L + 1/8 \cdot DELTA_L$$

$$RTTVAR_L = RTTVAR_L + 1/4 \cdot (|DELTA_L| - RTTVAR_L)$$

$$RTO_L = MAX(SRTT_L + 4 \cdot RTTVAR_L, 2ticks)$$

其中 RTO_L 要在完成每一个新的 RTT 测量后修改。SRTT(Smoothed RTT)称作平滑 RTT 估计器。$SRTT_L$ 是一个低通过滤器,它能以固定的加权因子 7/8($SRTT_L = SRTT_L + 1/8 \cdot DELTA_L = 7/8 \cdot SRTT_L + 1/8 \cdot RTT_{Sample}$)记忆连接的 RTT 历史。$DELTA_L$ 是最新的 RTT_{Sample} 与当前的 $SRTT_L$ 的差。RTTVAR 是平滑的 RTT 偏差的估计器。利用 RTTVAR,RTO 可以解释 RTT 的变化。也是一个低通过滤器,它能以固定的加权因子 3/4 ($RTTVAR_L = RTTVAR_L + 1/4 \cdot (|DELTA_L| - RTTVAR_L) = 3/4 \cdot RTTVAR_L + 1/4 \cdot |DELTA_L|$)记忆连接的 RTT 偏差的历史。1/8 和 1/4 是估计器增益,4 是 RTT 偏差权重。

REXMT 和 RTO 都是用时钟滴答来计量的,一个时钟滴答一般为 1 秒的几分之一,这与操作系统有关,时钟滴答也称作计时器粒度。RTO_L 最小为两个时钟滴答,因为 TCP-Lite 中一个时钟滴答为 500 ms,而实际的计时中,一个时钟滴答可能是 0～500 ms 的任何值,所以就用两个时钟滴答。TCP-Lite 为每一个 TCP 连接都分别维护一个 REXMT,当一个报文段被发送并 REXMT 没有变化时,就初始化 RTO_L。当对最老的报文段的响应到达并有很多老的报文段时,REXMT 重新用于初始化 RTO_L。

RTO 最小值对 Lite_Xmit_Timer(TCP-Lite 中实现的重传超时计时器)的性能有较大的影响,如果计时器粒度为 100 ms 或更低,其性能会更进一步提高。另外,RTT 取样率也影响 Lite_Xmit_Timer 的性能,当 RTT 取样率高、TCP 发送者负载较大时,估计器增益的选择和偏差权重的选择变得特别关键,而这两个值的取值应该依赖于 RTT 取样率。

TCP 的操作和性能很大程度上取决于路径特性:如可利用带宽、端到端时延、包丢弃模式。理想情况下,一个设计得很好的重传计时器应能在任何端到端路径上都执行得很好。但是在 Internet 上,路径的特性随时间的变化非常大。从上面的分析可见,Lite_Xmit_Timer 存在一些问题,这些问题已经严重影响了 TCP 的性能。Lite_Xmit_Timer 的 4 个重要问题是:

(1) 当 RTT 下降时有预测的错误。在计算 $RTTVAR_L$ 时,使用 $DELTA_L$ 的绝对值,这会导致当 RTT 下降而 RTO 增加。也就是说,它会导致连接的 RTT 已经下降到 SRTT 以下(即 $DELTA_L$ 变成负值)后,RTO 还增加。在这种情况下给人的感觉好像是 RTT 增加了同样幅度的值后对 RTO 的影响。这导致 RTO 过高预测了 RTT,因而需要花费一段时间才能变到合理的水平。

(2) 固定的增益和偏差权重不合适。Lite_Xmit_Timer 在定义时基于这样的假定,对每个 flight(即一个 RTT 时间内能够传送的报文段数),只有一个段被计时,估计器和偏差都调整到适应于这种情况。但如果 RTT 采样率较高(也即对每一个 flight,不只计时一个报文段)、发送方负载较大时,固定的增益和权重是失败的。问题在于在这种情况下,偏差权重太低无法使 RTO 达到足够的水平,估计器增益也太高。这会导致 $RTTVAR_L$ 变化得特别快,因而 RTOL 很快会变到 RTT,这显然太激进。

(3) REXMT-Restart Bug。REXMT 实现的问题是,当一个响应最老的报文段的响应到达时,它会被 RTO_L 重新初始化,但还有更多的段等待响应。这无法解释最老的报文段的生存期。因而发生第一次超时前 REXMT 是 RTO_L 与最老报文段的生存期的和,这使 REXMT 太保守。

(4) 计时器粒度。由于 RTO 是对 RTT 上限的预测,所以计时器粒度越高(即一个时钟滴答的时间较长),RTO 就越不精确,就越保守,因而需要一个低粒度计时器。TCP – Lite 的 500 ms 的计时器粒度是不够的。在最坏情况下,RTT 的数量级大概为几百 ms,因而计时器粒度应当为 10ms 或几十 ms。

REXMT 问题是基于 Heartbeat timer 的,Heartbeat timer 每 500 ms 过期并触发一个特定的中断例程修改每个活动的 TCP 连接的 REXMT(减少一个时钟滴答)。这样做没考虑 REXMT 实际上是否到达 0,仅增加 Heartbeat timer 的频度会增加处理器的开销,处理一些无意义的中断。这对于繁忙的服务器是一个很大的问题,这也是定义 RTO_L 最小值的理由。另外一个理由是,一个时钟滴答的 REXMT 的值可能在 0～1 个时钟滴答的任何时间过期。为了解决 Lite_Xmit_Timer 的这些问题,对 RTO 的估计可采用一种新的算法:

$DELTA_E = RTT_{Sample} - SRRT_E$

$FLIGHT_E = MAX(SSTHRESH, CWND/2) + 1$

$GAIN_E =$ / $FLIGHT_E$,如果 RTT 采样率为 1

2/ $FLIGHT_E$,如果 RTT 采样率为 2

1/3,如果在每个 RTT 内 RTT 取样为 1

$VARGAIN_E = GAIN_E$,如果$(DELTAR_E - RTTVAR_E) >= 0$

$GAIB_E$ * *2,如果$(DELTAR_E - RTTVAR_E) < 0$

$SRTT_E = SRTT_E + GAIN_E * DELTA_E$

$RTTVAR_E = RTTVAR_E + VARGAIN_E * (DELTAR_E - RTTVAR_E)$,如果 DELTAE >= 0

$RTTVAR_E$,如果 DELTAE < 0

$RTO_E = MAX((SRTT_E + (1/VARGAIN_E) * RTTVAR_E), RTT_{Sample} + 2ticks)$

上面用的计时器称之为 Eifel_Xmit_Timer。为避免 Lite_Xmit_Timer 的问题(1),在 Eifel_Xmit_Timer 中定义 $RTTVAR_E$ 在 $DELTA_E < 0$ 时保持不变。这样,RTOE 只跟 $SRTT_E$ 一样快地减少,对 RTTVAR 定义的细微变化就导致了在 RTT 下降时 RTO 不会出现尖峰。为避免 Lite_Xmit_Timer 的问题(2),在 Eifel_Xmit_Timer 中去掉了恒定的估计器增益。对 $SRTT_E$ 和 $RTTVAR_E$ 的每一个的增益都随发送端的负载变化,并依赖于 RTT 采样率。如果每个 RTT 内多于一个报文段被计时,这有助于把整个权值 1 平均分配到每个 flight 的 RTT 取样上,即把两个估计器的记忆限制到一个 RTT 上。取样率为 2,指的是延迟响应,即每收到两个报文段发送一个响应。把偏差权重定义为估计器增益的倒数,使得它也随发送方的负载变化。在 RTT 一直长时间保持恒定后突然增加的情况下,它能保证 RTO_E 是 $SRTT_E$ 和 $DELTA_E$ 的和。其中 $FLIGHT_E$ 的定义加 1 是因为一个拥塞避免周期中第一个 flight 的报文段数量为 SSTHRESH + 1 或 CWND/2。

在 Lite_Xmit_Timer 中,当 RTT 增加时,RTOL 也增加,但 RTO_L 的增加会在每一个 flight 的报文段发送到一半途中就结束增加,并在 flight 的后一半报文段发送期间迅速下降。这显然在发送方最大负载较小时成了一个问题:在每一个 flight 结束时,RTO_L 的值变得非常

接近于 RTT。为了避免这个问题,在 Eifel_Xmit_Timer 中定义了 $RTTVAR_E$ 的增益在 $RTTVAR_E$ 下降时为 $GAIN_E$ 的平方。同时使用与发送方负载成反比的 $GAIN_E$ 为增益,在每一个 flight 的后半部分报文段发送期间,RTO_E 的值保持平滑而不至于急剧下降。

RTO 的最小值对于保护不合理的超时是必要的,那时可能 RTT 非常接近于甚至低于计时器粒度。在其他情况下 RTO 的最小值是没有意义的。当使用 Heartbeat timer 时,RTO 最小值必须是两个时钟滴答。另外避免让 RTO 降到低于最近的 RTT 取样是合理的,这在 FreeBSD 操作系统中已经实现。

对于 Lite_Xmit_Timer 的问题(3),在 Eifel_Xmit_Timer 中通过仅仅在一个动态的数据结构中存储每一个报文段的时间戳就消除了。这样就能通过公式:REXMTE = RTOE - 最老报文段的生存期,精确确定 REXMT。

在一个连接的 flight 中没有足够的报文段触发快速重传算法的情况下,也就是说重传必须依赖于重传计时器时,$REXMT_E$ 和 $REXMT_L$ 相比能够大大改善端到端连接的性能。

通过大量的实验数据已经证明 Eifel_Xmit_Timer 的性能比 Lite_Xmit_Timer 性能更好,Eifel_Xmit_Timer 解决了 Lite_Xmit_Timer 存在的所有问题。当然 Eifel_Xmit_Timer 在付诸使用前,还需要在各种环境下进行测试。总之,在优化端到端的重传计时器时要注意以下问题:

① 低于 SRTT 的 RTT 采样不应当用于修改 RTTVAR;

② 估计器的增益和偏差权重应当依赖于采样率;

③ 如果每一个段被计时来测量 RTT,即使用时间戳可选项,那么估计器的增益和偏差权重需要因发送端的负载而变化。

4. TCP 协议的 RTO 估计和带宽估计策略的不足

对于传输协议,它利用网络状况的信息越多,就越能有效地用网络传输数据。在 Internet 中,传输协议要经常对连接进行测量来产生对网络特性的估计。两个最基本的估计问题是 RTO 计时器的设定和连接的可利用带宽的估计。对于 RTO 的估计,估计器的性能主要取决于它的最小值,计时器粒度的影响不大。带宽的估计远没期望的好,因而有必要开发一种新的算法来改善。

如果发生包丢失,说明网络出现了拥塞,因而发送者应该等待比最大的 RTT 更长的时间再发送包,这样可以使拥塞有更多的时间从网络中流出。如果 RTT 时间一过,发送者就重传,那重传的包也会被丢失,但稍后发送则能成功。在标准的 TCP 中,RTO 估计器的缺陷是:

(1) 在测量与重传的包相关的 RTT 时有二义性;

(2) 每个往返只重传一个丢失的包保守的 RTO 策略;

(3) 很难选择一个初始的估计;

(4) 在拥塞期间不能迅速跟踪增加的 RTT。

这些问题都已经得到了解决。SACK(Selective Acknowledgement)解决了问题(2)。Jacobson 引入了对 RTT 差值的 EWMA 估计(TCP-Lite 的估计器对 RTT 差值的估计算法),从而使 TCP RTO 估计更好。这些已经在 BSD 版的 TCP(现在最流行的 TCP 版本)中实现。但是 BSD 版的 TCP 实现仍然有一些限制:① 适应性的 RTT 和 RTT 差值估计器只能在一个往返用新测量的值修改一次,因此它们非常慢速地适应于网络状况的变化;② 测量使用 500 ms的时钟粒度,那必然产生粗略的估计;③ 结果 RTO 估计有一个大的最小值 1 s,这使它

必然保守。

随着高精度的时钟的出现和时间戳可选项的利用,这 3 个限制可以被消除。但如何更好地确定 RTO 估计器呢?在 BSD 版的 TCP 中,因为在任何时间内只对一个报文段和它的响应计时,所以每 RTT 时间内仅对 RTO 和 RTTVAR 修改一次,所有的测量都使用时钟粒度。RTO 被限定在 RTOmin 和 RTOmax 间,在 BSD 版中,时钟粒度 G = 0 .5 s, RTOmin = 2G = 1 s, RTOmax = 64 s。测量使用时钟粒度是非常粗略的,所以 RTOmin 没有设置为 0。

对 RTO 估计器的实现的 3 个建议是:

(1) 为每一个段的 RTT 计时,而不是仅对一个 flight 做一个 RTT 计时;

(2) 使用较小的时钟粒度;

(3) 降低 RTOmin 以便花费更少的时间等待超时。

RTOmin 的设置对 RTO 估计器性能的好坏有非常重要的影响。一般 RTOmin 被设置成两个时钟滴答,原因是一个时钟滴答可能是 0～G 秒的任何一个时间。但这样的设置是保守的,实际的 BSD 版使用的超时为 0 .5～1 s。对于 G = 1 ms 的情况, RTOmin 又显得太大,也会产生重要的影响。

时钟粒度越粗略,就越无法监视 RTT 的变化;而时钟精度越精密,估计器就越能反映变化。

坏的超时反映了 RTT 比 RTO 小得多或 RTO 小于或略大于 RTT。坏的超时不仅使有用的吞吐率遭受损失,更严重的是把 SSTHRESH 减半,开始一个新的慢启动。另外 TCP 要发送重传的包,除非使用 TCP 时间戳可选项,否则它不能安全地测出那些包的 RTT,因此为了改进它中断的 RTT 估计使 TCP 能够适应它的 RTT 估计,这需要花很长时间。

解决这一坏的超时问题的方法是:如果 TCP 使用了时间戳可选项,它能够记住上一次重传的 SSTHRESH 和 cwnd,从而能够在发现超时没有必要时恢复它们。如果没有时间戳或 SACK,可采用启发式思想:无论什么时候,当由于 RTO 引起 TCP 重传时,测量 ΔT(从重传开始到收到 ACK 的时间间隔),如果 ΔT 比迄今测量的 RTT 的最小值都小,那么可以推断 ACK 已经在重传发送之前就被传送,那么超时就是坏的,若 ACK 在最小超时之后,说明超时可能是必要的。

TCP 在不知道网路带宽情况下,必须形成对带宽的估计。具体做法是指数级增加发送速率直到出现包丢失。包丢失表明发送速率太大,因而速率减半,再以保守的方式增加。如果用 SSTHRESH 能够给出连接的可利用的带宽的精确估计,拥塞避免用于探测另外的带宽,即以保守的线性增加的方式增加 cwnd,就可以提高带宽的利用率。

5 . TCP 应用于非对称的网络中性能严重降低

在现代的网络中,带宽不对称是经常的,即在一个方向上的带宽跟在另一个方向上的带宽差一个或几个数量级。这样在高带宽方向上, TCP 的传输性能将会大大受到在相反方向上的响应包延迟的影响而锐减。卫星链路就是非对称的,下行带宽要比上行带宽宽得多。如果从接收端到发送端的带宽非常有限,那么一个响应可能要在接收端的传输点经历很长的排队时延,这会减慢 TCP 发送端的发送处理,降低吞吐率。在非对称的网络中, TCP 连接的下行流就受到影响,尤其是在上行流已经占用了很大部分的带宽的情况下,几乎没有剩余的带宽供响应(ACK)利用,从而更恶化了下行传输。

解决这些问题的方法有:

(1) 在每一个连接的上行流管理一个包队列。这使得各响应有平等的机会被传输而不被多个大的数据包延迟。

(2) 慢启动后延迟响应。即在慢启动后缩减响应的数量,对几个包发送一个响应。这在慢速启动前是不可取的。因为 cwnd 的增加依赖于每一个响应,延迟响应就意味着 cwnd 增加速度放慢,这会降低连接的性能。因此在 cwnd 较小时,响应频率高一些,而在 cwnd 较大时响应频率低一些。

(3) 响应过滤,也是一种延迟响应的方法,但它取决于发送方发送的速度,发送得快,返回的响应就少,发送得慢,返回的响应就多。响应是预先产生的,在发送方发送速度快时就把原生成的响应修改为对现在的数据段的响应,这称为过滤。

(4) 依赖于拥塞窗口响应,即接收方跟踪发送方的 cwnd,然后根据 cwnd 决定响应的速度。

(5) 头压缩,因为发送的许多包都有很多相同的字段值,因而可以对两边的协议作一修改,使得接收端发送的响应只有少量不相同的数据,然后在发送端自动恢复,可节省带宽。

这些方法在一定程度上均可改善非对称网络的下行传输性能,但由于涉及修改工作的复杂程度各不相同,需要从中筛选一种较好的方法。第(4)种方法就比较好,因为它仅需要修改接收端,并且实现起来也不复杂。那么如何在接收端估算发送端的 cwnd 呢?方法是:如果 RTT 比发送端发送窗口内数据段发送所需时间长很多,TCP 发送端是完全可能在一个 RTT 时间内发送接近于窗口内的包数的包。这样接收端就可以通过计算在 RTT 内接收到的包数来估计发送端 cwnd 的大小,对于 RTT 的估计,下面列出了几种方法:

(1) 如果接收端也在发送数据包,那么 RTT 可以从接收端的 TCP 发送进程获得。

(2) 在建立 TCP 连接时,通过计时 SYN(SYN 报文段)和 SYNACK(SYN 报文段响应)来估算 RTT。

(3) 接收端的 ICMP 发送带时间戳的 echo 请求,导致发送端的 ICMP 发送 echo 响应,那么 RTT 的值就可从请求与收到的响应时间间隔估算。

(4) 通过使用 TCP 时间戳可选项,发送端在每一个数据段设置一个时间戳,接收端在响应中回送那个时间戳。这样发送端仅做减法就可以精确估算 RTT。

方法(1)依赖于应用程序,如果应用程序没有要发送的数据包,该方法就不可用。另外,由于带宽不对称,发送上行流的发送端测量的 RTT 要比发送下行流的发送端测量的 RTT 要大。方法(2)只在连接开始时测量 RTT,而 RTT 是随时间变化的,因而使用从连接开始时测量的 1 次或 2 次值是不够的。方法(3)需要发送另外的包,但开销是合理的,几乎所有的主机都响应 echo 请求,因而它可以很容易地使用。方法(4)要求两边都支持 TCP 时间戳可选项,而旧版 TCP 不支持这一功能。另外有时发送端没有数据要发送,因而接收端就不可能收到时间戳响应。

对 RTT 的估计可以采用方法(1),在终端没有数据要发送时再采用方法(3)。

根据测量的 cwnd——est-cwnd,采用的响应策略:

ppa = 1,当 est_cwnd < = min_ack_per_win

MIN(FLOOR(est_cwnd/ min_ack_per_win), max_packets_per_ack),其他

其中 ppa(the numbers of packets per acknowledge)是每一个 ACK 响应的包数,min_ack_per_win = 3, max_packets-per_ack 大于 5。限制每一个 ACK 响应的包数有两个原因:

① 每一个 ACK 响应的包数大意味着接收端会抑制许多传输机会,这将降低吞吐率。

② 如果每一个 ACK 响应的包数较大,而每个响应将响应许多包,这可能导致发送端发出迸发包,而导致网络中出现包丢失。

这种对 TCP 的改进非常简单,但却能提高 TCP 在非对称网络中的传输性能,在卫星这种高代价的链路中,它是非常好的提高性能的方法。

6 . 基于 TCP/ IP 协议的路由器拥塞控制机制的缺撼

TCP/ IP 的设计基于这样一个假定: best-effort,即所有应用是协作的,采用相同的端到端拥塞控制机制,路由器采用简单的先来先服务调度算法和尾部丢弃缓存管理算法。但现在一些应用已不采用协作方式,如基于 UDP 的声音、视频应用就是非拥塞响应的,还有一些应用为获的更多带宽而不使用端到端的拥塞控制机制。这就需要路由器来增强自身的拥塞控制机制。

随机提前检测(Random Early Detection)缓存管理方法的缺点是它以相同的概率丢弃所有连接的数据报,这对未获取公平带宽的连接是不利的。

先来先服务调度算法对应于连接的缓存占用量分配带宽,因而它不能有效控制非拥塞响应的连接对带宽的过度占用。

公平排队调度算法可以隔离和保护 TCP 连接,限制非拥塞响应连接对别的连接的影响,但它只能给排队连接公平分配带宽。因此,还需要配合采用 FBD(Fair Buffer Drop)缓存管理算法来限制非拥塞响应连接的数据报在网络拥塞时过度地占用缓存,来保证 TCP 连接公平地占用缓存。

尾部丢弃算法是缓存占满时,路由器丢弃每个新接收的数据包。它的优点是实现非常简单,但缺点是网络拥塞时,路由器丢弃突发而来的数据报,使链路利用率急剧下降。

随机提前检测算法使路由器设置两个缓存门限参数 low 和 high,把拥塞控制过程分为正常运行、拥塞避免和拥塞控制三个阶段。随机提前检测路由器将计算平均缓存队列长度,如果平均缓存队列长度小于 low,为正常运行阶段,路由器不丢弃数据包;若平均队列长度超过 low 但小于 high 就为拥塞避免阶段,路由器就按照一个随平均缓存长度递增的概率丢弃每个接收到的数据包,如果平均缓存队列长度进一步增加并超过 high,则为拥塞控制阶段,路由器将丢弃每个新接收到的数据包。

尾部丢弃算法和随机提前检测算法对各数据包采用相同的丢弃策略,当存在非拥塞响应连接时,它们不能有效隔离和保护 TCP 连接。这样非拥塞响应连接将阻止 TCP 连接的数据包排队,而使公平排队调度算法难以发挥作用。

因此有必要对各个连接分别进行缓存管理,即路由器按照各连接分配缓存,当缓存占满时,根据各连接占用缓存情况,对各连接的数据包采用不同的丢弃策略,这样,一方面可以限制连接过分使用缓存,如网络拥塞时丢弃非拥塞响应连接的数据包来保证 TCP 连接能公平使用缓存;另一方面路由器仅丢弃少量数据包,可使连接利用率保持相对稳定。

7 . 操作系统内部保守的 TCP/ IP 协议的实现

报文处理的时间开销是由报文的大小决定的。大报文的处理时间主要决定于数据接触操作所消耗的时间(如拷贝和计算校验和),这些操作必须应用于每一个数据字节。小的报文只有少数几个数据字节,因而大部分处理时间用于非数据接触操作(如数据结构处理、错误检查、网络缓存管理、操作系统函数和特定协议的处理等)。数据接触操作开销时间随报文的大小呈线性变化,而非数据接触操作开销的时间相对稳定。对于大报文,校验和的计算要占据整个报

文处理时间的将近一半。现在的固定网络传输介质的可靠性是非常高的,在协议的每一层都有数据出错处理,这显得有点重复。另外经过大量的实验数据证明,在不计算校验和的情况下,网络的工作状况仍然非常好,因而有必要降低甚至消除大部分校验和处理。消除大部分校验和处理而不牺牲可靠性是能够很大程度上改善吞吐率的。对于非数据接触操作的时间开销是很难削减的,该时间几乎遍布于所有的操作,因此要实现这种开销时间的节省需要优化许多不同的机制。分段也会导致更多的非数据接触时间开销。对于大的报文段,网络缓存管理也需要较大的开销,在大部分 TCP/IP 实现中,采用了 BSD 版的网络缓存管理方法,使用 mbuf 结构进行内存分配,每一个只有 128 字节,如果数据报大,可能需要调用好几次 mbuf 结构的内存分配算法,才能容纳下大的数据报,因而要消耗较多的时间。如果采用更好的网络缓存管理办法,如采用较大的 mbuf,在一定程度上能够降低非数据接触操作的时间开销,提高 TCP/IP 协议的性能。

三、基于 TCP/IP 协议网络的安全问题

由于 TCP/IP 协议主要目的是用于科学研究,所以很少考虑安全性方面的问题。但随着应用的普及,它已经成为 Internet 网络通信协议的标准。它不仅用于一些要求安全性很高的军事领域,也应用于商业领域,因而对其安全性的要求越来越高。下面从 TCP/IP 协议本身来看它的安全漏洞,以便应用时参考。

基于 TCP/IP 协议的网络存在的安全问题包含:运行协议的操作系统的网络安全漏洞、防火墙自身也存在安全性问题、网络内部的用户的威胁、TCP/IP 协议族本身存在很多安全隐患。下面从各协议层来看存在的安全漏洞。

(1) 链路层存在的安全漏洞。在以太网中,信道是共享的,任何主机发送的每一个以太网帧都会到达别的与该主机处于同一网段的所有主机的以太网接口。而 CSMA/CD 协议使以太网接口在检测到数据帧不属于自己时,就把它忽略,不会把它发送到上层协议(如 ARP、RARP 层或 IP 层)。如果稍作设置或修改,就可以使以太网接口接收不属于它的数据帧。例如使用杂错接点(能接收所有数据帧的机器节点)。解决该漏洞的对策是:网络分段、利用交换器、动态集线器和桥等设备对数据流进行限制、加密(采用一次性口令技术)和禁用杂错接点。

(2) 网络层漏洞。几乎所有的基于 TCP/IP 的机器都会对 ICMP echo 请求进行响应。所以如果一个敌意主机同时运行很多个 ping 命令向一个服务器发送超过其处理能力的 ICMP echo 请求时,就可以淹没该服务器使其拒绝其他服务。另外,ping 命令可以在得到允许的网络中建立秘密通道从而可以在被攻击系统中开后门进行方便的攻击,如收集目标上的信息并进行秘密通信等。解决该漏洞的措施是拒绝网络上的所有 ICMP echo 响应。

(3) IP 漏洞。IP 包一旦从网络中发送出去,源 IP 地址就几乎不用,仅在中间路由器因某种原因丢弃它或到达目标端后,才被使用。这使得一个主机可以使用别的主机的 IP 地址发送 IP 包,只要它能把这类 IP 包放到网络上就可以。因此如果攻击者把自己的主机伪装成被目标主机信任的友好主机,即把发送的 IP 包中的源 IP 地址改成被信任的友好主机的 IP 地址,利用主机间的信任关系(Unix 网络软件的开发者发明的术语)和这种信任关系的实际认证中存在的脆弱性(只通过 IP 确认),就可以对信任主机进行攻击。这是所说的信任关系是指一个被授权的主机可以对信任主机进行方便的访问。所有的 r * 命令都采用信任主机方案,所以一个攻击主机把自己的 IP 改为被信任主机的 IP,就可以连接到信任主机并能利用 r * 命令开

后门达到攻击的目的。解决这个问题的办法是,让路由器拒绝接收来自网络外部的 IP 地址与本地某一主机的 IP 地址相同的 IP 包的进入。

(4) ARP 欺骗。ARP 协议在对 IP 地址进行解析时,利用 ARP 缓存(也叫 ARP 表)来进行。ARP 缓存的每一条目保存有 IP 地址到物理地址的映射。如果在 ARP 表中没有这样的对应条目,ARP 协议会广播 ARP 请求,获得对应于那个 IP 地址的物理地址,并把该对应关系加入到 ARP 表中。ARP 表中的每一个条目都有一个计时器,如果计时器过期,该条目就无效,因而从缓存中删除。显然,如果攻击者暂时使用不工作的主机的 IP 地址,就可以伪造 IP - 物理地址对应关系对,把自己伪装成像那个暂时不使用的主机一样。克服此问题的方法是,让硬件地址常驻内存,并可以用 ARP 命令手工加入(特权用户才可以那样做);也可以通过向 RARP 服务器询问来检查客户的 ARP 欺骗。因为 RARP 服务器保留着网络中硬件地址和 IP 的相关信息。

(5) 路由欺骗。在路由协议中,主机利用重定向报文来改变或优化路由。如果一个路由器发送非法的重定向报文,就可以伪造路由表,错误引导非本地的数据报。另外,各个路由器都会定期向其相邻的路由器广播路由信息,如果使用 RIP 特权的主机的 520 端口广播非法路由信息,也可以达到路由欺骗的目的。解决这些问题的办法有,通过设置主机忽略重定向信息可以防止路由欺骗;禁止路由器被动使用 RIP 和限制被动使用 RIP 的范围。

(6) DNS 欺骗。网络上的所有主机都信任 DNS 服务器,如果 DNS 服务器中的数据被攻击者破坏,就可以进行 DNS 欺骗。

(7) 拦截 TCP 连接。攻击者可以使 TCP 连接的两端进入不同步状态,入侵主机向两端发送伪造的数据包。冒充被信任主机建立 TCP 连接,用 SYN 淹没被信任的主机,并猜测 3 步握手中的响应(建立多个连接到信任主机的 TCP 连接,获得初始序列号 ISN(Initial Serial Number)和 RTT,然后猜测响应的 ISN,因为序列号每隔半秒加 64 000,每建立一个连接加 64 000)。预防方法:使所有的 r * 命令失效,让路由器拒绝来自外面的与本地主机有相同的 IP 地址的包。RARP 查询可用来发现与目标服务器处在同一物理网络的主机攻击。另外 ISN 攻击可通过让每一个连接的 ISN 随机分配配合每隔半秒加 64 000 来防止。

(8) 使用 TCP SYN 报文段淹没服务器。利用 TCP 建立连接的 3 步骤的缺点和服务器端口允许的连接数量的限制,窃取不可达 IP 地址作为源 IP 地址,使得服务器端得不到 ACK 而使连接处于半开状态,从而阻止服务器响应别的连接请求。尽管半开的连接会因过期而关闭,但只要攻击系统发送的 spoofed SYN 请求的速度比过期的快就可达到攻击的目的。这种攻击方法一直是一种重要的攻击 ISP(Internet Service Provider)方法,这种攻击并不会损害服务,而是使服务能力削弱。解决这种攻击的办法是,给 Unix 内核加一个补丁程序或使用一些工具对内核进行配置。一般的做法是,使允许的半开连接的数量增加,允许连接处于半开状态的时间缩短。但这些并不能从根本上解决问题。实际上在系统的内存中有一个专门的队列包含所有的半开连接,这个队列的大小是有限的,因此只要有意使服务器建立过多的半开连接就可使服务器的这个队列溢出,从而无法响应其他客户的连接请求。

四、结束语

TCP/ IP 协议的两个最重要的协议是 TCP 协议和 IP 协议,我们着重从 TCP 协议入手分析了它的慢速启动算法,延迟响应和固定增益与偏差权值的 RTO 估计器等对 TCP 性能的影

响,提出了一些应对措施,另外,就安全问题也作了详细分析,对大部分问题给出了预防对策。虽然这些应对措施和解决策略中的一些可能是权宜之策,但在很大程度上,它们确实能够解决目前所面临的问题。在文中描述的 RTO 估计器没有在广泛的环境中经过大量的实验论证,距正式使用还有待更进一步的研究和改进。在将来,移动主机和无线链路会迅速增加,而目前对 TCP/IP 的大量研究主要基于固定的有线网络,因而对 TCP/IP 在无线应用中的研究是一个非常有挑战性的工作,我们下一步要做的工作就是在这方面的研究,给出整体的解决方案。

参 考 文 献

1 eorge Xylomeros, George C. Polyzos. Internet Protocol Performance over Networks with Wireless Links [J]. IEEE NETWORK, 1999, 13(4)

2 Chris Hayes, Shawn Ostermann. An Evaluation of TCP with Larger Initial Windows[J]. Computer Communication Review, 1998, 38(3)

3 赵季中,宋政湘,齐勇.对基于 TCP/IP 协议的几个网络安全问题的分析与讨论[J].计算机应用研究,2000, 17(5):44～47

4 陈依群,顾尚杰,诸鸿文.一个基于连接的增强拥塞控制机制[J].计算机研究与发展,2000, 3

5 Jeong Geun Kim, Marwan M Krunz. Bandwidth Allocation in Wireless Networks with Guaranteed Packet-Loss Performance[J]. IEEE/ACM Transaction on Networking, 2000, 8(3)

6 Jonathan Kay, Joseph Pasquale. Profiling and Reducing Processing Overheads, in TCP/IP[J]. IEEE/ACM Transaction on Networking, 1996, 4(6)

7 Reiner Ludwing, Keith Sklower. The Eifel Retransmission Timer[J]. Computer Communication Review, 2000, 30(3)

8 Ivan Tam Ming-chit, Du jinsong, Weiguo Wang. Improving TCP Performance Over Asymmetric Networks [J]. Computer Communication Review, 2000, 30(3)

9 Andras G Valko. Cellular IP: A New Approach to Internet Host Mobility[J]. Computer Communication Review, 1999, 29(1)

10 e Macro Vivo, Gabriela O. De Vivo and Germinal Isern. Internet Vulnerabilities Related to TCP/IP and T/TCP[J]. Computer Communication Review, 1999, 29(1)

11 Mark Allman. TCP Byte Counting Refinements[J]. Computer Communication Review, 1999, 29(3)

12 Mark Allman, Vern Paxson. On Estimating End-to-End Network Path Properties[J]. Proceedings of ACM/SIGCOMM 99 CONFERENCE, Computer Communication Review, 1999, 29(4)

13 RFC 2581 TCP Congestion Control[R]. April 1999

14 RFC 2018 TCP Selective Acknowledgement Options[R]. October 1996

15 RFC 2582 The New Reno Modification to TCP s Fat Recovery Algorithm[R]. April 1999

选自《计算机应用研究》月刊,2001 年第 7 期

4.8 单片机的 MODEM 通信

西安交大长天软件有限公司 梁亚光

我们经常能见到关于 PC 的 MODEM 通信的文章,但关于单片机 MODEM 通信的文章却不多见。现在将我个人单片机 MODEM 通信的实践经历写出来供大家参考。

要写单片机的 MODEM 通信必须要有两个背景知识:一个是 AT 命令集;另一个是通用异步接收发送器(UART)。

一、AT 命令集

下面介绍我的通信程序例子中涉及到的 AT 命令。

Dn:拨号命令。该命令使 MODEM 立即进入摘机状态,并拨出跟在后面的号码。D 命令是基本的拨号命令,它受到其他命令的修饰可构成 MODEM 何时拨号以及如何拨号等操作。

T:音频拨号。例如,ATDT2245879,其中 2245879 为电话号码。

P:脉冲拨号。例如,ATDP2245879,其中 2245879 为电话号码。

,:标准暂停。我们常常碰到拨打外线电话时需要暂停一下,等听到二次拨号音(外线)之后才能再拨后续的号码。缺省时暂停时间为 2 s,它由 S8 寄存器指定。

Sn:表示 MODEM 内部的寄存器。

S0:自动应答。如果要求 MODEM 具有自动应答特性,则应该预先将 MODEM 的 S0 寄存器设置为非 0。

S8:逗号拨号修饰符的暂停时间。该寄存器决定了当 MODEM 在拨号中遇到逗号(,)时应该暂停的时间。

二、通用异步接收发送器 UART

深入理解 UART 内部结构以及内部寄存器各位的含义,详细了解数据发送和接收的过程,有助于编写出高效、稳定的程序。现以 GM16C550 为例介绍编写基本通信程序需要知道的寄存器。实际的 ADDRESS 由具体接线决定。表 4.8-1 为 GM16C550 寄存器的介绍。

(1) 波特率除数锁存器(LSB、MSB)

在通信之前要进行一些参数初始化,波特率是首先应该考虑的一项。该寄存器是一个 16 位的寄存器,分为低 8 位(LSB)和高 8 位(MSB)寄存器。

当 LCR.7 = 1,且 A2A1A0 = 000/001 时,单片机访问的是波特率除数锁存器 LSB/MSB。GM16C550 推荐的工作频率是 1.843 2 MHz。这个频率除以 16 就是波特率的时钟频率,用于控制发送和接收数据的速度。下面给出波特率除数锁存器值的计算公式:

波特率除数锁存器值 = 工作频率/(16×期望波特率) = 1 843 200/(16×期望波特率)

表 4.8-2 给出了常用波特率与波特率除数锁存器值。

表 4.8-1 GM16C550 寄存器

A2	A1	A0	ADDRESS	W/R	寄存器
0	0	0	FFF8	W	接收缓冲寄存器(RHR)
				R	发送保持寄存器(THR)
0	0	1	FFF9	W	中断允许寄存器(IER)
0	1	0	FFFA	W	FIFO 控制寄存器(FCR)
0	1	0	FFFA	R	中断状态寄存器(ISR)
0	1	1	FFFB	W	线路控制寄存器(LCR)
1	0	0	FFFC	W	MODEM 控制寄存器(MCR)
1	0	1	FFFD	R	线路状态寄存器(LSR)
1	1	0	FFFE	R	MODEM 状态寄存器(MSR)
1	1	1	FFFF	W/R	临时数据寄存器(SPR)

表 4.8-2 波特率除数锁存器

波特率 / baud	锁存器 (HEX)	MSB	LSB	波特率 / baud	锁存器 (HEX)	MSB	LSB
300	180	01	80	600	C0	00	0C
1 200	60	00	60	2 400	30	00	30
4 800	18	00	18	9 600	0C	00	0C
19.2K	06	00	06	38.4K	03	00	03
57.6K	02	00	02	11.5K	00	00	01

```
MOV     PTR, #LCR           除数锁定允许
MOV     A, #80H
MOVX    @DPTR, A
MOV     DPTR, #LSB          ;波特率为 9 600baud
MOV     A, #0CH
MOVX    @DPTR, A
INC     DPTR
CLR     A
MOVX    @DPTR, A
```

图 4.8-1 为 GM16C550 与 RS232 接线图。

(2) 接收缓冲寄存器和发送保持寄存器(transmit and receive holding register)

当 LCR.7 = 0,且 A2A1A0 = 000 时,读操作单片机访问接收缓冲寄存器(RHR),写操作单片机访问发送保持寄存器(THR)。

(3) 中断允许寄存器(interrupt enable register)

当 LCR.7 = 0,且 A2A1A0 = 001 时,单片机访问中断允许寄存器(IER)。

IER.0 = 1,允许接收器数据就绪中断。

IER.1 = 1,允许发送保持寄存器为空时中断。即当从发送保持寄存器把一个字节移到移

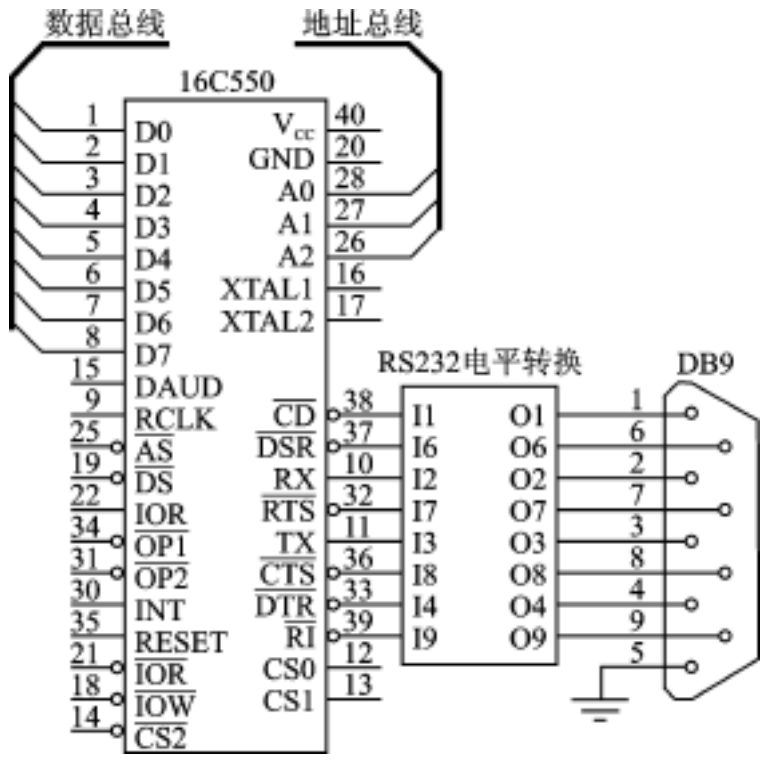


图 4.8-1　GM16C550 与 RS232 接线图

位寄存器时,产生一个中断,使发送保持寄存器能够接收下一个字节。

IER.2 = 1,表示允许接收有错信息或间断条件中断。

IER.3 = 1,MODEM 状态变化中断。

IER.4～7,没有使用,设置为零。

```
MOV   DPTR,#IER
MOV   A,#01H
MOVX  @DPTR,A
```

(4) FIFO 控制寄存器(FIFO control register)

数据发送和接收模式的选择。GM16C550 提供了两种模式:FIFO 和 DMA。其中 DMA 又有两种模式 DMA 的模式 0、DMA 的模式 1 可供选择。我的举例采用默认的 DMA 的模式 0。感兴趣的朋友可试一试其他模式,这里不再说明。

(5) 中断状态寄存器(interrupt status register)

前面介绍了几种中断,它们在各自的条件下产生中断,UART 都会输出一个高电平的中断请求信号,触发同一个中断请求。为了具体判断是哪一种中断,还应该检测 ISR,如表 4.8-3 所列。

表 4.8-3　中断状态寄存器

中断优先级	(ISR)				中　断　源
	D3	D2	D1	D0	
1	0	1	1	0	接收奇偶校验错、超越错、帧格式错、间断条件
2	0	1	0	0	接收寄存器就绪
2	1	1	0	0	接收数据超时
3	0	0	1	0	发送保持器准备好
4	0	0	0	0	MODEM 状态变化

ISR.0 = 1,表示没有中断产生。

ISR.4～5 没有使用。

ISR.6～7,当采用 FIFO 的接收和发送模式时,这两位都设置为 1;反之,都设置为 0。

(6) 线路控制寄存器(line control register)

LCR.0～1,表示发送和接收时的字节长度,如表 4.8-4 所列。

表 4.8-4 线路控制寄存器 LCR.0～1

LCR.0	LCR.1	发送和接收时的字节长度/bit
0	0	5
0	1	6
1	0	7
1	1	8

LCR.2,这一位与 LCR.0～1 共同定义了停止位的长度,如表 4.8-5 所列。

表 4.8-5 线路控制寄存器 LCR.2

LCR.2	发送和接收时的字节长度/bit	停止位长度/bit
0	5 6 7 8	1
1	5	1.5
1	6 7 8	2

LCR.3 = 1,进行奇偶校验。

LCR.4 = 0,进行奇校验;LCR.4 = 1,进行偶校验。

LCR.5 = 1,奇偶校验位恒为 1 或 0。

表 4.8-6 为线路控制寄存器 LCR.3～5。

表 4.8-6 线路控制寄存器 LCR.3～5

D5D4D3	奇偶校验	D5D4D3	奇偶校验
000	无奇偶	111	奇偶位恒为 1
101	奇偶位恒为 1	011	偶校验
001	奇校验		

LCR.6 = 1,表示允许间断,即允许发送器寄存器保持一个完整帧时间以上的空号状态。

LCR.7,用于区分访问除数锁存寄存器还是访问接收缓冲/发送保持和中断允许寄存器。

```
MOV   DPTR,#LCR        通信传输长度为 8 位,停止位为 1,偶校验
MOV   A,#1BH
MOVX @DPTR,A
```

(7) MODEM 控制寄存器(MODEM control register)

这是一个 MODEM 和外设接口的寄存器。

MCR.0 = 1 时,强制芯片引脚 DTR = 0;

MCR.0 = 0 时,强制芯片引脚 DTR = 1。

MCR.1 = 1 时,强制芯片引脚 RTS = 0;

MCR .1 = 0 时,强制芯片引脚 RTS = 1。

MCR .2 = 1 时,强制芯片引脚 OP1 = 0;

MCR .2 = 0 时,强制芯片引脚 OP1 = 1。

MCR .3 = 1 时,强制芯片引脚 OP2 = 0;

MCR .3 = 0 时,强制芯片引脚 OP2 = 1。

MCR .4 = 1 时,循环返回模式,可用于芯片自测。

其他位保留。

```
MOV   DPTR, # MCR        异步串口芯片的 DTR、RTS 引脚送出逻辑低电平
MOV   A, # 03H
MOVX @DPTR,A
```

(8) 线路状态寄存器(line status register)

LSR .0: 当接收移位寄存器接收到的字节完全移到接收缓冲寄存器时,该位置 1。如果读该寄存器,那么这一位被清零。

LSR .1: 出现超越错时,这一位被置 1。读该寄存器,这一位被清零。

LSR .2: 出现奇偶校验错时,这一位被置 1。读该寄存器,这一位被清零。

LSR .3: 出现停止位不完整、丢失、空号时,这一位被置 1。读该寄存器,这一位被清零。

LSR .4: 当接收寄存器检测到空号状态已持续一个完整帧传输时间时,这一位被置 1。读该寄存器,这一位被清零。

LSR .5: 当发送的字节从发送保持寄存器移到发送移位寄存器时,该位置 1。

LSR .6: 当发送保持寄存器和发送移位寄存器都没用字节时,该位置 1。

LSR .7: 当奇偶校验错、帧格式错、空号错有一个出现时,该位置 1。

```
Setdata:
            MOV     PTR, # LSR
            MOVX   A,@DPTR
            JNB     ACC .5, Setdata
            MOV     A, DataNumber          ;DataNumber 记录发送字符的个数
            MOV     DPTR, # AtCommand      ;AtCommand 定义 AT 命令参数的起始地址
            MOVC   A,@A + DPTR
            MOV     DPTR, # THR            ;发送保持寄存器
            MOVX   @DPTR,A
            INC     DataNumber
            MOV     A, DataNumber
            CJNE    A, # 0BH, Setdata      ;发送 11 个字符
AtCommand:  DB      " A "," T "," S "," 0 "," = "," 2 "," S "," 8 "," = "," 5 ", 0DH
```

(9) MODEM 状态寄存器(MODEM status register)

MSR .0～3 = 1 时,表示自上一次单片机读 MSR 寄存器之后,分别反映 MODEM 控制逻辑的四个输入信号的状态发生了变化。

MSR .0 = 1 时,清除发送信号(CTS)已经发生了变化。

MSR .1 = 1 时,数据设备就绪信号(DSR)已经发生了变化。

MSR.2 = 1 时,振铃信号(RI)已经发生了变化。

MSR.3 = 1 时,载波信号(DCD)已经发生了变化。

MSR.4～7 四位分别反映 MODEM 控制逻辑的四个输入信号的当前状态。

MSR.4 = 1 时,清除发送信号(CTS)有效。

MSR.5 = 1 时,数据设备就绪信号(DSR)有效。

MSR.6 = 1 时,振铃信号(RI)有效。

MSR.7 = 1 时,载波信号(DCD)有效。

(10) 临时数据寄存器(scratchpag register)

临时数据寄存器可以存储用户信息。

有了上面知识的准备后就可以轻松地写出单片机的 MODEM 通信程序。现在可把零散的东西组织起来了。

GM16C550 芯片初始化模块

```
Init_16C550:
            MOV    DPTR,#LCR          ;除数锁定允许
            MOV    A,#80H
            MOVX   @DPTR, A
            MOV    DPTR,#LSB          ;波特率为 9 600baud
            MOV    A,#0CH
            MOVX   @DPTR, A
            INC    DPTR
            CLR    A
            MOVX   @DPTR, A
            MOV    DPTR,#LCR          ;通信传输长度为 8 位,停止位为 1,偶校验
            MOV    A,#1BH
            MOVX   @DPTR, A
            MOV    DPTR,#MCR          异步串口芯片的 DTR。RTS 引脚送出逻辑低电平
            MOV    A,#03H
            MOVX   @DPTR,A
            MOV    DPTR,#IER
            MOV    A,#01H
            MOVX   @DPTR,A
```

采用中断的方式接收数据

```
Interrupt2:
            PUSH   ACC
            PUSH   DPH
            PUSH   DPL
            PUSH   PSW
            MOV    DPTR,#RHR          ;接收数据
            MOVX   A,@DPTR
            ……
```

```
POP     PSW
POP     DPL
POP     DPH
POP     ACC
RETI
```

至此,完成了一个小型的单片机 MODEM 通信系统。其中的检错寄存器、MODEM 状态寄存器等应用限于篇幅没有完全涉及。有兴趣的朋友可以试一试,那么你就会对 MODEM 通信有一个比较深刻的认识。

选自《单片机与嵌入式系统应用》月刊,2001 年第 8 期

4.9　无线串行接口电路设计

衡阳南华大学电气工程学院(421001)　黄智伟　朱卫华　陈　和

无线通信技术的发展与日益成熟,为各种潜在的工程技术应用提供了新的方法和手段。笔者所设计的无线串行接口电路由 TRF6900 单片射频收发器、MSP430F1121 微控制器和 75LV4737A 接口芯片组成,工作在 ISM 频段,有效距离 30～100 m,波特率 19 200 bps,可用于 PC 机与 PC 机之间、PC 机与单片机之间、单片机与单片机之间进行无线串行数据双向传输,在仪器仪表、无线抄表系统、计算机遥测遥控系统以及家庭网络系统等有广泛的应用。

一、电路组成及工作原理

所设计的电路如图 4.9-1、图 4.9-2、图 4.9-3 所示。射频收发电路选用 Texas Instru-

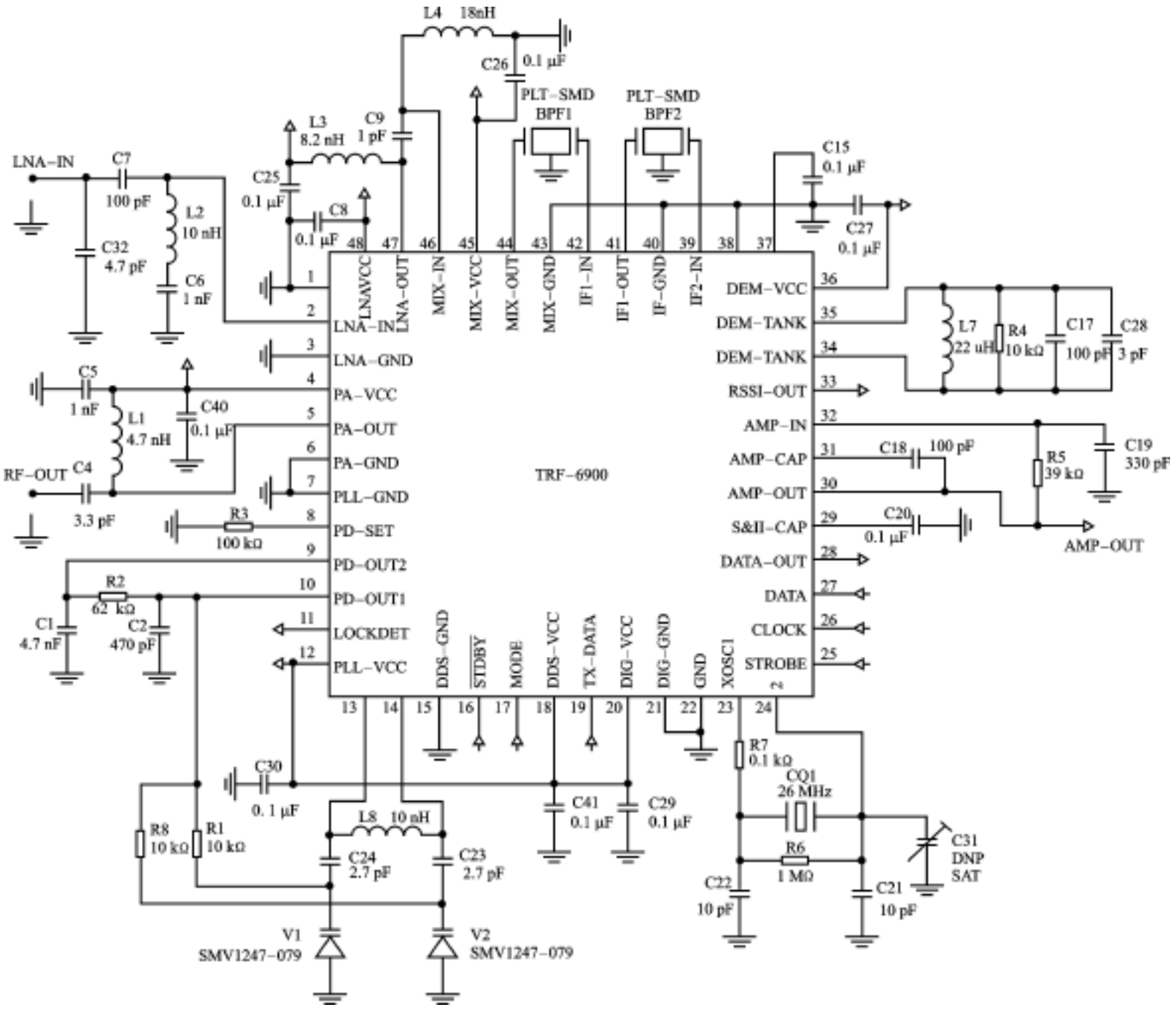


图 4.9-1　射频收发电路

ments 公司最新推出的单片射频收发器芯片 TRF6900。该芯片采用 48 脚 PQFP 封装,高集成度,工作频率为 850～950 MHz,具有 FM/ FSK 调制模式,供电电压 2 .2～3 .6 V,射频输出功率高达 + 5 dBm,待机模式电流消耗仅 0 .5～5 μA;采用高吞吐率 16 位 RISC 结构,最快可达 8 MIPS(每秒指令执行次数),采用三线制串行接口,方便与微控制器连接,可用于 ISM 频段进行数据的双向传输。

TRF6900 内部电路分为发射电路和接收电路两部分。发射电路包含有:RF 功率放大器、锁相环、压控振荡器、可编程的直接数字合成器、代功耗控制逻辑电路和串行接口电路。接收电路包含有:低噪声放大器、射频缓冲放大器、射频混频器、本机振荡缓冲放大器、第一级中频放大器、第二级中频放大器、限幅器、FM/ FSK 解调器、低通滤波放大器/ 后检波放大器、数据限制器、接收信号强度指示器等电路。

发射电路中,TRF6900 使用 3 线单向串行总线(CLOCK、DATA、STROBE)进行编程,来自微控制器的 24 位控制字通过串行口输入到直接数字合成器(DDS),在每一个 CLOCK 信号的上升沿,在 DATA 端上的逻辑值被写入 DDS 的 24 位移位寄存器中。设置 STROBF 为高电平,编程信息被装入所选择的锁存器,完成 DDS 模式、调制器、PLL 等设置。要发射的数据通过 TX DATA 端进入 DDS,直接数字合成器将数字信号通过 11 位数/ 模转换器、正弦波形成器等电路转换成模拟信号。基准振荡器输入频率 f_{ref} 为 15～26 MHz,可编程的 DDS 分配器比率 0～4 194 303 bits,分辨率 $\Delta f = N \times f_{ref} \div 2^{24}$,FSK 调制器寄存器比率 0～1 020 bits,分辨率 $\Delta f = N \times f_{ref} \div 2^{22}$。时钟电路采用外接晶体振荡器(25 .6～26 MHz)产生电路所需的基准频率。本机振荡用锁相环(PLL)方式,由在 DDS 基础上的频率合成器、外接的无源回路滤波器和压控振荡器组成。压控振荡器由片内的振荡电路和外接的变容二极管及 LC 谐振回路组民,频率范围为 850～950 MHz。RF 功率放大器具有高达 + 5 dBm 的输出功率。

接收电路中,低噪声放大器具有 13 dB 增益和 3 .3 dB 的噪声指数;具有两种工作模式:标准和低增益。在低 RF 输入电平时为了得到最大的灵敏度选择标准模式;在高 RF 输入电平时选择低增益模式。混频器采用常规的双重平衡式 Gilbert-Cell 混频器结构,用片上的压控振荡器操作,混频器的输出阻抗(MIX－OUT 端)是 330 Ω,这阻抗允许一个 330 Ω 的陶瓷滤波器直接连接到端点 MIX－OUT。第一级中频放大器具有大约 7 dB 的放大增益和 330 Ω 的输入输出阻抗,用来放大从混频器来的输出波形。第二级中频放大器和限幅器具有大约 80 dB 放大增益和 330 Ω 的输入阻抗,中频频率范围 10～21 .4 MHz,限幅器的输出连接到 FM/ FSK 解调器。对于频率 10～21 .4 MHz,接收信号强度指示器的斜率是 19 mV/ dB。FM/ FSK 解调电路完成 FM 或 FSK 信号的解调。解调器输出带宽 0 .3 MHz(IF = 10 .7 MHz),探测范围 300 kHz。

微控制器选用德州仪器公司的 MSP430F1121 超低功耗微控制器。MSP430F1121 具有 16 位 RISC 结构,16 位 CPU 寄存器和常数寄存器,4 KB ROM,256 B FLASH, 256 B RAM,指令周期 125 ns,超低电压工作(1 .8～3 .6 V),超低功率消耗(1 .3～160 μA),具有 5 种省电模式,可串行在线编程,程序代码由加密熔丝保护。

MSP430F1121 的 I/ O 口线 P2 .0、P2 .2、P1 .0、P1 .1、P1 .2 分别与接口芯片 75LV4737A 的 DIN1(RS232 的 CTS)、ROUT3(RS232 的 RTS)、STDBY、DIN2(RS232 的 RXD)、ROUT5(RS232 的 TXD)相连,完成接口芯片的特种控制和串行接口所需的数据发送、接收、请求、清除功能。

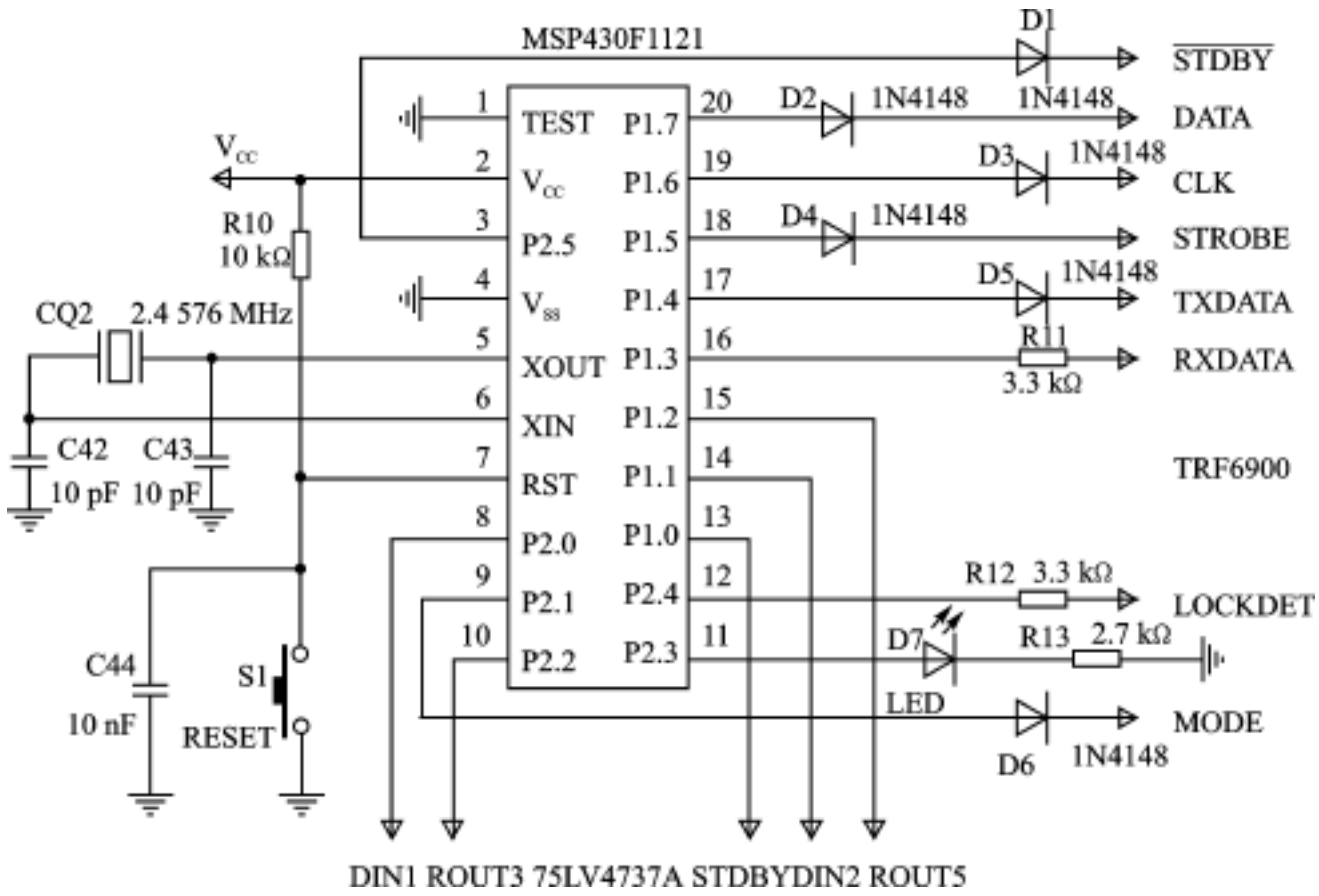


图4 9-2 微控制器电路

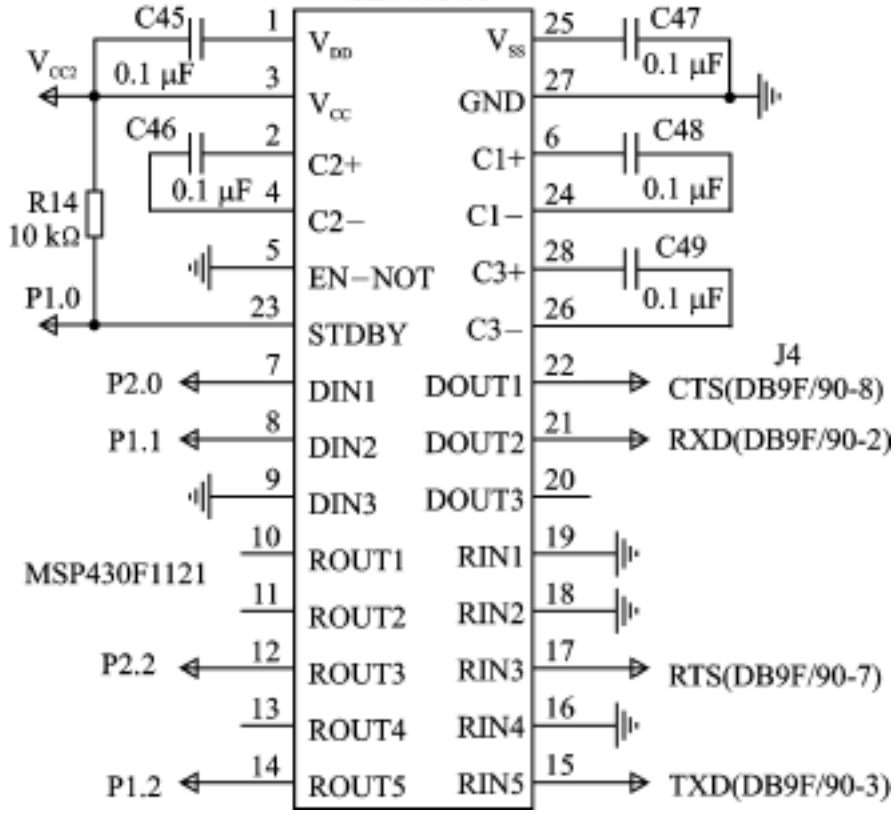


图4 9-3 串行接口电路

MSP430F1121的I/O口线P2.5、P2.4、P2.1、P1.7、P1.6、P1.5、P1.4、P1.3分别与TRF6900的控制端:STDBY(待机控制)、LOCKDET(同步检测)、MODE(模式选择)、DATA(串行控制数据)、CLK(串行控制时钟)、STROBE(串行控制读取脉冲)、TXDATA(发射数据)、RXDATA(接收数据)相连,完成TRF6900工作模式设置;接收和处理TRF6900接收电路所接收的数据,并传送到串行接口;控制TRF6900发射电路,将来自串行接口的数据发送出去。

二、电路设计时应注意的一些问题

1 . 印制电路板设计

印制电路板分成射频电路和控制电路两部分排列。射频电路以 TRF6900 为中心各元器件紧靠其周围,尽可能减少分布参数的影响。印制电路板(PCB)采用 4 层 PCB,顶层包含调整电路连接、RF 信号引入和芯片接地。第 2 层是单纯的电源接地板。第 3 层包含模拟功率电源以及馈送到所有板上的附加电路的电源连接。最下层包含为屏蔽目的而设计的接地。

2 . TRF6900 工作模式设置

方法 1:将 TRF6900 的控制端 STDBY(待机控制)、LOCKDET(同步检测)、MODE(模式选择)、DATA(串行控制数据)、CLK(串行控制时钟)、STROBE(串行控制读取脉冲)、TXDATA(发射数据)、RXDATA(接收数据)引出,通过所设计的并行通道接口电路(采用 2 片 SN54LV244)与 PC 机的并口相连,利用基于 Windows 的 TRF6900 编程工具软件(TRF6900 Programming Tool Software)在 PC 机显示器屏幕的提示下完成 TRF6900 的设置。

方法 2:根据 TRF6900 的工作模式要求,将串行的模式控制字 A 字、B 字、C 字、D 字分别写入 MSP430F1121 中,电路启动时,由 MSP430F1121 通过串行控制接口(DATA、CLOCK、STROBE)和控制端(STDBY、MODE)完成对 TRF6900 的设置。

试验调试时可选用方法 1。

3 . MSP430F1121 软件开发

MSP430F1121 软件开发可选用“ Kickstart ”开发工具,“ Kickstart ”是一个基于 Windows 的开发环境,用户可完成设计编码、模拟操作、下载软件和应用调试等工作。通用的接口源程序可以从“ www .ti .com/ sc/ docs/ produts/ micro/ msp430/ msp430 .htm ”或者“ MSP430 Application Peport Book (SLAA024) ”等处获得。TRF6900 的工作模式控制字按设计要求设置。

4 . PC 机数据/ 文本文件的接收/ 发射

当所设计的电路与 PC 相连时,使用“ TRF6900/ MSP430 系统模式软件(TRF6900/ MSP430 SestemSoftware),使用者可将数据/ 文本文件输入/ 调入在演示窗中,经由 PC 机的 com1 和 com2 接口,传送到无线串行接口电路中,MSP430 接收到数据包后发射出去。反之,TRF6900 接收到 FSK 的 RF 数据信号,经放大解调后,送入 MSP430 处理,经由 PC 机的 com1 和 com2 接口显示在演示窗中。形成 RF 双向串行数据连接。

三、结束语

实验表明:使用 TRF6900/ MSP430 组成的 RS232 无线串行接口电路,电路结构简单,工作可靠,由于有较好的软、硬件技术支持,可方便地构成一个双向的无线串行数据传输通道,用于检测仪器仪表和自动控制系统中。

参 考 文 献

1 exas Instruments (SLAS213C) . TRF6900 Single-Chip RF Transceiver[Z] . Texas Instruments, 2000(5)

2 Craig Bohren, Matthew Loy, John Schillinger . Designing with the TRF6900 Single-Chip RF Transceiver (SWRA033A)[Z] . Texas Instruments, 2001(1)

3 Texas Instruments (SLAS241C) . MSP430X11X1 Mxed signal Microcontroller[Z] . Texas Intruments, 2000(6)

4 Texas Instruments (SWRA032) . TRF6900/ MSP430EVK[Z] . Texas Instruments, 2000(5)

选自《电测与仪表》月刊,2001 年第 7 期

4.10　通用无线数据传输电路设计

衡阳南华大学电气工程学院(421001)　黄智伟　朱卫华

一、电路组成及工作原理

所设计的电路如图 4.10－1 所示。nRF401 采用 20 脚 SOIC 封装,引脚功能如表 4.10－1 所列。

nRF401 芯片内包含发射功率放大器(PA)、低噪声接收放大器(LNA)、晶体振荡器(OSC)、锁相环(PLL)、压控振荡器(VCO)、混频器(MIXER)、解调器(DEM)等电路。在接收模式中,nRF401 被配置成传统的外差式接收机。所接收的射频调制的数字信号被低噪声放大器放大,经混频器变换成中频,放大、滤波后送入解调器,解调后变换成数字信号输出(DOUT 端)。在发射模式中,数字信号经 DIN 端输入,经锁相环和压控振荡器处理后送入发射功率放大器射频输出。由于采用了晶体振荡和 PLL 合成技术,频率稳定性极好;采用 FSK 调制和解调,抗干扰能力强。

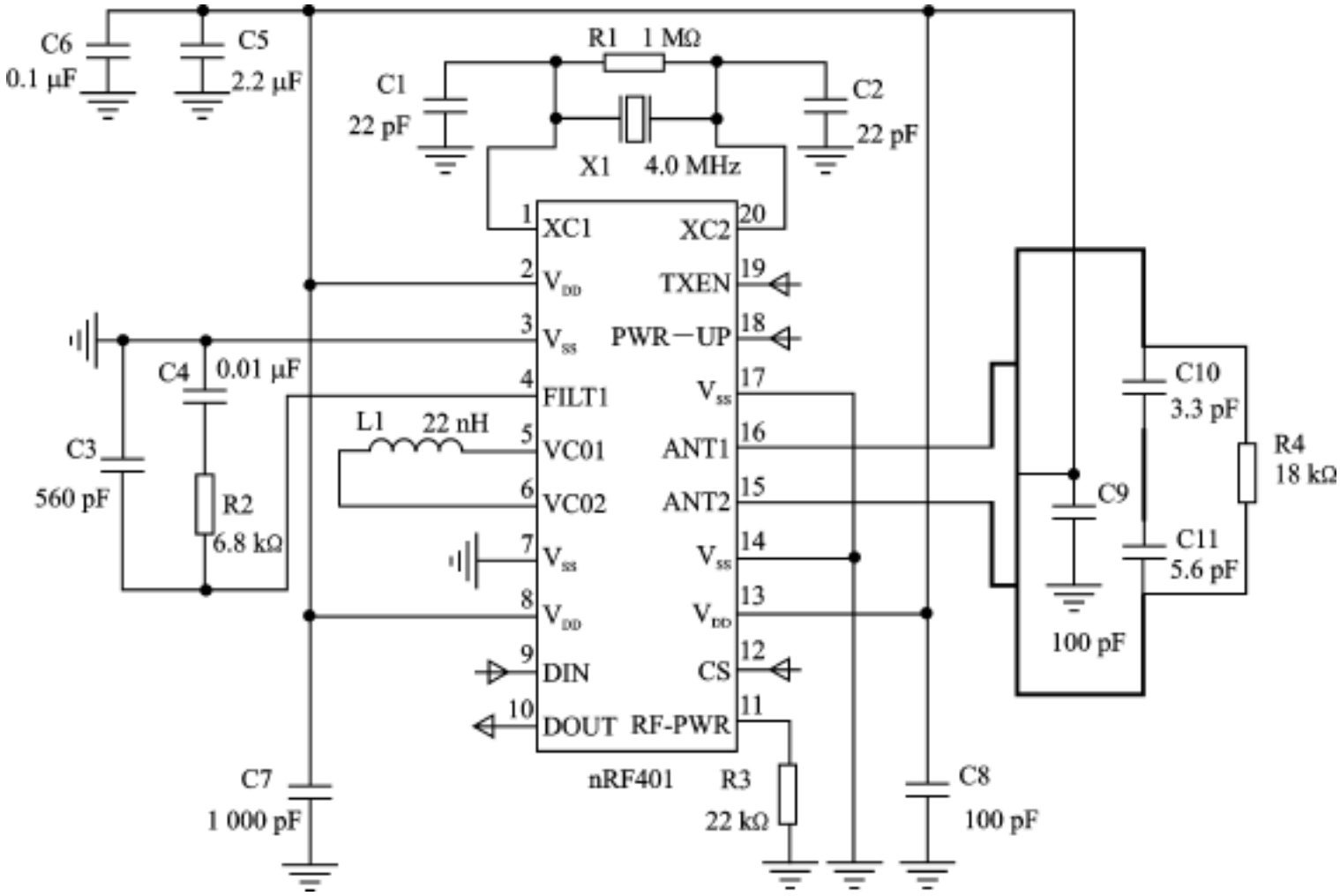


图 4.10－1　无线数据传输电路

表 4.10-1 引脚功能

引 脚	名 称	引脚功能描述	引 脚	名 称	引脚功能描述
1	XC1	晶振输入	11	RF_PWR	发射功率设置
2	V_{DD}	DC 电源(3～5V)	12	CS	通道选择
3	V_{SS}	地(0V)	13	V_{DD}	DC 电源(3～5V)
4	FILT	回路滤波器	14	V_{SS}	地
5	VC01	VC0 外接电感	15	ANT2	天线接头
6	VC02	VC0 外接电感	16	ANT1	天线接头
7	V_{SS}	地	17	V_{SS}	地
8	V_{DD}	DC 电源(3～5V)	18	PWR_UP	电源开关
9	DIN	数据输入	19	TXEN	发射允许
10	DOUT	数据输出	20	XC2	晶振输出

注:9 脚及 10 脚:DIN 输入数字信号和 DOUT 输出数字信号均为标准的逻辑电平信号,需要发射的数字信号通过 DIN 输入,解调出来的信号经过 DOUT 输出;12 脚通道选择:CS="0"为通道#1(433.9 MHz),CS="1"为通道#2(434.33 MHz);18 脚电源开关:PWR_UP="1"为工作模式,PWR_UP="0"为待机模式;19 脚发射允许:TXEN="1"为发射模式;TXEN="0"为接收模式。

二、典型应用电路

1.与编码器/解码器连接的电路

所设计的无线数据传输电路可以直接与常用的编码器/解码器如 MC145026/MC145027、VD5026/VD5027、PT2262/PT2272 等连接,实现编码数据的无线传输,连接电路如图 4.10-2 所示。

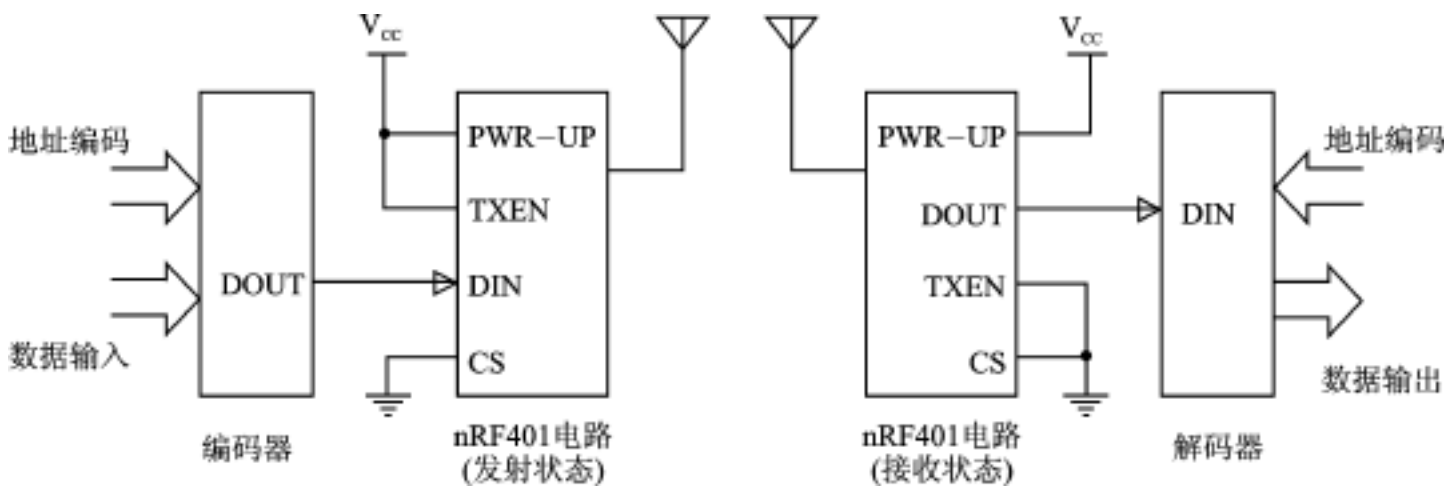


图 4.10-2 与编码器/解码器连接的电路

2.与计算机串行接口连接的电路

所设计的无线数据传输电路通过 MAX232A 接口芯片可以与计算机串行接口连接,实现计算机与计算机之间的数据无线传输,连接电路如图 4.10-3 所示。

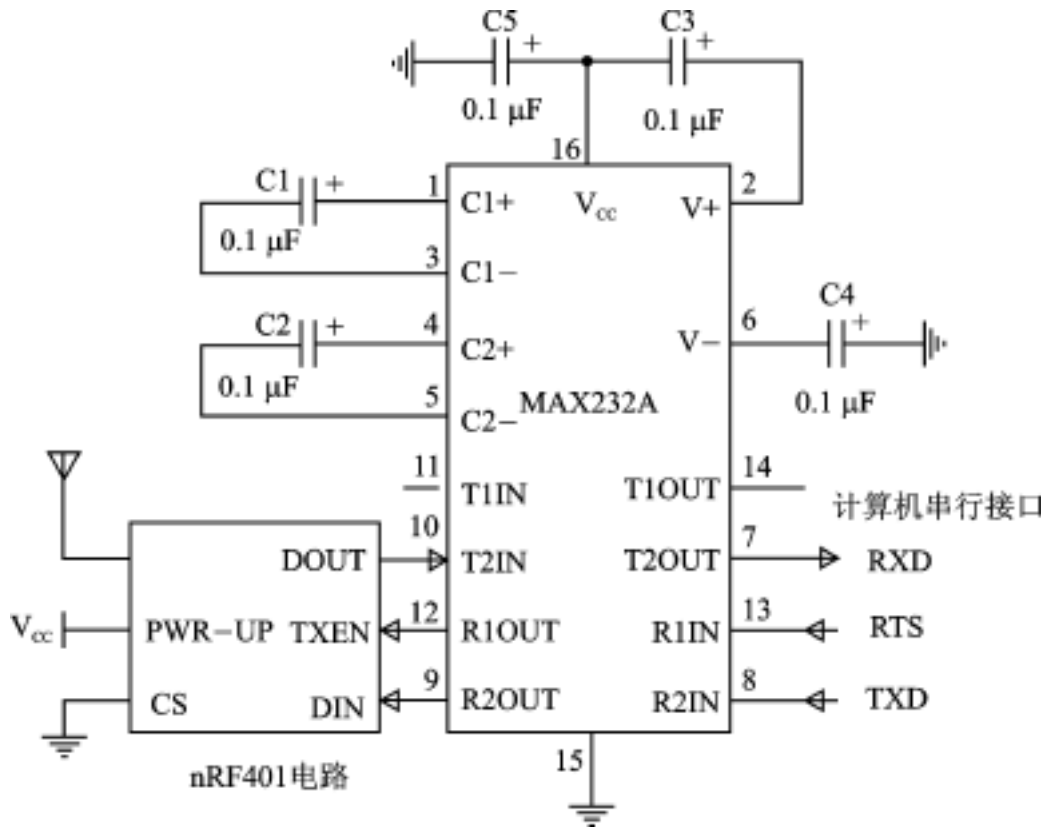


图 4 .10 - 3 与计算机串行接口连接的电路

三、结束语

实验表明:使用 nRF401 组成的无线数据传输电路,电路结构简单,工作可靠,可方便地嵌入仪器仪表和自动控制系统中,构成一个点对点或一点对多点的双向无线串行数据传输通道。

使用中应注意的问题是:在发射模式:通信速率最高为 20 Kbps;发送数据之前需将电路置于发射模式(TXEN = 1);接收模式转换为发射模式的时间至少 5 ms;可以发送任意长度的数据;发送结束后应将电路置于接收模式(TXEN = 0);发射模式转换为接收模式的时间至少 5 ms。在接收模式接收到的数据可以直接送到单片机串行接口或者经电平转换后送入计算机。PWR - UP 为 0 时,电路进入待机模式,工作电流 8 μA,在待机模式下电路不接收和发射数据。设计通信程序应考虑双方通信的协议、有效数据识别标志及数据的检错、纠错和校验。

参 考 文 献

1 ong Kong Regal Distributors inc . 433 MHz Single Chip RF Transceiver nRF401[Z] . 2000

2 李朝青 .PC 机及单片机数据通信技术[M] .北京:北京航空航天大学出版社,2001

选自《仪表技术》双月刊,2001 年第 5 期

4 .11　FX909 在无线高速 MODEM 中的应用

西安空军工程大学工程学院研究生大队(710038)
张益强　郜兴峰　王红卫

无线通信广泛应用于移动通信、空中交通管制和导航系统,数字化是通信发展的必然要求。从我国目前情况来看,广泛应用的大量 VHF/ UHF 电台多为模拟话音电台,通信手段仍以短波、超短波话音通信为主,数据传输能力很低,不能适应当前数字化数据传输的要求。而专门设计数字电台又有设计周期长、费用高等弊端。本文提出了一种设计方案,通过话音接口使 MODEM 与常规 FM 电台连接,同时还可与主控计算机或其他具有标准 RS－232C 接口的数据设备相连,从而实现数据的无线传输,有效地利用了现有设备,在一定程度上满足了日益增长的高速数据传输的要求。我们设计的 MODEM 是针对常规 FM 电台设计的,信道间隔为 25 kHz 的电台可实现 9 600 bps 或更高速率数据传输。

一、FX909 简介

1 . FX909 芯片特征

FX909 是英国康舒微电路有限公司(CML)推出的调制解调芯片,具有实现 GMSK 无线数据调制解调所需的基带信号处理功能,并具有相应的协议。它可方便地和主控器及电台调制/ 鉴频电路相接,实现无线数据链中的半双工数据传输。

在发送模式下,FX909 从主控器接收数据,计算并加上前向纠错码(FEC)及循环冗余校检码(CRC),并采用交织(突发错误保护)及扰码技术提高其性能。然后加上位同步及帧同步码,将数据打包并转变为模拟 GMSK 信号送往电台调制器接口。在接收模式下,FX909 完成相反的过程,从接收机鉴频器接收模拟 GMSK 信号,解码并纠错后送主控器。

其主要特征如下:

- GMSK 调制;
- 最大速率 19 .2 Kbps;
- 数据打包格式;
- 24 脚小型封装;
- 灵活的操作模式;
- 低电压(3 .3/ 5 V)工作;
- Mobitex 兼容;
- 主控器接口。

2 . FX909 原理框图

FX909 原理框图如图 4 .11－1 所示。

3 . FX909 封装形式及管脚说明

FX909 为 24 管脚封装,有贴片和双列直插两种封装形式,各管脚说明如表 4 .11－1 所列。

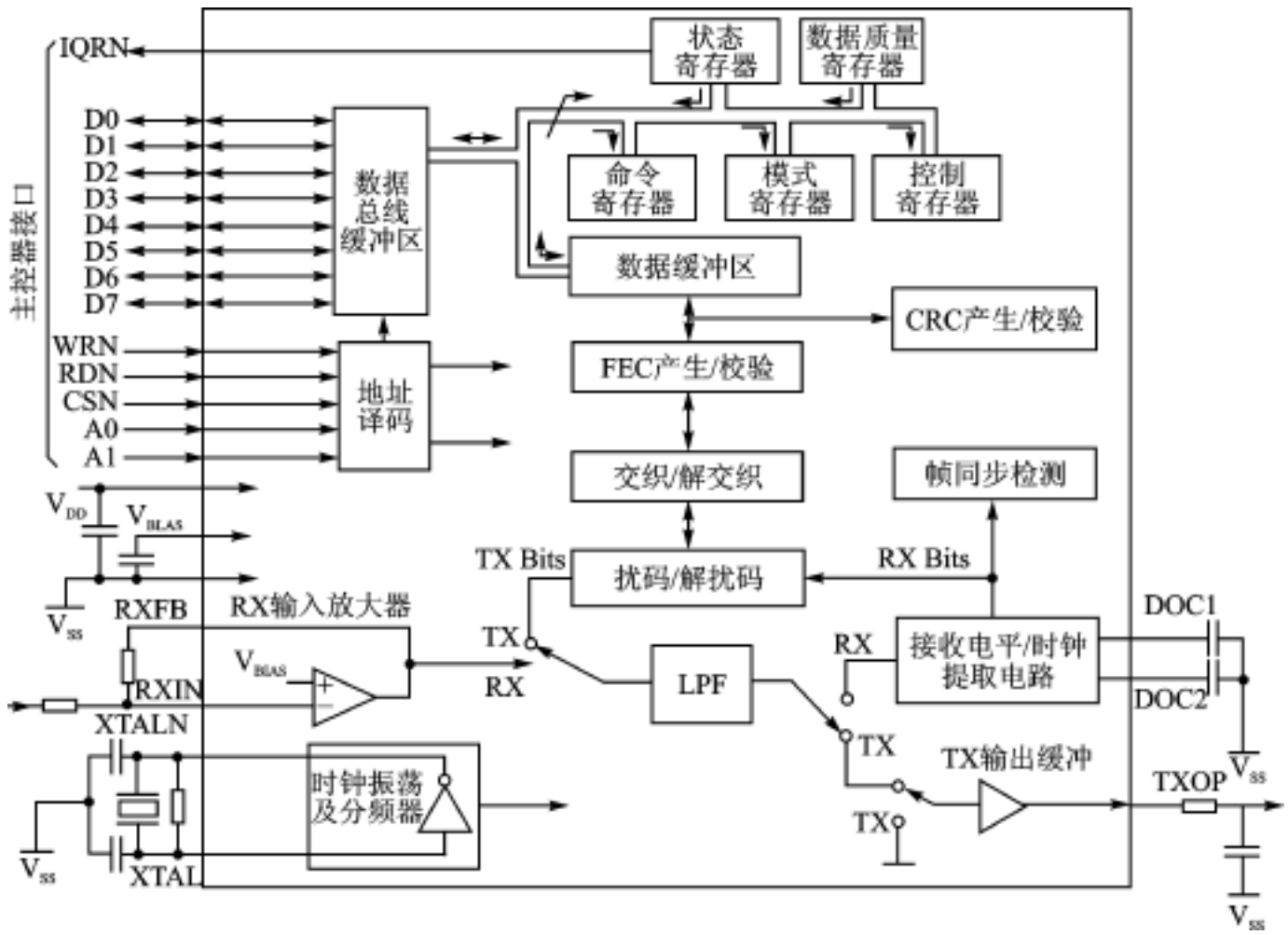


图 4 .11 - 1 FX909 原理框图

表 4 .11 - 1 FX909 管脚说明

管 脚	名 称	类 型	功 能
1	IRQN	O/ P	接到主控器的中断请求输入端,低电平有效,无效时呈高阻态
2	D7	BI	8 位 3 态双向数据总线
3	D6	BI	
4	D5	BI	
5	D4	BI	
6	D3	BI	
7	D2	BI	
8	D1	BI	
9	D0	BI	
10	RND	I/ P	低电平有效,可使主控器从 FX909 读取数据
11	WRN	I/ P	低电平有效,可使主控器向 FX909 写入数据
12	V_{SS}	Power	负电源端(地线)
13	CSN	I/ P	片选信号
14	A0	I/ P	地址线,选择 FX909 不同寄存器
15	A1	I/ P	
16	XTALN	O/ P	片内晶振的输出端
17	XTAL/ CLOCK	I/ P	片内晶振的输入端

续表 4.11－1

管　脚	名　称	类　型	功　能
18	DOC2	O/P	与接收信号电平测量电路相连
19	DOC1	O/P	
20	TXOP	O/P	发送数据输出端
21	V_{BIAS}	O/P	内部电路产生的偏置信号,保持 $V_{DD}/2$,和地之间应接去耦电容
22	RXIN	I/P	接收信号放大器的输入端
23	RXFB	O/P	接收信号放大器的输出端和 RX 滤波器输入端
24	V_{DD}	Power	正电源端。FX909 的供电端,接去耦电容

4.寄存器及其选择

FX909 有 3 个只读寄存器和 4 个只写寄存器。通过 A0、A1 和读写信号进行选择。如表 4.11－2 所列。

表 4.11－2　寄存器的选择

A1	A0	写(WRN 为 0)	读(RDN 为 0)
0	0	数据缓冲区	数据缓冲区
0	1	命令寄存器	状态寄存器
1	0	控制寄存器	数据质量寄存器
1	1	模式寄存器	保留未用

其中数据寄存器为主控器和 FX909 之间数据及命令传送的缓冲区。命令寄存器的不同设置使得 FX909 执行相应的任务,控制寄存器及模式寄存器决定了 FX909 的操作模式和不同设置,状态寄存器反映了 FX909 的当前状态,数据质量寄存器的值反映了接收数据信号的质量。

这里需要特别指出的是,FX909 的 18 字节数据缓冲区比较特殊,在写入或读取多个数据时,只需对同一地址进行顺序的读写,FX909 的内部电路可以保证数据的正确性。

二、MODEM 的硬件框图

该 MODEM 主要包括串行口 RS－232C 接口、主控器 89C51 与 FX909 接口、电台接口三部分,其硬件框图如图 4.11－2 所示。

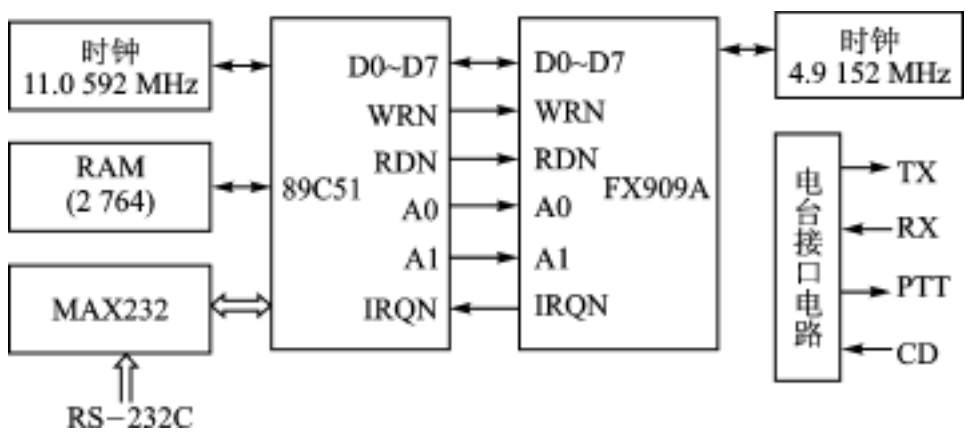


图 4.11－2　MODEM 硬件框图

RS－232C 接口包括标准 232C 接口信号线 RXD、TXD、CTS 和 RTS。电台接口包括发送信号线(TX)、接收信号线(RX)、电台收/发控制信号(PTT)和电台载波检测信号(CD)。主控器 89C51 与 FX909 接口有数据总线(D0～D7)、地址线(A0, A1)、读写信号线(WRN、RDN)、中断请求信号线(INT0)、片选信号(CSN)。

1．FX909 的外部电路

FX909 的外部电路如图 4.11－3 所示。

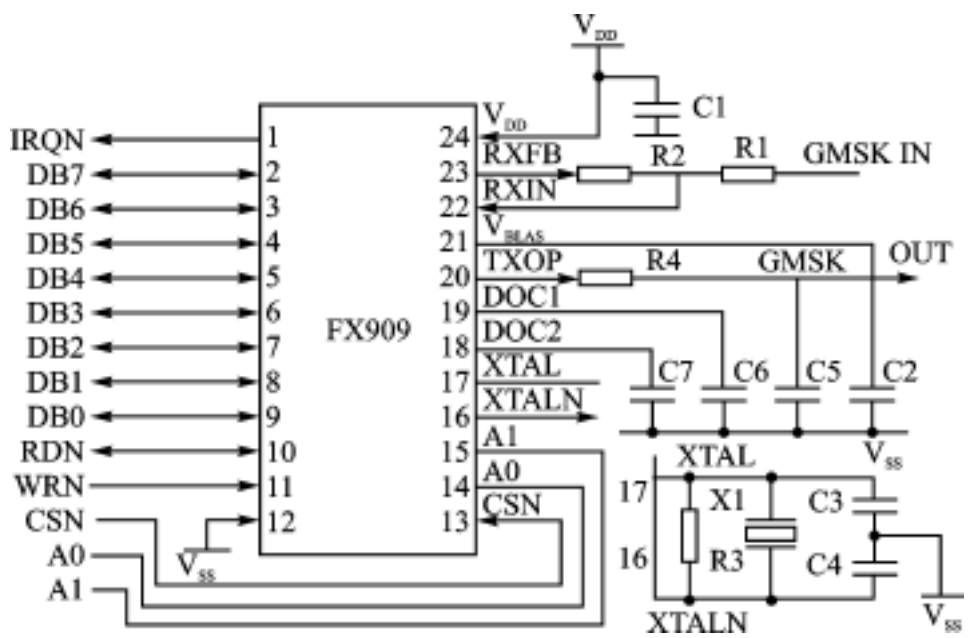


图 4.11－3　FX909 的外部电路

2．主控器接口及电台接口

MODEM 的主控器接口及电台接口电路如图 4.11－4 所示。

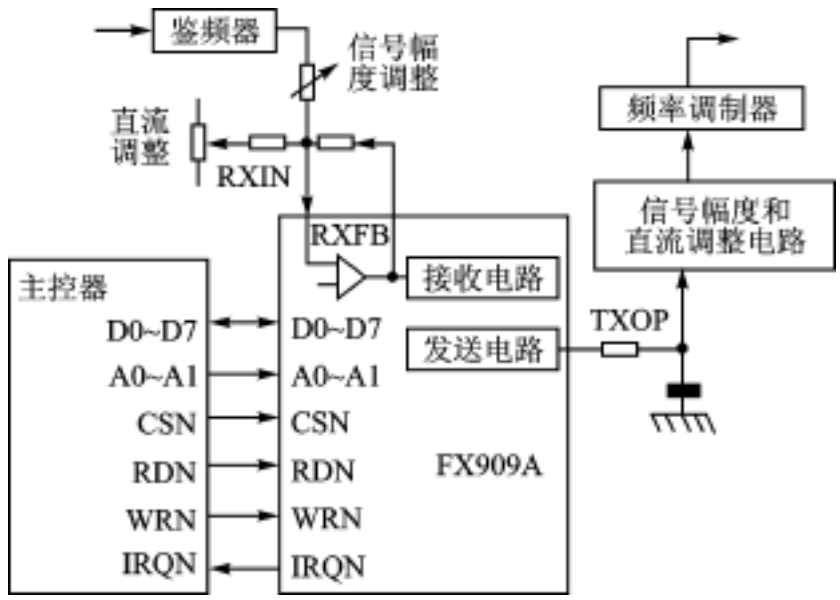


图 4.11－4　FX909 接口电路

数字信号由主控器进入 FX909 编码并调制后形成模拟信号，经发送接口送往电台。电台接收的信号，经接收电路，转换成与 FX909 匹配的模拟信号，经过解调解码成原数字信号，送入主控器。

三、MODEM 的程序组成

程序主要由主循环程序和串行中断、外部中断 0(INT0)、定时中断(T0)等子程序组成。

串行中断完成与RS－232C接口间的数据传送。外部中断(INT0)完成对FX909A的操作,包括设置命令、读取状态、数据传送与接收等功能。定时中断(T0,定时周期设置为1 ms)检测是否有数据要发送(条件是串行口收到新的数据),若有则置电台到“发送”状态,主循环检测到“发送”标志位后,即调用发送初始化程序。另外,主循环程序检测是否有数据要接收(条件是CD信号为1),若有则调用接收初始化程序。

主程序流程图如图4.11－5所示。

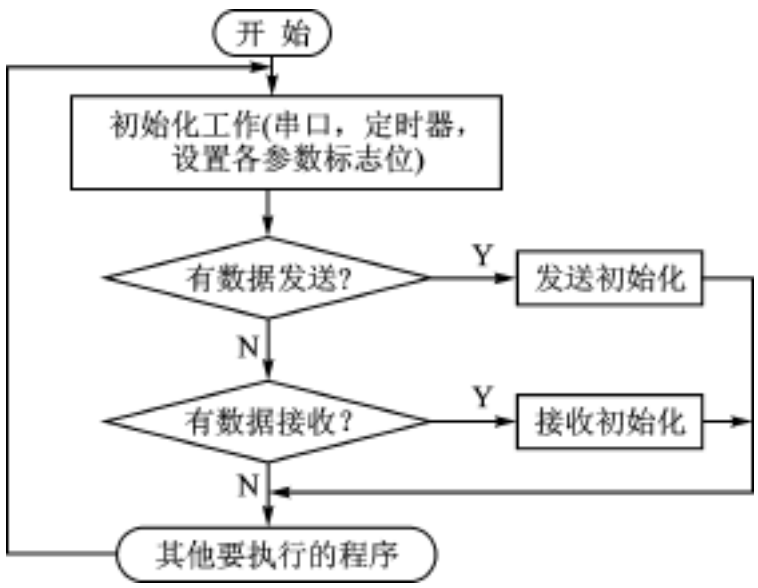


图4.11－5　主程序流程

按本方案设计的MODEM已成功地应用于某动态监控系统中,效果良好。采用该设计方案有设计简单、易实现、开发周期短等优点,可用于实时监控系统、远程数据采集系统、GPS车辆定位、调度、报警等系统,具有较高的实用价值和推广价值。

参 考 文 献

1　ML. CML Semiconductor Products Wireless Modem Data Pump FX909A. 1996

2　CML Semiconductor Products Application Information. GMSK Modem Application Notes. Nov. 1996

3　李广弟 单片机基础 北京:北京航空航天大学出版社,1998

4　郭梯云,杨家玮,李建东 数字移动通信 北京:人民邮电出版社,1995

选自《电子技术应用》月刊,2000年第6期

4 .12 蓝牙——短距离无线连接新技术

郑州解放军信息工程大学(450053) 曾繁景 王振兴 王 剑

蓝牙(Bluetooth)是语音和数据无线传输的开放性通信标准,其目的是建立低成本、短距离的无线连接。蓝牙技术是通过一个 9 mm×9 mm 的微芯片和蓝牙无线收发器嵌入各种便携式设备,诸如笔记本电脑、移动电话、台式机、个人数字助理(PDA)、数字相机、打印机、便携式游戏机、手表等,利用短距离无线连接技术替代许多专用的电缆连接。

一、蓝牙的由来

Bluetooth 是以一千多年前一位丹麦皇帝哈拉德·布鲁斯(Harald Buletooth)的名字命名的。据史书记载,这位哈拉德国王反对战争,主张以对话来解决纠纷,是一位擅长于与人交流的君主。他将当时四分五裂的瑞典、芬兰、丹麦统一。

1994 年,爱立信无线通信部门开始了一项针对移动电话及其周边配件的具有低功率、低成本的无线电界面的可行性研究开发。其最主要目标是提出能消除电话与个人电脑卡、无线手机间干扰的电缆,这项研究即蓝牙计划的前身。1998 年 2 月,以爱立信为首,联合东芝、IBM、Intel 和诺基亚成立了特别兴趣小组(Special Interest Group);5 月 SIG 共同发布了蓝牙无线电接入技术,即在全球范围内建立和运用通用、先进的无线接口技术标准。1999 年 7 月,蓝牙正式公布了蓝牙技术规范 Bluetooth Version 1 .0(事实上的标准),用以确保不同设备相互之间的通信。该技术规范文档由二部分组成:卷 1 为核心(Core)文档,详细说明了诸如无线通信、基带、链路管理、服务发现协议、传输层以及不同通信协议之间的互操作性;卷 2 为外围(Profiles)文档,详细说明了不同类型的蓝牙应用所必需的各种协议和程序。

到目前为止,已有 1500 多家公司加入了 SIG,其中包括电脑、通信、家电等产品的制造商,如宏基、佳能、Motorola、三洋、西门子、Philips 等也亦成为合作伙伴。

二、蓝牙的技术特点

蓝牙技术利用短距离、低成本的无线连接技术替代了电缆连接,从而为现存的数据网络和小型外围设备接口提供了统一的连接桥,形成了不同于固定网络基础设施的小型、专用的无线连接群。例如,蓝牙无线技术可以嵌于蜂窝电话、膝上电脑、打印机、PDA、传真机、键盘、游戏杆等设备之间,可取消设备之间不方便的连线;蓝牙技术的工作频段选在全世界范围内都可以自由使用的 2 .4 GHz 的 ISM(即工业、科学、医学)频带,这样用户不必经过申请便可以在 2 400～2 500 MHz 范围内选用适当的蓝牙无线电收发器频段;为了解决噪声干扰和抗信号衰减,蓝牙技术采用了跳频技术和短分组技术,即在接收或发送一分组数据后,则跳至另一频点,避免了其他信号源的干扰。快速跳频和短分组数据还可以保证链路传输的可靠性;蓝牙技术采用了前向纠错(FEC)技术和优化编码技术,减小了远距离传输时的伪随机噪声的影响,使其能适应各种无线传输环境;蓝牙无线发射机采用 FM 调制方式,从而能降低设备的复杂性;蓝

牙的数据传输率为 1 Mb/s,采用时分复用的全双工方式;蓝牙基带协议是电路交换和分组交换的组合。1 个跳频频率发送 1 个同步分组,每个分组占用 1 个时隙,也可扩展到 5 个时隙;蓝牙技术支持 1 个异步数据通道,3 个并发的同步话音通道,或 1 个同时传送异步数据和同步话音的通道。每 1 个话音通道支持 64 Kb/s 的同步话音,异步通道支持最大速率为 721 Kb/s,反向应答速率为 57.6 Kb/s 的非对称连接,或者是速率为 432.6 Kb/s 的对称连接。

三、蓝牙的网络拓扑结构

蓝牙系统支持点对点及点对多点通信。几个相互独立的、以特定方式连接在一起的锯齿网构成分散式网络,各锯齿网由不同频率的跳频序列来区分。在同一锯齿网中,所有的用户均同步于同一跳频序列。拓扑结构如图 4.12-1 所示。

锯齿网(Piconet):采用蓝牙技术的设备以特殊方式连接成的集合。锯齿网的建立由 2 台设备(如便携式电脑和蜂窝电话)的连接开始,最多可拓展至 8 台设备。所有的蓝牙设备都是对等的,并且遵循相同的工作方式。然而,当锯齿网建立和持续期间,只有 1 台设备为主设备,其他均为从设备。

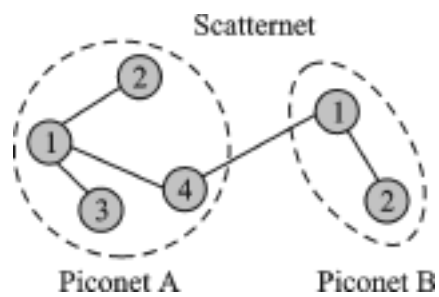


图 4.12-1 网络拓扑结构

分散式网络(Scatternet):由多个独立、非同步的锯齿网形成一个分散式网络。

四、蓝牙系统的功能模块

蓝牙系统基本功能模块如图 4.12-2 所示。它的功能模块包括无线单元、链路控制单元、链路管理、软件功能。

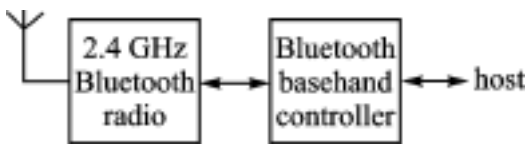


图 4.12-2 蓝牙系统基本功能模块

1. 无线技术规范

蓝牙无线接口是基于常规无线发射功率 0 dBm 设计的,符合美国联通信委员会(FCC)的 ISM 频段的规定。扩展频谱技术的应用使得功率可增至 100 dBm,可满足不同国家的需要。系统的最大跳频速率为 1 600 跳/s,在 2.402～2.480 GHz 之间,每采 1 MHz 带宽内有 79 个跳频点,从而实现了频谱扩展。在日本、法国、西班牙,由于当地规定的频段相对较窄,可用内部软件转换来实现。系统的设计通信距离为 10 cm～10 m,如果增加发射功率,可以达到 100 m。

2. 基带技术规范

基带描述了设备的数字信号处理部分,即蓝牙链路控制器,它完成基带协议和其他底层的链路规程。主要包括以下几方面内容:

(1) 网络连接的建立

(2) 链路类型和分组类型

链路类型决定了哪种分组模式能在特定的链路上使用,蓝牙基带技术支持 2 种链路类型:即同步面向连接类型 SCO(主要用于语音)和异步非连接类型 ACL(主要用于分组数据)。

(3) 纠　错

基带控制器采用 3 种纠错方式:1/3 速率前向纠错编码(FEC)、2/3 速率前向纠错编码

(FEC)、对数据的自动请求重传(ARQ)。

(4) 认证和加密

认证与加密服务由物理层提供。认证对任何一个蓝牙系统都是重要的组成部分,它允许用户自行添加可信任的蓝牙设备,认证采用口令—应答方式。蓝牙系统采用流密码加密技术,适于硬件实现,密钥长度可能是 0 位、40 位或 64 位,密钥由高层软件管理。

3 . 链路管理协议

链路管理器(LM)软件实现链路的建立、认证、链路配置及其他协议。链路管理器可发现其他链路管理器,并通过连接管理协议(LMP)建立通信联系,LM 利用链路控制器(LC)提供的服务实现上述功能。LC 的服务项目包括:接收和发送数据、设备号请求、链路地址查询、建立连接、认证、链路模式协商和设置(如传数据或是传数据/ 话音)、链路管理器确定分组的帧类型、设置设备处于 Sniff 模式、设置设备处于 Hold 模式、设置设备处于 Park 模式。

4 . 软件框架

为了适应各种用户环境,蓝牙设备应具有支持底层协议的互操作性。对某些设备,要求包括无线电兼容模块和空中接口、应用层协议和对象交换格式;对另外一些设备(如头戴式设备等)的要求则宽松得多。蓝牙计划的目标就是要确保任何带有蓝牙标记的设备都能进行互操作。软件的互操作性始于链路层协议的多路传输、设备和服务的发现,以及分段和重组。蓝牙设备必须能够彼此识别,并通过安装合适的软件识别出彼此支持的高层功能。互操作性要求采用相同的应用层协议栈。不同类型的蓝牙设备(如 PC 机、手持设备、头戴设备、蜂窝电话)对兼容性有不同要求,蓝牙的兼容性是指具有无线电兼容性,有话音收发能力及发现其他蓝牙设备的能力,更多的功能则要由手机、手持设备及笔记本电脑来完成。为实现这些功能,蓝牙软件构架将利用现有的规范,如 OBEX、vCard/ vCalendar、HID(Human Interface Device)及 TCP/ IP 协议等,而不是再去开发新的规范。设备的兼容性要求能够适应蓝牙规范和现有的协议。软件框架将实现以下功能:配置及诊断、蓝牙设备的发现、电缆仿真、与外围设备的通信、音频通信及呼叫控制、名片交换和电话号码的网络协议。

随着蓝牙技术的推广和应用,蓝牙的市场前景将十分乐观。蓝牙技术在各种无线设备中的广泛使用,将为人们提供一种更为方便、快捷、安全、可靠的通信联络方式。

选自《微型机与应用》月刊,2000 年第 11 期

4 .13　蓝牙技术——一种短距离的无线连接技术

北京工业大学信号与信息处理研究室(100022)　卓　力　沈兰荪

近十年来,微电子技术的不断进步极大地推动了计算机和通信设备的普及和迅猛发展,一些电子消费产品如 PC 机、掌上电脑、移动电话、无绳电话等进入人们日常的生活和工作中,成为人们生活中不可缺少的一部分。在这些设备之间传送文件时往往是通过线缆来进行,而且需要一些软件的支持才能进行连接。人们希望有一个能够取代线缆的短距离无线连接技术。为了解决这个问题,一项新的技术——蓝牙,便应运而生了。蓝牙技术提供了一种不需要线缆就可以在一定距离内进行无线连接的方法。

1998 年初,爱立信、诺基亚、IBM、INTEL、东芝等几家公司,组成了蓝牙技术特别兴趣小组 SIG(Special Interest Group)。1999 年 7 月,SIG 推出了蓝牙技术标准的第一个版本。目前已有 1 000 多家公司成为蓝牙技术的支持者,并已有一批支持蓝牙技术的设备投入市场。

“蓝牙”(Bluetooth)概念最早是由爱立信公司提出的,指的是一个单芯片、低成本、低功耗的无线通信模块。这个模块安装在移动电话、掌上电脑、数码相机和无绳耳机等设备的内部,使用这些设备的用户不再需要线缆就可以方便快速地进行通信。蓝牙技术通过一个短距离的无线连接来达到不同设备之间的互联,使得打印机、传真机、键盘以及其他的设备都可以成为蓝牙系统的一部分。

蓝牙工作在 2 .4～2 .5 GHz 频带,即工业-科学-医疗频带,ISM(Industrial-Scientific-Medical)频带。蓝牙一般的连接范围是 10 m 以内,通过扩展可以达到 100 m。采用了蓝牙技术的设备可以被自动发现,并在一定的距离内实现无线互联。

蓝牙技术定义了计算机和通信设备之间如何进行通信,也定义了如何将某一个应用映射成硬件以便和蓝牙技术相融合。采用了蓝牙技术的设备可以与其他的蓝牙设备进行无线通信而不管这个设备是由哪个厂商制造的。

一、蓝牙系统协议体系结构

蓝牙系统是一个开放性的系统,其协议层模型如图 4 .13 - 1 所示。

1 . 无线连接(Radio)

无线连接层定义了对工作在 2 .45 GHz ISM 频带的蓝牙接收机的要求。

2 . 基带(Baseband)

基带层和连接管理层负责在多个蓝牙设备间建立物理 RF(Radio Frequency)连接组成一个 Pico 网(piconet)。这一层控制蓝牙设备的同步和跳频序列,并管理蓝牙中定义的两个不同的连接类型 SCO 和 ACL。

3 . 连接管理协议(LMP, Link Manager Protocol)

LMP 负责蓝牙设备连接的建立和释放、控制,数据包的控制以及电源模式的管理、消耗、蓝牙设备的状态等。最后,LMP 还要负责认证、加密的密码问题。

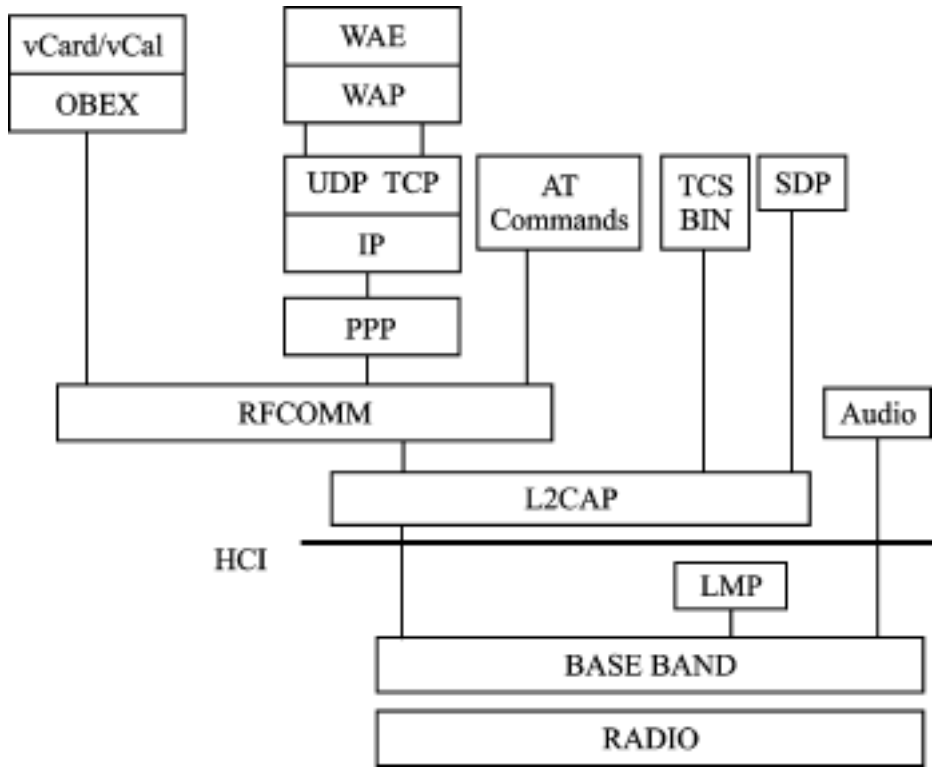


图 4.13-1 蓝牙系统协议体系结构

4.主控制器接口(HCI, Host Manager Interface)

HCI 提供了一个接入蓝牙硬件的统一接口方法,它包括一个到基带控制器和连接管理器以及进入硬件状态的命令接口。另外还包括控制和事件寄存器。

5.逻辑连接控制和自适应协议(L2CAP, Logical Link Control and Adaptation Protocol)

L2CAP 在基带层之上,是基带与高层协议间的接口,处理更高层协议的复用、大数据包的拆分和重组、传送以及服务质量(QoS)信息等问题。

6.串口仿真协议(RFCOMM)

RFCOMM 协议是一个串口仿真协议,它支持蓝牙设备之间点对点的通信。作为一个线缆替代协议,在基带上仿真 RS-232 的控制和数据信号,为一些将串行线缆作为传输机制的高层业务提供传输能力。这个协议基于 ETSI 的 TS07.10 标准。

7.服务发现协议(SDP, Service Discovery Protocol)

SDP 工作于 L2CAP 层之上,用于发现一定范围内的蓝牙设备上的服务,通过该协议可以知道哪些服务可用,以及这些服务的属性。这些服务既包括一般的打印、传真等服务,还包括一些复杂的如远程会议、网络接口等服务。

8.电话控制协议

包括电话控制协议二进制(TCS BIN)和 AT Commands。TCS BIN 是面向比特的协议,定义了蓝牙设备间建立语音和数据呼叫所需的呼叫控制信令,AT Commands 是电话控制命令,用于控制移动电话和调制解调器等,由 SIG 在 ITU-TQ.931 基础上开发而成。

9.被采纳的其他协议

除了 SDP、L2CAP、RFCOMM 等蓝牙专用协议外,蓝牙技术还采用了其他一些已有的协议,如 PPP、UDP/TCP/IP、WAP、WAE、OBEX、vCard、vCalendar 等。PPP 工作于 RFCOMM 之上,用于实现点到点的连接。UDP/TCP/IP 用于 Internet 上的通信。OBEX 采用客户-服务器模式提供与 HTTP 相同的基本功能。WAP 是无线应用协议,将 Internet 的电话业务引入移动电话和其他无线终端。vCard 和 vCalendar 则定义了电子商务卡和个人日程表的格式。

二、蓝牙系统的框架结构

除了保证两个蓝牙设备之间可以相互通信的协议外,SIG 还定义了框架(Profile)。其目的是为了描述如何实现用户模块,以及如何将应用和设备映射为蓝牙设备。框架与具体的应用有关,它详细说明了对于某一具体应用来说,协议中的哪些部分是必须包括的,也给每一个协议定义了参数范围。定义框架也可以防止一些内存小、处理能力差的设备占用蓝牙全部的堆栈,这些设备往往只需要占用全部堆栈的一小部分就足够了。在这个意义上说,对于新应用来说框架是动态的,新的框架可以添加到蓝牙标准中。同时,定义框架也降低了不同厂商制造出来的产品之间不能互用的风险。

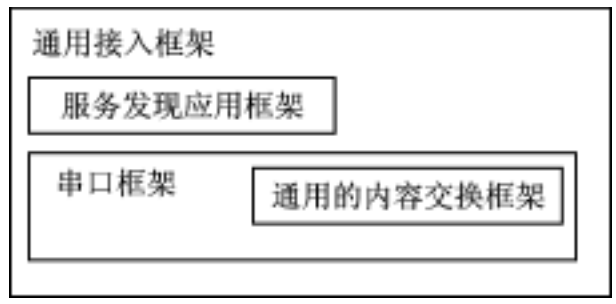


图 4.13-2 蓝牙系统的框架结构

蓝牙标准化组织定义了四个主要的框架,这四个框架分别是:通用的接入框架、串口框架、服务发现应用框架、通用的对象交换框架。框架结构如图 4.13-2 所示。这四个框架是用户模块以及相应框架的基础,同时也考虑了未来的用户模块和框架的发展。

1. 通用的接入框架 GAP(Generic Access Profile)

GAP 定义了一个蓝牙设备如何发现另一个设备并与之建立连接,主要处理未连接的设备之间发现对方以及建立连接的问题。这个框架定义的是通用的操作,可以被与 GAP 有关的框架和实现多个框架的设备使用。

GAP 保证任意厂商生产的两个蓝牙设备之间可以交换信息,并发现这些设备可以提供什么类型的服务。一个蓝牙设备可以不遵守其他蓝牙框架的约定,但必须遵守 GAP 框架以保证基本的互用和共存。

2. 服务发现应用框架 SDAP(Service Discovery Application Profile)

SDAP 定义了发现一个蓝牙设备可用服务的方法,主要处理对已有服务的搜索。SDAP 包括一个用户服务发现应用,这个应用用来在一个蓝牙设备中对服务进行定位,它与服务发现协议接口。SDAP 框架依赖于 GAP 框架,采用了 GAP 框架的一部分。

3. 串口框架 SPP(Serial Port Profile)

串口框架定义了如何在两个蓝牙设备上建立虚拟串口,然后将这两个串口连接起来。采用这个框架,可以为蓝牙设备提供一个使用 RS232 控制信令的串行线缆(RS232 是数据通信设备普遍采用的接口标准)。串口框架就象 SDAP 框架一样依赖于 GAP 框架,也采用了 GAP 框架的一部分。

4. 通用的内容交换框架 GOEP(Generic Object Exchange Profile)

GOEP 定义了一套用于内容交换的协议和过程,比如将数据从一个蓝牙设备如何传送到另一个设备,以及如何从另一个蓝牙设备接收数据等。一些用户模块如文件传输和同步等都基于这个框架。典型应用这个框架的蓝牙设备是笔记本电脑、PC 机、移动电话和智能电话等。

GOEP 依赖于串口框架。

三、蓝牙主要技术

1 . 蓝牙无线技术

(1) 无线频带

蓝牙无线频带的选择要满足两个要求:一是这个频带必须是公开的,用户使用时无需特别申请;二是必须全世界公用。为此,蓝牙选择了 2 .4～2 .5 GHz 之间的频带,即工业-科学-医疗频带,ISM 频带。在欧洲和美国,这个频带是 2 .4～2 .483 5 GHz。蓝牙设备必须能够在 ISM 频带内工作,也能够在 ISM 频带内的某一部分工作。

(2) 跳频技术

为了抗干扰,蓝牙采用了跳频(FH, Frequency Hop)扩谱技术,在 ISM 频带上以 1 600 跳/ s的速率进行跳频,这项技术非常适合于低成本、低功耗的无线通信。在 ISM 频带定义了 79 个跳频载波,每个跳频载波占用 1 MHz 带宽,这样可以有 79 个 1 MHz 带宽的跳频信道。理论上讲,79 个跳频载波可以支持 79 Mbps 的传输速率,即单个信道所能支持的速率为 1 Mbps,但实际上最大传输速率为 721 Kbps。

在时域,FH 系统被分成若干个时隙,每个时隙为 625 μs。每个信道使用 625 μs 后,随机地跳到另一个信道上,一直重复进行,这样就使得 FH 系统具有很好的抗干扰能力。

蓝牙还定义了大量的伪随机跳频序列。如果一个蓝牙主设备占用了一个 FH 信道,那么一个跳频序列将被指定给这个主设备,主设备的内部时钟决定跳频序列的相位。占用同一跳频信道的是从设备,这些从设备采用主设备 ID 来选择同一个跳频序列,并在各自的时钟上加上一个偏移量与跳频同步以便于跳频。

为了降低接收机的复杂度,蓝牙采用时分复用(TDD)技术进行全双工通信,发射、接收在不同的时隙内进行。

(3) 调制方式

蓝牙采用高斯二进制 FSK 调制方式以降低接收机的复杂度。蓝牙的基带协议是电路交换与包交换的结合,对于同步数据包采用预留时隙的办法。一个数据包一般只用一个时隙,但蓝牙中也定义了多时隙方法,多时隙数据包可以占用 3～5 个时隙。数据包一般是在单个的跳频信道中发送。

(4) 蓝牙网络

蓝牙中定义了 Pico 网。一个 Pico 网最多包括 8 个蓝牙设备,一个是主设备,负责建立 Pico 网,其余 7 个是从设备,这 8 个设备共享同一个 FH 跳频信道。一个 Pico 网只能有一个主设备,由主设备决定 FH 跳频信道。这样定义的目的是为了保证设备间高速、可靠的连接。

蓝牙跳频技术的核心就是主设备的系统时钟和 ID(Identity)。蓝牙定义了大量的伪随机跳频序列,一个跳频信道由跳频序列和这个序列的相位来确定。Pico 网主设备的 ID 决定跳频序列、系统时钟决定跳频序列的相位,不同的信道有不同的主设备从而有不同的跳频序列和相位。对于从设备,系统时钟要加上一个偏移量以保证和主设备的系统时钟一致。这样,一个 Pico 网的所有设备的时钟都是同步的。

除了确定 Pico 网,主设备还负责 Pico 网内所有的流量控制和接入控制。蓝牙根据从设备的特点采用智能化的安排算法由主设备来安排上行和下行的流量,这样可以避免信道中的传输冲突。

2.数据包格式定义

蓝牙技术采用基于包的传输方式,所有的蓝牙数据包都有相同的格式:72 位的接入码、54 位的包头和 2 745 位的数据部分,一个数据包最多可达 340B。

接入码具有随机的特性,它取决于主设备的 ID 和主设备的系统时钟,它提供了用于同步的手段,接入码中包含 Pico 网主设备的 ID。对于一个具体的信道来说,这个接入码是惟一的,在这一信道内传送的所有数据包都采用同一个接入码。只有数据包的接入码与主设备的接入码相同,这个数据包才能被接收。这样即使另一个 Pico 网的设备也用同一个跳频载波,它也无法接收到这个数据包。

包头包括连接控制信息:从设备地址、数据类型、纠错、重传、码流控制信息等。

3.连接类型

蓝牙定义了两种类型的连接:异步、无连接类型 ACL(Asynchronous Connectionless Link)和同步、面向连接的类型 SCO(Synchronous Connection Oriented)。

SCO 连接支持对称的、电路交换、点到点的连接,所以主要是用于语音。两个固定间隔内的时隙预留给了 SCO 连接。SCO 连接的数据速率在两个方向都是 64 Kbps。

ACL 连接是为突发的数据传输定义的,支持对称、非对称、包交换、点对多点的连接。在没有使用纠错的情况下,多时隙的数据包采用 ACL 连接类型可以达到的最高速率一个方向是 721 Kbps,另一个方向是 57.6 Kbps。主设备控制 ACL 连接的带宽,而且决定从设备可以使用多少带宽。ACL 连接也支持从主设备到从设备的消息广播。

SCO 和 ACL 可以重用同一个 RF 连接。

4.蓝牙系统安全问题

虽然蓝牙技术主要目的是为了进行短距离内的无线连接,但蓝牙技术中还是加入了认证和加密技术以防止被监听以及未被授权的用户使用。这两个技术与跳频技术结合起来,再加上蓝牙设备的传输范围只有 10 m,保证蓝牙技术有很好的保护用户不被监听的能力。

两个蓝牙设备之间一旦建立了连接,就有一个鉴权过程来鉴别设备的身份。鉴权过程的中心部分是一个 128 位的连接密码,这个连接密码可以看作是为两个设备之间提供了协议。蓝牙在较低层提供了一些安全措施,在较高层还采用了更先进的安全措施,比如公共密码、认证等,以尽可能为用户提供安全通信。

5.Pico 网之间的通信

蓝牙系统允许多个 Pico 网共存在同一区域内而不会造成明显的通信质量下降。在同一区域内的多个 Pico 网组成了分散网(Scatternet)。所有的 Pico 网共享 80 MHz 的带宽,每一个 Pico 网用 1 MHz。每一个 Pico 网有不同的跳频序列,在不同的跳频信道中传输。

由于蓝牙采用了在时隙上连接的基于数据包的传输方式,这使得一个设备可以加入多个 Pico 网,但某一具体时刻只能加入一个 Pico 网。只要调整 Pico 网信道的参数(比如主设备 ID 和时钟),一个设备就可以从一个 Pico 网跳到另一个 Pico 网。在从一个 Pico 网跳到另一个 Pico 网时,设备还可以改变自己的身份,比如从主设备变为从设备等。但是根据定义,一个设备在一个 Pico 网内是主设备,跳到另一个 Pico 网后,它只能是从设备而不能是主设备,因为是主设备确定 Pico 网的 FH 信道。

蓝牙设计了一个跳选择机制允许 Pico 网间的通信,只要改变跳选择机制中的 ID 和时钟输入,马上就可以跳到一个新的 Pico 网中。

四、发展前景

随着计算机技术和移动通信技术的迅速发展,人们对移动电话、笔记本电脑、掌上电脑的需求量越来越大,而对这些设备之间能无线连接的要求也越来越高。蓝牙作为一个开放的无线应用标准,能通过无线连接的方式将一定范围内的多个设备连接起来,使人们能够方便、快速地进行语音和数据信息的交换与传输,无疑将成为未来信息领域的一个研究热点。它必将逐步进入人们的日常生活和工作中,成为我们生活中不可缺少的一部分。

参 考 文 献

1 AAP C .H, The Bluetooth Radio System, IEEE Personal Communication, February 2000

2 Bluetooth overview, http:\\www .bluetooth .com 2000,12

3 Bluetooth profiles, http:\\www .palowireless .com 2000,12

4 Bluetooth tutorial, http:\\www .palowireless .com 2000,12

5 Bluetooth whitepaper, http:\\www .ausystem .com 2000,12

选自《电子技术应用》月刊,2001 年第 3 期

4 .14　蓝牙芯片及其应用

西安电子科技大学信息科学研究所(710071)　刘　兴　李建东

一、关于蓝牙

“蓝牙”是一项令人振奋的技术,它利用微波取代传统网络中错综复杂的电缆,使家庭或办公场所的移动电话、便携式计算机、打印机、复印机、键盘、耳机及其他手持设备实现互联互通,将人们从无数的连接电缆中解放出来,自由方便地构成自己的个人网络。有了蓝牙,你甚至不用掏出你的移动电话,就可以用 PDA(个人数字助理)通过口袋中的移动电话查阅新闻、订票以及进行其他电子商务活动,无拘无束、自由自在。这个由爱立信公司于 1995 年提出的概念已广泛地为业界所接受,从 SIG(蓝牙特殊利益集团)的成员就可以看出业界对它的重视程度。SIG 的九个成员包括爱立信、诺基亚、摩托罗拉、3COM、IBM、INTEL 等,都是各自行业的‘领导者’。目前,这一技术已经有 2 000 多家支持厂商。蓝牙技术的应用非常广泛,来自 IDC 的数据预测,到 2005 年全球围绕移动设备、桌面设备和其他设备将有 40 亿件蓝牙产品被广泛应用,具有极大的市场潜力。

作为取代数据电缆的短距离无线通信技术,蓝牙支持点对点及点对多点的通信,它以无线方式将家庭或办公室中的各种数据和语音设备联成一个微微网(Piconet),几个微微网还可以进一步实现互联,形成一个分布式网络(Scatternet),从而在这些联接设备之间实现快捷而方便的通信联系。蓝牙的工作频段为全球开放的 2 .4 GHz ISM(工业、科学、医学)频段,这样就保证了旅行者可以毫无障碍地使用蓝牙设备。由于 ISM 频段是对所有无线电都开放的频段,汽车、微波炉等将有可能成为其不可预测的干扰源,因此对蓝牙特别设计了快速确认和跳频方案以确保连接的稳定和数据保密。在目前公布的蓝牙规范‘Bluetooth V1 .0’中,数据传输速率最大为 721 Kbps,通信距离为 10 m,若加大发射功率,通信距离可达 100 m。

二、蓝牙系统结构

MT1020 基带控制器和 PH2401 无线收发器分别由 MITEL 公司和 PHILSAR 公司提供,两者配合可构成完整的低功耗的蓝牙模块,提供高至 HCI(主机控制接口)层的功能。它们在蓝牙系统中的位置如图 4 .14－1 所示。

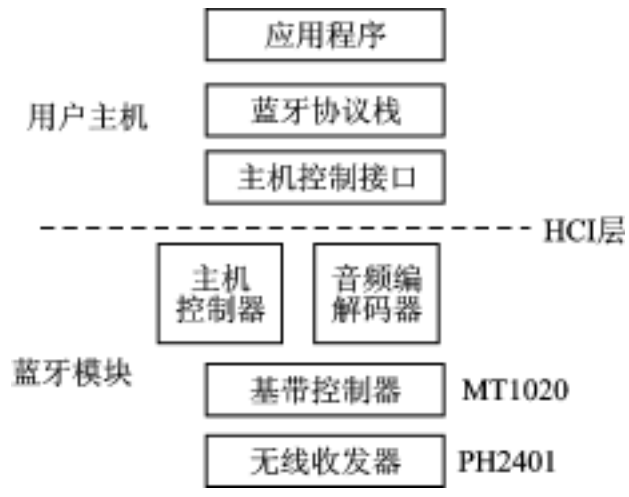


图 4 .14－1　蓝牙系统结构

MT1020 基带控制器负责蓝牙基带部分的功能,完成基带以及链路的管理,包括对 SCO(同步)和 ACL(异步)连接方式的支持、差错控制、物理层的认证与加密、链路管理等;PH2401 实现数据的无线接收和发送;虚线以上部分由用户根据不同的应用需求来实现。特别值得一

提的是,在该蓝牙模块解决方案中,即将推出的改进型基带控制器 MT1020B 可提供 20 KB 的用户 ROM,使用户可以利用其内嵌的低功耗、高性能的 32 位 ARM7TDMI 内核,从而简化用户设计,实现最低功耗、最高集成度的蓝牙产品。

三、蓝牙芯片组简介

1 . MT1020 的内部结构及各功能块介绍

MT1020 由嵌入式微处理器和蓝牙基带外设组成,如图 4 .14 - 2 所示。在该芯片中,系统内部时钟可以低至 5 MHz、内核供电电压为 2 V、硬件解码、支持 DMA 传输,所有这些使得该芯片具有超低功耗。

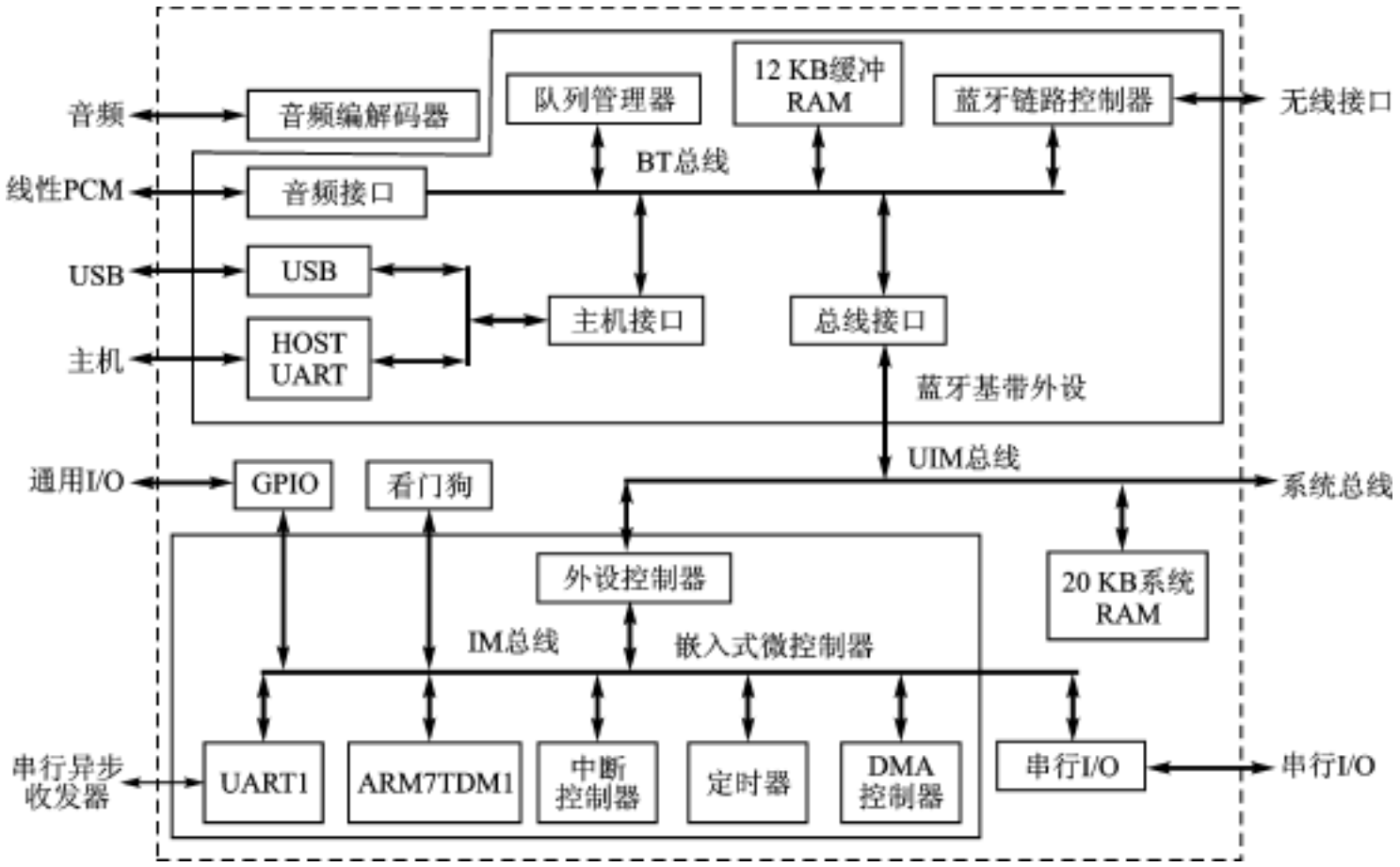


图 4 .14 - 2 MT1020 内部结构

(1) 基带外设

基带外设以最小的开销完成重要的蓝牙操作,它挂在“向上集成模块总线(Up-Integration Module Bus) ”上,由以下几个功能块组成。

① 总线接口

总线接口完成微处理器与基带外设之间的通信,基带外设内各个模块之间使用专门的 BT 总线传送数据。

② 链路控制器

链路控制器与 PH2401 无线收发器接口,需要发送的数据在链路控制器中被装配,加上同步字、帧头以及 CRC 校验字,并且被白化,是否进行加密可由用户选择;收到的数据则在此被解码、检错。

③ 队列管理器

队列管理器完成缓冲 RAM 与链路控制器以及 USB 或串行主机与音频接口之间的智能

DMA 传输。它能识别不同格式的蓝牙数据包,并能进行相互转换。

④ 缓冲 RAM

专门用于存储蓝牙数据包和变量,容量为 12 KB。

⑤ 音频编解码器

音频编解码器是一个全双工的编解码器,包括麦克风放大器和耳机驱动器,其中的数字转换器能够进行线性 PCM、A 律 PCM、μ律 PCM 及 CVSD(连续可变斜率增量调制)之间的相互转换。

(2) 嵌入式微控制器内核

嵌入式微处理器由 32 位 RISC 架构的 ARM7TMDI 中央处理单元、专用的模块交互总线(Inter-Module Bus)和其他一些功能块组成。

① ARM7TDMI 处理器

ARM7TDMI 是一款性能优异的嵌入式 CPU,具有极快的运算速度和很低的功耗,利用其内部的 Thumb 指令压解器可支持 16 位指令,并支持扩展调试、快速乘法等功能,它通过模块交互总线与其他功能快交换数据。

② 外设控制器

外设控制器是 MT1020A 中内部总线与外部总线进行通信的主要通道,它支持动态总线宽度,并能产生访问外设所需要的控制信号。

③ 串行 I/O

串行 I/O 用来连接各种串行接口器件,例如与串行 EEPROM、串行时钟等器件接口。

④ 中断控制器

ARM7TDMI 处理器接受两种中断请求:普通中断请求和快速中断请求。根据用户所需要的优先级,所有中断都可设置成两种类型中的一种。中断控制器能处理八个外部中断和两个内部中断。外部中断可被编程设置成电平触发或沿触发。为减少中断响应的延迟时间,对每种类型的中断,中断控制器能进行硬件优先级判断,从而加快对中断的实时响应。

⑤ 定时器/计数器

MT1020A 提供两个双独立 32 位定时器/计数器,它们与系统时钟同步,可以在程序中轮询,也可设置成溢出中断,并能自动重装。

⑥ DMA 控制器

在该控制器中有两个 DMA 引擎,它们可以配置成一对,从而支持 ARM7TDMI 中任意位置两个内存块的 DMA 传输。当然,它们也可独立使用。

⑦ 通用异步收发器

通用异步收发器接口形式为 RS-232,支持硬件握手和 XON/XOFF 软件协议,收发通道上各有一个缓冲器,可以在程序中轮询,也可使用中断形式。器件内部的波特率发生器用来产生需要的数据速率。

⑧ 系统存储器

系统存储器挂在 UIM 总线上。MT1020 有 20 KB 的内部静态 RAM,用于程序变量的存储。用户需要外挂一个外部 ROM/FLASH 来存储蓝牙链路控制和管理协议代码。在 MT1020 的下一个版本中,将提供片上 ROM,并具备存储用户代码的能力,从而简化用户设计,降低功耗。

2 . PH2041 蓝牙无线收发器功能

PH2401 单片无线收发器用砷化镓工艺制造,具有高集成度、超低功耗、体积小等优点,专门优化用于 2 .4 GHz 无线个人系统,完全兼容蓝牙规范“ Bluetooth V1 .0 ”。它工作于 2 .4 GHz 的 ISM 频段,以每秒 1 600 次的速度在 79 个频道(2 .402～2 .408 GHz)上快速跳频,最大位传输速率可达 1 Mbps。

PH2401 采用调制指数为 0 .3 的高斯频移键控(GFSK)调制方式,信道带宽为 1 MHz,频偏在 140～175 kHz 之间,满足蓝牙 2 级和 3 级操作,发送功率可在 - 10～ + 2 dBm 之间编程设定,发射范围为 10～100 m。接收器由 RF-IF 下变频器、自动增益控制(AGC)、滤波器、双通道模/ 数转换器及调制器组成。

基带控制器通过串行总线与 PH2401 接口,通过对其内部寄存器的读写实现跳频、调谐等其他控制。

四、蓝牙无绳电话

由 MT1020 和 PH2401 构成的蓝牙模块提供高至 HCI 的功能,因此可以很方便地利用它构成蓝牙系统。我们采用该芯片组设计了蓝牙无绳电话。

根据蓝牙规范对无绳电话的协议要求,无绳电话实现协议栈如图 4 .14 - 3 所示。

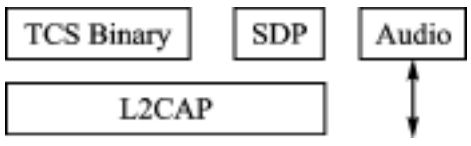


图 4 .14 - 3 蓝牙无绳电话协议栈

通过服务发现协议(SDP),子机寻找通信范围内所有蓝牙设备信息和服务类型,从而与无绳电话主机建立连接。语音呼叫的控制信令则在二元电话控制协议(TCS Binary)中定义。逻辑链路控制应用协议(L2CAP)向上层提供面向连接和无连接的逻辑链路,传输上层协议数据。语音流不经过逻辑链路控制应用协议(L2CAP),直接与基带控制器连接,使用连续可变斜率增量调制(CVSD)技术,以获得高质量传输的音频编码。

蓝牙无绳电话子机的基本电路框图如图 4 .14 - 4 所示。

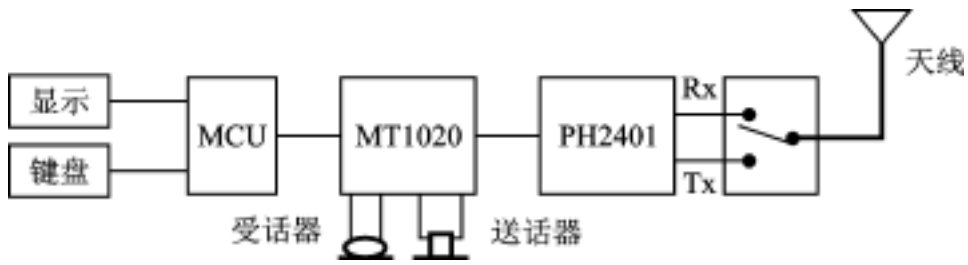


图 4 .14 - 4 蓝牙无绳电话子机电路框图

MCU 不仅完成对键盘、显示器的控制,而且实现 TCS Binary、SDP 和 L2CAP 协议,受话器和送话器直接与 MT1020 基带控制器连接,系统简洁、可靠,具有很好的性能价格比。

参 考 文 献

1 金 纯 .蓝牙技术 .北京:电子工业出版社,2001

2 Mitel Semiconductor Inc . MT1020 Data Sheet . 2000

3 Philsar Semiconductor Inc . PH2401 Preliminary Pruduct Sheet . 2000

选自《电子技术应用》月刊,2001 年第 7 期

4 .15　BlueCore™ 01 蓝牙芯片的特性与应用

广州华南理工大学电子与通信工程系(510641)　高智衡　韦　岗

蓝牙技术是一项新兴的计算与通信方面的短距离(约 10 m)无线电信号传输标准[1]。在这种技术的支持下,各种各样的电子设备(包括便携设备、家用设备、办公设备以及汽车和航空设备等)可以在不需电缆的条件下,轻轻松松地自动连接起来,实现相互通信和数据共享。

蓝牙技术所描绘的诱人蓝图已逐渐引起了各大软硬件公司的注意。爱立信、IBM、英特尔、诺基亚和东芝五大企业于 1998 年共同组建蓝牙特殊兴趣组 BSIG(Bluetooth Special Interest Group),经过短短两年的发展,成员已由 5 个骤升到近 2 000 个。根据 Frost & Sullivan 所进行的一项研究,2000 年蓝牙的销售额将从 1999 年的几乎为零猛增到 3 670 万美元,到 2006 年更有望增至 6 亿 9 900 万美元[2]。目前已有不少公司生产出高集成度的蓝牙芯片组,例如朗讯的 W7020 + W7400 芯片组、飞利浦的 LMX3162 芯片、ATMEL 的 T2901 和 AT76C551 芯片等。本文介绍 CSR 公司推出的 BlueCore™ 01 低复杂度、低成本、单芯片蓝牙方案。

一、BlueCore™ 01 的特性

目前大多数厂家所提供的蓝牙方案在硬件上都分为三部分:无线电收发单元、链路控制单元以及链路管理及主机 I/O 单元,如图 4 .15 -1 所示。BlueCore™ 01 是在一片 8 mm×8 mm 的 CMOS 芯片上集成了射频单元和基带控制器,只要和内含蓝牙软件栈的 Flash Rom 配合,即可向数据和语音设备提供全兼容的蓝牙接口。

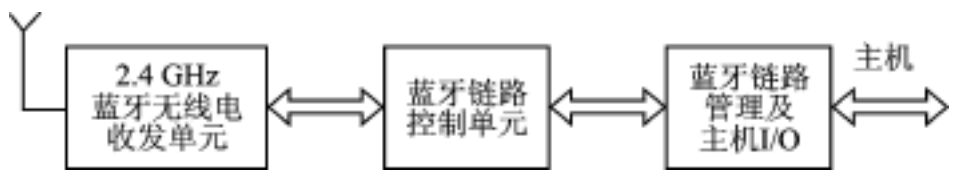


图 4 .15 -1　蓝牙系统方框图

BlueCore™ 01 芯片的方框图如图 4 .15 -2 所示,该芯片由射频接收器、射频发射器、射频合成器、物理层 DSP 硬件引擎、猝发状态控制器、微处理器、内存管理单元等部分组成[3]。

1 . 射频接收器(RF Receiver)

接收器所采用的极低中频频率结构使得信道滤波器可集成到片内。低噪放大器(LNA)的输入端具有 1 dB 带外阻塞特性,因此即使与 900 MHz、1 .8 GHz 或 1 .9 GHz 的蜂窝移动电话发射器很接近也不需在射频输入端采取特殊滤波处理。数字式的鉴频器提供了抗同频干扰和邻频干扰的良好性能。而且,每个时隙都测量接收信号的强度并调整前端 LNA 的增益以限制混频器的输入信号幅度,从而实现了快速的自动增益控制(AGC)。

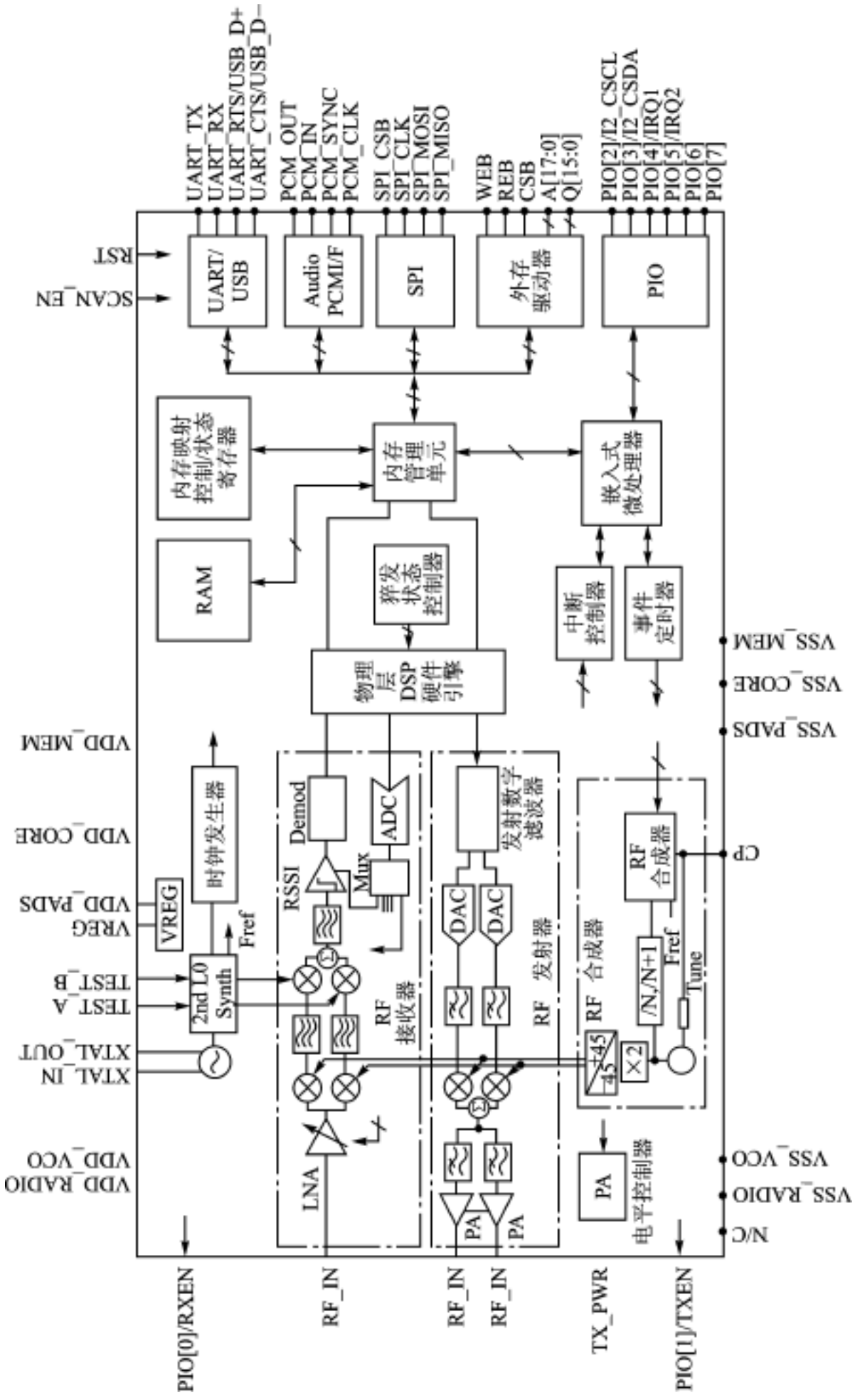


图 4.15-2 Blue Core™01 原理方框图

2 . 射频发射器(RF Transmitter)

发射器采用了 2 .4 GHz 的直接正交调制,使得频率漂移极小而且调频指数稳定。基带发射数字滤波器保证了频谱形状满足要求。最大发射功率可达 + 4 dBm,符合二级(4 dBm)和三级(0 dBm)功率要求。由于片内提供了发射增益控制,因而只需简单地增加一级或两级射频功放即可符合一级(20 dBm)要求。

3 . 射频合成器(RF Synthesiser)

合成器采用了新颖的电路形式,完全集成到片内,省却了外部的压控振荡器、变容二极管或者 LC 振荡电路。合成器的工作频率仅为发射频率的一半,减少了与射频放大器的耦合。

4 . 物理层 DSP 硬件引擎(Physical Layer DSP Hardware Engine)

该硬件引擎用专门硬件电路来实现,能够在高速运算的同时维持低功耗。它所实现的功能包括:前向纠错(Forward Error Correction)[1] 、帧头错误控制、加密、数据白化(Data Whitening)[1] 以及访问码比较等。此外,还能完全兼容 A 律、μ 律、线性和 CVSD(连续可变斜率增量调制[1])四种语音格式的数据。

5 . 猝发状态控制器(Burst Mode Controller)

其功能是:发送时,把先前已保存到内存映射寄存器(MMR)中的头信息和 RAM 缓冲区中的数据/ 语音合成为数据包以供发送;接收时,则完成相逆的功能,把数据包分解为头信息和数据/ 语音,分别存入 MMR 和 RAM 缓冲区。猝发状态控制器的采用进一步减轻了微处理器的工作量。

6 . 微处理器(Microprocessor)

片内集成的通用微处理器是 16 位 RISC 结构,有足够的能力运行完整的蓝牙软件栈和 OEM 应用程序。微处理器的结构具有占用面积小和消耗电流小的特点。

7 . 内存管理单元(Memory Management Unit)

内存管理单元能为用户的数据/ 语音信号动态地分配环状缓冲区,提高了可用 RAM 的利用率。

二、BlueCore™ 01 的应用

BlueCore™ 01 的典型应用如图 4 .15 - 3[3] 所示。TX_A、TX_B 为射频发射输出脚,RF_IN 为射频接收输入脚,与天线相连。主机可以通过 UART 或 USB 与 BlueCoreTM01 通信,也可用专门的 PCM 接口来收发 PCM 语音信号。蓝牙协议栈的软件存储在片外的 FLASH ROM 中,由 BlueCore™ 01 内嵌的微处理器执行。此外,BlueCore™ 01 还提供了 I^2C 接口,可直接与 I^2C 芯片相连,而 8 条通用 IO 引脚则提供了方便的控制功能。

BlueCore™ 01 是 CSR 提供的蓝牙解决方案的第一代芯片,目前正在开发第二代和第三代芯片(代号为 BlueCore™ 02/ 03)。第二代芯片将采用 0 .25 μm 的 CMOS 工艺,并且只需 64 引脚(BlueCore™ 01 为 81 引脚),使芯片面积更小。而最重要的改进是把 FLASH ROM 也集成到片内,使之成为彻底的单芯片方案。第三代芯片除了更进一步缩小芯片面积以外,将在第二代的基础上增加语音编解码功能和面向应用的定制 I/ O 口[4] 。

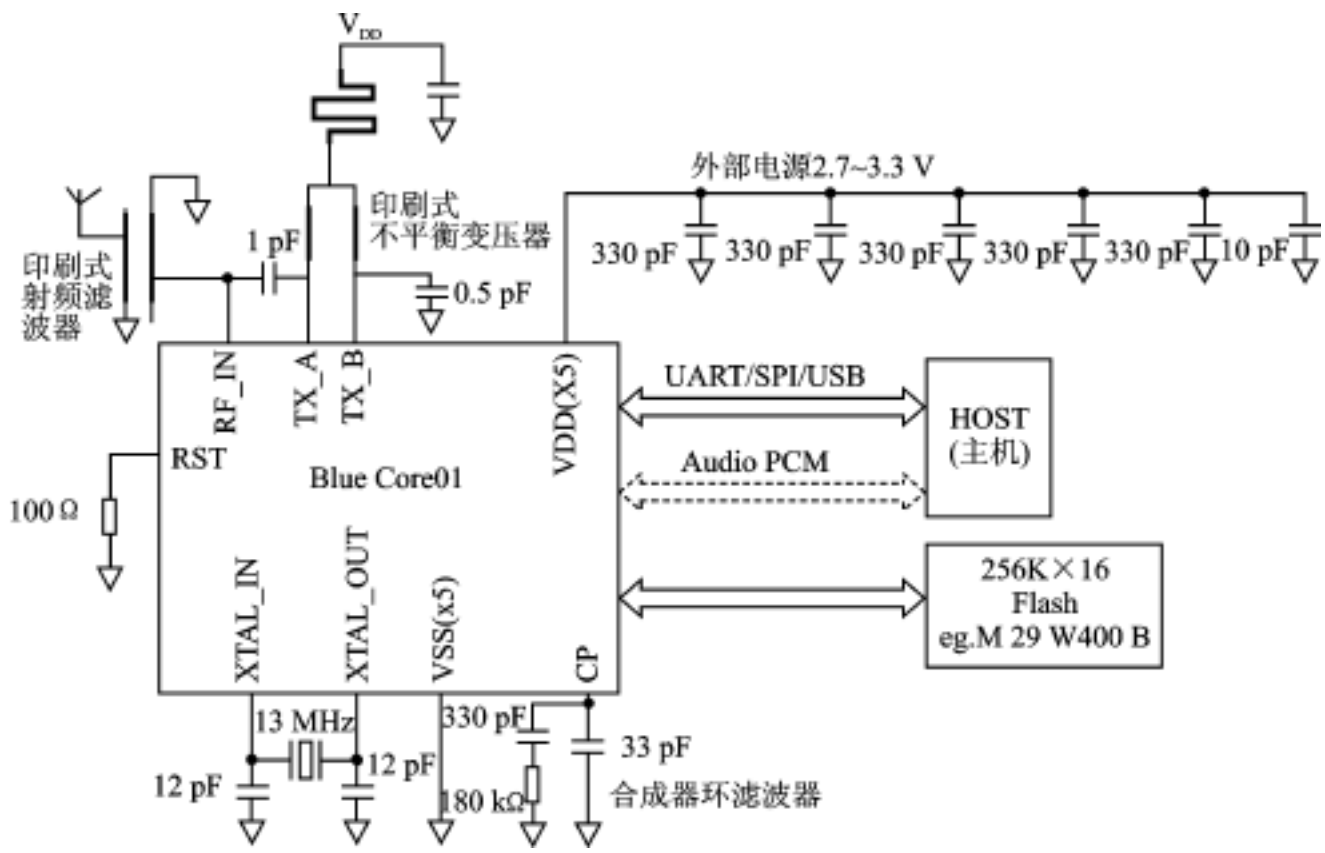


图 4 .15 - 3 基于 Blue Core01 的蓝牙系统

参 考 文 献

1 pecification of the Bluetooth System - Core, Version 1 .0 b . http:/ / www .bluetooth .com/ developer/ core_10_b .pdf

2 研究报告:蓝牙产品销售将大规模增长 . http:/ / www .waterwood .com .cn/ news/ 2000/ 2000-01/ 011401 .htm

3 BlueCore™ 01 Single Chip Bluetooth System . http:/ / www .cambridgesiliconradio .com/ bc 01 .pdf

4 Three Design Options for Implementing Bluetooth Products . http:/ / www .cambridgesiliconradio .com/ options .htm

选自《电子技术应用》月刊,2001 年第 1 期

4 .16　内嵌微控制器的无线数据发射器的特性及应用

衡阳南华大学电气工程学院　黄智伟　朱卫华　陈　和

AT86RF401 是单片集成内嵌 AVR RISC 微控制器的 RF 无线数据发射器,输出频率为 250～450 MHz,最大输出功率 + 6 dBm,发射率 10 Kbps。芯片内嵌 AVR RISC 微控制器、2 KB(1K× 16b)的 Flash 程序存储器、128 B 的可下载的 EEPROM 数据存储器、128 B 的 RISC SRAM、看门狗定时器、6 个通用 I/ O、在系统可编程。工作电压 2 .0～3 .5 V。待机电流 0 .1～0 5 μA,发射电流 17 mA,可用 CR20332/ CR201B $LiMnO_2$ 电池供电。可用于遥控无键入口(RKE)发射器、车库门开门器、遥测(轮胎压力,水、电、气表,贵重物品跟踪)、无线安防系统、无线电遥控等应用领域。

一、引脚及功能

AT86RF401 采用 20 脚 TSSOP 封装,各引脚功能如表 4 .16 - 1 所列。

表 4 .16 - 1　AT86RF401 引脚功能

引　脚	符　号	功　　能
1	ANTB	天线输出
2	LOOPFIL	外接 VCO 回路滤波器
3	L1	外接 VCO 电感
4	L2	外接 VCO 电感
5	RESETB	SPI 复位输入
6	NC	空　脚
7	SDI/ IO0	SPI 数据输入、输入/ 输出 0
8	SDO/ IO1	SPI 数据输入、输入/ 输出 1
9	SCK/ IO2	SPI 时钟输入、输入/ 输出 2
10	XTAL	晶振输入
11	XTALB/ CLK	晶振/ 时钟输入
12	IO3	输入/ 输出 3
13	IO4	输入/ 输出 4
14	IO5	输入/ 输出 5
15	DGND	数字地
16	AGND	模拟地
17	DVDD	数字电源
18	AVDD	模拟电源
19	CFIL	外接数据滤波器
20	ANT	天线输出

二、基本结构和特性

AT86RF401 内部结构框图如图 4.16-1 所示,包括一个完整的发射器电路和微控制器电路。

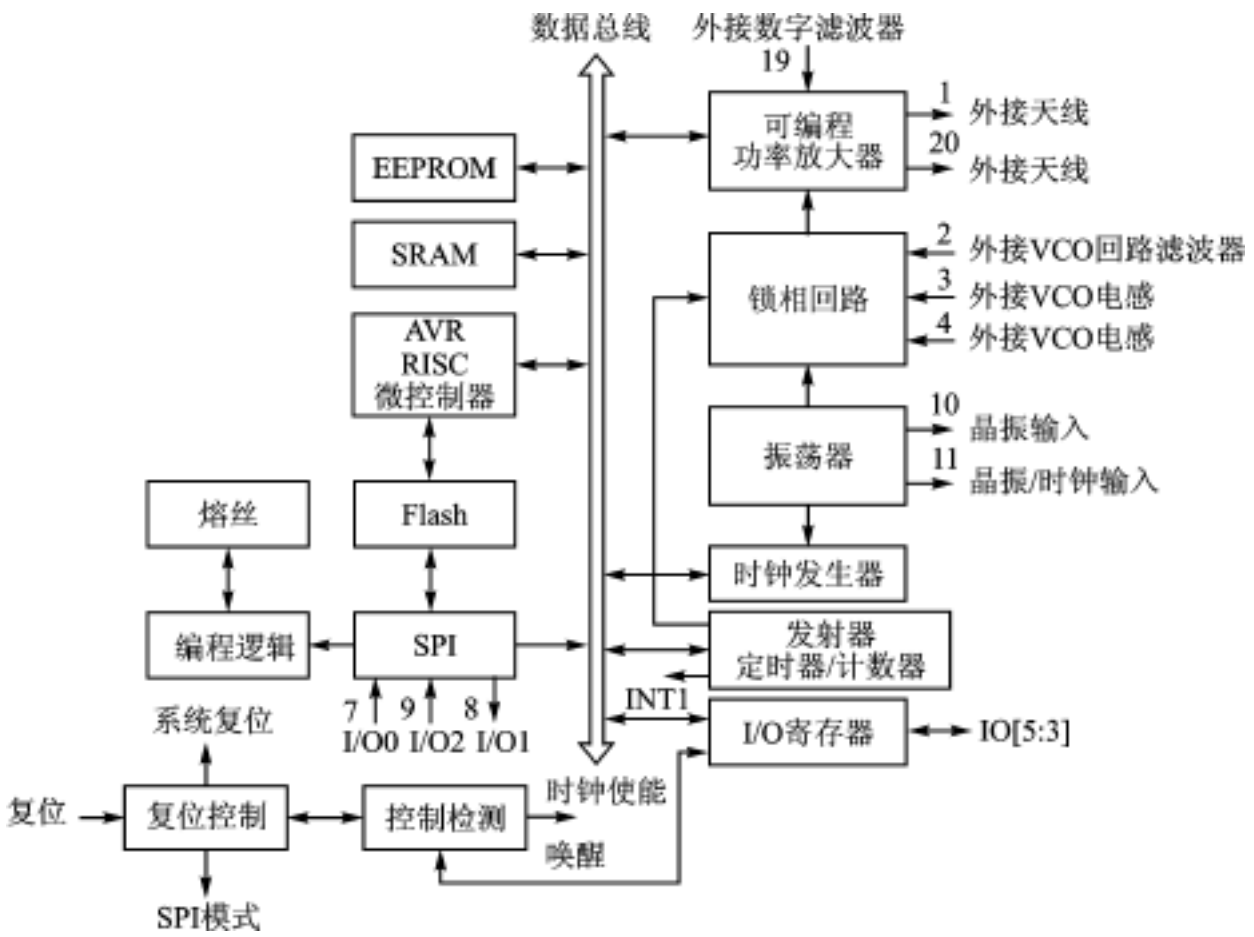


图 4.16-1 AT86RF401 内部结构框图

1.发射器

晶体振荡器振荡频率是 6～20 MHz,为整个芯片提供主时钟,并使用一个可编程的分频器为 AVR 系统提供时钟。PLL 输出信号经 RF 功率放大器产生一个适合驱动在片调谐回路天线的差动输出(RF 发射)。PLL 输出信号在发射前被选通(导通-截止 ASK)或被调幅,使用 RF 载波发射数据流。RF 功率放大器输出功率能够使用软件调节。

2.AVR 微控制器结构

I/O 和控制寄存器:I/O 空间地址和功能如表 4.16-2 所列。AT86RF401 I/O 和外设被放置在 I/O 空间内。各 I/O 存储单元利用输入和输出指令在 32 个通用工作寄存器和 I/O 空间之间传送数据。地址为 $00～$1F 内的 I/O 寄存器,使用 SBI 和 CBI 指令,可直接进行存取。在这些寄存器中的每位数值都能够使用 SBIS 和 SBIC 指令检验。

表 4.16-2 AT86RF401 I/O 空间地址和功能

地 址	名 称	功 能
$3F	SREG	状态寄存器
$3E	SP	堆栈指针高位寄存器
$3D	SPL	堆栈指针低位寄存器
$35	B_CONFIG	低电池组低寄存器

续表 4.16-2

地 址	名 称	功 能
$34	B_DET	按键检测寄存器
$33	PWR_CTL	功率控制寄存器
$32	IO_DATIN	I/O DATA 输入寄存器
$31	I_DATOUT	I/O DATA 输出寄存器
$30	IO_ENAB	I/O 使能寄存器
$22	WDTCR	看门狗定时控制寄存器
$21	BTCR	位定时控制寄存器
$20	BTCNT	位定时控制寄存器
$1E	DEEAR	数据 EEPROM 地址寄存器
$1D	DEEDR	数据 EEPROM 数据寄存器
$1C	DEE	数据 EEPROM 控制寄存器
$17	TXCR7	发射组态寄存器 7
$16	TXCR6	发射组态寄存器 6
$15	TXCR5	发射组态寄存器 5
$14	TXCR4	发射组态寄存器 4
$13	TXCR3	发射组态寄存器 3
$12	CTL0	
$11	TXCR2	发射组态寄存器 2
$10	TXCR1	发射组态寄存器 1

AVR 状态寄存器 SREG 为 8 位,包括:总中断使能、位复制存储、半进位标志、符号位、溢出标志、负数标志、零标志和进位标志。可读可写,初始值为 00H。复位和中断处理,必须设置 SREG 中的中断控制使能位。

对于位定时器,AT86RF401 使用 2 个中断。这些中断和复位在程序存储器空间有各自的编程向量。复位和中断向量如表 4.16-3 所列。

表 4.16-3 AT86RF401 复位和中断向量

向量编号	编程地址	源	中断定义
1	$000	复位、看门狗、按键	硬件端、看门狗或按键复位
2	$002	位定时器	位定时器标志 2 中断
4	$004	I/O3	位定时器标志 0 中断
5	$006	TBD	未使用

对于复位和中断处理,最典型和通用的程序设置是:

```
地址表    码                    释
$000    jmp RESET          ;复位处理
$002    jmp BT_F2_ISR      ;位定时器 2 中断服务程序
```

```
$004    jmp BT_F0_ISR          ;位定时器 0 中断服务程序
$006    MAIN: < instr > XXXX   ;主程序开始
```

AT86RF401 复位源有:① 上电复位。当电源电压加到 V_{DD} 和 GND 端时,AT86RF401 复位。② 外复位。当逻辑低电平加在 RESETB 端时,AT86RF401 复位。这个复位复位所有的 I/O 寄存器和置通道进入 SPI 模式。在 2 个 AVR 系统时钟脉冲后,I/O 寄存器可通过 SPI 接口读和写。③ 看门狗复位。类似上电复位,由看门狗定时器引起。④ 按键复位(软件复位)。是由软件设置的专门复位,在按键复位时大多数的 I/O 寄存器不被复位。⑤ 电压下降复位。当电池电压低于设定值时,产生复位。在上电复位和看门狗复位时,所有 I/O 寄存器是被设置在初始状态,程序开始执行的地址是 $000。设置在 $000 单元的指令必须是 RJMP 或 JMP 转向复位处理程序。

存储器编程:AT86RF401 MCU 提供 2 个程序存储器允许编程/不允许编程锁定位。锁定位保护模式如表 4.16-4 所列。AT86RF401 提供 2 KB(字节)的可反复编程的 Flash 程序存储器和 1 Kb(位)的 EEPROM 数据存储器。存储器可由串行 SPI 接口编程。当 RESETB 端接地时,程序存储器和数据存储器可使用串行 SPI 总线编程。串行接口由 SCK、SDI(输入)和 SDO(输出)组成。在 RESETB 被设置为低后,在编程/擦除操作执行前,必须首先执行编程使能指令。

表 4.16-4 AT86RF401 锁定位保护模式

编程锁定位			保护类型
模式	LB1	LB2	
1	1	1	无编程锁定 < 可编程 >
2	0	1	EEPROM 不可编程
3	0	0	EEPROM 不可编程也不可校验

AT86RF401 包括一个 128 B EEPROM。这个存储器利用在 I/O 存储器中的 3 个寄存器 DEECR、DEEDR、DEEAR 访问。

串行编程和校验电路如图 4.16-2 所示。串行编程时序波形如图 4.16-3 所示。当写数据到 AT86RF401 时,数据在 CLK 的上沿被选通;当从 AT86RF401 读数据时,数据在 CLK 的下沿被选通。

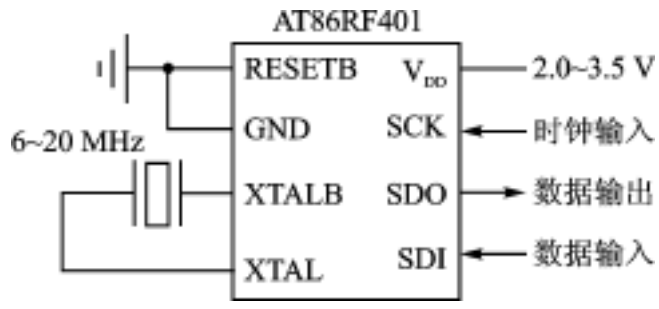


图 4.16-2 AT86RF401 串行编程和校验电路图

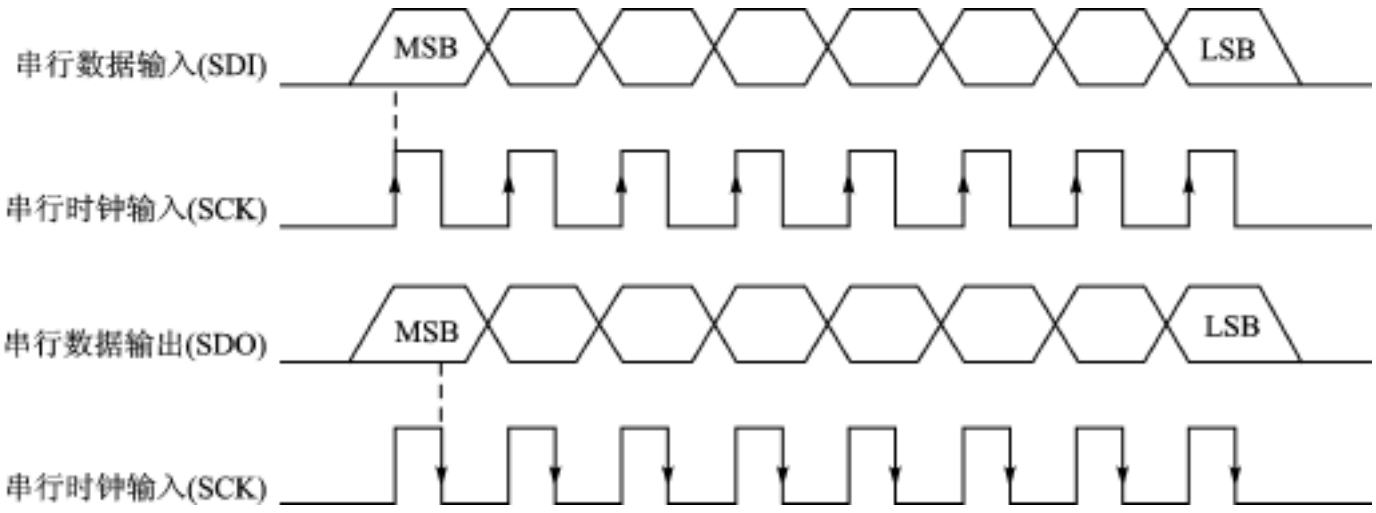


图4.16-3 AT86RF401串行编程时序图

三、应用电路

典型的应用电路如图4.16-4所示。图4.16-4中芯片工作频率为315 MHz,如需工作在433.92 MHz,则电路中元件参数R1为9.1 kΩ,C3为6.5 pF,C8为120 pF,Y1为18.08 MHz。发射天线可印制在PCB上。

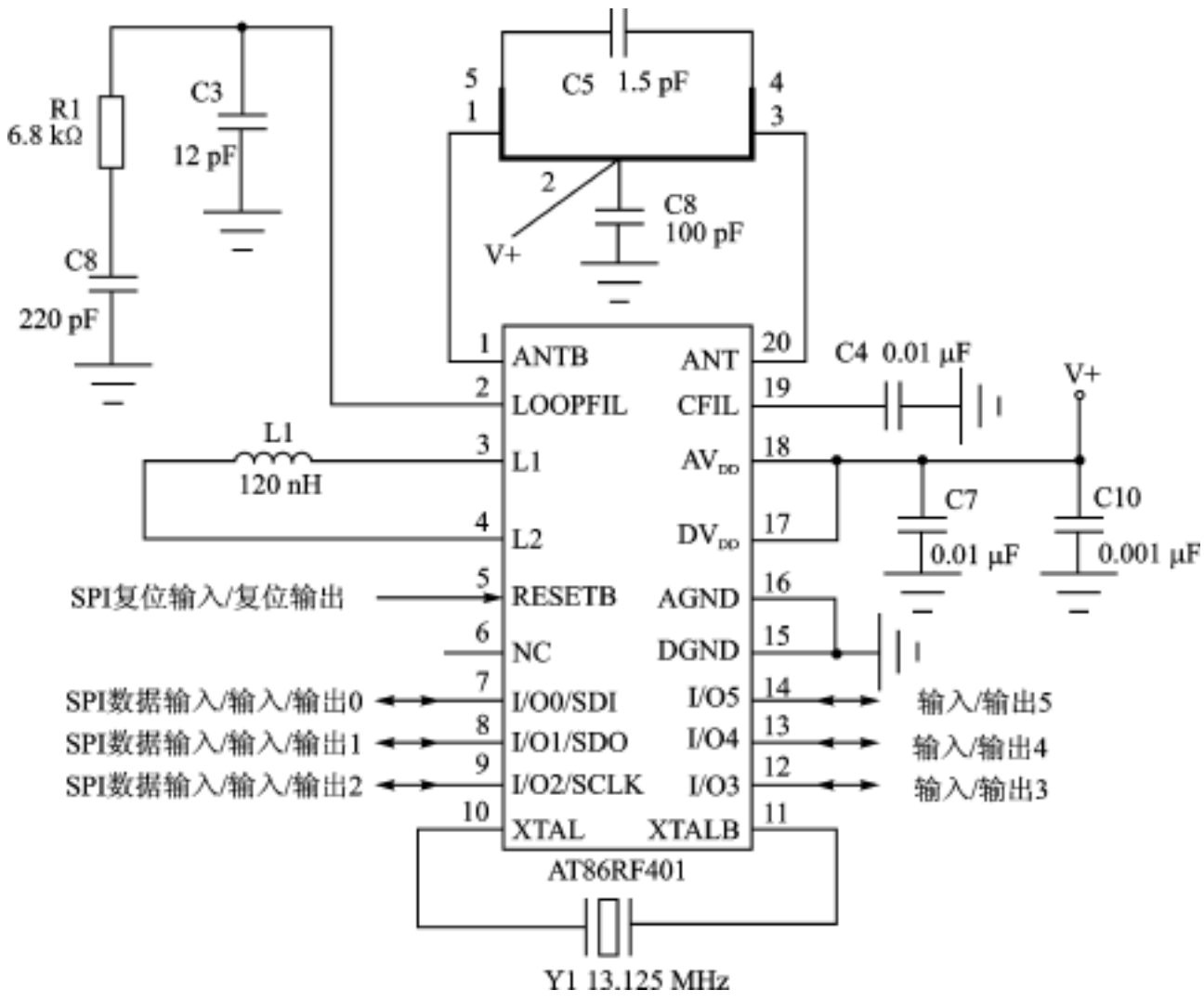


图4.16-4 AT86RF401典型应用电路

选自《单片机与嵌入式系统应用》月刊,2001年第11期

第五章

新器件及其

应用技术

5.1 一种全新结构的微控制器——Triscend E5

长沙国防科技大学 ATR 实验室(410073) 吴 东 卢焕章 沈振康

一、前 言

近年来,随着以 FPGA 为代表的可重构逻辑器件在价格、速度和逻辑容量等指标方面的不断提高,人们对可重构逻辑器件应用的认识也不断加深。由过去的简单地把 FPGA 作为 ASIC 的替代品到把 FPGA 作为一种动态实现逻辑功能的计算平台,并且由此发展出了一门新兴的技术——可重构计算技术(reconfigurable computing)。可重构计算在计算性能方面优于依赖微处理器的纯软件方式,在灵活性和通用性方面又优于依赖硬连线的 ASIC 纯硬件方式,在某些计算密集型的应用领域中具有极大的潜力。但是由于可重构计算器件结构的特点决定了它在处理具有复杂控制流程的应用方面能力不足。

因此如何将通用微处理器、ASIC 器件和可重构逻辑器件结合,取三家之长,补各自之短成为计算机体系结构研究领域的热点。今年年初,一家位于硅谷的美国公司 Triscend 推出了世界上第一种商用的将微控制器、ASIC 和可重构逻辑阵列集成在单一芯片上的器件 E5,在这个方向上作出了勇敢的尝试。

下面我们将分别对 E5 的体系结构、主要特点,对可重构逻辑阵列的支持和开发系统设计流程作详细介绍。

二、E5 的主要特点

Triscend E5 的主要特点有:

(1) 它是工业界第一个完整的可重构片上系统(CSoC);

(2) 它的主控制器是一个增强型的 8032 微控制器,与工业界标准的 8032/ 8051 微控制器在指令集上兼容;

(3) 它包含一个嵌入式可重构系统逻辑矩阵(CSL);

(4) 拥有一个高性能的内部系统总线(CSI),连接微控制器、系统存储器和可重构系统逻辑矩阵;

(5) 拥有最多有 315 个用户可编程 IO 端口,用作微控制器、片内专用外设和可重构逻辑矩阵与外部电路的接口;

(6) 增加了一个存储器接口单元(MIU),负责连接外部的存储器,也可以作为访问外部设备的接口。

三、E5 的体系结构

E5 的体系结构框图如图 5.1-1 所示。从图中我们可以看到 E5 的主要部件是 8032 增强

型微控制器、可重构系统逻辑矩阵、内部系统总线、在片 RAM 和可编程 I/O 端口以及一些专用外设部件。

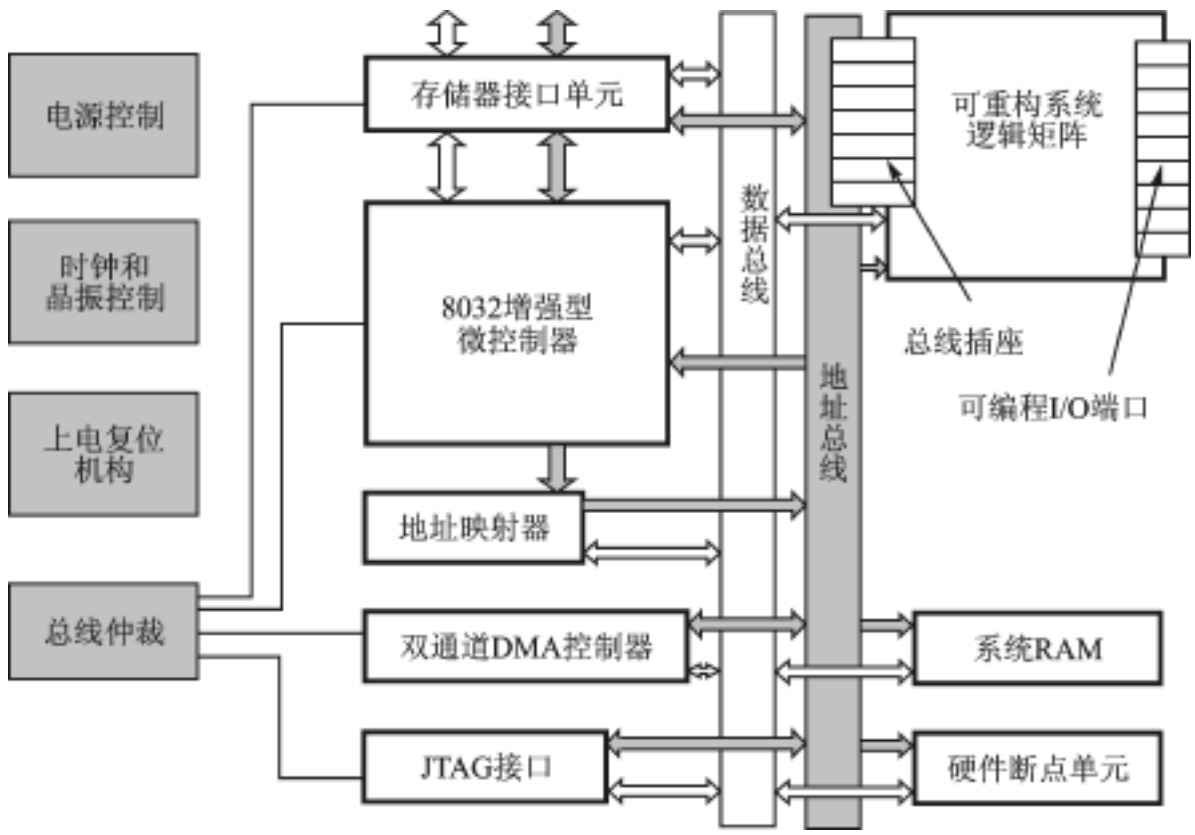


图 5.1-1 E5 的体系结构框图

1. E5 增强的 8032 微控制器核

E5 的核心部件是一个增强的 8032 微控制器,它与业界标准 8032/8051 微控制器在指令集上兼容,使得以前为 8032/8051 编写的软件不经修改就可以在 E5 上执行,同时还在许多方面有所增强,比如,E5 中的 UART 是原来 UART 的超集;E5 包含 3 个 16 位的时钟和一个看门狗时钟;在中断方面,E5 中断源和中断矢量的个数也比原来有所增加;另外原来 8032 有 4 个 8 位 I/O 端口,而 E5 中的微控制器实际上可以拥有多达 315 个 I/O 端口。

2. E5 中的可重构逻辑阵列

可重构逻辑矩阵(CSL)是 E5 的特色。它作为一个虚拟的外部设备,最多包含 3200 个可重构逻辑胞元,相当于 40 000 逻辑门,可重构逻辑阵列可以根据用户的需要,实现逻辑、算术、存储、时序等多种外设的功能。用户可以自己利用开发工具设计这些“虚拟外设”,也可以调用 Triscend 公司提供的“软模块”库中的电路。在保证性能的同时大大增加了 E5 的通用性,使得 E5 可以适应各种嵌入式应用环境。

3. E5 的可重构系统互联总线

E5 中的可重构系统互联总线,简称 CSI,在设计时独立于 8032 微控制器,可以适应今后 Triscend 产品的升级,其中的 Socket 部分负责与可重构逻辑矩阵接口。有了 Socket,就给可重构逻辑矩阵中的“软模块”提供了一个统一的与外界接口的界面,一方面降低了可重构逻辑矩阵中“软模块”的复杂度,另一方面也提高了系统的可靠性,同时也节省了宝贵的可重构硬件资源。CSI Socket 由 8 位读数据线、8 位写数据线、32 位地址线、一组片选信号线、DMA 应答信号、总线时钟信号、等待状态和硬件断点的控制和检测信号线组成。

CSI 有一个总线仲裁结构,支持多个总线主设备。微控制器、DMA 控制器、JTAG 和 MIU 都可以作为 CSI 的主设备。CSL 通过 DMA 控制器作代理也可以控制 CSI 总线。

4 . E5 的 DMA 控制器

E5 的 DMA 有两路独立的通道和一组分布式的 DMA 控制寄存器,负责存储器与外设之间的数据传输。每个外设(包括 CSL)与一个 DMA 控制寄存器相连,通过 DMA 控制寄存器的请求,应答信号与 DMA 通道联系。由于在可重构逻辑矩阵中实现一套总线控制逻辑十分复杂,又要耗费大量宝贵的可重构逻辑资源,因此设计者安排 DMA 控制器作为 CSL 的代理,由 CSL 借用 DMA 控制器中的控制逻辑来实现对总线的控制,使得 CSL 可以间接地作为总线的主设备。这样的用 ASIC 电路来完成固定功能,节省可重构逻辑资源的设计思路,我们在 CSI 的 Socket 设计中已经见到过。

5 . E5 的存储器接口单元

存储器接口单元(MIU)也是原来 8032 中没有的部分,它提供了一个 E5 芯片和外部存储器接口的单元。原来 8032 中需要占用两个 I/ O 端口,作为与外部存储器的接口,并且地址信号和数据信号分时复用这两个 I/ O 端口,而在 E5 中 MIU 提供了专门的独立数据和地址信号线。另一方面,MIU 也可以作为 E5 芯片内部的外设访问外部功能单元的接口。

6 . 可编程 I/ O

E5 的可编程 I/ O(PIO)最多可达 315 个,分别可以配置为输入、输出和双向数据传输。PIO 可以作为可重构逻辑矩阵的输入输出单元,也可以通过 CSI 被微控制器和其他外设使用。与原来 8032 的仅有 4 个 I/ O 端口不同,E5 中的任何 PIO 都可以通过 CSI 被微控制器使用,实际上,E5 中的微控制器可以拥有所有的 I/ O 端口,这对嵌入式应用来说是非常有利的。

四、E5 的开发设计流程

Triscend 公司为 E5 提供了开发系统 FastChip, FastChip 针对不同的应用目的支持两种开发设计流程。不论何种设计流程,都可分为处理器配置、程序开发和调试验证三个阶段。

第一种流程称为“ Derivation On Demand”,在处理器配置阶段,它利用 Triscend 提供的可重构逻辑库“ Soft Module”来满足应用需求。Triscend 公司为 E5 的开发提供了“ Soft Module”库,包括时钟/ 计数器、可编程 I/ O、串行通信(SP1, I2C, UART)、脉宽调制(PWM)、存储块等等。在程序开发阶段,由于 E5 中的微控制器核与 8032/ 8051 指令集兼容,因此在软件开发的工作中,原来的代码都可以继续使用,也可以使用标准的第三方软件开发工具。在调试验证阶段,可以采用工业标准的调试工具来进行逻辑调试和实时在线调试。这种开发流程的特点是可以最大限度地加快开发速度,提早上市时间。

另一种流程称为“ System On Chip”,它的特点是尽可能对系统的各部分进行精心定制,使得系统的性能最优。这种开发流程与“ Derivation On Demand”的区别在处理器配置阶段,在这一阶段设计师不使用系统提供的软模块库,而是自己利用第三方标准的 EDA 工具设计软模块,设计过程与 ASIC 设计相似,在经过验证之后,就可以加载到 E5 的 CSL 矩阵中。其他设计阶段与第一种流程相同。

五、结 论

E5作为一种全新结构的微控制器,利用了可重构计算技术来提高系统的性能和灵活性。加上处理器核与工业标准兼容,为嵌入式应用领域提供了一种强大的工具。其设计思想和体系结构必然会对今后的微控制器设计产生深远的影响。

参 考 文 献

1 riscend E5 Configurable System—on—chip family . Triscend Corp, 1999

2 Barr, Michael . A Reconfigurable Computing Primer . Multimedia Systems Design, 1998: 44～47

选自《微处理机》季刊,2000年第2期

5.2 PSD8XXF 的在系统编程技术

空军雷达学院科研处(430010) 颜荣江

采用模块化设计技术,由 WSI 公司研制的可编程微控制器系统外围器件 PSD 系列芯片,将构成一个单片机应用系统所需的多个外围功能块,如 EPROM、SRAM、I/O、PLD 等,集成在单一硅片上,为简化嵌入式应用系统的设计、缩短产品的开发周期、提高系统的可靠性、降低系统的成本、缩小产品尺寸提供了一条便利的捷径。PSD 系列器件的可编程特性及其潜在的优点已逐步为广大的产品设计和应用工程师所理解和掌握,并将其应用在各自的产品设计中,从 PSD2XX、PSD3XX、PSD4XX、PSD5XX、PSD6XX、PSD7XX 各系列产品中,用户可根据实际电路工作的需要,选择合适的 PSD 芯片来完成各自的功能。为进一步方便工程师的研发及产品的更新、升级,WSI 最近又推出了新一代在系统可编程的微控制器外围器件:PSD8XXF 系列芯片。该系列产品具有完整的在系统可编程特性,从而可使嵌入式系统的设计更加简单、灵活。

PSD8XXF 除提供与各种微控制器直接连接的总线接口逻辑外,片内还集成了优化的"宏单元"逻辑阵列块,它包括输入宏单元和输出宏单元。这些宏单元,既可传递组合逻辑信号,又可在时钟同步下寄存输出,以完成一定的时序要求。其中,输入宏单元可将来自引脚的输入信号经锁存、寄存后或直接传至内部 PLD 阵列的输入总线。一个典型的应用就是分离地址/数据复用总线上的地址信息,或在两个 CPU 通过共享存储体传递数据时提供握手通信信号。而输出宏单元既可作为一个内部节点,通过器件内部的反馈网络将宏单元的输出信号反馈至 PLD 的输入总线,也可直接通过引脚输出,一个典型的应用是可利用这些输出宏单元构成一个定时器/计数器,或实现状态机逻辑功能。以 PSD813F1 为例,单个芯片内集成有:

· 64 B 的 E^2 PROM(OTP)存储器,由用户存入特殊的文档信息;

· 128 KB 的 Flash 存储器,分成 8 个大小相同的块,每个块均可被设定为程序空间或数据空间,以分别用来存放程序或数据;

· 32 KB 的 E^2 PROM 存储器,分成 4 个大小相同的块,每个块均可被设定为程序空间或数据空间,以分别用来存放程序或数据;

· 2 KB 的 SRAM 存储器,掉电时可自动切换到备用电源供电,以保持数据;

· 24 个输入宏单元和 16 个输出宏单元,可有效地实现多种逻辑功能,包括状态机和地址译码器;

· 27 个独立可配置输入/输出引脚,可用作标准 MCU I/O,PLD I/O,锁存地址输出及特殊功能 I/O;

· 8 个页面寄存器,使微控制器的寻址空间扩展 256 倍;

· 内置的 JTAG 从属串口,可在现场对空白器件进行在系统编程或对已编程器件进行重新编程;

· 10 万次的擦除/写入,10 年的数据保存期。

零功耗 CMOS 技术,可编程的电源管理单元,支持两种独立的低功耗工作方式,+ 5 V 时,待机状态的电流典型值为 10 μA,且芯片可自动检测微控制器活动的停止,并使 PSD 进入低功耗方式。

该系列芯片为真正的现场可编程器件,芯片的 PSD 配置、PLD 阵列、E^2 PROM 存储器和 Flash 存储器均能在系统中编程,通过 WSI 公司提供的 PSDsoft 软件包,可生成各种编程器所需的目标文件,包括标准的 BSDL 和 SVF 文件格式。由 WSI 公司自身提供的编程器有三种,分别是:MagicPro Ⅲ、PSDPro 和 FlashLink。其中 FlashLink 为低成本的编程适配器,其一端与 PC 机的并行打印口相连,另一端则是标准 14 引脚的 JTAG 带状连接器,可直接与目标系统板上的 JTAG 接口相连,通过 JTAG 信号线对 PSD8XXF 器件进行在系统编程,从而省去专用的器件编程器。当电路设计中不使用 JTAG 信号对器件进行编程时,可使用另外两种编程器完成对器件的编程任务。此时对应的引脚可用作其他目的。

由于 PSD8XXF 片内集成有多个功能块,受篇幅限制,本文只着重介绍与 JTAG 接口有关的内容。

一、JTAG 技术规范

PSD8XXF 遵守 IEEE1149 .1 JTAG 技术规范的基本要求,但不支持该技术规范内定义的边界扫描功能,而支持 JTAG 接口的在系统配置(ISC)规范,并允许 PSD8XXF 与其他工作于边界扫描方式的器件一起存在于 JTAG 链中,此时,PSD8XXF 支持 BYPASS 命令。

IEEE JTAG－ISC 规范仍在制订中,尽管如此,许多应用系统均采用通用可接受数据文件格式来反映最近的 ISP 规范。同样,在 PSD 编程环境,需要两个文件来定义 ISC 功能,一个文件是边界扫描定义语言(BSDL)文件,它用来定义被编程器件的引脚和内部寄存器(JTAG 编程方式);另一个文件是串行矢量格式(SVF)文件,它用来定义所产生的动作。这两个文件均可由 PSDsoft 软件包产生(关于 JTAG 和文件格式的详细信息,请参见德州仪器网址 www .ti .com 搜索 JTAG 和 PRIMER)。

标准 JTAG 采用 4 个基本信号:TCK、TMS、TDI、TDO。其中,TCK 为同步时钟,TDI 和 TDO 分别为串行数据输入和串行数据输出,TMS 为 JTAG 方式选择。WSI 还提供增强的 JTAG 引脚信号,即在上述 4 个标准信号的基础上,又附加两个信号线:$\overline{\text{TSTAT}}$和$\overline{\text{TERR}}$。其中,$\overline{\text{TSTAT}}$代表正在进行的当前动作的瞬时状态,TERR 指示对字节或扇区的编程/ 擦除是否超时。到目前为止,还没有一个标准的 JTAG 引脚信号分配方案,由 WSI 公司定义的连接器引脚信号如图 5 .2－1 所示。这里,除上述已介绍的 6 个信号外,附加的信号有:$\overline{\text{JEN}}$提供低电平有效的使能输入信号,以便使 PSD8XXF 的 JTAG 功能在端口 C 引脚上有效;$\overline{\text{RST}}$,复位目标系统;$\overline{\text{TRST}}$,复位目标板上的 JTAG;CNTL,由用户定义的信号。

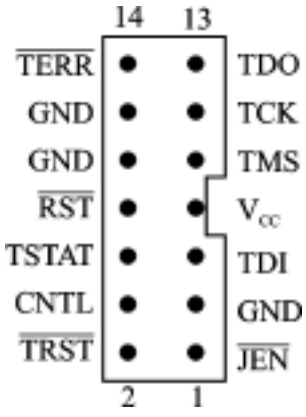


图 5 .2－1　JTAG 连接器引脚信号

二、编程/ 擦除流程控制

有三种方法控制 JTAG 器件编程器和 PSD 器件之间的数据传输,其中 option2 和 option3

是 WSI 增强型 ISP 特性的选择。

option1:无流程控制(使用标准的 4 个 JTAG 引脚)。通常不予推荐。它涉及并考虑每次编程或擦除动作的最差情况下的最长时间。

option2:软件流程控制(使用标准的 4 个 JTAG 引脚)。用软件方法扫描硬件信号 TSTAT 和 TERR,使用串行方式传输信息,不必等待每次编程/擦除动作的最长时间,此方法较 option1 有较大改进。

option3:硬件流程控制(使用增强的 6 个 JTAG 引脚)。对于每次编程/擦除动作,器件编程器将直接轮询 TSTAT 和$\overline{\text{TERR}}$信号,相对 option2 有可观的改进。

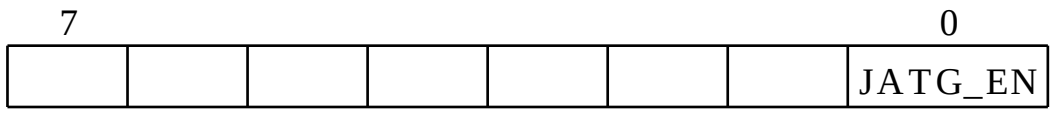
三、JTAG 信号引脚的使能控制

在 PSD8XXF 器件内部,所有与 JTAG 接口有关的信号均可从 PSD8XXF 的端口 C 引脚上输出。由于端口 C 是一个多功能口,当 JTAG 编程完毕后,可将端口 C 用作其他目的。为了允许这种情况的出现,在 PSD8XXF 器件内部,提供了三个不同的条件来使能标准的 JTAG 信号。这三个条件在逻辑上组织成“或”的关系。即只要其中任何一个条件为真,均使能 JTAG 信号连至端口 C,其逻辑表达式如下:

JTAG_ON = ! Jtag_en + Jtag_FF + Jtag_PT

(1) Jtag_en 由 PSDsoft 软件包中→PSD configuration (或 Device config)→JTAG configura‾tion→JTAG Functions 中的第一个选择框 EnableTMS/TCK/TDI/TDO on PC0/PC1/PC5/PC6 确定。允许时,Jtag_en = 0,未选择时,Jtag_en = 1。当器件空白时,Jtag_en = 0,即 JTAG 在端口 C 上被使能,以允许 FlashLink 对器件进行首次编程,一旦器件被编程且 JTAG 功能被禁止,则无法通过 Flash Link 对已编程器件中的内容进行第二次更新。在此情况下,只能通过 WSI 的专用编程器对其进行后续的擦除/编程操作。

(2) Jtag_FF 为 PSD 器件内部的 JTAG 使能寄存器的位 0,其地址为 CSIOP + C4H,其配置如下:



JTAG 使能寄存器(地址为 CSIOP + C4H)

在该寄存器中,位 1～7 未用,应置为 0,位 0 的定义如下:

JTAG_EN = 1 JTAG 端口被使能 即 Jtag_FF = 1

JTAG_EN = 0 JTAG 端口被禁止 即 Jtag_FF = 0

(3) Jtag_PT 是在 PSDabel 内定义的 JTAG 使能乘积项,该乘积项由 PSD 器件内部的一个节点输出,并由 PSDabel 提供一个保留名 jtag sel。

四、专用 JTAG 引脚功能

将 PSD8XXF 的端口 C 专用于 JTAG 功能的基本连接方法如图 5 2‾2 所示。

对于 option1 和 option2,可不连接 TSTAT 和$\overline{\text{TERR}}$,此时 PSD8XXF 的 PC3、PC4 可用于其他功能。这里假定 PC 口被设定为 CMOS 输出,若设置为开路输出,应分别在每一个信号的引出端连接 100 kΩ 的上拉电阻。为了减少与电缆有关的尖峰或振铃造成的虚假 TCK

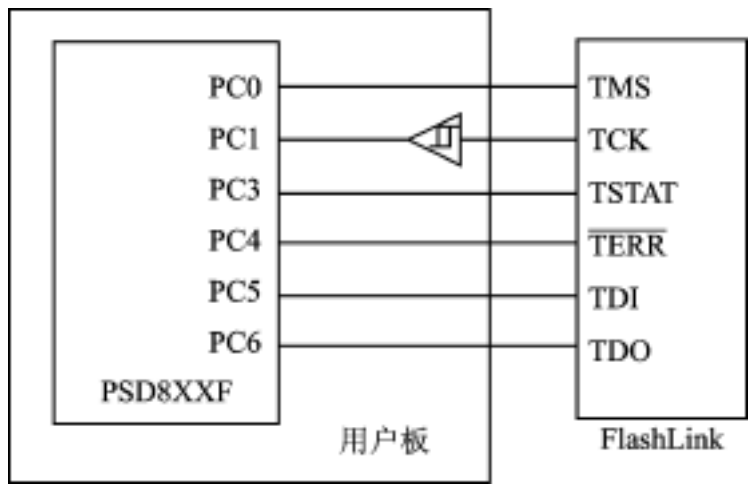


图5.2-2 端口C专用于JTAG功能与Flash编程器的连接

信号,推荐在TCK的输入端使用施密特触发器进行缓冲。

专用于JTAG引脚功能对芯片的JTAG配置如下:

(1) 在PSDabel中,书写方程:jtag sel = 0,即无需任何乘积项。

(2) 在PSD configuration中,选中Enable TMS/ TCK/ TDI/ TDO on PC0/ PC1/ PC5/ PC6以保证JTAG信号在端口C上始终有效。

五、复用JTAG引脚功能

仅当使用FlashLink编程器通过JTAG连线编程PSD8XXF时,片内的JTAG信号才被连至端口C相应的引脚上,而当编程结束后可将端口C用于其他目的。这样,可使同一端口用于多种目的。而IEEE1149.1规范并不支持JTAG引脚与其他功能多路复用,这是WSI的PSD8XXF器件特有的功能。为保证该功能的正确实现,可利用FlashLink编程适配器上提供的JEN信号,该信号为低电平有效,并将其作为PSD8XXF内部PLD阵列的一个逻辑输入,参与JTAG乘积项的控制,其基本连接方法如图5.2-3所示(这里假设$\overline{\text{JEN}}$被连至PC7)。

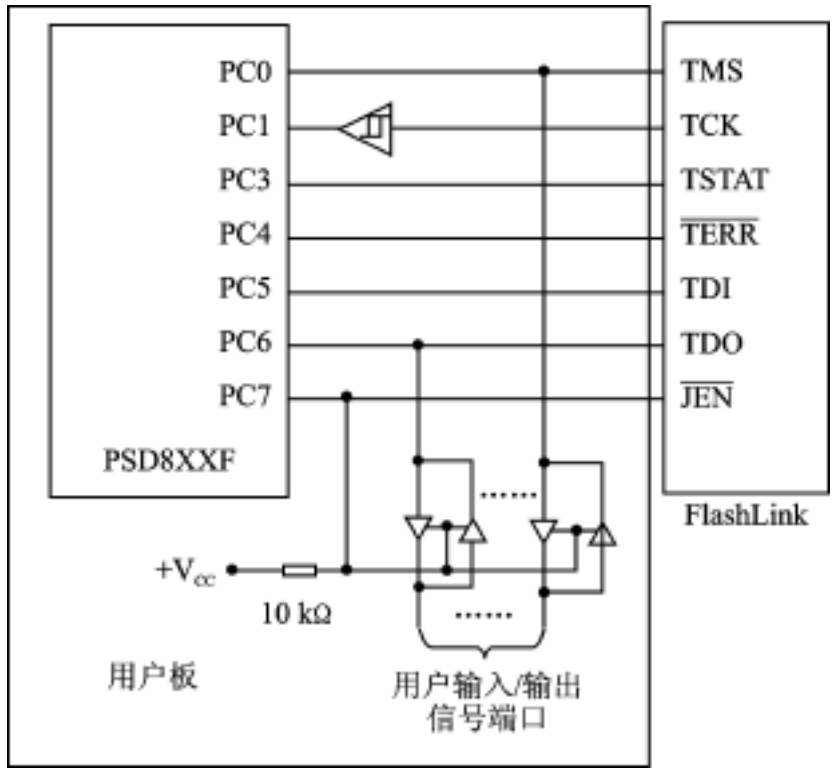


图5.2-3 端口C复用JTAG功能与FlashLink编程适配器的连接

对于 option 1 和 option 2,开路输出及 TCK 施密特触发器的缓冲的说明与专用 JTAG 引脚部分的说明相同。特别需要注意的是,要确保在编程期间内,用户电路组件上的专用 I/O 信号与 JTAG 信号不发生冲突。即在 JTAG 工作期间没有用户指定的逻辑电平驱动 JTAG 信号或没有 JTAG 信号以有害的方式驱动用户指定的逻辑。这里是在用户 I/O 与 PSD8XXF 端口 C 之间插入双向三态缓冲器,当 JEN 信号有效时,使缓冲器处于高阻态,从而阻断用户 I/O端口与 PC 口 JTAG 信号之间的联系。用户可根据实际电路工作的需要,选择合适的三态缓冲器,以确保得到正确的逻辑状态。

该方案对芯片的 JTAG 配置如下:

(1) 在 PSDabel 中,书写方式为

JEN pin 11 ;这里指定 PC7 用于 JEN 输入,也可将 JEN 连至其他引脚

jtag sel = !JEN ;仅当 JEN 信号有效时,才将芯片内部 JTAG 功能连至端口 C 相应引脚

(2) 在 PSD configuration 中,禁止 Enable TMS/ TCK/ TDI/ TDO on PC0/ PC1/ PC5/ PC6,以便于端口 C 在 JTAG 无效时可用于其他目的。

六、使用 JTAG 接口在线编程

WSI 的 PSDsoft 软件包支持在系统编程,只需将 FlashLink 编程适配器连接在 PC 机并口与用户 JTAG 接口之间,即可进行实时在系统编程,其物理连接如图 5.2-4 所示。

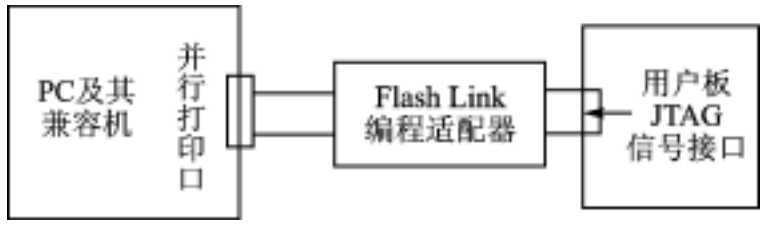


图 5.2-4 FlashLink 编程适配器与 PC 机和用户板的连接

为了实现在系统编程,在启动 PSDsoft 软件后,选择 JTAG Programming 进入 JTAG chain setup 对话窗,在此窗中可完成 PSD8XXF 器件的在系统编程操作。其操作步骤如下:

(1) 在 chain Information 框的 File Name 栏下,输入目标文件名(其文件扩展名为 .obj),也可点击 Browse 按钮框,以浏览并选取相关的文件,然后打开该文件。如选中 JTAG_TEST .obj 文件名,并打开。

(2) 在 chain Information 框的 Device Name 栏下,选定器件型号,如 PSD813F1。

(3) 点击 Add 按钮,则该芯片的 JTAG 链路信息被加入,并在 Operation 栏下显示 By-Pass,表示该链路目前正处于“旁路”状态。

(4) 将鼠标指向刚出现的行上,右击以选择属性配置 Properties,此时弹出 JTAG chain setup Properties 对话窗。

(5) 点击 Set Pin/ Flow Control,选择编程/擦除的流程控制及端口 A、B、C、D 上电时的状态,一般可选 option3 或 option2 流程控制。对 JTAG Attributes 和 user code 栏可暂不设置,采用当前的内定值即可,并返回至链路设置对话窗。

(6) 同步骤(4),右击,此时选 program,然后按 Go 按钮,即可开始在线编程。

(7) 点击 Create SVF 按钮,保存当前的配置信息至扩展名为 .JCF 的文件,以便下次调用时通过 JTAG chain File 显示框下的 File Name 栏,将所配置的信息调入使用。

PSD8XXF 系列芯片为真正的在系统可编程单片机外围芯片,包括 MCU 接口、flash 存储器和 E^2 PROM 存储器,均可通过 JTAG 进行重新编程。另外,还可由 MCU 控制器来编程 PSD8XXF 内的 Flash 和 E^2 PROM 存储器,利用在一个块内执行程序,编程/ 擦除另一个块的方法,也可完成对整个片内存储器内容的更新。Flash 存储器的 8 个块和 E^2 PROM 或第二 flash 存储器(取决于所使用器件的型号)的 4 个块均可分别被擦除和编程,从而为用户的应用设计带来了很大的灵活性。

参 考 文 献

1 WSI PSD8XXF 系列数据手册 . 武汉力源电子股份有限公司,1999 .2

选自《电子技术应用》月刊,2000 年第 1 期

5.3　PSD813F1 及其接口编程技术

西安交通大学电信学院(710049)　胡国鹏　武自芳

由 WSI 公司研制的新一代可编程微控制器系统外围器件 PSD813F1 芯片,具备完整的在系统可编程(in system programmable)特性。其在系统可编程不仅仅是对程序存储器而言,而是指整个芯片内的所有功能块的配置,而且可对器件的逻辑和功能进行随时组态或重组。该芯片采用模块化设计技术,可集成一个单片机应用系统所需的多个外围模块,如将 FLASH-MEM、EEPROM、SRAM、PLD 等集成到单一硅片上,为简化微控制器应用系统的设计、缩短产品的开发周期、提高系统的可靠性、降低系统的成本、缩小产品尺寸、增强系统保密性提供了一条捷径。

一、PSD813F1 芯片的性能特点

PSD813F1 的内部结构如图 5.3-1 所示。

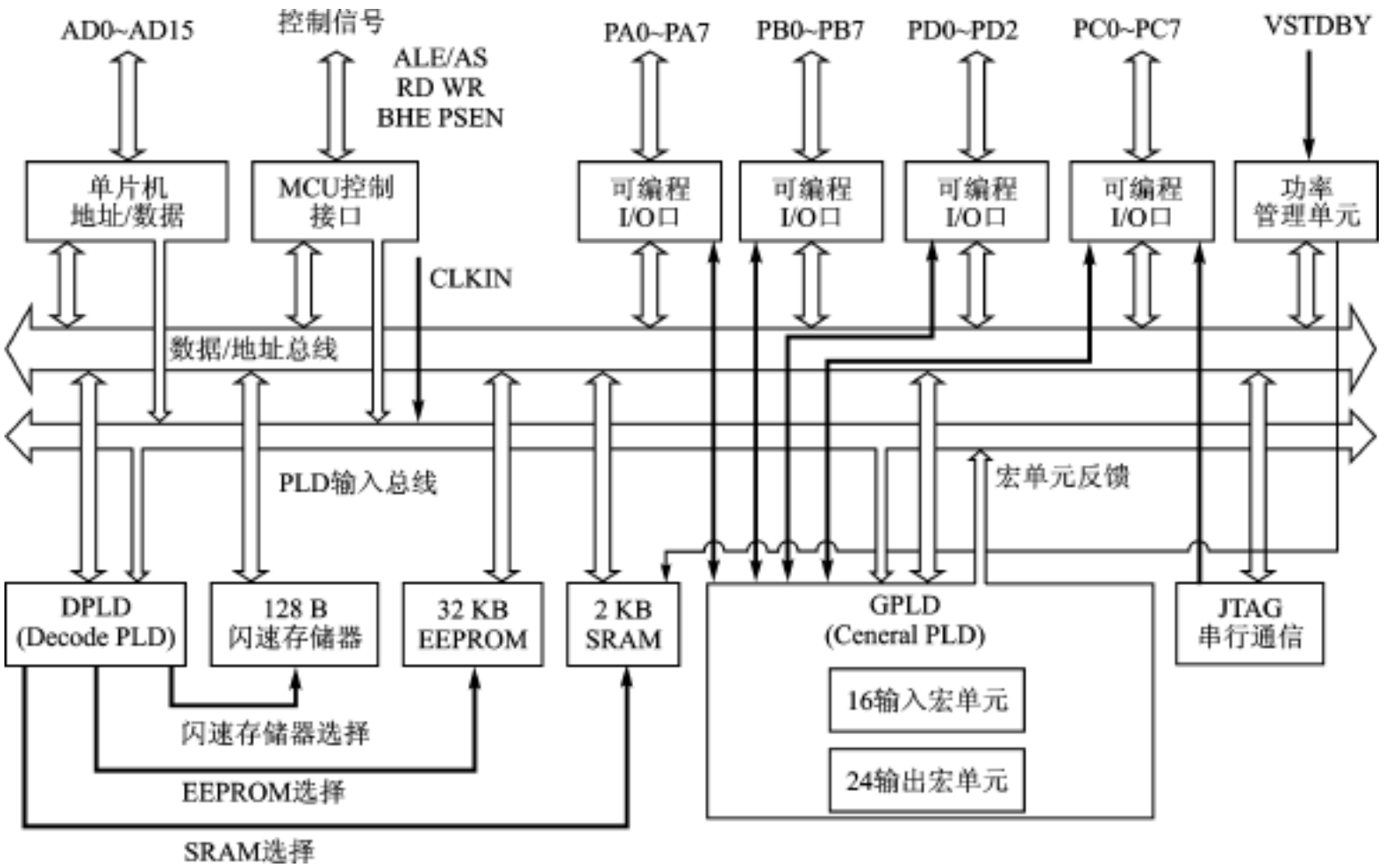


图 5.3-1　PSD813F1 内部结构图

PSD813F1 内部集成了可分区段保护的 128 KB FLASH MEM(闪速存储器)和 32 KB EEPROM,以及掉电时通过切换到备用电源以保持数据的 2 KB SRAM。每一个存储器块可以被用户通过配置定位在不同地址空间。FLASH MEM 分成 8 个大小相同的块,EEPROM

分成 4 个大小相同的块,以上各块均可被设定成程序空间或数据空间;2 KB 的 SRAM 存储器可完成数据缓存的功能。

PSD813F1 具有可与微控制器相匹配的总线接口逻辑,片内还集成了优化的"宏单元"逻辑结构,包括 24 个输入宏单元和 16 个输出宏单元。这些宏单元,既可作为内部 PLD 阵列的一个组合逻辑输入信号,又可用来完成寄存输出,以完成一定的时序要求。其中输入宏单元可将来自引脚的输入信号经锁存、寄存后或直接传至内部 PLD 阵列的输入总线。而输出宏单元既可作为一个内部节点反馈至 PLD 的输入总线,也可直接通过引脚输出。

芯片包括 4 个可编程 I/O 端口,共 27 个允许独立配置的端口引脚,端口 A、B 和 C 为 8 位,端口 D 为 3 位,端口可设置为 MCU I/O、PLD I/O、外设 I/O、地址输入输出、数据端口等不同的配置,而且其中的 16 个 I/O 引脚可以设定为 CMOS 电平或漏极开路工作方式。这些端口的工作方式由数据输入、数据输出、方向、控制、驱动选择等寄存器决定,这些寄存器位于以 CSIOP 为基地址的 256 字节空间里,CSIOP 基地址由配置软件决定,不同寄存器具有惟一的相对于 CSIOP 基地址的偏移地址,对寄存器的操作可在程序中完成。

PSD813F1 的译码逻辑由内部 FLASH PLD(包括 FDPLD 和 FGPLD)实现,具体结构由与阵列组成。FDPLD 为内部功能部件提供地址译码,例如内部的 FLASH MEM、EEPROM、SRAM、寄存器,以及 I/O 端口的选择。FGPLD 用于实现系统逻辑,如状态机功能块和组合逻辑的实现。

PSD813F1 的加密技术为系统的加密提供了便捷的实现方法,当 PSD 配置寄存器中的保密位被置位时,不允许在器件编程器或通过 JTAG 端口读器件,当使用 JTAG 端口时,只有全芯片擦除命令允许,其他命令被阻断。

二、JTAG 编程

PSD813F1 为真正的现场可编程器件,包括芯片的 PSD 配置、PLD 阵列、EEPROM 和 FLASH MEM 均能在系统中编程。

基于 PSD 技术的硬件设计方法是采用硬件描述语言来设计复杂的数字逻辑系统,生成符合 PSD 芯片要求、在电路上可行的数字逻辑,通过 WSI 公司提供的 PSDSOFT 软件包,可生成编程器所需的目标文件(包括程序及系统配置文件),经 JTAG 接口可下载到芯片中。

端口 C 是可复用的端口,可配置成 JTAG 编程端口,在现场对空白器件进行在系统编程或对已编程器件进行重新编程,而无需使用微控制器。另外,也可以通过微控制器执行来自 EEPROM 的编程算法,FLASH MEM 也可在系统内被编程。通过执行来自 FLASH MEM 的算法,EEPROM 也可用同样的方法进行编程。PLD 逻辑或其他 PSD813F1 配置同样也可通过 JTAG 口或器件编程器编程。

PSD813F1 遵守 IEEE1149.1 JTAG(Joint Test Action Group 联合测试行动小组)技术规范的基本内容,支持 JTAT 接口的 ISC(In System Configurition 在系统配置)规范,允许 PSD813F1 与其他工作于边界扫描方式的器件一起存在于 JTAG 链中,但不支持该技术规范内定义的边界扫描功能。

标准 JTAG 采用 4 个基本信号:TCK, TMS, TDI, TDO。其中,TCK 为同步时钟,TMS 为 JTAG 方式选择,TDI、TDO 分别为串行数据输入和串行数据输出。WSI 公司的 JTAG 接口除以上 4 个基本信号外,还附加提供了用于增强功能的 TSTAT 和/TERR 信号,其中

TSTAT 代表正在进行的当前动作的瞬时状态,/ TERR 指示对字节或扇区的编程/ 擦除是否超时。此外的附加信号为:/ JEN 提供低电平有效的使能输入信号,以便使 PSD813F1 的 JTAG 功能在端口 C 引脚上有效;/ RST 复位目标系统,/ TRST 复位目标板上的 JTAG, CNTL 是由用户定义的信号。

三、PSD813F1 在数据采集系统中的应用

PSD813F1 可与 8 位或 16 位 MCU 构成一应用系统,本文在此介绍了采用 PSD813F1 与 80C196KC[2] 构成的现场数据采集电路。

1 . 硬件结构

系统硬件结构如图 5 .3 - 2 所示,核心器件为 80C196KC 和 PSD813F1,图中简化了复位等电路。

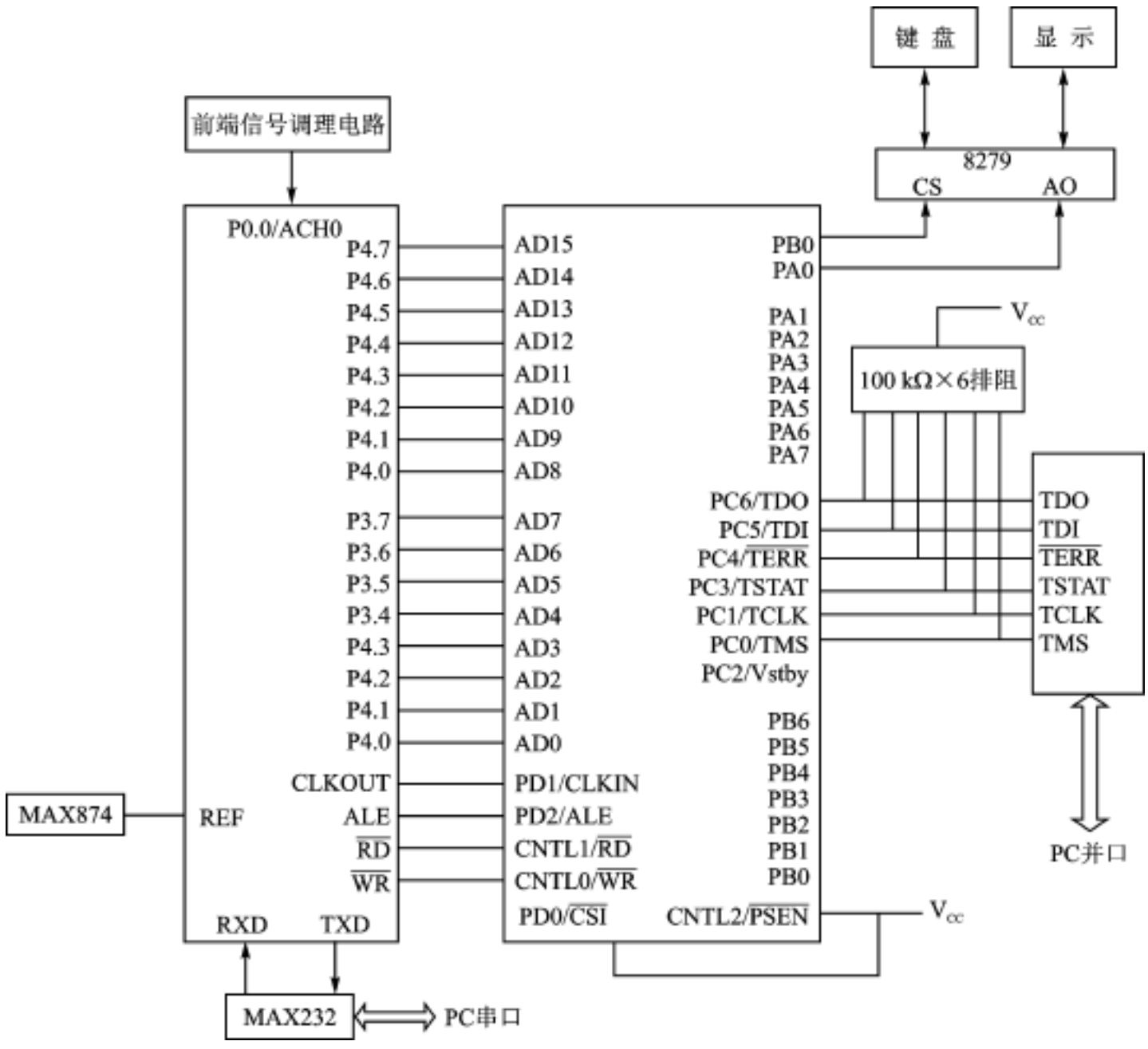


图 5 .3 - 2 PSD813F1 与 80C196KC 的接口图

80C196KC 为 INTEL 公司 16 位单片机。该单片机具有 16 位多路复用的地址/ 数据总线,工作于 12 MHz 的时钟频率,系统主体程序采用 C96 语言设计,程序空间大小占用 32 KB,系统设计要求采用非易失性存储器进行数据存储,另外需外扩 SRAM 用于数据缓存。程序存储、数据存储和数据缓存分别由 PSD813F1 的 FLASH MEM、EEPROM 和 SRAM 完成。

A/D 转换采用 80C196KC 自身的 10 位 8 路逐次比较型 A/D,键盘/显示接口芯片采用 8279,与 PC 机通信采用 MAXIM 公司的 MAX232 用于 RS-232 通信,由 MAX874 经精密调整后给 80C196KC 提供 4.096 V 电压。8279 的片选和地址译码由 PSD813F1 的 FGPLD 完成。

2.软件设计

(1) PSD813F1 配置文件设计

该电路在 PSDSOFT(WSI 公司提供的下载软件)中的配置为:

· 多路复用工作方式;

· ALE 高电平有效;

· 读/写控制选择 RD/WR。

PSD813F1 以 ABEL 语言为硬件描述语言,在本电路中的设计方法为:

① 内部译码实现:

```
fs0 = !a15 * !a14 * !a13 * !a12;          * s 为设计中 FLASH MEM 的保留名,fs0 地址空间为 0000h～0fffh */
fs1 = !a15 * !a14 * !a13 * a12;           /* fs1 地址空间为 1000h～1fffh */
fs2 = !a15 * !a14 * a13 * !a12;           /* fs2 地址空间为 2000h～2fffh */
fs3 = !a15 * !a14 * a13 * a12;            /* fs3 地址空间为 3000h～3fffh */
ees0 = !a15 * a14 * !a13 * a12;           /* es 为 EEPROM 的保留名,ees0 地址空间为 4000h～4fffh */
ees1 = !a15 * a14 * a13 * !a12;           /* ees1 地址空间为 5000h～5fffh */
ees2 = !a15 * a14 * a13 * !a12;           /* ees2 地址空间为 6000h～6fffh */
ees3 = !a15 * a14 * a13 * a12;            /* ees3 地址空间为 7000h～7fffh */
rs0 = a15 * !a14 * !a13 * !a12 * !a11 * !a10;  /* s 为 SRAM 的保留名,地址空间为 8000h～8fffh */
csiop = a15 * !a14 * !a13 * a12;          /* csiop 基地址为 9000h */
```

② 外部译码的实现:

将 PB 的最低位端口设计为片选信号的设计方法为

```
cs1 pin7;                                 /* PB 的 0 端口 */
cs1 = !(a15 * a14 */ a13 * a12);          /* 8279 的地址空间为 d000h～dfffh */
```

(2) 程序设计

程序设计语言为 C96,其实现如下:

对 SRAM 的操作为:

```
static const unsigned char * pointer_sram = 0x8000;  /* 设置 SRAM 数据单元的基址指针 */
pointer_sram = 0xaa;                      /* 将数据写入基址单元 */
pointer_sram[1] = 0xbb;                   /* 将数据写入下一地址单元 */
……
```

对 EEPROM 的操作为:

```
static const unsigned char       * pointer_ee = 0x4000;
```

```
                                        /* 设置 EEPROM 空间的基址指针 */
pointer_ee = 0xaa;
……                                      /* 延时等待,对 EEPROM 的单元写入数据时必
                                           须增添一定的时间延时 */
pointer_ee[i] = 0xbb;                   /* i为地址增量,即相对于基地址的偏移量 */
……
```

端口 A、B 的地址及各控制寄存器的地址由 CSIOP 的基地址加偏移地址来决定,本设计中 PA .0 用于为 8279 的 A0 提供地址信号。如不采用 PSD813F1,则需在 80C196KC 和 8279 之间接 74373 用于地址锁存输出,采用 PSD813F1 则只需配置端口 A 的 PA .0 工作于地址锁存工作方式,配置由 A 口的控制寄存器(占据一字节空间)决定,实现方法如下:

```
static const unsigned char      * csiop = 0x9000;
csiop[2] = 0xfe;                        /* 端口 A 的控制寄存器相对于 CSIOP 基地址的
                                           偏移地址为 02H;此时 PA .0 在控制寄存器中
                                           的相应位为 0,因此 PA .0 工作于地址锁存输
                                           出方式,其他高 7 位相应位为 1,因此工作于
                                           MCU I/O 方式 */
```

对 8279 的初始化操作为:

```
static const unsigned char  * pointer_8279 = 0xd000h;
                                        /* 基址为数据口地址 */
pointer_8279[1] = 0xd1;                 /* 对命令端口写入清除命令 */
do { }
while( (pointer_8279[1]|| 0x7f = = 0x7f);   /* 等待清除完成 */
pointer 8279[1] = 0x2a;                 /* 送程序时钟分频常数 */
pointer_8279[1] = 0x08;                 /* 写键盘/显示器工作方式命令 */
pointer_8279[1] = 0x90;                 /* 写显示 RAM 命令 */
pointer_8279 = 0x10;                    /* 向数据口送入显示数据 0x10 */
```

总之,本文通过设计该数据采集系统,表明 PSD813F1 芯片集 FLASH MEM、EERPOM、SRAM、PLD 等于一体,可代替电路设计中的程序存储、数据存储、数据缓存和译码等芯片,从而可使单片机系统大大简化,一个实用系统可简化为“两片”系统,而其 JTAG 技术更是为该芯片的使用增添了便捷的设计和使用手段。

参 考 文 献

1 PSD8XXF 系列数据手册及应用笔记 . 武汉:武汉力源,1998

2 孙涵芳 . INTEL 16 位单片机 . 北京:北京航空航天大学出版社

选自《电子技术应用》月刊,2000 年第 7 期

5 .4　一种优越的可编程逻辑器件——ISP 器件

长沙国防科技大学 407 教研室(410073)　杜青松　张　炜

一、前　言

在数字系统的设计中,常常要用到可编程逻辑器件(PLD),尤其是可以反复编程的 PLD (如 EPROM、E^2 PROM、FPGA 等等)。采用 E^2 PROM 编程技术的 PLD 器件具有可反复编程且数据可长期保存的优点,但编程时需采用专用的编程器;而采用静态 RAM 技术的 PLD 器件具有灵活的现场编程的特点,但其结构编程数据必须存储在外加的 EPROM 中或在每次使用时重新下载。ISP 器件则解决了这两种 PLD 器件存在的问题,将二者的优点完美地结合在一起。

ISP 是指在用户自己设计的目标系统中或印刷电路板上为了重新配置逻辑而对逻辑器件进行反复编程的能力。具有这一功能的器件在编程时无需专用的编程器,而只需通过器件内部所具有的系统内编程电路直接接收编程的命令和数据,然后在编程状态机的控制下对器件进行编程。

与传统的数字电路系统设计和制造方法相比,ISP 技术有如下优点。

1 .简化了数字电路系统的设计和制造

采用系统内可编程(ISP)器件,在印刷电路板上就可以对 ISP 器件进行重新编程,修改器件的内部逻辑以实现逻辑功能的改变,也不需修改与之相关的印刷电路板,因而减少了试制过程中因反复修改逻辑而造成的人力和物力耗费,并且缩短了研制周期。

2 .减小系统的冗余度

在数字电路的系统中,经常要用到许多逻辑功能大致相同、仅仅是具体指标有所差异的电路模块。在传统的设计方法中,必须按各电路模块的具体要求选择不同的逻辑器件,设计不同的电路,并制作出不同的印刷电路板。而用 ISP 技术设计时,这些功能相似的模块可用同一种印刷电路板加上同一个 ISP 器件来实现,其不同的逻辑功能只需通过软件对其进行编程即可实现,这样就减少了电路板的种类。

3 .有利于产品的升级

在传统的设计方法中,如果遇到产品需升级的情形,必须按新的逻辑功能重新进行设计、制板,加大了重复投资。而采用 ISP 技术,则可在设计时预先保留重新编程的功能,当此产品需要升级时,可通过软件对 ISP 器件进行重新编程,使其具有新的逻辑功能,几乎可以不需附加任何硬件上的投入。

4 .简化生产流程

在数字系统生产的过程中,对于普通的可编程逻辑器件,其生产流程是先对器件进行编程,然后对器件加贴标签,再用此器件进行装配。而采用 ISP 技术时,可先将 ISP 器件预先装配在电路板上,再根据系统的要求用软件方法对其进行编程,从而简化了生产流程。

二、ISP 器件结构

下面我们以 Lattice 公司生产的 ispLSI1032 为例来看看 ISP 器件的结构。

ispLSI1032 内部具有 6 000 个可编程逻辑门、192 个寄存器，外部有 64 个可编程的 I/O 口、8 个特定输入。其结构框图如图 5.4-1 所示。

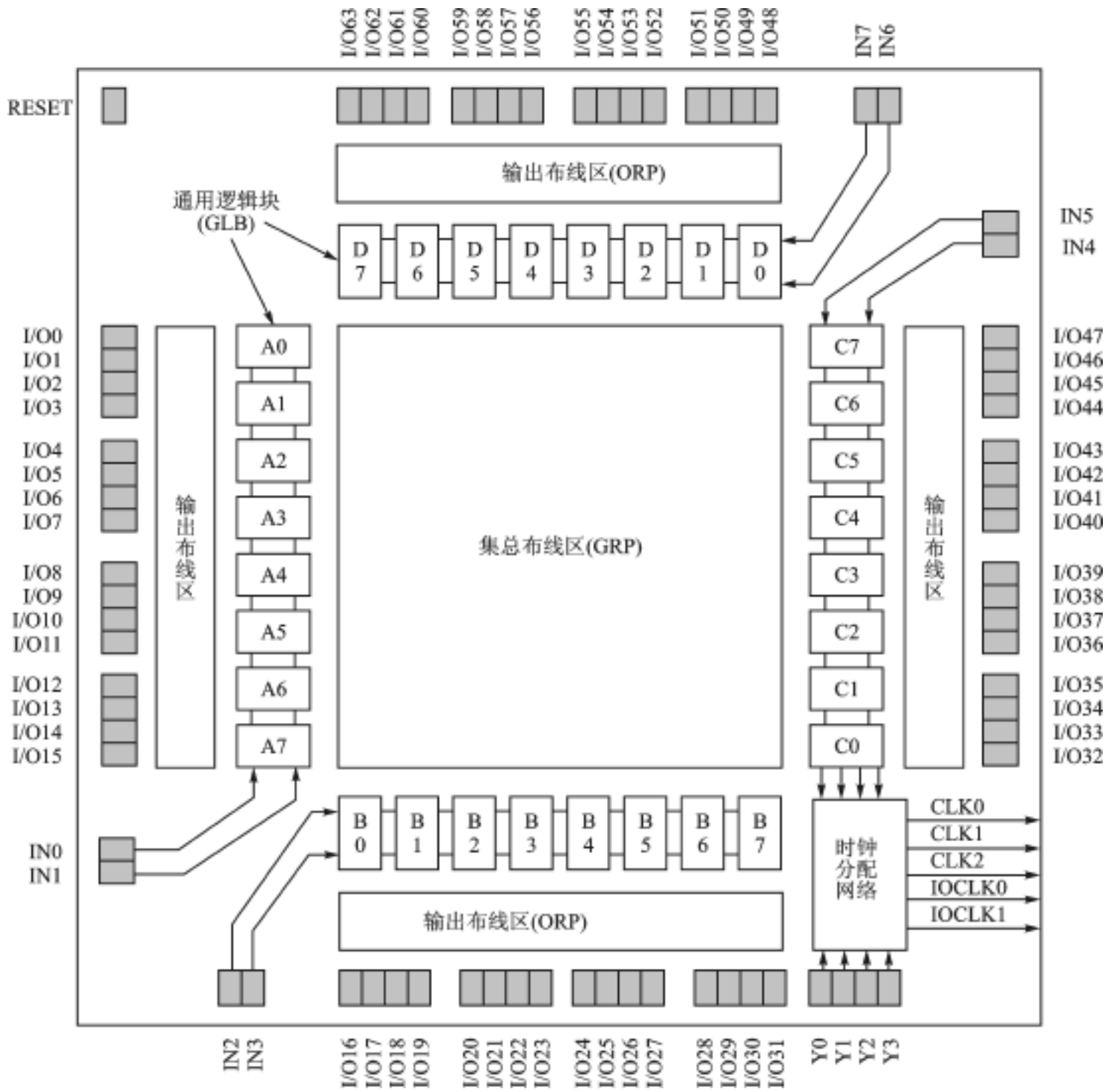


图 5.4-1 ispLSI1032 结构框图

在图中我们可以看到，ispLSI1032 由总布线池(GRP)、输出布线池(ORP)、输入总线、时钟分配网络、I/O 单元、编程控制单元以及通用逻辑块(GLB)等组成。其中 GRP 用于内部各 GLB 输入、输出的相互连接(完成内部逻辑的连接)，它的延时或为固定延时，或为可预知的延时，有利于实现高性能的复杂的系统设计；ORP 用于 GLB 输出到输出引脚的分配和连接，它可使管脚的排列非常灵活；时钟分配网络用于产生 I/O 单元和 GLB 所需的时钟(全局同步时钟、单 GLB 异步时钟、I/O 时钟)。ISP 器件的核心逻辑单元是 GLB，每个 GLB 由逻辑与阵列、乘积项共享阵列(PSTA)和输出逻辑宏单元(OLMC)组成，如图 5.4-2 所示。

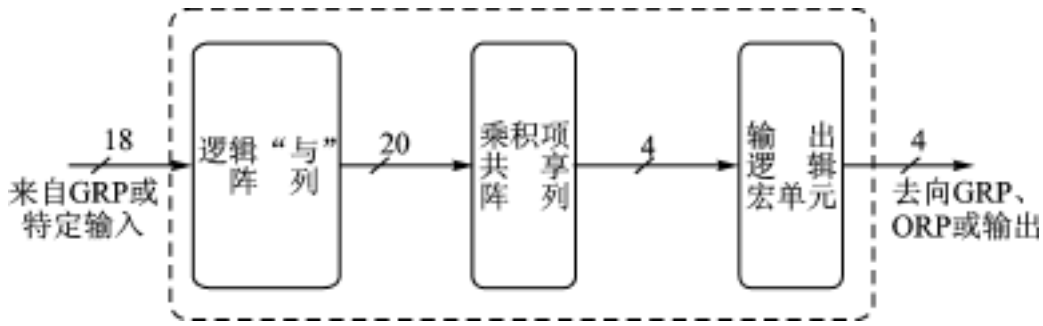


图 5.4-2 GLB 结构示意图

每个 GLB 有 18 个输入端,它的“与”阵列可产生 20 个积项,由积项共享阵列分配至 OLMC 中的 4 个输出单元,每个输出单元可构成 D 型触发器、JK 型触发器或 T 型触发器。GLB 完成每个 ISP 器件的主要逻辑功能。

三、ISP 技术设计流程

采用 ISP 技术的数字系统设计流程如图 5.4-3 所示。

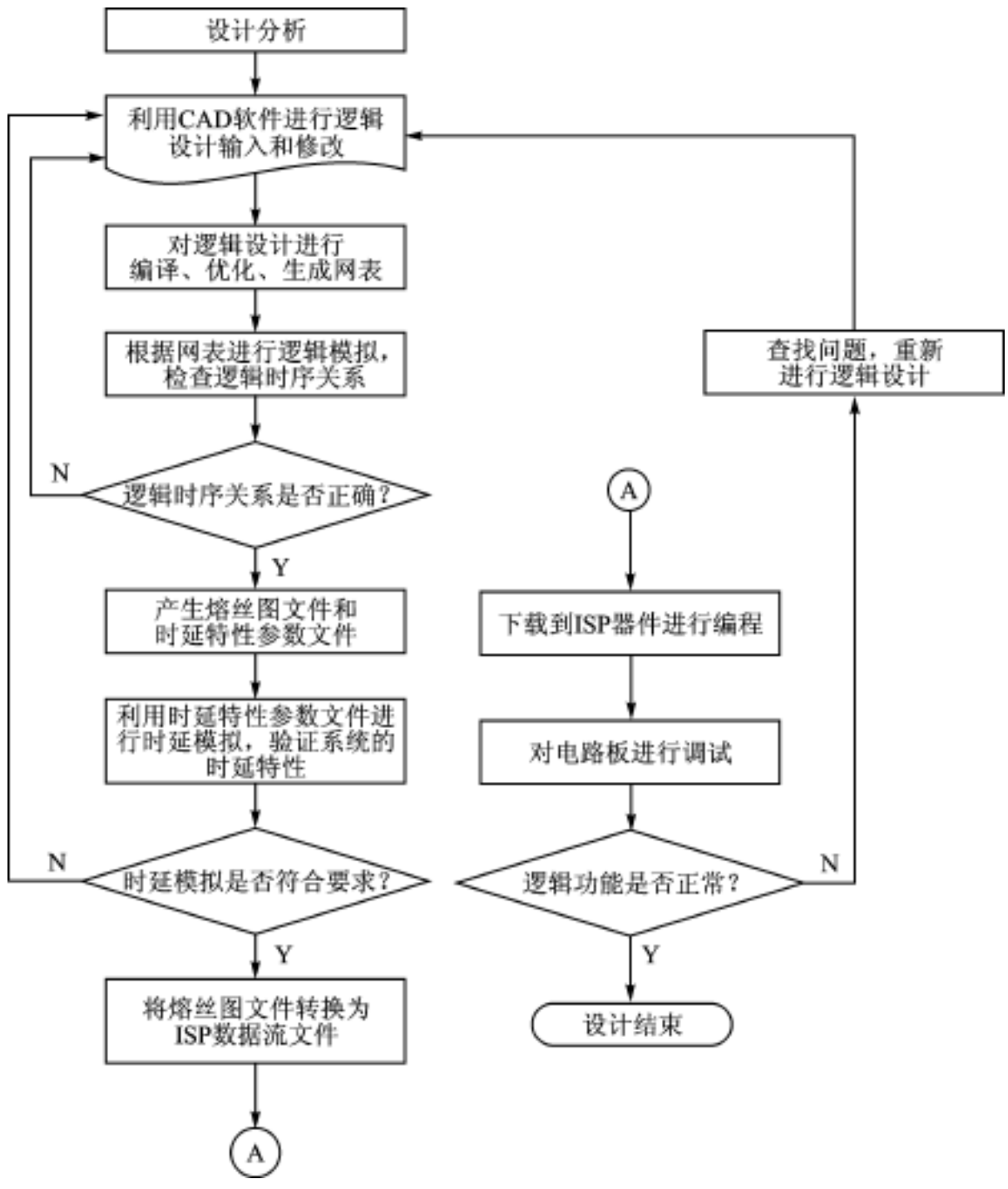


图 5.4-3 ISP 数字系统设计流程

四、ISP 器件的编程

如前所述,当逻辑电路的设计、模拟完全通过以后,用 ISP 器件的专用编译软件根据逻辑电路的网表产生熔丝图文件 JEDEC,就可以对 ISP 器件进行编程。对 ISP 器件的编程首先是由 ISP 下载软件将 JEDEC 文件转换为 ISP 器件专用的数据格式——ISP 数据流,然后通过专门的 ISP 器件控制电路向 ISP 器件发出控制命令,使其处于编程状态,将 ISP 数据流传送至 ISP 器件,在器件内的状态机控制下完成对器件的编程。

ISP 器件编程控制电路用以产生器件所需的编程接口信号,通常是将 ISP 器件的编程管脚通过 ISP 下载电缆连至计算机并行接口来实现。下面以本系统中使用的 ispLSI 器件为例,对 5 个编程信号的描述如下。

MODE——模式控制信号,输入。它与 SDI 信号一起对编程状态机的编程模式进行控制。

SDI——数据串行输入信号,输入。当 MODE 信号为低电平时,SDI 是片内串行移位寄存器的输入信号,用于将编程命令和数据写入移位寄存器;而当 MODE 信号为高电平时,SDI 成为编程状态机的控制信号,控制编程命令的装载和执行。

SDO——数据串输出信号,输出。它与片内串行移位寄存器的输出相连。

SCLK——串行时钟信号,输入。提供片内串行移位寄存器的时钟信号,同时也是编程状态机的状态切换驱动信号。当 MODE 信号为高时,SCLK 控制编程状态机的切换;而当 MODE 信号为低时,SCLK 则成为移位寄存器的移位时钟。

$\overline{\text{ispEN}}$——系统内编程使能信号,输入。当其为高电平时,器件处于正常工作模式;当其为低电平,则器件进入编程模式,此时 ISP 器件的所有 I/O 引脚将处于高阻态。

系统内编程是在编程状态机的控制下进行的,通过编程状态机的控制,可向 ISP 器件写入数据或读出数据。编程状态机有三种状态:闲置/ 识别码状态、命令移位状态和命令执行状态,如图 5.4-4 所示。这三种状态的相互转换是由 MODE、SDI 信号在 SCLK 的驱动下实现的。

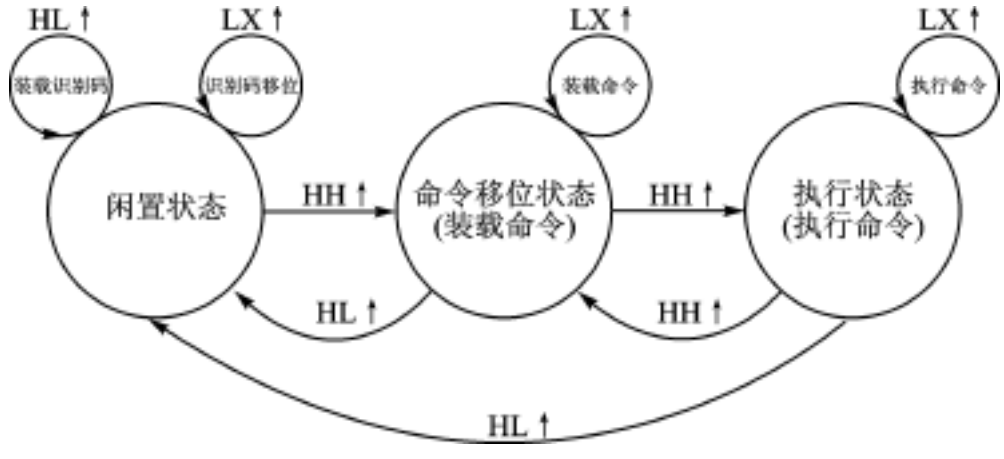


图 5.4-4 ISP 器件的编程状态机

H、L、↑分别表示 MODE 信号、SDI 信号、SCLK 信号的状态;H 表示高电平,L 表示低电平;X 表示任意态,↑表示上跳沿

闲置/ 识别码状态:当器件进入编程状态时,其状态机的初始状态为闲置/ 识别码状态;当器件处于状态机的其他状态时,只要 MODE 为高,SDI 为低,在 SCLK 上跳沿的控制下均进入此状态。此状态用于器件在编程时闲置,或用于装载器件识别码(每个 ISP 器件都有一个特定

的识别码 ID)。

命令移位状态:当器件处于编程状态机的闲置/ 识别码状态时,如 MODE 信号为“H”,SDI 为“H”,则在 SCLK 上跳沿的控制下进入此状态,此后,MODE 为低电平,SDI 成为编程命令输入端,在移位时钟 SCLK 的控制下将 SDI 端的编程命令及数据串行地移入器件内的移位寄存器,为编程状态机的命令执行状态装载编程命令及数据。

命令执行状态:当编程命令和数据被移入器件内的移位寄存器后,控制 MODE 及 SDI 信号使 MODE 为“H”,SDI 为“H”,则在 SCLK 上跳沿的控制下进入此状态。进入此状态后,将 MODE 置为“L”,使得 SCLK 成为时钟信号,控制状态机执行已装载的命令。当命令执行完之后,将 MODE 和 SDI 均置为高,在 SCLK 的控制下转入命令移位状态,装载下一条命令以备执行。

ISP 器件的编程流程如图 5.4-5 所示。

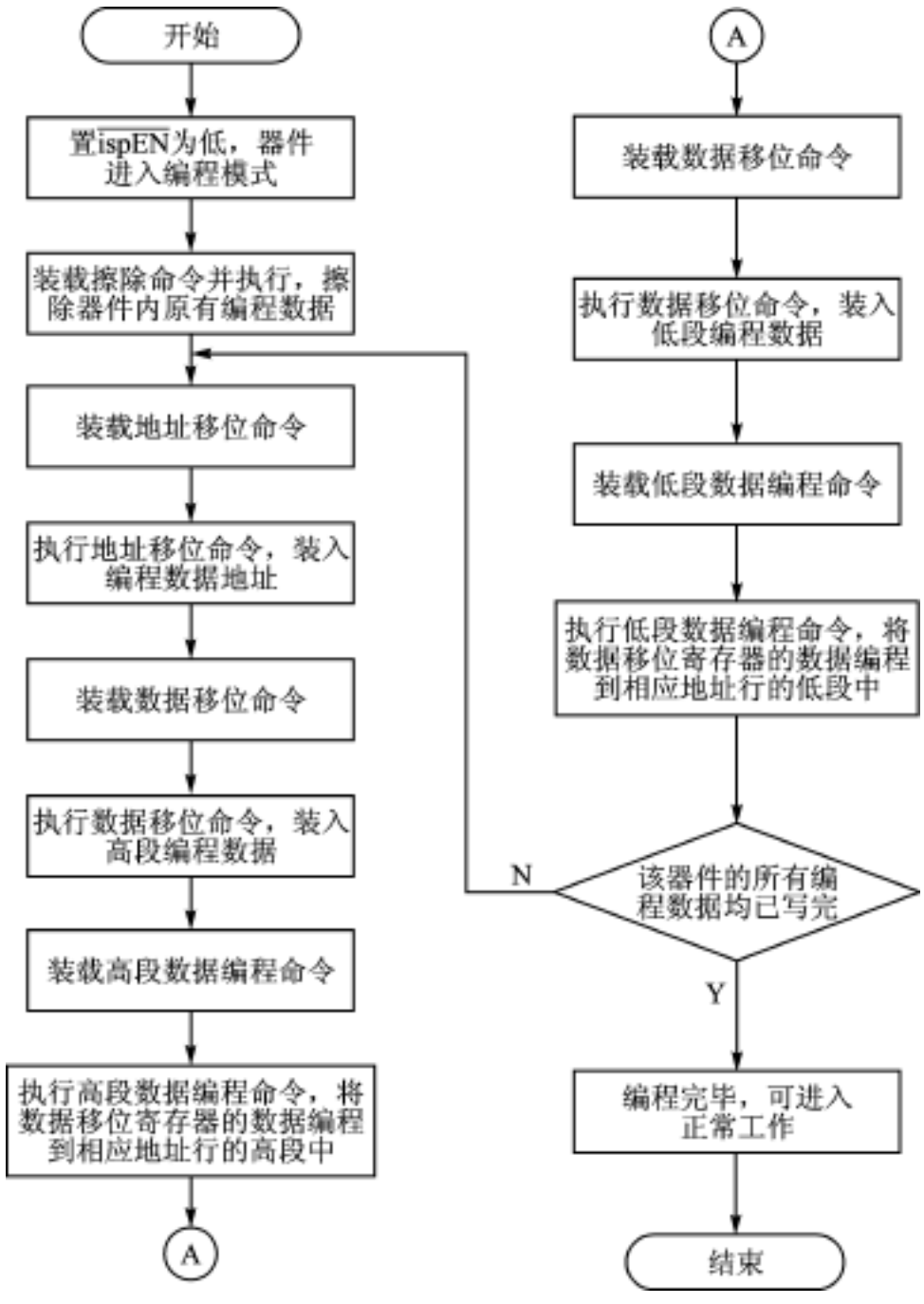


图 5.4-5　ISP 器件编程流程图

五、结束语

ISP 器件作为一种新型的可编程逻辑器件,它的优越性是非常明显的。它相对于 FPGA 来说,容量较小,加载过程有很大区别。FPGA 掉电后,片内逻辑丢失,再启动时需要通过 EPROM 等方式再加载;而 ISP 的片内逻辑一经加载后,不会因掉电而丢失,若需改变逻辑,则需重新加载。总而言之,ISP 器件是一种方便、快捷的科研工具。

参 考 文 献

1 Lattice 公司 . Lattice Semiconductor Data Book . 1996

2 杜青松 . 遥科学通信与数据管理仿真系统——音频/ 视频数据处理分系统的设计与实现[硕士论文] . 长沙:国防科技大学,1998

选自《微处理机》季刊,2000 年第 1 期

5.5 ISP－PLD原理及其设计应用

石家庄河北师大职技学院电子系(050031) 王秀青
哈尔滨东北农业大学工程学院(150030) 王立舒

一、引 言

传统的可编程逻辑器件(PLD)在使用中通常是先插在专用编程器上进行编程,编程后再拔下装配在目标板。这样对于管脚很密的器件,在插拔的过程中易使管脚弯曲。若在同一系统中使用多个PLD,必须分批对其进行编程。使用传统的PLD,在对系统逻辑功能进行更改的时候不够灵活,系统维护不够方便。

20世纪80年代末Lattice公司提出了“在系统编程”技术(简称ISP)并生产出具有ISP能力的PLD(这种PLD称作ISP－PLD)。所谓ISP技术是指对器件、电路板或整个电子系统的逻辑功能随时进行修改或重构的能力。这种重构或修改可在产品设计、制造过程中的每个环节,甚至在交付用户使用之后进行。

ISP－PLD与传统的PLD不同,它可先装配再编程,而且不需专用的编程器,只需通过计算机接口和相应的编程电缆,即可直接在目标系统或印刷线路板上进行编程。它解决了传统的PLD较难解决的问题,例如:多个器件同时编程,管脚间距很密,器件编程和管脚弯曲等问题。所以ISP－PLD提高了系统的可靠性,并且便于系统板的调试和维修。ISP－PLD的出现使数字电路系统的设计方法得到改进。通过ISP技术修改ISP－PLD的内部逻辑功能,从而实现整个系统的逻辑功能的改变,使系统的硬件功能可象软件一样,通过编程来配置。ISP技术的出现在电子系统中引入“软”硬件这样一个全新的概念,为数字电路的设计开创了一个全新的局面。

ISP技术是下一代PLD的首选编程手段,世界各著名半导体公司纷纷推出自己的ISP产品,如:Lattice公司的ispLsi、ispGAL、ispGDS、ispGDX,Xilinx公司的XC产品系列,Vantis公司的MACH系列产品,Altera公司的Flex、Max、Classic等系列产品。众多公司间的相互竞争,不断促进可编程集成电路技术的提高,使其性能不断发展,在结构密度、功能、速度和性能等各方面均得到进一步发展,产品更加丰富。

二、ISP原理及ISP－PLD的开发

1.ISP原理

ISP技术是一种串行编程技术,下以Lattice公司的ispLSI器件为例说明其编程原理:器件的编程信息数据用E^2CMOS元件存储,E^2CMOS元件按行和列排成阵列,编程时通过行地址和数据位对E^2CMOS元件寻址。编程的寻址和移位操作由地址移位寄存器和数据移位寄存器完成。两种寄存器都按先入先出的方式工作。ISP－PLD插在目标系统或线路板上进行编程,编程时芯片与外系统脱离。

2．ISP 实现方法

目前 ISP 编程方法大致有三种:(1) 利用计算机接口和下载电缆对器件编程;(2) 利用目标板上的单片机或微处理器对 ISP－PLD 编程;(3) 多片 ISP－PLD 用菊花链编程。

其中,方法(1)只需要一个简单的编程电缆和一台 PC 机就可完成器件编程,如图 5.5－1 所示。ISP 编程接口非常简单,共有 5 根信号线:模式控制输入 MODE、串行数据输入 SDI,串行数据输出 SDO,串行时钟输入 SCLK 和在系统编程输入 ispEN。PC 机可通过这 5 根信号线完成编程数据传送和编程操作。

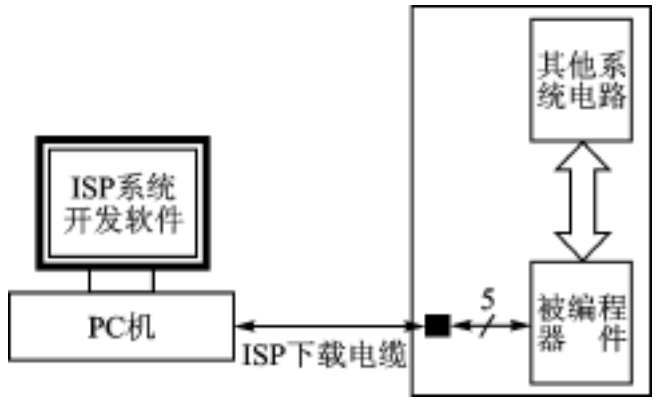


图 5.5－1 ISP 编程方法

3．ISP－PLD 的开发

对于 ISP－PLD 的开发,不同公司的产品采用不同的软件开发系统,如 Lattice 公司专门研制的开发软件有:ISPSynario System,在此基础上又推出了新的、更完整的开发软件 ispEXPERT。Vantis 公司生产的 MACH 系列 ISP－PLD 所用的开发软件为 Design Direct。Altera 公司的 ISP－PLD 的开发采用自行设计的软件:早期 A + plus,第二代 Max + plus 发展到目前的第三代 Max + plusⅡ和第四代 Quartas。后二者均具有完全集成化的易学易用的可视化设计环境,工业标准 EDA 接口,并可运行在多种操作平台上。

不同厂家的 ISP－PLD 产品的开发过程大致相同,一般都包括以下几个过程:(1) 输入设计项目;(2) 设计处理;(3) 设计校验;(4) 下载编程文件;(5) 编程器件的功能测试。各种 ISP－PLD 的开发系统一般都支持图形设计输入、硬件描述语言输入或两种输入的混合方式。

三、设计实例

本设计利用 ISP－PLD 实现一交通灯控制系统,此系统可管理东西主干道南北支干道方向行驶的车辆通过十字路口。在十字路口面对各个方向悬挂红、绿、黄三色信号灯及表示禁止(或允许)通过时间的计时牌。HG, HY, HR:分别代表东西主干道绿、黄、红三色灯。FG, FY, FR:分别代表南北支干道绿、黄、红三色灯。

此控制系统状态机共有 6 个状态 S0～S5。其中 S0:东西绿灯亮,南北红灯亮;S1:东西绿灯闪烁,南北红灯亮;S2:东西黄灯亮,南北红灯亮;S3:东西红灯亮,南北绿灯亮;S4:东西红灯亮,南北绿灯闪烁;S5:东西红灯亮,南北黄灯亮。其状态转换由 S0 S1 S2 S3 S4 S5 S0 如此周而复始地进行下去。其时序图可参见图 5.5－4 仿真时序图。

1．设计方案

设计方案框图如图 5.5－2 所示。

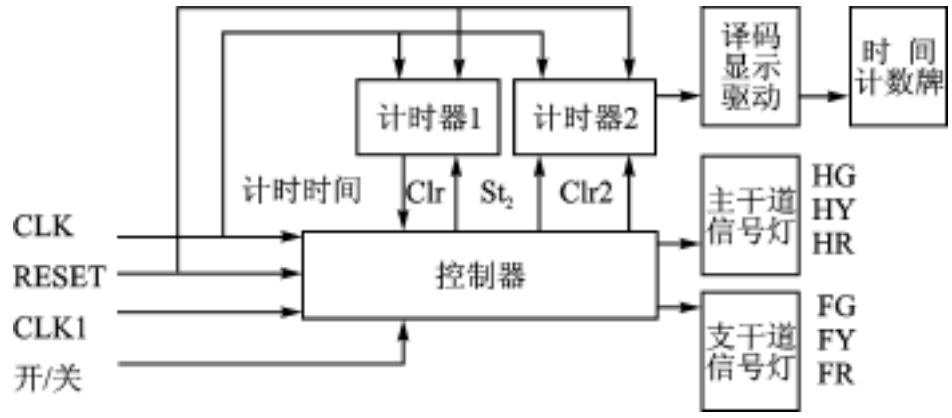


图5.5-2 交通灯控制系统方框图

2.实现方法

本设计选用Atlera的Flex系列EPF10K10LC84-4芯片来实现控制器、计时器1和2、译码及显示等功能。时间计数牌的计时显示用外接数码管实现,信号灯外接。采用层次化设计,顶层采用原理图输入,设计如图5.5-3所示。图中各模块均采用AHDL硬件描述语言设计,其功能如下:

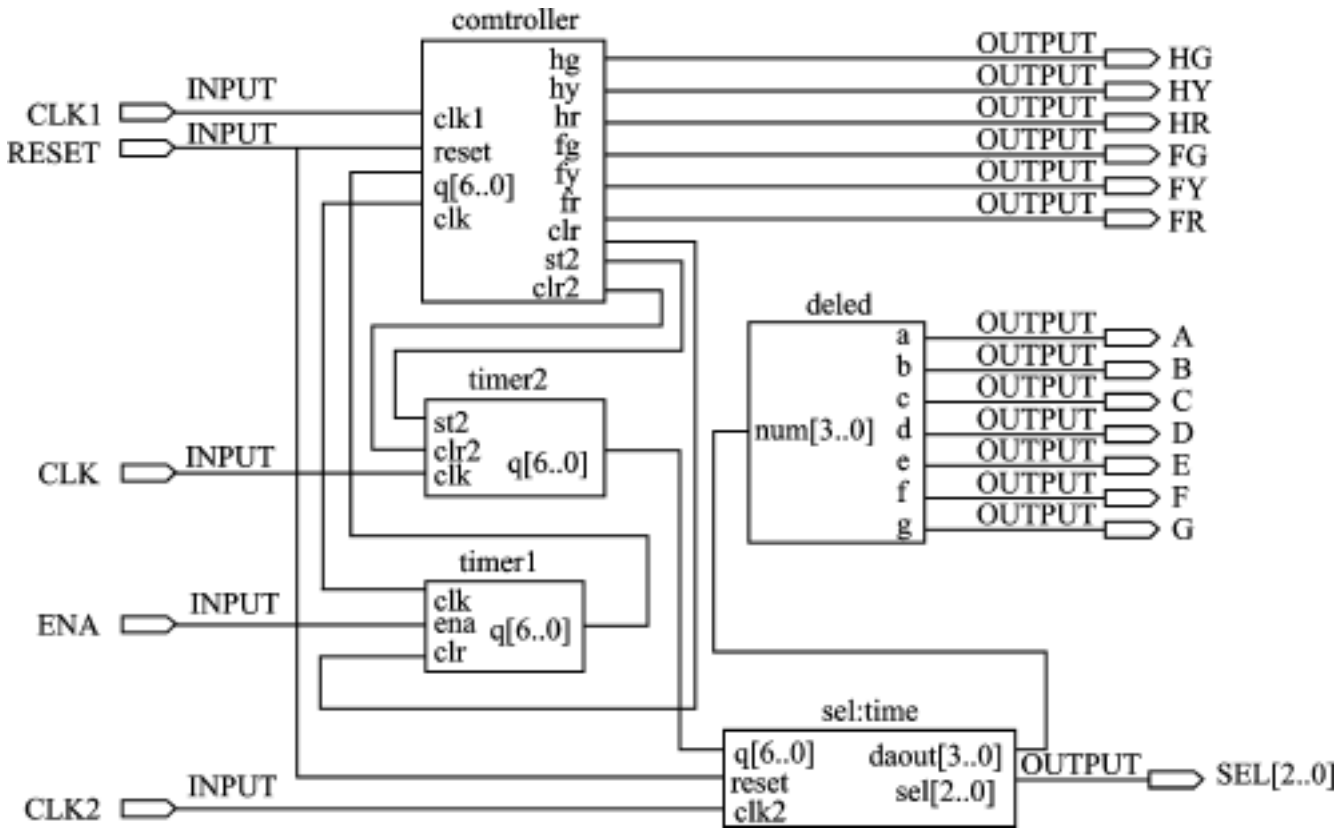


图5.5-3 交通灯控制系统顶层输入原理图

timer1模块:控制系统开始工作后,发送计时数据到主控模块controller。

controller模块:当接受到timer1发送来的计时数据后,据计时数据及状态机所处状态,确定状态机是否发生相应的状态转换。状态转换完毕,控制器发出clr=1信号,计时器1开始从零计时。

timer2模块:当st2=1时进行禁止(或允许)通行时间的计时,clr2=1时计时器2清零。

seltime模块:将timer2模块输出的计时数据转换为BCD码输出到译码显示deled模块,同时输出与十进制每一位数据相对应的数码管显示的片选信号。

deled模块:对Seltime模块传送来的信号进行七段数码管显示的相应译码。

图5.5-3中CLK为计时器1、2的计时时钟和绿灯处于闪烁状态的闪烁时钟。CLK2为

计时牌显示的扫描显示频率。CLK1 为状态机时钟,周期为 T_1。在程序设计时若状态发生转变,则状态转换控制信号应至少持续 $T_1/2$,否则尽管程序逻辑无错误,但实际工作中状态机来不及响应状态转换信号而仍旧处于原状态。ENA 为交通灯控制系统进行工作的开关信号。

3.仿真与验证

仿真:输入与设计项目顶层文件相对应的仿真通道文件 .SCF 文件,利用 MaxplusⅡ中的 Simulator 模块进行仿真,得到仿真时序图如图 5 5-4 所示。图 5 5-4 的仿真图为取绿灯闪烁时间 4 s 和黄灯亮时间 2 s,东西绿灯亮 20 s、南北绿灯亮 14 s 时所得到的仿真曲线。(图中 fCLK1 = 10 fCLK,由于显示比例较小,因此 CLK1 曲线线条过密连成一片成黑色。)

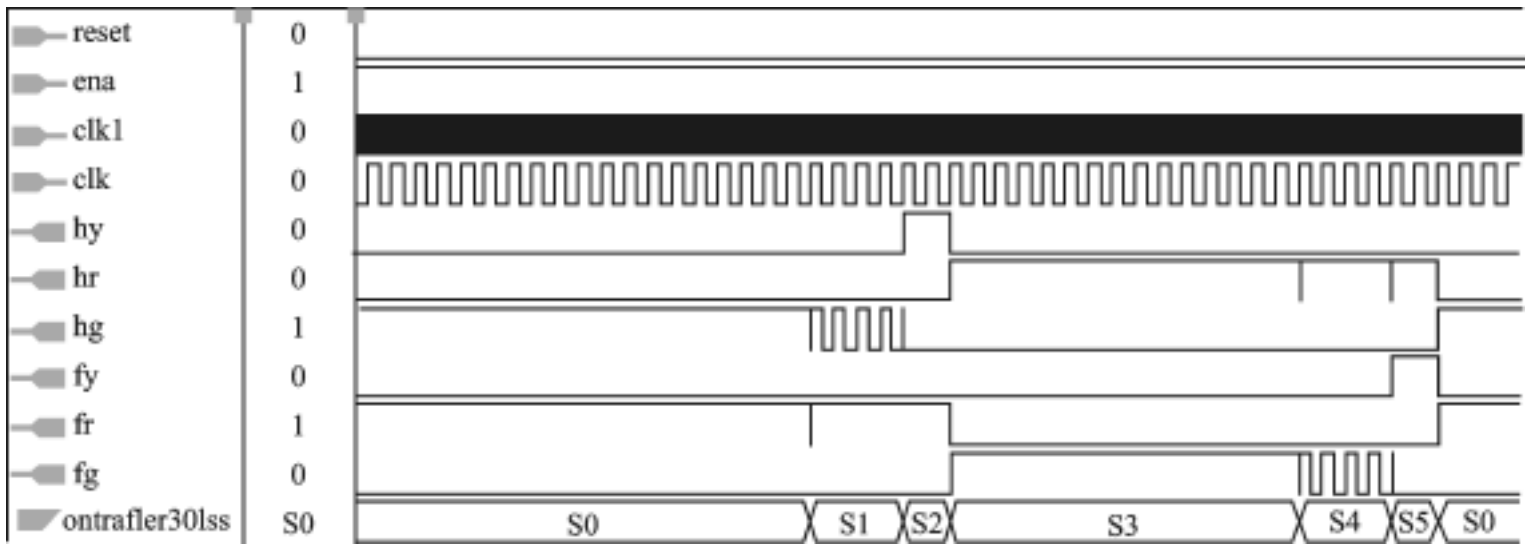


图 5.5-4 仿真时序图

实验验证:在连接相应的具体硬件后,将所编设计项目下载到 Atlera 芯片中,实验结果证明本设计满足设计要求。

四、结束语

电测仪表和控制系统中数字系统的实现可采用中规模集成电路实现,也可用各种可编程逻辑器件实现。通过上述设计可见:单片 ISP-PLD 可完成传统设计中多片中规模集成电路才能实现的电路功能,ISP-PLD 简化了系统结构,减小了设计单元的规模。ISP-PLD 应用于仪表、测控系统可减小其体积,利于实现整个系统的小型化。采用硬件描述语言描述计数器、定时器、显示、控制器等的电路的工作过程可简化设计过程、缩短设计周期、提高设计的可靠性及可移植性。随着 ISP 技术的不断成熟,及 ISP-PLD 的不断发展,将 ISP-PLD 应用于电测仪表和控制系统中的数字系统设计不失为一种好选择。

参 考 文 献

1 赵曙光,郭万有,杨颂华.可编程逻辑器件原理、开发与应用[M].西安:西安电子科技大学出版社,2000
2 康华光,邹寿彬.电子技术基础 数字部分[M].北京:高等教育出版社,2000

选自《电测与仪表》月刊,2001 年第 11 期

5.6 ispPAC10 在系统可编程模拟电路及其应用

天津大学精密仪器与光电子工程学院 梁 磊 宋光德 粟大超

一、概 述

随着大规模数字集成电路的发展,数字电路的设计产生了革命性的变革。各类 FPGA、CPLD 的出现,使得数字电路设计人员可以在微机辅助软件的协同下,编制出高效的电路,而且可以根据现场实际情况的需要,随时在线调整电路。

然而,传统的模拟电路设计,常常需要进行反复多次的试验,才能最终确定各个元器件的参数值,而且一旦设计要求有所变化,就不得不重新进行试验,既耗时又耗力。由美国 Lattice 公司在 2000 年率先推出的 ispPAC10 从根本上改变了这种局面,其在系统可编程能力,使模拟电路的设计简单化、程控化,被誉为模拟系统中的 FPGA。

ispPAC10 内部共含有 4 个可编程模拟功能单元,称之为 PACblock 单元。其中每一个 PACblock 单元包括 1 个加法器、2 个差动式输入仪表放大器和 1 组反馈电容阵列。可变增益的输入仪表放大器能够在程序控制下以整数为步长单位,在±(1～10)之间进行增益的调节。ispPAC10 无需外挂元器件即可实现一些基本的模拟功能,如精密滤波、加法/ 减法运算、增益/衰减及积分功能。如果把 PACblock 单元组合起来使用,该集成电路将可以完成一些更为复杂的信号处理功能。

通过使用基于 Windows 的图形输入风格的软件包 PAC－Designer 来对 ispPAC10 进行编程配置。PAC－Designer 可以在配置器件之前对电路功能和性能进行仿真、校验。用户可以随时通过编程电缆将程序由微机下载到 ispPAC10。一旦下载成功,电路的拓扑结构和元器件的参数值将存储在 ispPAC10 的非易失性 E^2CMOS 配置存储器中。

二、ispPAC10 的功能框图及引脚功能

ispPAC10 的功能框图及引脚排列如图 5.6－1 所示。各引脚的功能如表 5.6－1 所列。

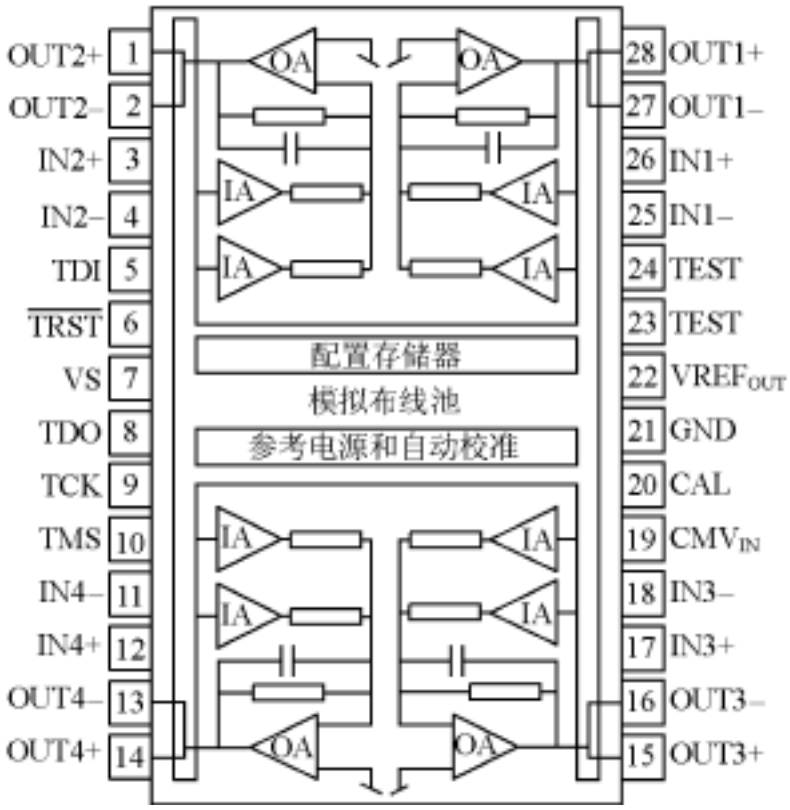


图 5 .6 - 1 ispPAC10 功能框图引脚排列

表 5 .6 - 1 ispPAC10 引脚功能

引脚号	引脚名	功 能
1	OUT2 +	差动输出端, V_{OUT+}
2	OUT2 -	差动输出端, V_{OUT-}
3	IN2 +	差动输入端, V_{IN+}
4	IN2 -	差动输入端, V_{IN-}
5	TDI	JTAG 串行输入端,输入信号在 TCK 上升沿时有效,编程接口
6	$\overline{\text{TRST}}$	JTAG 串行复位端,编程接口
7	VS	电源电压(5 V),用 1 μF 及 0 .01 μF 的电容作旁路滤波
8	TDO	JTAG 串行输出端,输入信号在 TCK 下降沿时有效。编程接口
9	TCK	JTAG 串行时钟端(输入),编程接口
10	TMS	JTAG 串行模式选择端(输入),编程接口
11	IN4 -	差动输入端, V_{IN-}
12	IN4 +	差动输入端, V_{IN+}
13	OUT4 -	差动输出端, V_{OUT-}
14	OUT4 +	差动输出端, V_{OUT+}
15	OUT3 +	差动输出端, V_{OUT+}
16	OUT3 -	差动输出端, V_{OUT-}
17	IN3 +	差动输入端, V_{IN+}
18	IN3 -	差动输入端, V_{IN-}
19	CMV_{IN}	可选择的模拟共模电压输入端
20	CAL	自动校正命令序列输入端

续表 5.6-1

引脚号	引脚名	功 能
21	GND	电源地端。通常与模拟地相连
22	$VREF_{OUT}$	共模参考电压输出端(+2.5 V),用 0.1 μF 的电容作旁路滤波
23	TEST	工业制造测试端,通常与地相连
24	TEST	工业制造测试端,通常与地相连
25	IN1-	差动输入端,V_{IN-}
26	IN1+	差动输入端,V_{IN+}
27	OUT1-	差动输出端,V_{OUT-}
28	OUT1+	差动输出端,V_{OUT+}

三、ispPAC10 电路结构及特点

PACblock 作为 ispPAC10 的主要模拟功能单元,由 2 个输入仪表放大器和 1 个输出加法器构成,如图 5.6-2 所示。两路仪表放大器各自具有 10 阶的可编程跨导值,与 10 阶跨导值相对应的是 2～20 μs/V 的阶跃灵敏度。PACblock 的输出端具有短路保护措施,且能够驱动最小为 300 Ω 的阻性负载或最大 1 000 pF 的容性负载。

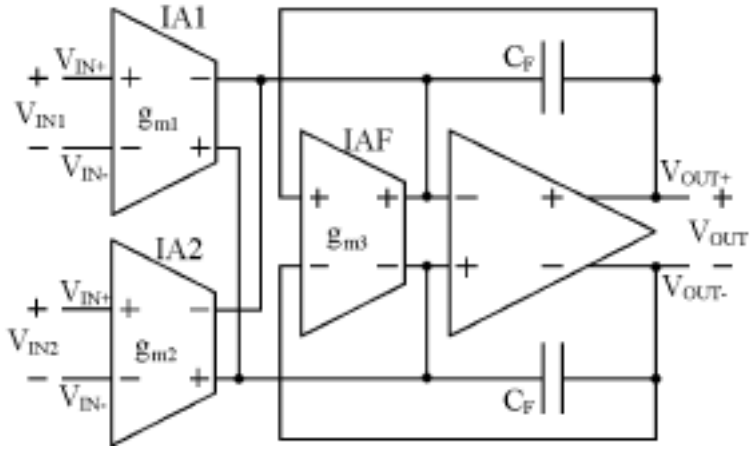


图 5.6-2 PACblock 单元内部结构

由图 5.6-2,并根据基尔霍夫电流定律,最终可以得出如下传输特性方程:

$$\frac{V_{OUT}}{V_{IN}} = \frac{K_1 g_m V_{IN1} + K_2 g_m V_{IN2}}{g_{m3} + \frac{sC_F}{2}}$$

其中,输入放大器具有可程控的增益值,其跨导值分别为 g_{m1}、g_{m2},且 $g_{m1} = g_{m2} = g_m$。K_1、K_2 为 -10～10 的整数。反馈放大器的跨导值为 g_{m3}。反馈电容在 1～62 pF 的域值内,通过编程选择。

输入偏差自动校正是 ispPAC10 的一个特有性质:每次开机上电,或通过外部 CAL 引脚命令,或通过 JTAG 编程命令均能产生一自动校正序列。通过自动校正,就能够大大降低因时间、温度而产生的偏移误差。当 PACblock 单元增益为 1 时,输入偏移误差不超过 1 mV;在增益为 10 的情况下,输入偏移误差将小于 100 μV。

四、实际使用中应注意的问题

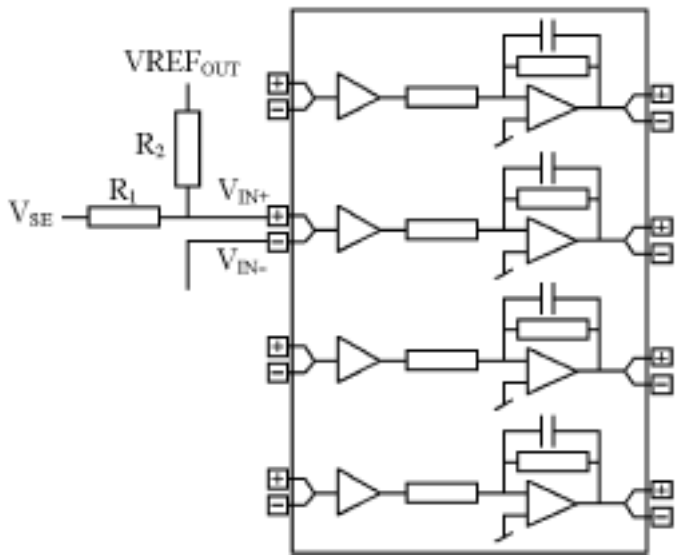
1. 关于 $VREF_{OUT}$ 的使用

$VREF_{OUT}$ 输出引脚具有高输出阻抗，所以在作为参考电压的时候，需加缓冲电路。PACblock 单元本身就可作为 $VREF_{OUT}$ 的缓冲器。在该种情况下，输入仪表放大器的输入引脚与 $VREF_{OUT}$ 相连。这样，PACblock 单元的输出端就可提供 2.5 V 的参考电压，并且具有良好的驱动能力。

当然，若用 $VREF_{OUT}$ 作为具有更高输入阻抗的电路的参考电压时，$VREF_{OUT}$ 可以直接与电路相连。

2. 关于输入接口的使用

当信号通过直流耦合方式输入时，若输入信号不提供直流偏置电流回路，那么就必须利用分压电路为直流偏置电流提供回路，以满足 ispPAC10 的输入要求，如图 5.6-3 所示。



$$V_{IN+} = \frac{V_{SE} R_2}{R_1 + R_2} + \frac{VREF_{OUT} R_1}{R_1 + R_2}$$

图 5.6-3 直流输入偏置电路

当然信号也能通过交流耦合方式输入，如图 5.6-4 所示。该电路不但提供了满足输入要求的直流偏置电流回路，而且还起到了高通滤波器的效果，其截止频率为 $1/2\pi RC$。若 $VREF_{OUT}$ 经过缓冲器后作为参考电压，R_{IN} 的最小阻值为 600 Ω；若 $VREF_{OUT}$ 直接提供 2.5 V 的参考电压，R_{IN} 阻值为 200 kΩ。

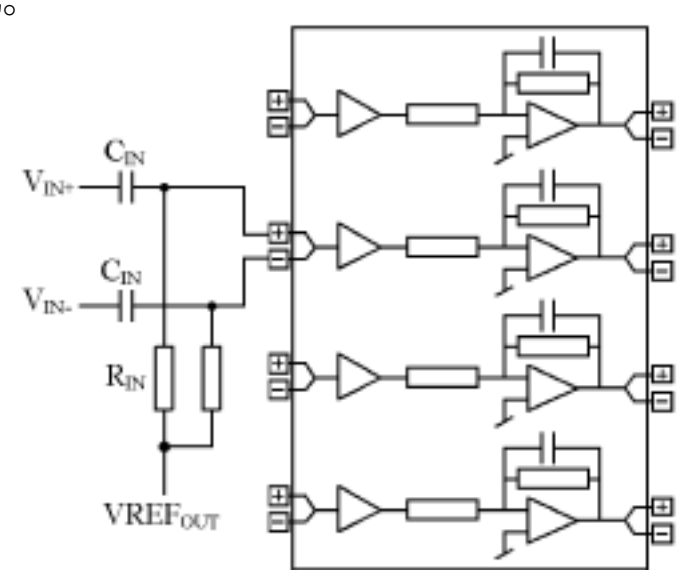


图 5.6-4 交流耦合直流偏置输入电路

3．关于单端输入的使用

ispPAC10 对于单端输入信号来说，要求其中一个输入引脚接成直流偏置，最好是接 $VREF_{OUT}$；另一个输入引脚接输入信号。该输入信号或通过交流耦合方式，或通过直流耦合方式与第二个输入引脚相连。这种连接方式，可以有效地抑制因单端输入而造成的共模误差。

五、ispPAC10 在模拟系统设计中的应用实例

下面结合几个基本的模拟电路功能单元，讨论 ispPAC10 的具体应用。通过对 ispPAC10 的灵活配置，可以实现模拟电路基本功能模块的自由组合，从而构建出复杂多样的模拟电路来，以适应不同模拟系统设计的需要。

1. 低通滤波器

低通滤波器的原理图如图 5.6-5 所示。

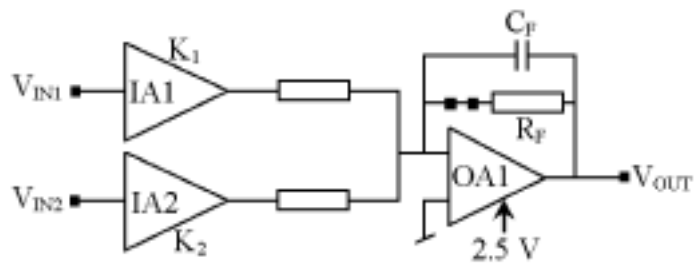


图 5.6-5 低通滤波器原理图

该电路的传输特性如下：

$$V_{OUT} = -\frac{K_1 V_{IN1} + K_2 V_{IN2}}{1 + \frac{sC_F}{2g_m}}$$

在截止频率之内，该电路可看作一个增益单元。其截止频率为 $(1/2\pi)(2g_m/C)$。由于 $g_m = 2u/V$，且 $1\ pF \leq C_F \leq 62\ pF$，可推得 $600\ kHz \geq f_p \geq 10\ kHz$。根据对选频反馈电容的选择，在 10～100 kHz 之间，有至少 120 级的截止频率。其中每一级都能保证其相应的幅频特性平坦度不超过 3.2%。事实上，若所选的截止频率在 48 kHz 左右，就能够得到小于 1.0%的幅频特性平坦度。由于反馈电容值的偏差，在实际使用过程中，PACblock 单元能够做到 5%的幅频平坦度。

2. 积分电路

将 R_F 从 OA1 的反馈成分中除去，即可得到积分电路，如图 5.6-6 所示，其传输特性如下：

$$V_{OUT} = -\frac{K_1 V_{IN1} + K_2 V_{IN2}}{\frac{sC_F}{2g_m}}$$

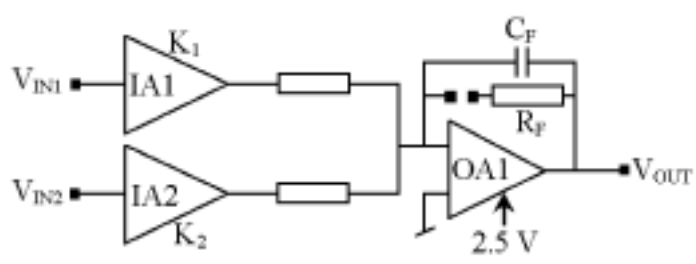


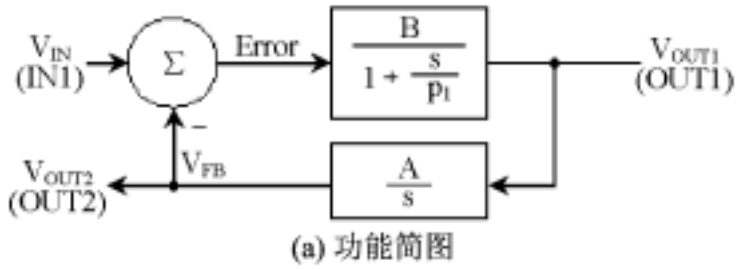
图 5.6-6 积分电路原理图

一般情况下,该积分电路不作为一个独立的环路单独使用。我们常利用它的直流反馈来确保输出不致饱和,图 5 .6 - 6 即为一例。

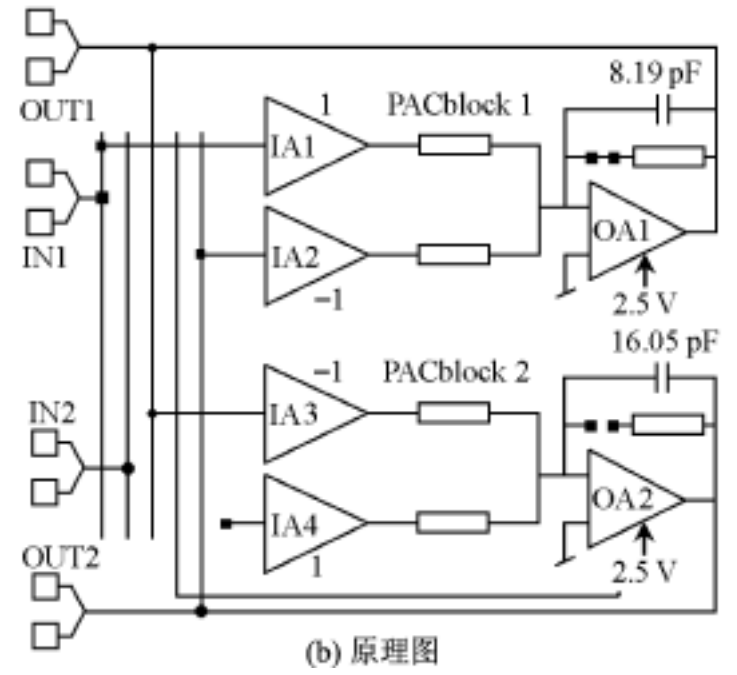
3. 双二次带通滤波器

将两个 PACblock 功能单元连接起来,即构成 1 个有用的环路,如图 5 .6 - 7(a)所示。该环路能够完成带通功能。

若设低通滤波器的截止频率为 100 kHz,积分器的截止频率为 10 kHz,则整个环路就构成了一个由 10～100 kHz 的带通滤波器。其设计原理图如图 5 .6 - 7(b)所示。



(a) 功能简图



(b) 原理图

图 5 .6 - 7　双二次带通滤波器

4. 衰减电路

图 5 .6 - 8 所示电路可实现衰减功能。

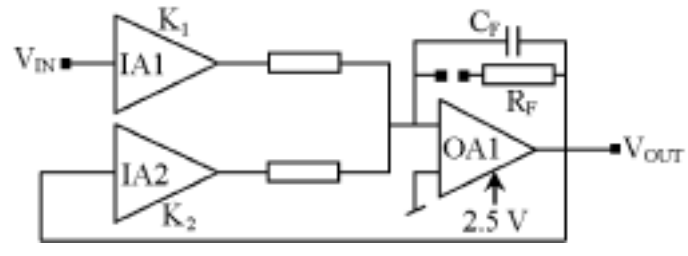


图 5 .6 - 8　衰减电路原理图

其传输特性如下:

$$\frac{V_{OUT}}{V_{IN}} = -\frac{k_1}{k_2 + 1 + \frac{sC_F}{2g_m}}$$

增益 K_1、K_2 可根据用户需要进行设置,该环路既可用来放大,又可用来衰减输入信号。分母中的 1 与 R_F 有关,若断开 R_F,分母中的 1 将略去。在 R_F 连接时,最低的衰减率可到 1/ 11(- 20 .8 dB);在 R_F 断开时,最低衰减率可到 1/ 10(- 20 dB)。

若把 PACblock 单元配置成衰减电路,应提高反馈电容的电容值,以维持电路的稳定度。

5. ADC 的前端信号调理电路

将 ispPAC10 内部的 4 个 PACblock 单元进行组合,就可构成如图 5.6-9 所示的 ADC 的前端信号调理电路。

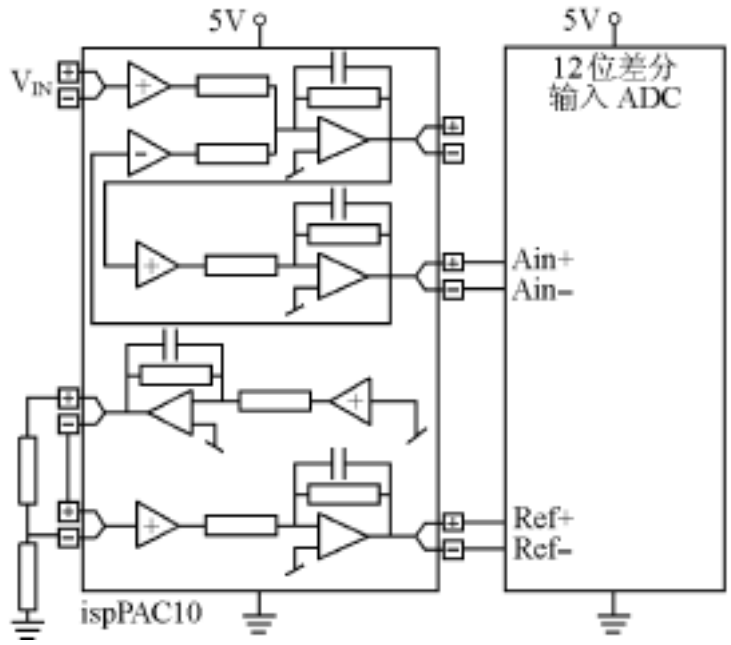


图 5.6-9 ADC 的前端信号调理电路

采样所得的模拟信号可以通过 V_{IN} 输入 ispPAC10。根据实际对信号的需要,选择合适的参数,该电路即可完成带通及信号的放大。同时,ADC 的参考电压也直接由 ispPAC10 提供。

以上应用实例,只是 ispPAC10 在系统可编程模拟电路应用中的一小部分。用户可以根据实际要求,构建出各种不同的模拟电路,并且一旦设计要求有所变化,只须简单地调整电路参数,重新将变化后的电路参数及拓扑结构下载到 ispPAC10 中去即可。由此可见,ispPAC10 有着相当广泛的应用前景,必将成为以后模拟电路设计的主流。

参 考 文 献

1 童诗白 模拟电子技术基础 北京:人民教育出版社,1982

选自《单片机与嵌入式系统应用》月刊,2001 年第 7 期

5.7 在系统可编程器件 ispPAC80 及其应用

济南山东大学电子工程系(250100) 王祖强 葛 敏 王照君

1992 年美国 Lattice 半导体公司发明了在系统可编程技术(In-System Programmability),彻底改变了传统数字电子系统的设计和实现方法,开创了数字系统设计的里程碑。1999 年 11 月,美国 Lattice 公司又推出了在系统可编程模拟电路(ispPAC: In-System Programmable Analog Circuits),翻开了模拟电路设计方法的新篇章。与数字在系统可编程大规模集成电路一样,在系统可编程模拟器件允许设计者使用开发软件在计算机中设计、修改模拟电路,进行电路特性模拟,最后通过编程电缆将设计方案下载到芯片中。目前已经推出的在系统可编程模拟器件有三种:ispPAC10、ispPAC20 和 ispPAC80。本文主要介绍 ispPAC80 器件及其应用。

一、ispPAC80 结构与特性

ispPAC80 器件包括一个增益为 1、2、5 或 10 可选的差分输入仪表放大器 IA、一个差分输出求和放大器 OA、5 阶低通滤波器、两个非易失性在系统配置存储器、ISP 接口、参考电压及自校正电路,如图 5.7-1 所示。

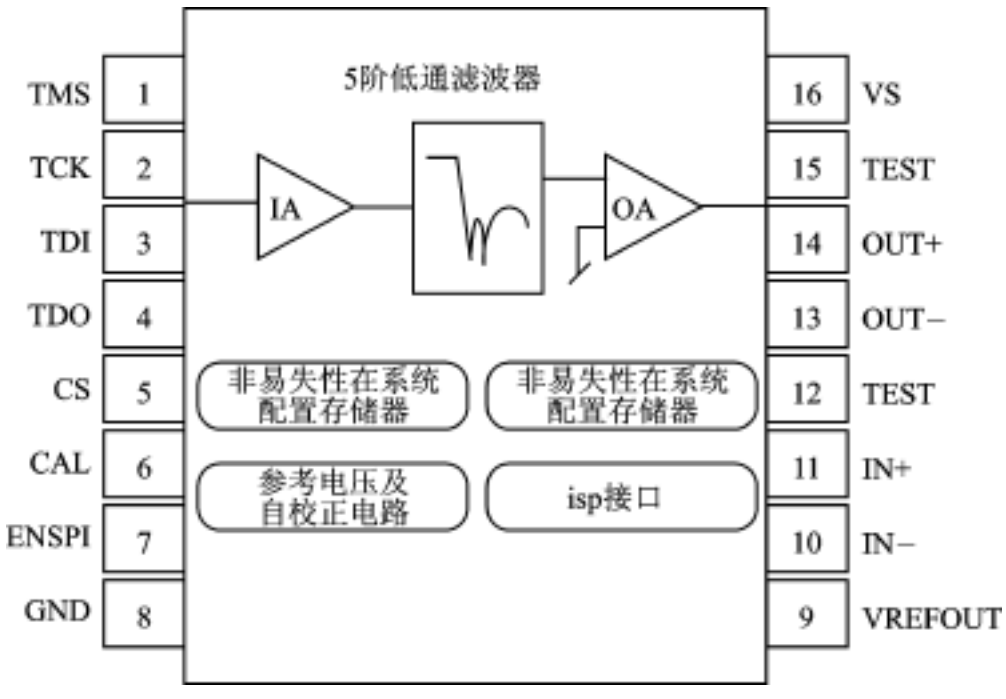


图 5.7-1 ispPAC80 结构图

ispPAC80 中所包含的 5 阶连续时间低通滤波器如图 5.7-2 所示。改变它的电容、电阻值便可得到具有不同幅频特性和相频特性的滤波器。这些值的改变有两种途径:① 设计者自行设定;② 设计者对滤波器性能(截止频率、滤波器类型等)提出要求,由应用软件计算出所需的参数值。通常采取第二种途径。这些值及放大器增益等信息均被直接存储在芯片上的非易失 E^2CMOS 存储器,并且设计者可以保存两种配置,因此在修改模拟电路时无需变动外部

元件。

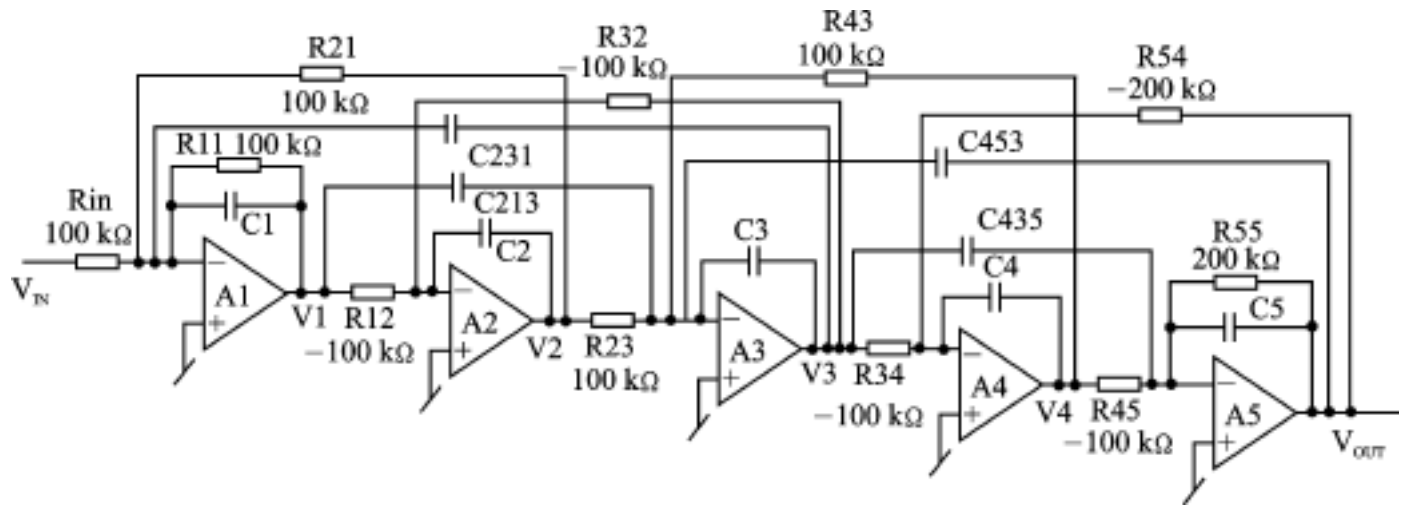


图 5.7－2　5 阶低通滤波器结构图

ispPAC80 的主要性能特点如下:

① 精密有源滤波(50～750 kHz);

② 双配置存储器使得用户可以在不对电路板做任何改动的情况下,完成两个完全不同的滤波器设计;

③ 所有的配置参数均存储在非易失 E^2 CMOS 存储器中;

④ 在系统可编程模拟电路;

⑤ 电路增益可以通过编程的方法改变,增益为 0～20 dB;

⑥ 可实现多种形式的滤波器设计:Chebyshev、Butterworth、Gaussian、Bessel 等;

⑦ 真正的差分Ⅰ/O,输入阻抗为 10^9 Ω,共模抑制比为 58 dB;

⑧ 失真低,总谐波失真 THD< -74 dB;

⑨ 系统包含大约 3 000 个独立的电容器,相当于 7 个分辨率为 9～12 位的可编程电容器,因此设计精度非常高;

⑩ 单电源 +5 V 供电,内部可提供 2.5 V 共模参考电压。

二、ispPAC80 的封装与引脚

ispPAC80 采用 16 脚塑封 PDIP 或 SOIC 封装,如图 5.7－3 所示。各引脚说明如表 5.7－1所列。

ispPAC80 的型号说明规则如图 5.7－4 所示。

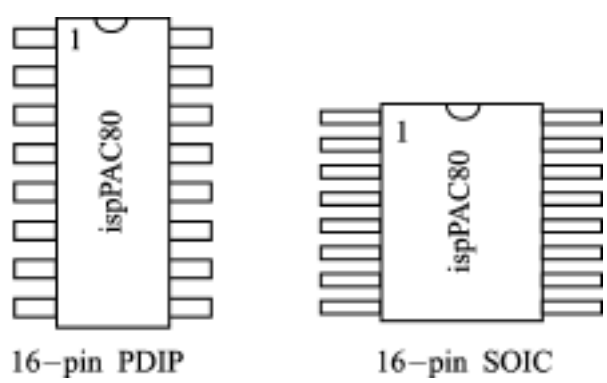


图 5.7－3　ispPAC80 两种封装形式

表 5.7-1 ispPAC80 引脚说明

引脚号	符 号	功能说明
1	TMS	测试模式选择,只能为 JTAG 模式
2	TCK	测试时钟,只能为 JTAG 模式
3	TDI	测试数据输入,可为 JTAG 或 SPI 模式
4	TDO	测试数据输出,可为 JTAG 或 SPI 模式
5	CS	片选
6	CAL	自校正
7	ENSPI	SPI 模式使能,ENSPI=1 时为 SPI 输入模式
8	GND	接地
9	VREFout	共模参考电压输出,正常时为 2.5 V,必须通过 1 μF 电容接地
10,11	IN	差分输入
12,15	TEST	测试脚,TEST=1 时测试,TEST=0 时正常工作
13,14	OUT	差分输出
16	VS	电源(正常时为 5 V),应通过 1 μF 电容接地

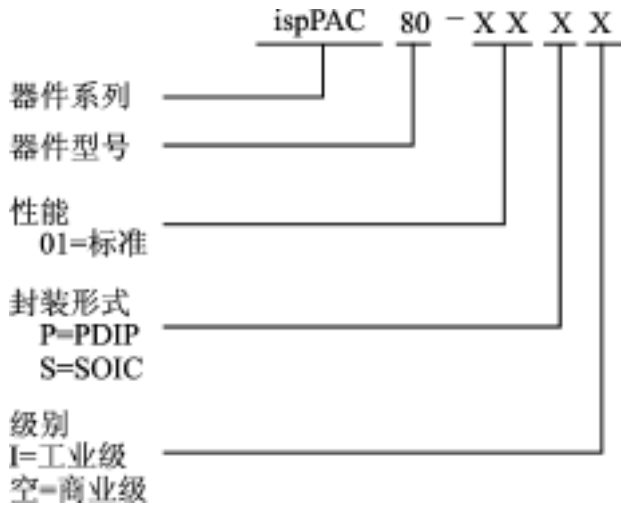


图 5.7-4 器件型号说明规则

三、ispPAC80 的开发环境

Lattice 半导体公司提供的基于 Windows 系统的 PAC-Designer 软件设计工具,具有非常友好的设计界面,且易学易用。用户可在用户图形界面 GUI 的原理图上或在滤波器表的输入项里描述属性或参数,然后在仿真窗口仿真滤波器的特性。在此仿真窗口中,设计者可同时查看至多 4 个输入输出的组合的增益和相位波形,且可用十字光标直接读取任意的增益和相位。一旦设计属性被确认,便可通过一条连接在 PC 机并口与 ispPAC80 串行编程口之间的下载电缆将其下载到 ispPAC80 器件中。连接接口符合 IEEE 1149.1 JTAG 标准。

四、应用举例

在一个实际的电子系统中,它的输入信号往往因干扰等而含有一些不必要的成分,应当把它衰减到足够小的程度;在一些场合,需要的信号却和别的信号混在一起,应当设法把前者挑选出来。上述问题均可利用 ispPAC80 来解决。图 5.7-5 所示电路是 ispPAC80 器件的一个典型应用,其功能就是先通过 ispPAC80 滤除有用信号中所含的干扰成分,再对其进行 A/D 转换。

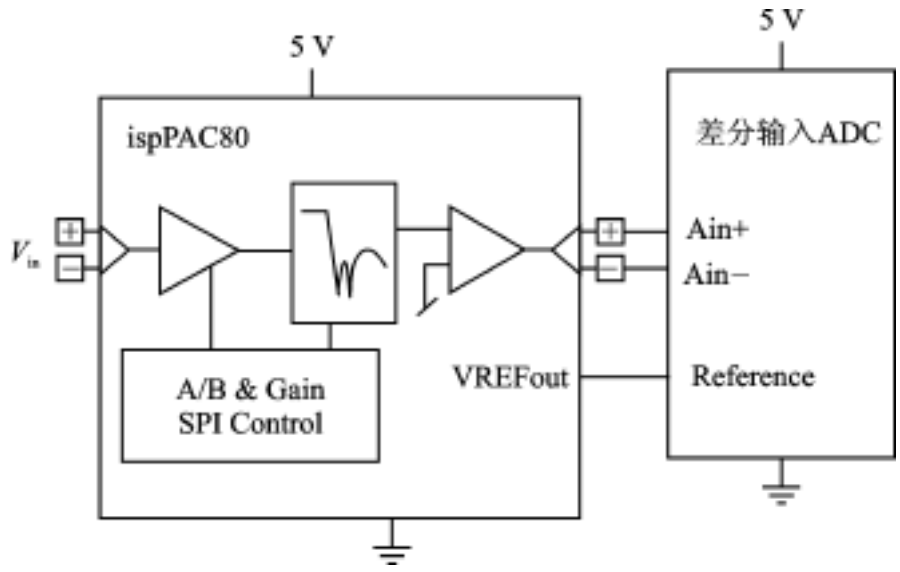


图 5.7-5 ispPAC80 典型应用电路图

五、ispPAC 系列器件比较

Lattice 公司推出的三种在系统可编程模拟器件分别是 ispPAC10、ispPAC20 和 ispPAC80。ispPAC10 的输入、输出全差分结构及高增益性能(0～80 dB),使得它非常适合仪器、仪表测量;ispPAC20 比 ispPAC10 少了两个 PAC 块,增加了两个可编程双差分比较器和一个 8 位的 D/A 转换器,所以 ispPAC20 的增益可调范围比 ispPAC10 小,只有 0～40 dB。但它却可以接收 8 位的数字信号并将其转换为模拟信号,此模拟信号的输出也是完全差分形式,可以与器件内部的比较器或仪用放大器相连,也可以直接输出;ispPAC80 器件则主要集中在滤波器的应用上,它使滤波器可以工作在很小的容差范围内。虽然使用 ispPAC10 或 ispPAC20 也可以进行滤波器的设计,但其精度和工作频率范围都不及 ispPAC80。例如 ispPAC10 有 4 个 PAC 块,最多可实现 4 阶滤波器,并且只能是截止频率在 17～100 kHz 之间的 Butterworth 或 Chebyshev 滤波器;而 ispPAC80 的截止频率却可以高达 500 kHz。

由于 isp 技术可以在器件被焊接在电路板上的情况下对系统编程或重构,因而当需要改变设计时,无需改动元器件或印制板,只需通过 Lattice 的开发系统软件在数分钟内即可完成。可见利用 isp 技术设计系统时,虽然设计的是硬件,却像软件一样方便,这就缩短了设计试制的周期,降低了试制成本,省去了编程器等设备和器件插拔的手续,是一种全新的设计方法。因此,IspPAC 器件的出现必将使电子系统设计、生产、维护及更新诸环节发生革命性的变化。

参 考 文 献

1 黄正瑾.系统编程技术及其应用.南京:东南大学出版社,1999

2 Lattice Semiconductor Corporation. ispPAC Handbook. 2000(3)

选自《电子技术应用》月刊,2001 年第 12 期

5.8 采用 ispLSI1016 设计高精度光电码盘计数器

上海交通大学机器人研究所(200030)
董高云 栾 楠 陈建平 赵锡芳

一、引 言

光电码盘是一种高精度的位置检测装置,广泛用于需要精确检测位置的控制系统中。笔者从事国家‘863’计划智能机器人主题项目——“精密 I 号”装配机器人后续研究课题中的控制系统改造工作,为了使该机器人能完成精密装配作业,该控制系统中采用特制的高精度光电码盘实时检测机器人各关节的运动角度。该种光电码盘为增量式旋转编码器,边沿采用高密度排列的小孔对光电头产生遮挡和透光,再辅以增速齿轮的增速,使其位置检测精度相当高,精度最高的 II 关节的脉冲数达到 115 200 个/转,最低的 III、IV 关节脉冲数也有 7 680 个/转,分辨率已达超高脉冲范围,从而能适应该精密装配机器人控制系统位置闭环的要求。为了与码盘的高精度相适应,要求设计多位(16 位)的计数器。用标准的 IC 芯片或一般的可编程逻辑器件 PLD(如 GAL,PAL 等)可以完成这一设计。但是此类芯片的功能与管脚有限,而设计要求的逻辑复杂,使设计所需的芯片个数会过多,电路板尺寸过大,时延问题严重,难以保证时序,从而加大设计难度。笔者采用新型大规模可编程芯片——ispLSI1016,充分利用其强大的可编程能力和众多的 I/O 口,用很少的器件,设计出高集成度的 16 位光电码盘计数器,将原来的许多由硬件实现的功能交由程序设计完成,达到了硬件设计软件化的效果。

二、ispLSI1016 芯片简介

在系统可编程大规模集成电路 ispLSI 系列芯片(in-system programmable Logic Scale Integration circuit)是 Lattice 公司生产的一种新型的大规模集成芯片,它不需要编程器,可直接在系统上编程,且具有众多的 I/O 口和内部逻辑阵列。在编程操作中该芯片编程方便,既可用抽象的逻辑表达式编程,也可以用逻辑图编程,还有强大的标准宏库可以调用的优点。因此,该芯片近来在数字电路的设计中得到了越来越广泛的应用。ispLSI1016 的引脚如图 5.8-1所示,为 44 脚双层四方结构。它由 32 个 I/O 单元(I/O0～I/O31),4 个专用输入单元(IN0～IN3),16 个通用逻辑块 GLB(A0～A7、B0～B7,每个 GLB 有 4 个可编程输出)及 I/O 总线、全局布线池、时钟分配单元等组成。

ispLSI1016 芯片的编程软件采用 Lattice 公司提供的 pDS 软件包,它是综合性的低成本逻辑开发软件包,采用 Windows 支持的图形界面,可以用 Boolean 方程输入开发程序,也可用库函数输入开发程序。同时 pDS 软件还提供仿真、时序测试等功能。在 pDS 开发系统中,32 个 I/O 模块和 16 个 GLB 的每一个都可由用户单独编写一小段程序,最后产生 JEDEC 文件,联调后直接写入 ispLSI1016,程序还可以方便地擦除和改写。pDS 开发系统规定每个 GLB 模块必须有一个时钟说明和最多四个可编程输出,共五句赋值语句,且整个程序必须有一个时钟

单元统一定义该芯片程序中所用到的时钟。

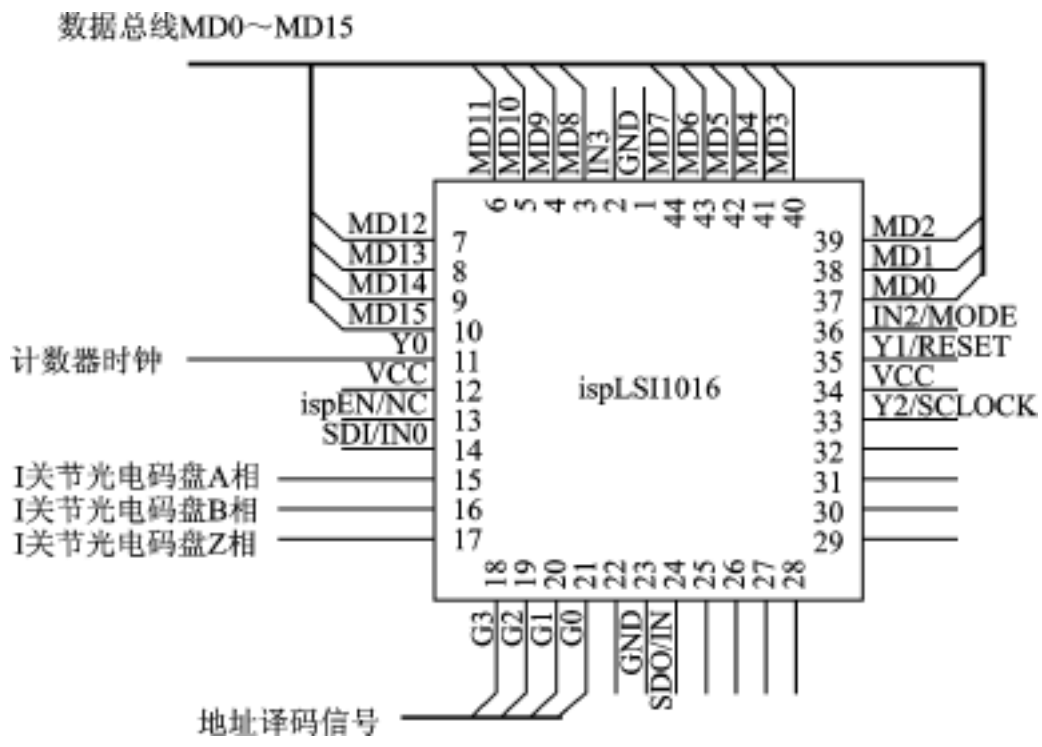


图 5.8-1 ispLSI1016 引脚及与码盘连接图

三、系统硬件连接情况

“精密 I 号”装配机器人为 SCARA 型机器人,其四个关节均采用高精度光电码盘检测位置,而码盘的精度有所不同。由于 ispLSI1016 有多达 32 个 I/O 接口,因此仅需一片该芯片即可完成一个关节的 16 位光电码盘计数器所需的鉴相、倍频、计数、锁存、译码和输出操作,整个机器人仅需四片 ispLSI1016。由于各关节码盘均采用 16 位计数器计数,计数器在程序设计上实际并无差别,故这里仅以控制 I 关节的一片 ispLSI1016 为例加以阐明。其与 I 关节光电码盘的连接在图 5.8-1 中示出。为集中说明计数器设计原理,该图仅示出了 ispLSI1016 在计数器设计中所必要的 I/O 端口,并略去了本片 ispLSI1016 与控制其他关节的 ispLSI1016 的串联接口。

本设计的硬件电路板插于 PC 工控机的通用 ISA 总线扩展槽中,图 5.8-1 中 MD0～MD15 为 16 位数据线,通过 74LS245(图中未示出)直接与 PC 机 ISA 总线的 MD0～MD15 数据线对应相连,PC 机通过它们读取码盘的 16 位计数值。A、B、Z 三相为来自光电码盘的计数波形,A、B 两相波形完全相同,分别来自两个不同位置的光电头,承担光电码盘增量计数工作,通过调整这两个光电头的位置可使这两相波形产生 90°的相位差,利用这样两相波形可完成鉴相和倍频操作。Z 相波形记录 Z 码(零码),这是该光电码盘在控制机器人关节运动中的所需要的特殊波形,其作用是在机器人开机上电过程中实现对极和标定操作,以使机器人运动前能准确定位。Z 相波形是不连续的,每当 Z 码出现时,就产生一个脉冲输出,光电码盘每圈的 Z 码个数依码盘的精度不同而各异,但这不影响本系统的软件设计。G3～G0 是地址译码信号,ispLSI1016 根据不同的 G3～G0 值的组合执行不同的操作,G3～G0 由另外的译码电路(由 GAL20V8 实现)直接根据 PC 机地址总线译出。Y0 为计数器时钟,由电路中的石英晶振经分频后输入,控制码盘计数的采码频率。其他引脚中,ispEN/NC、SDO/IN1、IN2/MODE、IN3 为在系统编程功能的预设管脚,它们须遵循特定的接法接到为编程而设置的特定插座上,

以使系统编程功能得以实现;Y1/ RESET、Y2/ SCLOCK 为本设计未启用的时钟分配单元,VCC/ GND 为电源和地。

四、光电码盘 16 位计数器编程分析

采用 ispLSI1016 实现 16 位计数器的程序由时钟单元、鉴相和倍频单元、计数逻辑单元(GLB)、计数输出单元几个模块组成,各单元编程分析如下。

1 .时钟单元

程序清单如下:

```
SYM GLB B2   1;
SIGTYPE CLKO OE;          P = CLKO;
SIGTYPE CP OUT;          CLKZ = Z;
SIGTYPE CLKZ OUT;          END;
EQUATIONS                END;
CLKO = ! G3& ! G2 &G1;
```

注:pDS 开发系统程序中的 &、# 、!、$ $ 分别为与、或、非、异或操作,下同。

时钟单元统一定义芯片计数程序的时钟。其中增量码盘的计数三态门由地址信号进行开关控制。在本系统中,规定 G3～G0 的初级译码值为 02H 时,开始 I 关节增量码盘正常计数;为 03H 时,则为该关节的 Z 码计数。由于上两种计数情况 G3～G1 值均相同,故设定 CLK0 = !G3& ! G2&G1,即可将地址值用作计数时钟控制计数值读取了。但同时,只要系统开始工作,ispLSI1016 即启动计数操作,且计数操作本身不受地址信号的影响。计数值的读取操作过程如下:由系统的译码器件 GAL20V8 先将地址值 A9～A0 译成 G3～G0,当地址初级译码值选通计数操作的地址后,执行读取计数值命令,将此时的计数值读入下位机,当再次选中计数操作所在的地址后,则再次读入当时的计数值,两次读码操作时间间隔内,计数操作仍然照常进行。Z 码的计数时钟则直接由 Z 码脉冲控制,当码盘处来一 Z 码信号后,即将当时的码盘计数值锁入 Z 码寄存器,没有 Z 码信号则该计数值保持不变,下一次 Z 码来后,再将该时刻的码盘计数值锁入 Z 寄存器内。

2 .鉴相和倍频单元

鉴相和倍频操作是光电码盘波形处理的第一步,其过程为:对波形相同相差为 90°的 A、B 两组输入波形进行处理,一方面将其进行逻辑运算后得出四倍频合成波形,进一步提高了码盘的计数精度,另一方面,比较 A、B 两组波形,判断码盘回转的方向(即计数值的正负):当 A 相超前 B 相 90°时,表明码盘正转,当 A 相落后 B 相 90°时,表明码盘反转。因语句较多,写入两个模块中。程序中的时钟 CLK 即为引脚 11 所接的计数器时钟 Y0。程序中引入了中间波形 GA1,GA2,GB1,GB2。GA1,GB1 由 A、B 波形经时钟触发后得出,与时钟的上升沿对齐,GA2 和 GB2 分别比 GA1 和 GB1 推迟一个时钟周期,由于 CLK 频率远高于 A、B 两相频率,故一两个时钟周期的改变并不影响码盘计数的准确性。正反转输出波形由下面的逻辑关系式给出:

```
UP = (GA1&( ! GA2)&( ! B)) # (GB1&( ! GB2) &A) # (( ! GA1) &GA2&B) # (( ! GB1) &GB2 &
( ! A));
```

```
DOWN = ((!GA1)&GA2&(!B)) # ((!GB1)&GB2&A) # (GA1&(!GA2)&B) # (GB1&(!GB2)
&(!A));
```

根据上面的逻辑关系得出的 UP 和 DOWN 的输出波形频率即为 A、B 波形的四倍,分别代表正转和反转,于是鉴相和倍频操作最终完成。

3.计数逻辑单元

程序清单如下:

```
SYM GLB B3 1 ;
    Q1 = Q1 $ $ ((Q0&UP) # (!Q0&DOWN));
SIGTYPE[Q0..Q3]REG OUT;
    Q2 = Q2 $ $ ((Q1&Q0&UP) # (!Q1&!Q0&DOWN));
EQUATIONS
    Q3 = Q3 $ $ ((Q2&Q1&Q0&UP) # (!Q2&!Q1&!Q0&DOWN));
[Q0..Q3].CLK = !CLK;              ND
Q0 = Q0 $ $ (UP # DOWN);          END;
```

该单元由 8 个 GLB 模块组成(B3～B6, A0～A3),每个 GLB 模块包括四位计数位和一个时钟,以上程序仅列出 B3 单元,其他单元从略。以 B3 模块为例:在时钟说明语句中,将计数时钟取反,使其由 CLK 的上升沿触发计数开始,变为下降沿触发计数。因为 UP 或 DOWN 的信号均与 CLK 上升沿对齐,故 CLK 信号与 UP DOWN 信号的翻转的先后顺序可能会不一样,由此可能造成误触发,现改为下降沿触发,则 CLK 下降沿来临时,UP 和 DOWN 信号已完成翻转。变为稳态,故能很好地完成计数操作的触发。计数器的翻转逻辑说明如下:只要计数器开始启动计数,Q0 就会发生翻转,且其翻转逻辑与加减计数无关,于是有 Q0 = Q0 $ $ (UP # DOWN)。当计数器进行加减法计数时,Q0 从 1 翻转到 0 表明产生了向 Q1 的进位,Q1 必须翻转,而 Q0 从 0 到 1,则无进位产生,Q1 不翻转。而当计数器进行减法记数时,而 Q0 从 0 翻转到 1 表明产生了向 Q1 的借位,Q1 必须翻转,而 Q0 从 1 到 0 则为无借位产生,Q1 不翻转。于是有 Q1 = Q1 $ $ ((Q0&UP) # (!Q1&DOWN));依此类推,可依次得出 Q2、Q3 的逻辑式和 B4～B6 号 GLB 单元的程序。对 Z 码的计数要求比较简单,只要求当有一 Z 码脉冲产生时,把此时的计数值另外锁存并记录下来,因此直接将 Q15～Q0 赋给 DZ15～DZ0,程序从略。

4.计数输出单元

程序清单如下:

```
SYM GLB A4 1;                          ND;
SIGTYPE[D0..D3]REG OUT;                END;
EQUATIONS                                  …
    [D0..D3].CLK = CP;                       …
    D0 = (Q0&!G0) # (DZ0&G0);
    D1 = (Q1&!G0) # (DZ1&G0);          SYM GLB A7 1;
    D2 = (Q2&!G0) # (DZ2&G0);          SIGTYPE[D12..D15]REG OUT;
    D3 = (Q3&!G0) # (DZ3&G0);
EQUATIONS          D14 = (Q14&!G0) # (DZ14&G0);
```

```
[D12 . .D15] .CLK = CP;      D15 = (Q15 & ! G0) # (DZ15 & G0);
D12 = (Q12 & ! G0) # (DZ12 & G0);          END;
D13 = (Q13 & ! G0) # (DZ13 & GO);          END;
```

本单元包括 A4～A7 四个 GLB 模块,用计数地址 CP 作为开门时钟来打开计数三态门。由前述,I 关节的译码值区别仅在 G0 位(地址值为 02H,开始增量码盘正常计数,为 03H 时,则为 Z 码计数),前三位已在时钟单元作出规定,故仅需用 G0 的 0 和 1 来区别这两种计数情况。于是当 G0 = 0 时,执行读 Z 码计数值的操作,最后的输出计数值 D15～D0 = DZ15～DZ0。当 G0 = 1 时,执行读增量码盘计数值的操作,D15～D0 = Q15～Q0。

该 ispLSI1016 程序其他部分包括 I/O 单元的编程,由于 I/O 单元的编程仅仅将 GLB 单元的部分变为输出信号,输出到各 I/O 引脚,比较简单,这里不再列出。此外,本片 ispLSI1016 除用作 16 位计数器外,实际上其多余的引脚将被赋与另外的逻辑,执行其他操作,因与本文无关,有关程序语句这里也未列出。

五、结 论

综上所述,本文采用 ispLSI1016 芯片来编写 16 位计数器,具有以下优点。

(1) 充分利用了 PLD 器件的可重复编程和可电擦除的优势,和 isp 器件的系统编程功能,无需专用的编程器,仅需在电路板上加装简单的接口,使程序的改写变得更加灵活方便。

(2) ispLSI1016 的高集成度和多 I/O 口使本计数器与传统的利用计数、锁存器件的计数器相比,大大减少了所用标准器件的数目,既简化了印刷板的设计,又减少了干扰和接触不良的故障隐患,提高了系统的可靠性,加快了响应时间(为 7 .5～15 ns),减小了驱动卡的面积,仅增加了编程的工作量。

目前利用该器件设计的 16 位计数器已成功应用于“精密 I 号”装配机器人关节控制印刷板(JCB)中,并已经通过下位机编程,成功实现了该机器人下位机关节运动的控制。在当前 ispLSI 系列器件得到越来越广泛应用的背景下,相信本文会对该类器件在机电系统测试中的应用有所启发。

参 考 文 献

1 牛秀卿,王朝英,等 . 用 ispLSI 实现一个四位二进制计数器[J] . 南开大学学报(自然科学),1997,30(4):99～102

2 孟传伟,蒋平 . 基于 ispLSI 的机器人控制器设计[J] . 电气自动化,1997(5):18～20

3 栾楠,陈建平,言勇华,刘宝生 . AC-RV 新型驱动单元研制[J] . 机器人,1999,21(5):370～374

4 林乐忠,王永青,胡力耘 . 采用 ispLSI1016 芯片设计数控系统的位置板[J] . 组合机床与自动化加工技术,1998(8):41～44

选自《计算机自动测量与控制》双月刊,2000 年第 4 期

5.9　基于 ADμC812 的一种仪表开发平台

呼和浩特内蒙古大学自动控制系(010021)　仲兆楠

仪表是工业自动化控制过程中非常重要的组成单元。本文介绍在一定的硬件基础上,通过软件的编程可以构成数显仪表、单回路调节器、巡检仪、电流输出的手操器等工业仪表。

一、仪表开发平台的硬件实现

作为可以构成闭环控制的工业仪表,其硬件应具备显示、键盘、模数转换、数模转换、具有掉电保护的参数存储器以及看门狗等电路。针对仪表需要完成的功能又使硬件结构简单化,单片机选择 AD 公司的全集成化单片机 ADμC812,其在单个芯片内包含有 12 位高性能的自校准 8 通道 ADC,两个 12 位 DAC,与 8051 兼容的 8 位可编程的 MCU。片内 8 KB 闪速/电擦除(Flash/EE)程序存储器、640 字节闪速/电擦除数据存储器以及 256 字节数据 SRAM,另外 MCU 支持的功能包括看门狗定时器、电源监视器以及 ADC DMA 功能。并具有 32 条可编程 I/O 接口和 SPI、UART 等串行接口。价格不足百元,是构成仪表的理想的器件。

电路原理如图 5.9-1 所示,由传感器送来的 4～20 mA 的输入信号,经 250 Ω电阻转换为 1～5 V 的电压信号,通过 RC 低通滤波器送入 ADμC812 的 ADC 通道(图中只给出 1 个通道),两个二极管用于 ADC 输入过压保护。ADμC812 的 DAC 为 12 位电压输出型 D/A,通过 AD694 实现的 V/I 转换为 4～20 mA 的标准电流信号,以驱动执行机构。IMP705 构成了复位电路,提高单片机工作的可靠性。由两片 8D 锁存器 74LS377 及 P3.2 组成动态扫描的键盘、显示电路。P3.3 与光耦构成开关量信号的输入电路。P3.4、P3.5、三极管及继电器构成开关量信号的输出电路,可用于上下限报警输出等。

MAX232 及 P3.0、P3.1 构成了与计算机通信的 RS232 接口。其有两个功能,其一为实现计算机与仪表之间的参数和数据的通信。另一功能是与开关 K1 配合,完成将程序从计算机下载到 ADμC812 片内的 8 KB 闪速/电擦除程序存储器中。当下载程序时,需将单片机系统的电源关掉,闭合开关 K1,单片机系统再上电,ADμC812 进入程序下载状态。运行计算机软件就可将 HEX 文件下载到 ADμC812 内。这正是本仪表的方便之处,在修改程序时,既不需要昂贵的编程器,也不需打开机壳插拔芯片。只需一条电缆将计算机的 COM1 或 COM2 连接到仪表即可。

整个仪表开发平台的各部分电路及电源都安装在 96 mm×96 mm×125 mm 的国际标准的仪表壳内。

二、仪表开发平台的软件设计

仪表的软件设计采用模块化的设计方法。建立一个子程序库,包括 ADC、DAC、显示、键处理、数字滤波器、控制算法等等子程序。对于不同功能的仪表在主程序中调用需要的子程序,修改、补充一些程序即可完成。在工作中不断的补充新的子程序,能提高开发的工作效率。

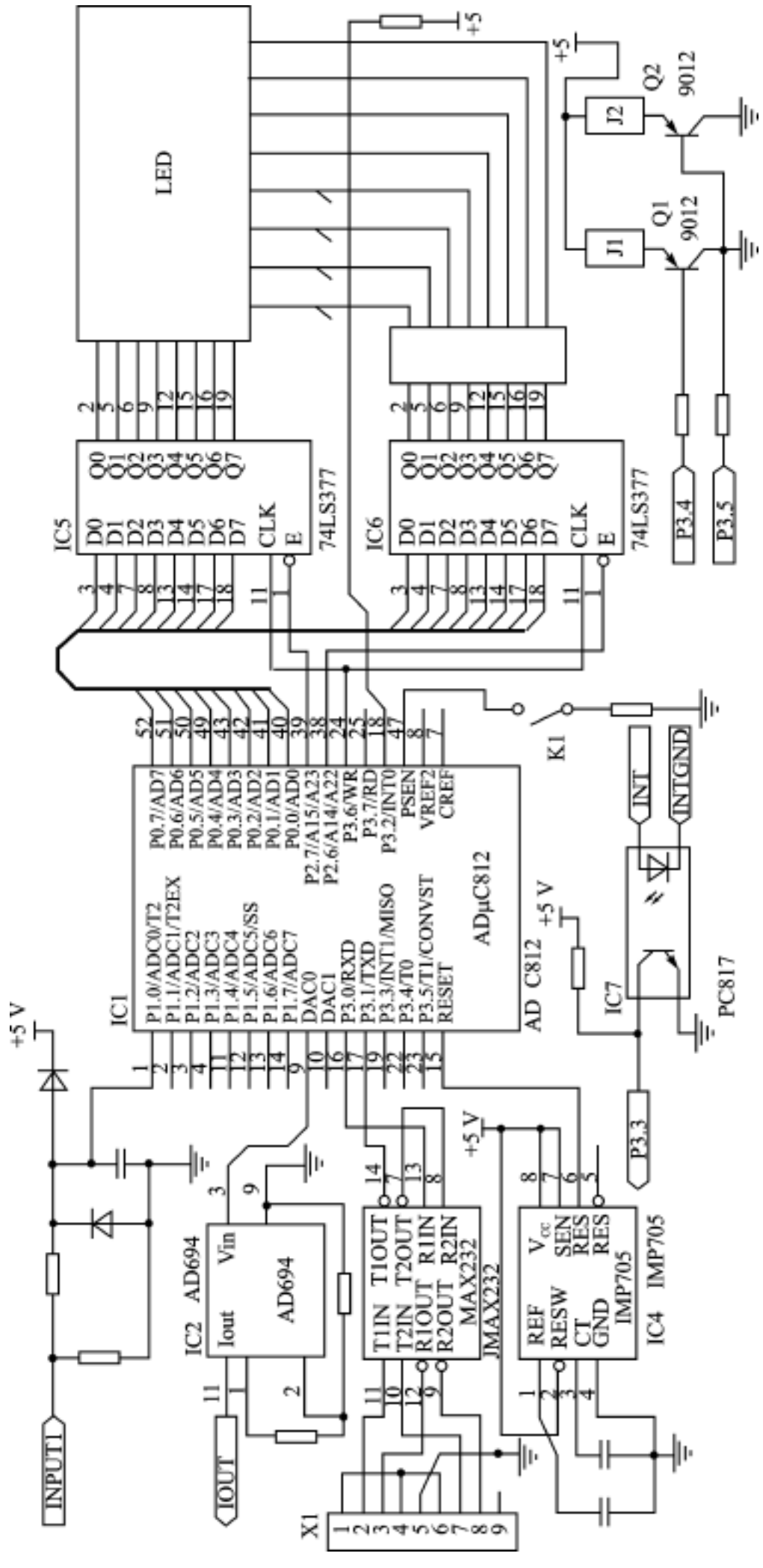


图 5 9-1 硬件构成原理图

由于 ADμC812 与 8051 兼容的内核,对于原来的 8051 的用户使用就非常的方便。本文只介绍内部新的单元(ADC、DAC、闪速/电擦除数据存储器)的使用方法及子程序。

1.ADC 使用方法及子程序

ADC 有两个控制寄存器和一个状态寄存器,ADCCON1 寄存器控制转换和采集时间、硬件转换模式及掉电模式。ADCCON2 寄存器控制 ADC 通道选择和转换模式。ADCCON3 寄存器对用户软件给出 ADC 忙状态的指示。通过对两个控制寄存器的设置,ADC 可工作在单次转换、ADC 中断方式及 ADC DMA 方式。本子程序采用单次转换方式,8 个通道 AD 采样值分别存放在 ADμC812 内部 RAM 50H～5FH 单元中。

```
ADC: PUSH    RO
       USH      1
     PUSH     R2
MOV  ADCCON1, #68H
     MOV      R0, #50H
     MOV      R1, #10H
     MOV      R2, #8
AD1: MOV      ADCCON2, R1
     LCALL    DELAY    ;延时
     MOV      @R0, ADCDATAL
     INC      R0
     MOV      R0, ADCDATAH
     INC      R0
     INC      R1
     DJNZ     R2, AD1
     POP      R2
       POP      R1
       POP      R0
     RET
```

2.DAC 使用方法及子程序

ADμC812 组合了两个片内 12 位 DAC,通过设置 DAC 控制寄存器 DACCON,可以使 DAC 工作在 8 位或 12 位模式、电压输出范围在 0～V_{DD} 或 0～V_{REF} 之间、强制 DAC 输出电压为 0 V 及上电或掉电方式。子程序中 DAC 工作在 12 位模式,电压输出范围在 0～V_{DD}。ADμC812 内部 RAM 40H 存放通道 0 的低 8 位输出量,41H 存放通道 0 的高 4 位输出量;42H、43H 存放通道 1 的 12 位输出量。

```
DAC:  OV     ACCON, #8FH
      MOV    DAC0L,40H
      MOV    DAC0H,41H;DAC0
      MOV    DAC1L,42H
      MOV    DAC1H,43H;DAC1
      RET
```

3．电擦除数据存储器使用方法及子程序

闪速/电擦除数据存储器阵列包括640 B,被配置为160(00H～9FH)页,每页4 B。其操作有4个数据寄存器组(EDATA1～4)用于保存读、写的4 B页数据,EADRL为页地址寄存器,ECON是8位控制寄存器,值为01H读命令、02H写命令、04H校验命令、05H擦除命令,以上操作都是按页进行。

读子程序:(A存放读出数据)

```
ERD:   OV      ADRL,R1                R1 页地址
       MOV     ECON, #01H
       CJNE    R0, #1,ER1             ;R0 页内字节号
       MOV     A,EDATD1               ;(1～4)
       RET
ER1:  CJNE R0, #2,ER2
       MOV     A,EDATD2
       RET
ER2:  CJNE R0, #3,ER3
       MOV     A,EDATD3
       RET
ER3:  MOV      A,EDATD4
RET
```

写子程序:(A存放写入数据)

```
EWD: MOV      EADRL,R1               ;R1 页地址
       MOV     ECON, #01H
       CJNE    R0, #1,EW1             ;R0 页内字节号
       MOV     EDATD1,A
       AJMP    EW4
EW1: CJNE R0, #2,ER2
       MOV     EDATD2,A
       AJMP EW4
EW2: CJNE R0, #3,ER3
       MOV     EDATD3,A
       AJMP    EW4
EW3: MOV      EDATD4,A
EW4: MOV ECON, #05H
       MOV ECON, #02H
       RET
```

4．软件开发工具

AD公司提供的软件开发工具有汇编语言编译器ASM51、调试器DEBUG、程序下载工具DOWNLOAD以及软件仿真器ADSIM812。尤其是软件仿真器,不但仿真ADμC812单片机程序,还可以仿真8031单片机的程序。软件开发工具可从网上下载。

三、结束语

该仪表适用于科研单位的工业自动化控制项目及高等学校自动化专业的实验教学工作等。具有结构简单、操作方便、成本低等优点。

参 考 文 献

1 西安虹飞公司. ADμC812 产品介绍[Z].

2 nalog Devices. MicroConverter, Multi-Channel 12 bit ADC with Embedded MCU ADμC812 Preliminary Technical Data[Z].

选自《电测与仪表》月刊,2001 年第 11 期

5.10 基于 P87LPC764 的 Σ－Δ ADC 应用设计方法

广州周立功单片机发展有限公司

一、简 介

Σ－Δ ADC 主要使用数字技术,具有数字系统高可靠性、高稳定性的优点,也为用单片机软件实现 ADC 提供了可能性。

Σ－Δ ADC 具有分辨率高、线性度好、抗干扰能力强(对噪声的抑制能力不亚于双积分 ADC)、成本低等不可多得的优点。Σ－Δ ADC 以很低的采样分辨率(1 位)和很高的采样速率将模拟信号数字化,通过使用采样、噪声整形和数字滤波等方法增加有效分辨率,然后,对 ADC 输出进行采样抽取,以降低有效采样速率。

本文研究用低成本单片机实现 8 位精度的 Σ－Δ ADC 的软件实现方法,并给出具体线路及源代码。关于 16 位分辨率 ADC 实现的技术较为复杂,后续将有文章详细介绍。

本方案适用于 PHILIPS 51LPC 系列 P87LPC760/761/762/764 等内含比较器的单片机,经在摩托车车速里程表、电子秤(人体秤、脂肪秤、厨房秤)、电热水器、电饭锅、消毒碗柜、电冰箱等一系列产品中实际使用证明:不受 RC 器件的误差、PCB 板的分布电容及环境温度变化的影响,软件便于无障碍地移植,测量结果一致性好,实用、可靠、价格低廉,且能够满足大多数速度要求不太高的产品。

二、原 理

Σ－Δ ADC 主要由调制器、数字滤波和采样抽取等几部分组成,如图 5.10－1 所示。调制器本质上是一个高速低精度(1 位)的 ADC,调制器以非常大的过采样率采样模拟信号。在这个阶段,调制器将输入和输出之间的差值(Δ)进行一阶或多阶积分(Σ),结果通过一个量化器(1 位 ADC)输出二进制码流。该码流一方面输出给数字滤波,另一方面通过一位 DAC 后和输入信号比较产生差值信号,继续反馈循环。通过数字滤波和采样抽取,滤除经过调制器整形后的量化噪声,提高系统精度。采样抽取的底限是满足信号无损重建的采样定律,采样频率大于奈奎斯特频率两倍($f_n > 2f_b$)。

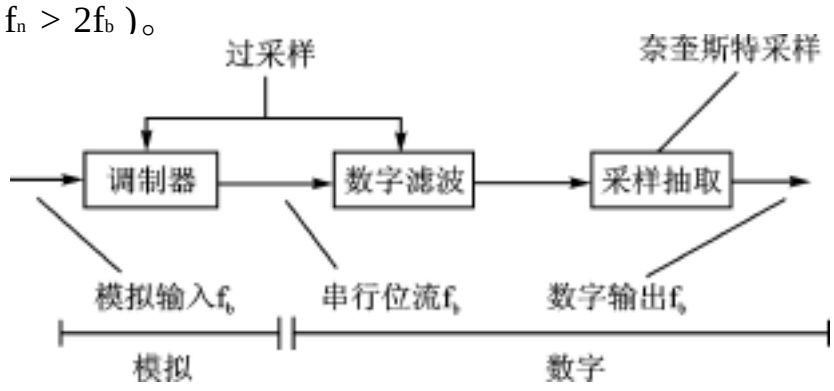


图 5.10－1 Σ－Δ ADC 组成框图

1. 调制器数学模型

图 5.10－2 所示为调制器的组成框图,其中一位 A/D 的量化器由比较器构成,检测出差值积分后的正负结果转换成 1、0 的位流。转换过程中会产生基于转换精度的量化误差(±1/2LSB),它是分布在整个频域范围内的白噪声。

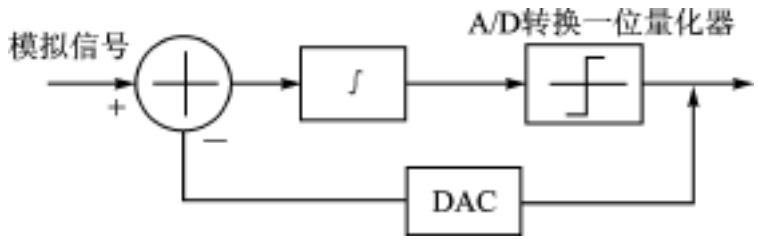


图 5.10－2　调制器数学模型

调制器的离散描述如图 5.10－3 所示。

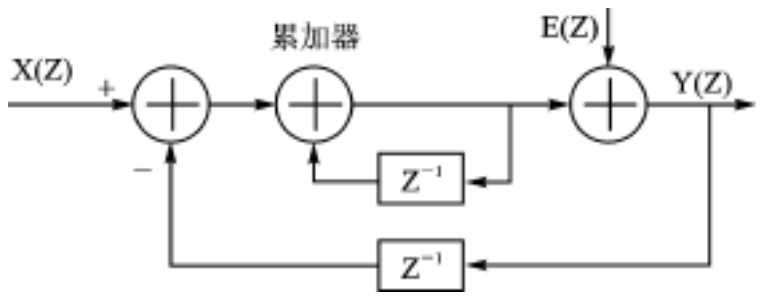


图 5.10－3　调制器离散性描述图

积分器在 Z 域由具有如下传递函数的累加器构成:

$$H(z) = \frac{1}{1 - z^{-1}}$$

系统的 Z 域描述如下:

$$Y(z) = \frac{H(z)}{1 + z^{-1}H(z)}X(z) + \frac{1}{1 + z^{-1}H(z)}E(z) = X(z) + (1 - z^{-1})E(z)$$

输入信号无损地传到输出端 W(z)=1。量化噪声的传递函数 $N(z) = 1 - z^{-1}$,对应于频域的一阶高通滤波器(HP1),传输特性如图 5.10－4 中 HP1 线所示。

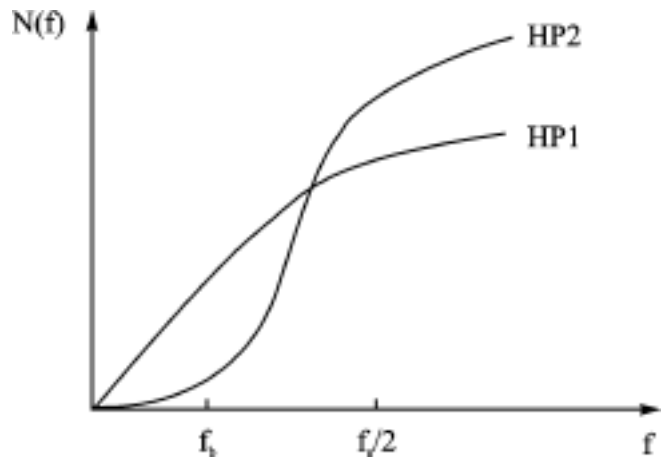


图 5.10－4　噪声传输特性图

噪声传输特性图 5.10－4 中 HP2 线是二阶系统的噪声传输特性。当采用二阶以上系统时,通带内的量化噪声已基本滤除,噪声被转移到所关心的频带以外,将通过数字滤波去除。

2. 过采样对量化噪声的影响

奈奎斯特采样时,量化噪声的能谱密度为

$$S_E(f) = \frac{1}{f_a} \cdot \frac{1}{\Delta}\int_{-\Delta/2}^{+\Delta/2} e^2\,de = \frac{\Delta^2}{12f_a}$$

其中 Δ 为量化步长，f_a 为模拟信号经抗混叠低通滤波后的通带频率(＝奈奎斯特频率)。通带范围内，噪声电压的有效值为 $\Delta/\sqrt{12}$，如图 5.10－5 所示。

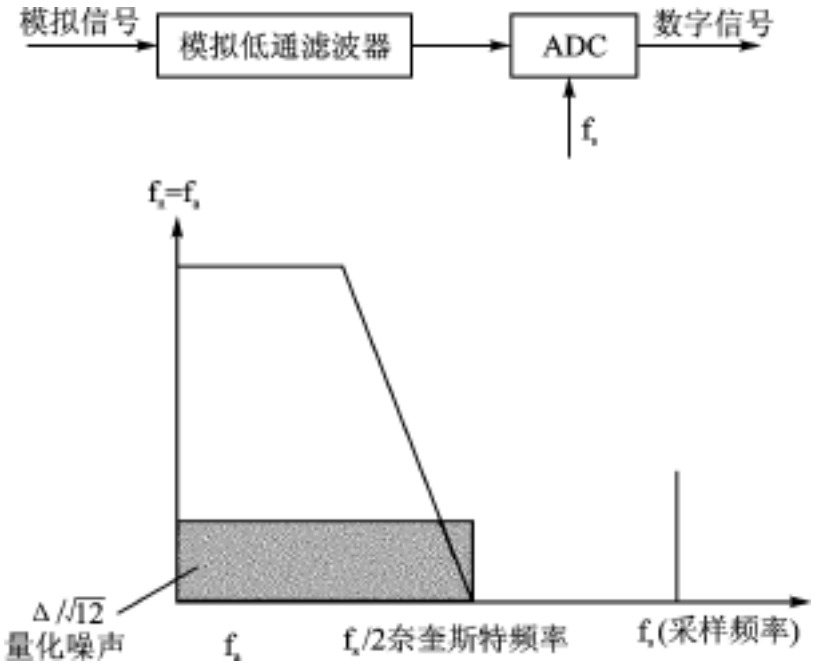


图 5.10－5 带模拟低通滤波器采样频谱图

当以 Kf_s 的采样速率进行过采样时，奈奎斯特频率增至 $Kf_s/2$，量化噪声位于 0～(1/2)Kf_s 之间，其有效值降为原来的 $1/\sqrt{k}$，调制器将噪声转移到 $f_s/2$ ～$kf_s/2$ 之间，这部分噪声由数字低通滤波器滤除，如图 5.10－6 所示。

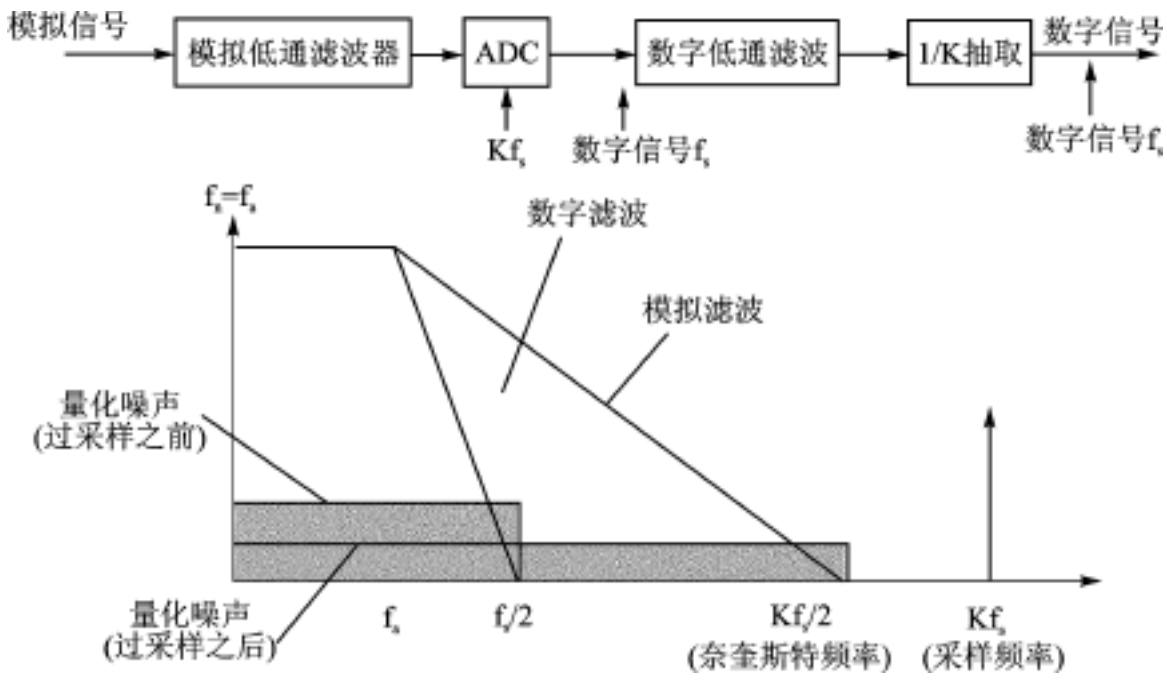


图 5.10－6 带模拟低通及数字滤波器的采样频谱图

3．信噪比以及精度转换

通过以上分析可知，提高调制器的阶数(n)和过采样率(OSR)均可提高系统的信噪比 SNR。

$$SNR = 10\lg\frac{3(2n+1)}{2\pi^{2n}}\text{dB} + (2n+1)\cdot 10\lg(OSR)\ \text{dB} \qquad (1)$$

b 位 ADC 的理想信噪比(动态范围 DR)可用如下方式求得。

信号功率：

$$P_S = \frac{A^2}{2} = \frac{(2^{b-1}\Delta)^2}{2}$$

噪声信号功率(白噪声):

$$P_R = \frac{1}{\Delta}\int_{-\Delta/2}^{+\Delta/2} e^2\,de = \frac{\Delta^2}{12}$$

信噪比:

$$SNR_{max} = 10\log\frac{P_S}{P_R} = b \cdot 6.02\ dB + 1.76\ dB \qquad (2)$$

由以上关系可得图 5.10－7 所示关联图。

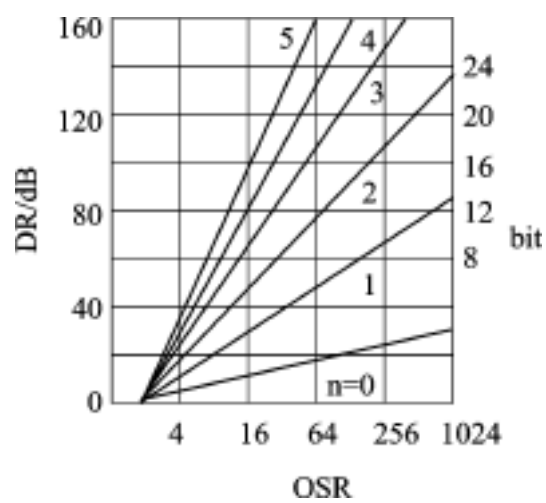


图 5.10－7 信噪比以及精度转换关联图

图 5.10－7 中,左边纵轴为信噪比 SNR(或动态范围 DR),单位为 3 dB;右边纵轴为位数;横轴为过采样率(OSR)。函数线由式(1)计算所得,n 为调制器的阶数。右纵轴的分布由式(2)计算所得。

DR 为信号的动态范围,即最大信号与杂散信号之比。由于此处输入的为理想信号,无谐波分量,杂散信号只有量化噪声,故动态范围即为信噪比 SNR。

由图 5.10－7 可知,8 位 ADC 的理想信噪比大致为 45 dB,采用一阶调制器实现需要大于 64 的过采样率。实现 12 位精度需要大于 380 的过采样率,16 位需大于 2415。

三、实　现

1. 基本方案

如图 5.10－8 所示为用单片机实现一阶 Σ－Δ ADC 的原理图,虚线框内为 MCU。

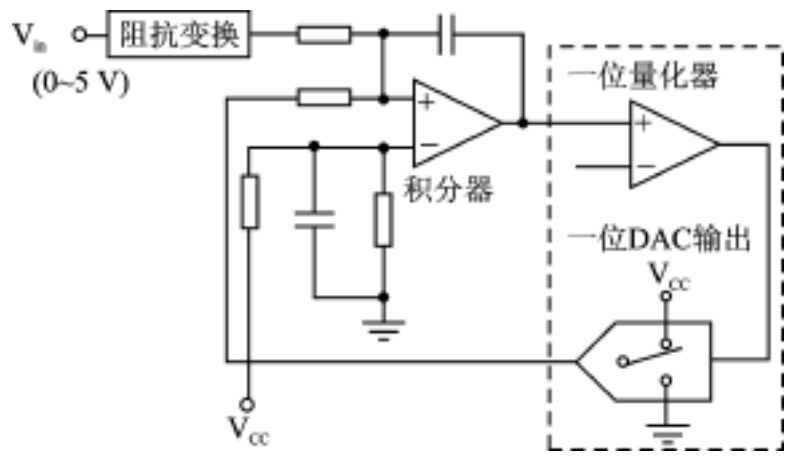


图 5.10－8 一阶 Σ－Δ ADC 的基本方案原理图

其中,阻抗变换器实际为一跟随器,用于减小信号源内阻的影响。如果不考虑边缘的非线性,积分器可用简单的 RC 网络替代,如图 5 .10 - 9 所示。

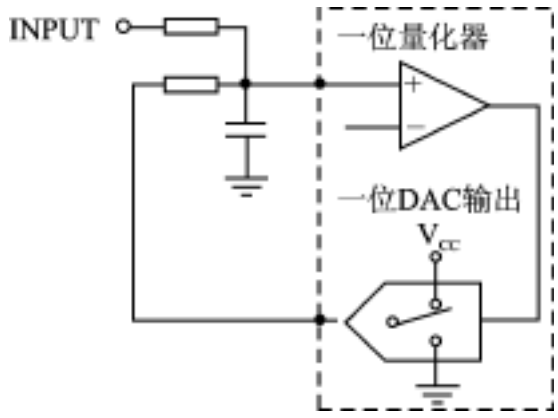


图 5 .10 - 9　一阶 Σ - Δ ADC 的简化方案原理图

一位量化器可用单片机的内部比较器,也可以用普通口线替代,只是不同的单片机口线翻转电平不一样,测量结果差别很大。一位 DAC 用口线置高或置低来模拟,需较强的高电平驱动能力。图 5 .10 - 10 所示为用 P87LPC764 内部比较器实现的 8 位 ADC 原理图,比较器偏置在(1/ 2) V_{CC},输入信号的内阻必须足够小,必要时可加跟随器。

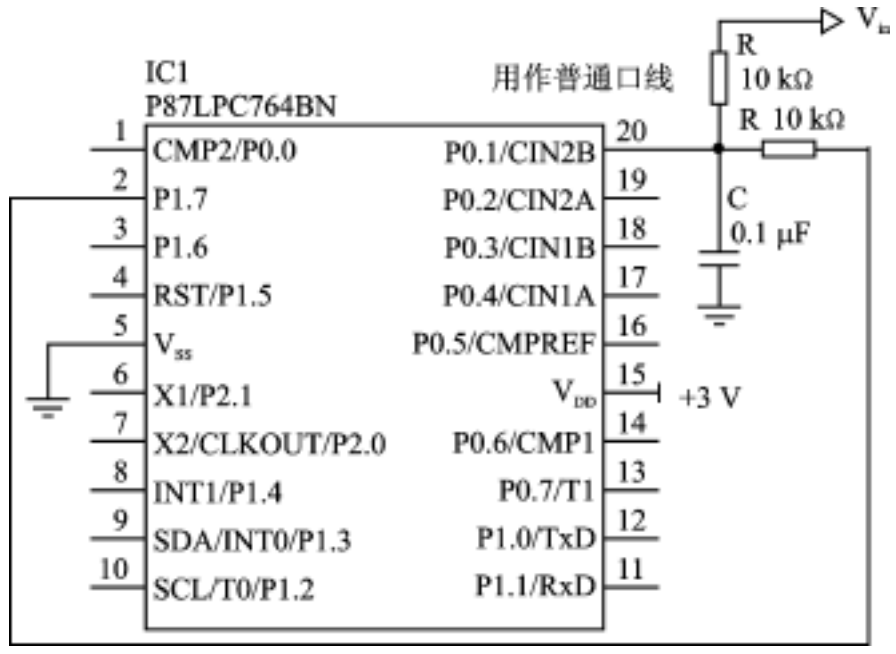


图 5 .10 - 10　8 位 Σ - Δ ADC 基本方案原理图

```
;特殊功能寄存器定义
  P1M1    DATA  91H      ;P1 方式寄存器 1
  P1M2    DATA  92H      ;P1 方式寄存器 2
  PT0AD   DATA  0F6H
  P0M1    DATA  84H      ;P0 方式寄存器 1
  P0M2    DATA  85H      ;P0 方式寄存器 2
  CMP1    DATA  0ACH     ;比较器 1 控制寄存器
  CMP2    DATA  0ADH     ;比较器 2 控制寄存器
;变量定义
;端口定义
  AD_CON   BIT   P1 .6
```

```
    ORG     0000H
    AJMP    START

;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
START:
    MOV     SP,#60H
    MOV     PT0AD,#24H          ;禁止 CMPREF、CIN2A
    ANL     P0M2,#0DBH
    ORL     P0M1,#24H
    MOV     CMP2,#20H           ;CIN2A 为输入端
MAINLOOP:
    LCALL   AD_CONV
    SJMP    MAINLOOP
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
;Σ-Δ ADC 转换程序
;占用 R2,R3,R4,R5 结果于 A 中
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
AD_CONV:
    ORL     P1M2,#40H
    ANL     P1M1,#0BFH          ;AD_CON 设为上拉
AD_START:
    MOV     R4,#2               ;循环次数
    MOV     R2,#0               ;总计数值 256 次
AD_TEST:
    MOV     R3,#0               ;TON 计数值 256,开始转换
AD_LOOP:
    MOV     A,CMP2
    JB      ACC.1,AD_HIGH
    SETB    AD_CON              ;输出正脉冲
    NOP
    NOP
    SJMP    AD_COUNT
AD_HIGH:
    CLR     AD_CON              ;输出负脉冲
    CJNE    R3,#255,AD_INC
    SJMP    AD_COUNT
AD_INC:
    INC     R3
    NOP
AD_COUNT:
    LCALL   DELAY10us
    DJNZ    R2,AD_LOOP          ;未完成 256 次循环,继续
    DJNZ    R4,AD_TEST
```

```
    MOV     A,R3                    ;转换完成,存结果于 A
    RET
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
DELAY10us:
    MOV     R5,#10
    DJNZ    R5,$
    RET
```

2.试验结果

表 5.10-1 为试验测试报告。测试条件:V_{CC} = 4.92 V,晶振 12 MHz,采用以上线路和源代码。由于没有采用跟随器,结果有些偏差。

表 5.10-1 测试报告

输入电压/V	测量值	期望值	误差
0	0	0	0
0.05	6	2.59	3.41
0.50	29	25.91	3.09
1.00	54	51.81	2.19
1.50	78	77.72	0.18
2.00	104	103.63	0.37
2.50	128	131.58	-3.58
3.00	153	155.44	-2.44
3.50	180	181.35	-1.35
4.00	204	210.53	-6.53
4.50	229	236.84	-5.84
4.92	251	255	-4

3.扩充方案

(1) 原 理

图 5.10-11 所示为一阶 Σ-Δ ADC 原理图。该方案由于相对于被测信号量化,被测信号即为比较基准,被测信号和基准之间无电压差,其积分为零,故只对反馈信号求积。电容 C 的信号围绕 V_{in} 上下变化,完成 Σ-Δ ADC 差值(Δ)和积分(Σ)的作用,实现比较巧妙,完全符合一阶 Σ-Δ 模型。输入信号端加一电阻电容,形成一阶 RC 低通滤波器,滤除高频杂散信号,有抗混叠滤波的作用,同时也可补偿比较器的失调误差。

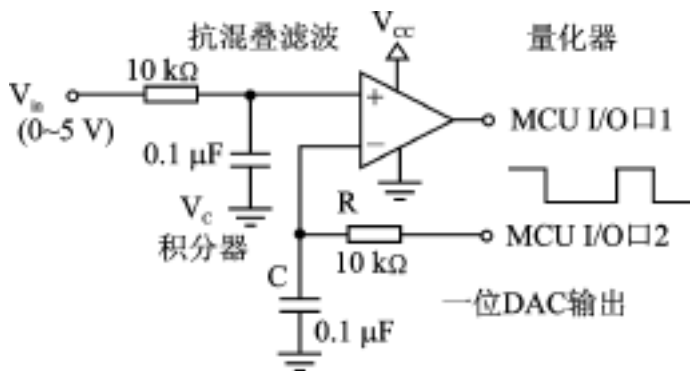


图 5.10-11 一阶 Σ-Δ ADC 扩充方案原理图

图 5.10－11 所示电路,采样频率 500kHz(20μs 采样一次),过采样率 OSR = 256 > 64,输出数据速率 1.95 kHz,采用一级 256 阶梳状滤波后完全可满足 8 位精度的要求。

另外,这种电路形式对信号源的输入阻抗没有要求,因而可达到更高的精度。

为了进一步认识其工作过程,作一下时域分析。1 位 DAC 输出可看作 PWM 脉冲。

对于 RC 时间常数远大于脉冲宽度标准 PWM 信号,有

$$V_{in} = V_{ref}[T_{on}/(T_{on} + T_{off})] \tag{3}$$

如果分级地调整波形使较大的时间常数稳定下来,这将消耗很长的时间。如采用对分搜索,可大大加快计算过程,可能效果会比较理想,但要花费较大的软件资源。

这里,我们采用一种新的思路(Σ－Δ),使用相对小的时间常数,每 20μs 对比较器作一次采样。$V_C < V_{in}$ 时,输出为正脉冲,反之为负脉冲。这种方法在固定周期内把 PWM 波形分解成数个小脉冲,代替表示占空比的单脉冲。

$$V_{in} = V_{ref}[\Sigma T_{on}/\Sigma(T_{on} + T_{off})] \tag{4}$$

我们把该系统看作一个动态的控制系统,系统动态平衡时,电容的充电电荷等于其放电电荷。V_C 在 V_{in} 上下波动,$V_C \approx V_{in}$,故充放电公式写作:

$$V_{charge} = V_{out} - V_C)[1 - \exp(-t/RC)] \approx (V_{out} - V_{in})[1 - \exp(-t/RC)]$$

$$V_{discharge} = V_C - V_0)[1 - \exp(-t/RC)] \approx (V_{in} - V_0)[1 - \exp(-t/RC)]$$

令 = T_{on} 脉冲的总次数;

m = T_{off} 脉冲的总次数;

t = 采样时间。

则 $n(V_{out} - V_{in})[1 - \exp(-t/RC)] = m(V_{in} - V_0)[1 - \exp(-t/RC)]$

解得

$$V_{in} = (nV_{out} + mV_0)/(n + m) \tag{5}$$

其中 $V_{out} = V_{ref} - V_{os}$,$V_{os}$ 为口线的失调电压,$V_{ref} = V_{CC}$,V_0 为口线的低电平电压。所以

$$V_{in} = nV_{ref}/(n + m) - (nV_{os} - mV_0)/(n + m) \tag{6}$$

前面一项为测量结果,后面为误差,方程中不存在 RC 值,它不是误差因子。比较器的失调电压和口线的失调电压均会影响测量精度。前者使 V_C 不能正确地跟踪 V_{in} 而造成误差。

(2) 利用 P87LPC764 内部比较器实现 8 位 Σ－Δ ADC

这里,用 P87LPC764 的内部比较器来实现,如图 5.10－12 所示。在正确测量之前,必须

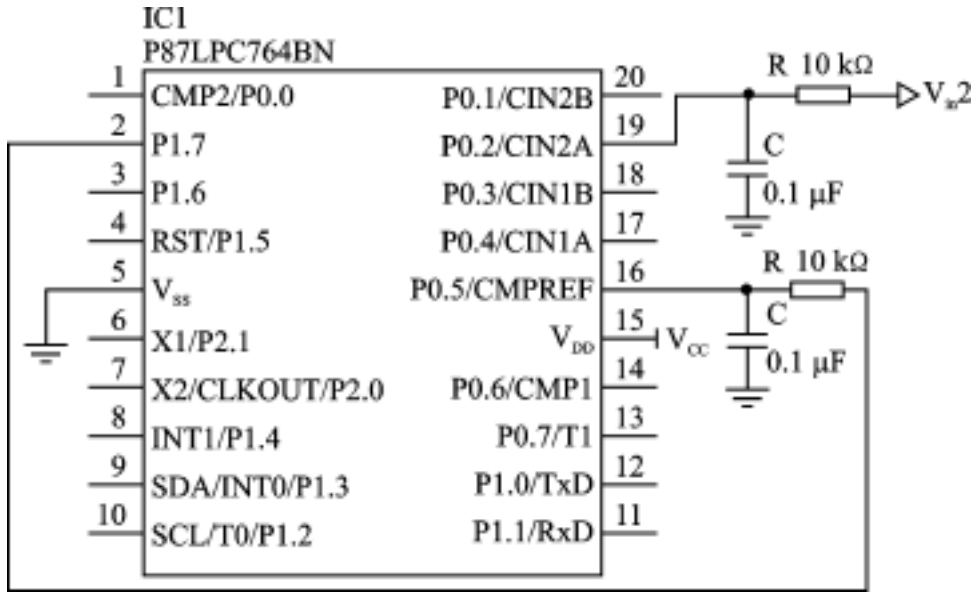


图 5.10－12 8 位 Σ－Δ ADC 扩充方案原理图

保证 $V_C = V_{in}$ (实际有一微小误差)。为此,进行两次测量:第一次使 V_C 锁定 V_{in},扔掉测量结果;第二次进行真正测量。8 位精度需 256 个计数。

图 5 .10 - 13 是 8 位 Σ - Δ ADC 实现原理流程图。

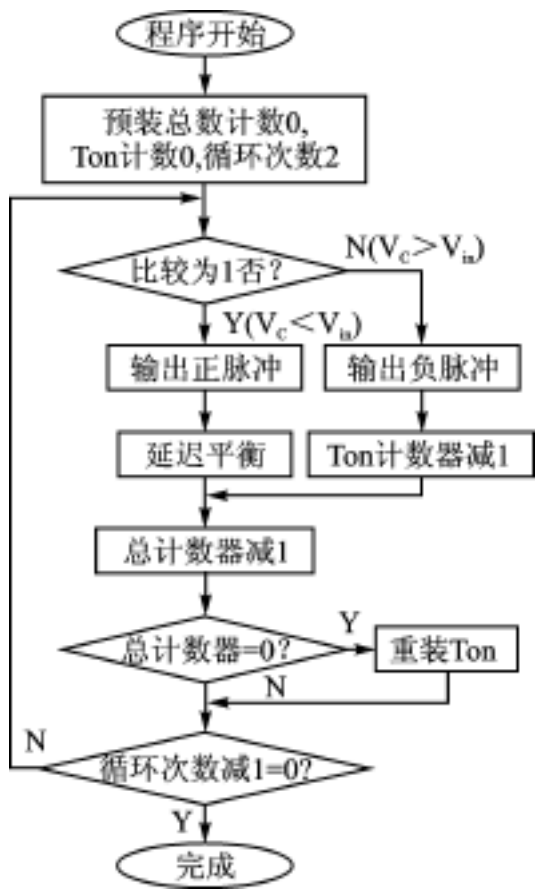


图 5 .10 - 13　8 位 Σ - Δ ADC 实现原理流程图

```
;特殊功能寄存器定义
  PT0AD  DATA  0F6H
  P0M1   DATA  84H      ;P0 方式寄存器 1
  P0M2   DATA  85H      ;P0 方式寄存器 2
  P1M1   DATA  91H      ;P1 方式寄存器 1
  P1M2   DATA  92H      ;P1 方式寄存器 2
  CMP1   DATA  0ACH     ;比较器 1 控制寄存器
  CMP2   DATA  0ADH     ;比较器 2 控制寄存器
;变量定义
;端口定义
  AD_CON   BIT   P1 .7
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;Σ - Δ ADC 转换程序
;占用 R2,R3,R4,R5 结果于 A 中
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
AD_CONV:
  ANL    P1M1, #7FH
  ORL    P1M2, #80H      ;AD_CON 设为上拉
  MOV    PT0AD, #24H     ;禁止 CMPREF、CIN2A
  ANL    P0M2, #0DBH
  ORL    P0M1, #24H
```

```
    MOV     CMP2, #20H      ;比较器 2 的 CIN2A 为输入端
AD_START:
    MOV     R4, #2          ;循环次数
    MOV     R2, #0          ;总计数值 256 次
AD_TEST:
    MOV     R3, #0          ;Ton 计数值 256 次,开始转换
AD_LOOP:
    MOV     A,CMP2
    JB      ACC.1,AD_HIGH
    CLR     AD_CON          ;输出负脉冲
    DEC     R3
    SJMP    AD_COUNT
AD_HIGH:
    SETB    AD_CON          ;输出正脉冲
    NOP
    NOP
    NOP                     ;平衡时间
AD_COUNT:
    LCALL   DELAY10us
    DJNZ    R2,AD_LOOP      ;未完成 256 次循环,继续
    DJNZ    R4,AD_TEST
    MOV     A,R3            ;转换完成,存结果于 A
    RET
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
DELAY10us:
    MOV     R5, #10
    DJNZ    R5, $
    RET
```

表 5.10-2 为测试报告。测试条件:V_{CC} = 4.87 V,晶振 6 MHz 采用以上线路和源代码。

表 5.10-2 测试报告

输入电压/V	测量值	期望值	误差
0	0	0	0
0.05	3	2.63	0.37
0.50	27	26.19	0.81
1.00	53	52.36	0.64
1.50	79	78.53	0.47
2.00	105	104.71	0.29
2.50	131	130.89	0.21
3.00	157	157.07	-0.07
3.50	183	183.25	-0.25
4.00	209	209.42	-0.42
4.50	235	235.60	-0.60
4.87	254	255	-1.00

由以上结果可知,Σ－Δ ADC 转换具有非常高的准确度和线性度。

(3) 利用 P87LPC764 双比较器实现 4 路 A/ D

图 5 .10－14 为 4 路 8 位 Σ－Δ ADC 原理图。

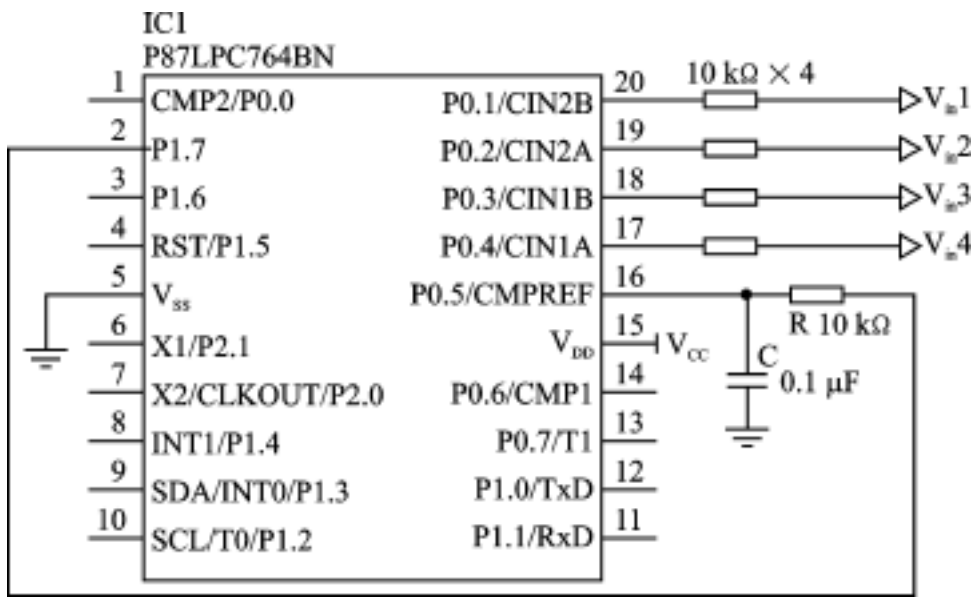


图 5 .10－14　4 路 8 位 Σ－Δ ADC 原理图

```
;特殊功能寄存器定义
  PT0AD  DATA  0F6H
  P0M1   DATA  84H      ;P0 方式寄存器 1
  P0M2   DATA  85H      ;P0 方式寄存器 2
  P1M1   DATA  91H      ;P1 方式寄存器 1
  P1M2   DATA  92H      ;P1 方式寄存器 2
  CMP1   DATA  0ACH     ;比较器 1 控制寄存器
  CMP2   DATA  0ADH     ;比较器 2 控制寄存器
;变量定义
  AD_BUF  DATA  30H     ;结果单元
;端口定义
  AD_CON  BIT   P1 .7
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
AD_4_CH:
  ANL    P1M1, #7FH
  ORL    P1M2, #80H     ; AD_CON 设为上拉
  MOV    PT0AD, # 3EH   ;禁止 CMPREF、CIN1A、
                        ;CIN1B、CIN2A、CIN2B 数字输入、输出
  ANL    P0M2, #0C1H
  ORL    P0M1, #3EH
  MOV    R0, # AD_BUF
  MOV    CMP2, #30H     ;比较器 2 的 CIN2B 为输入端
  LCALL  AD_START2
  MOV    @R0, A         ;存第一路结果
  INC    R0
  MOV    CMP2, #20H     ;CIN2A 为输入端
```

```
    LCALL  AD_START2
    MOV    @R0,A          ;存第二路结果
    INC    R0
    MOV    CMP1,#30H      ;CIN1B 为输入端
    LCALL  AD_START1
    MOV    @R0,A          ;存第三路结果
    INC    R0
    MOV    CMP1,#20H      ;CIN1A 为输入端
    LCALL  AD_START1
    MOV    @R0,A          ;存第四路结果
AD_START1:
    MOV    R4,#2          ;循环次数
    MOV    R2,#0          ;总计数值 256 次
AD_TEST1:
    MOV    R3,#0          ;TON 计数值 256 次,开始转换
AD_LOOP1:
    MOV    A,CMP1
    JB     ACC.1,AD_HIGH1
    CLR    AD_CON         ;输出负脉冲
    DEC    R3
    SJMP   AD_COUNT1
AD_HIGH1:
    SETB   AD_CON         ;输出正脉冲
    NOP
    NOP
    NOP                   ;平衡时间
AD_COUNT1:
    LCALL  DELAY10us
    DJNZ   R2,AD_LOOP1    ;未完成 256 次循环,继续
    DJNZ   R4,AD_TEST1
    MOV    A,R3           ;转换完成,存结果于 A
    RET
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
AD_START2:
    MOV    R4,#2          ;循环次数
    MOV    R2,#0          ;总计数值 256 次
AD_TEST2:
    MOV    R3,#0          ;TON 计数值 256 次,开始转换
AD_LOOP2:
    MOV    A,CMP2
    JB     ACC.1,AD_HIGH2
    CLR    AD_CON         ;输出负脉冲
    DEC    R3
```

```
    SJMP    AD_COUNT2
AD_HIGH2:
    SETB    AD_CON          ;输出正脉冲
    NOP
    NOP
    NOP                     ;平衡时间
AD_COUNT2:
    LCALL   DELAY10us
    DJNZ    R2,AD_LOOP2     ;未完成 256 次循环,继续
    DJNZ    R4,AD_TEST2
    MOV     A,R3            ;转换完成,存结果于 A
    RET
;- - - - - - - - - - - - - - - - - - - - - - - - - - - -
DELAY10us:
    MOV     R5,#10
    DJNZ    R5,$
    RET
```

5 .11　MP3 解码芯片组及其应用

湖南大学电气与信息工程学院(410082)
周志刚　黎福海　欧阳典勇　王永学

一、MP3 概述

作为一场堪称音乐工业的千禧年风暴,MP3 音乐已通过互联网络,以雷霆万钧之势席卷全球,给数字音频领域带来了一股新的冲击。MP3 是 MPEG Layer3 的缩写,它是一种超级声音文件的压缩方法。MPEG 由音频和视频两部分组成,可以分别进行压缩。MPEG 在音频上压缩分为 MPEG Layer1、MPEG Layer2、MPEG Layer3,其中 MP3 具有最高的压缩比 12∶1。在一般没有压缩数据的情况下,音频被数字化时,采样频率必须在实际声音最高频率的 2 倍以上。在 CD 中,声音最高频响是 20 kHz,采样频率定为 44 .1 kHz,16 位量化。要获得 CD 音质的立体声,每秒钟的数据量将超过 1 .4 Mbit。采用 MP3 压缩,数据量可以缩小到 1/ 12,音质却没有损失。如果再进一步压缩数据量到 1/ 24 或更多,依然可以维持相当好的音质,比起通过降低采样频率、缩短采样深度的方法要好得多。

采用 MPEG 音频压缩方法,获得与 CD 一样的音质。要求的典型数据如表 5 .11 -1 所列。

表 5 .11 -1

MPEG 音频层	压缩比	立体声信号数据量
MP1	1∶4	384 kbps
MP2	1∶6～1∶8	192～256 kbps
MP3	1∶10～1∶12	112～128 kbps

在 MPEG 音频编码模式中,MP3 功能最强大。在同样的音质条件下,MP3 需要的数据量最小。在同样的数据量条件下,MP3 音质最好。

MP3 作为高质量音乐压缩标准,正在进入越来越多人的生活。在因特网上,MP3 已成为事实标准,有众多可供下载 MP3 音乐文件的站点。今天,有数以千计的艺术家以 MP3 文件格式发布他们的音乐作品,并不断涌现 MP3 编解码软件和硬件设备。MAS3507D 和 DAC3550A 解码芯片组是专为便携式系统而设计的,并且采用了超大规模集成电路和超低功耗省电模式设计。

对于便携式系统,主要考虑的是其体积小巧、低功耗、高容量、低价格。MP3 播放机主要结构如图 5 .11 -1 所示。

图 5 .11 -1　MP3 播放机的结构

二、MP3 解码芯片组内部结构

MAS3507D 和 DAC3550A 是 MICRONAS INTERMENTALL 公司专为个人音频播放器以及 MP3 因特网音频播放器设计的芯片组。MAS3507D 在一块芯片上嵌入了 RISC DSP CORE,除此之外还有电源管理器(POWER MANAGEMENT)、程序存储器(INTERNAL ROM)、时钟管理器(CLOCK MANAGEMENT)、音频基带处理器(BASEBAND PROCESSOR),并有 I²S、I²C、PIO 等多种接口。MAS3507D 的强大功能,使它可以轻松完成 MP3 音频解码,内部结构如图 5.11-2 所示。DAC3550A 是与 MAS3507D 匹配的高质量的多采样率 DAC,可支持采样率为 8～50 kHz,具有内部振荡器和耳机放大器,它把 MP3 播放机结构中的后两部分集成在了一起,使之结构更紧凑。并且它还利用了 MICRONAS INTERMETALL 独特的多位 Σ-Δ 技术,使 D/A 转换具有高线性、出色的 S/N(可达 103 dBA)。其内部结构如图 5.11-3 所示。

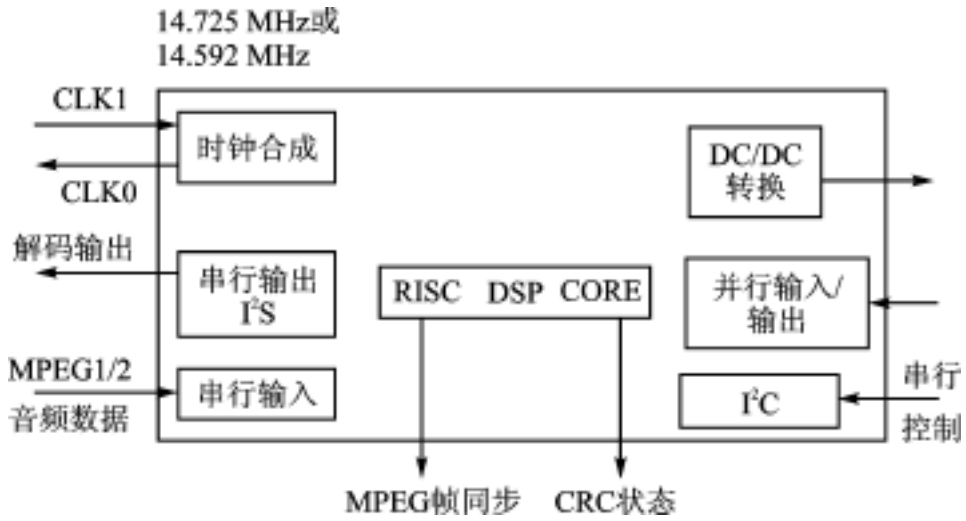


图 5.11-2 MAS3507D 内部结构图

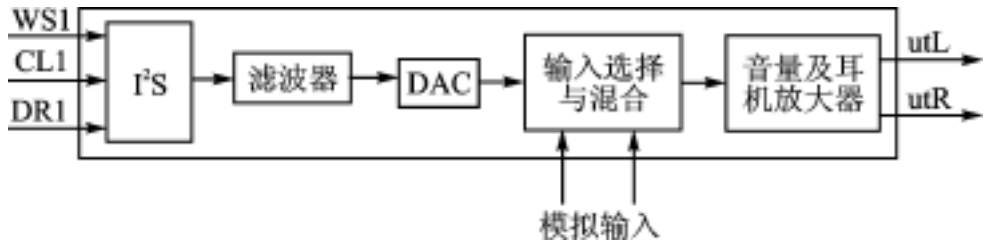


图 5.11-3 DAC3550A 内部结构图

三、MP3 解码芯片组性能特点

1. MAS3507D 的性能

MAS3507D 是单芯片解码芯片,片上具有 MPEG Layer2/3 音频解码可供选择使用,可支持音频广播和基于存储器的多媒体回放应用。

数据处理是由高性能的 RISC DSP 内核完成的,并具有嵌入式存储器,通过下载软件还可实现 CELP 语音解码和 ADPCM 编/解码。

串行异步 MPEG 数据流输入,具有 CRC 检测及 MPEG 帧同步检测,并有相应的辅助状态数据输出。

完备的电源管理,具有内置的升压变换器,工作电压范围很宽,1.6～3.6 V 均可正常工

作,可由两节电池来供电,功耗极低。在 3 V 工作电压情况下,当 $f_s \leq 12$ kHz 时,系统功耗低于 53 mW;当 $f_s < 24$ kHz 时,功耗为 90 mW;当 $f_s > 24$ kHz 时,功耗为 165 mW,并且还具有自动 Power－off 功能,非常适合于便携式 MPMan。

集成 I^2S 的音频输出,可方便地与 DAC3550A 连接,也可与其他低价格、低功耗、高品质的 DAC 连接,并可改变输出的数据格式。

集成了数字、音量、立体声、声道混合、低音和高音控制。

2 . DAC3550A 的性能

DAC3550A 是与 MAC3507D 匹配的音频 DAC,内部集成了立体声耳机放大器和单声道扬声器放大器,信噪比可达 103 dBA;内置时钟振荡器并有时钟输出提供给 MAS3507D,支持采样率 8～50 kHz;具有 I^2S 总线音频输入和 I^2C 控制总线;宽电压设计,支持 2 .7～5 .5 V 和低功耗模式。

四、应　用

MAS3507D 和 DAC3550A 的出现,使 MP3 的硬件解码变得更容易,噪声降至最低,使应用工程师的产品软、硬件设计变得更简单,产品设计周期大大缩短。应用 MAS3507D 和 DAC3550A 解码设计的 MP3 便携式系统其主板的基本电路框图如图 5 .11－4 所示。

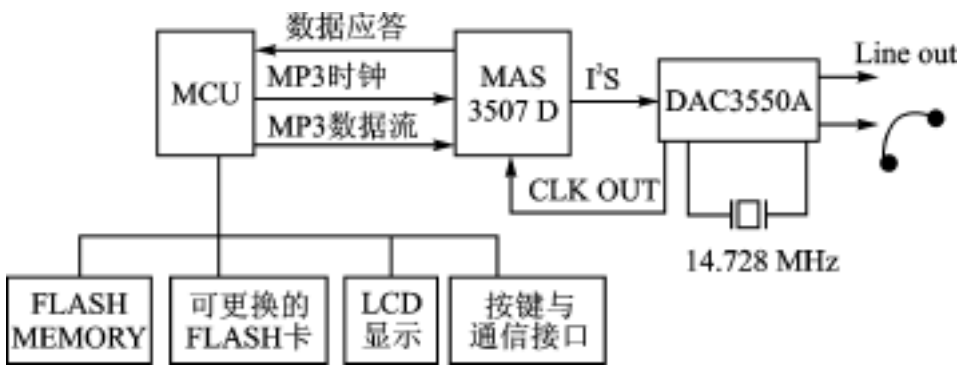


图 5 .11－4　MP3 便携式系统主板电路框图

通过 MCU 软件的设计,可使本已在 MAS3507D 中存在的功能能在面板上体现,方便地供用户使用。整机的操作可设置电源开关、随机播放、音乐简介播放、重复播放、快进、后退、音量调节、高低音调节、曲目预览等。MP3 解码芯片组强大功能的体现,可以使 MP3 便携式系统成为一个性价比较高的优秀产品。

参 考 文 献

1　Micronas Intermentall Data Book, 1999 .7

2　MP3 技术及系统构成 . 电子产品世界,1999(9)

3　Electronics Systems Technology&Design, 1999(4)

选自《电子技术应用》月刊,2000 年第 4 期

5 .12　射频 IC 卡 E5550 原理及应用

太原理工大学(030006)　苏　斌

德国 TEMIC 生产公司的 E5550 芯片早期是应用于轿车防盗安全系统的一种 ID(身份)识别芯片,该芯片由 EEPROM 及射频接收、发射电路、逻辑加密电路等构成,其数据的读写操作为非接触方式,由该芯片经塑封生产的非接触射频 IC 卡可广泛应用于各个领域,同时由于该器件所配套的读写控制装置与其他非接触 IC 卡的读写控制装置相比,具有结构简单、体积小、成本低廉等显著特点,在许多应用场合完全可以取代目前的接触式 IC 卡。

一、E5550 卡的主要特点

1 . E5550 卡的存储结构

如图 5 .12 - 1 所示,由低电压、低功耗 COMS 电路组成,无线射频频率为 100～150 kHz,其存储容量为 264 B,共分成 8 个数据块,每块 33 B。其中一块(block 0)用于模式设置,一块(block 7)用于口令设置。其余 6 块(block 1 至 block 6)为用户数据存储区。在每块用户数据存储区中,bit0 为锁定(lock)位,该位一但锁定后,该块的其余 32 B 位将变为只读而不能改写。口令块(block 7)可由模式设置决定是否存放口令还是用户的一般数据。

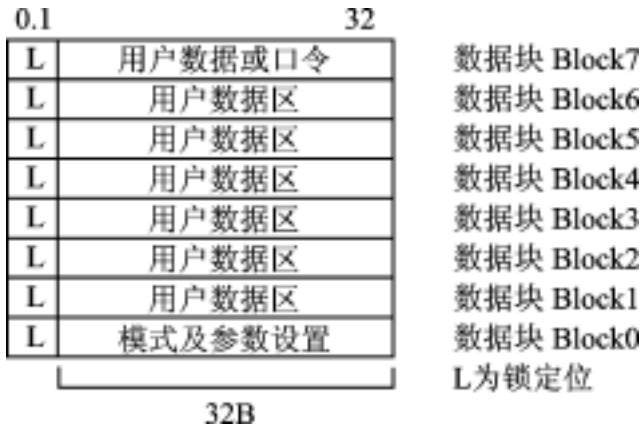


图 5 .12 - 1　E5550 存储器结构

2 . E5550 卡与读写器的信息和能量传递过程

图 5 .12 - 2 中所示,在数据的读写过程中,通过联接在 IC 卡读写器上的 LC 线圈产生高压感应电势(100～150 kHz),耦合到 IC 卡的天线回路中,该信号一方面由调制解调电路处理后通过控制电路对卡进行操作,另一方面通过卡内部的整流电路得到 IC 卡工作所需的电能量。

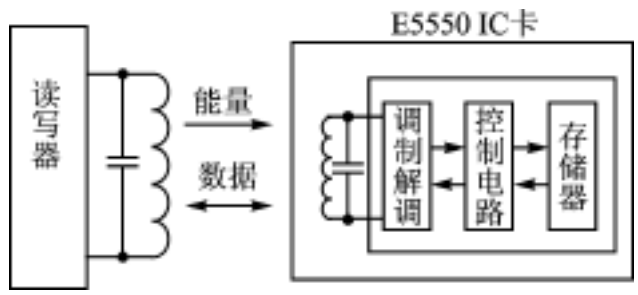


图 5 .12 - 2　E5550 IC 卡信息传递过程

3 . E5550 卡工作参数的设定

在 block 0 中的 33 bit 数据位可以对卡的工作模式进行设置:其主要内容为:

① 传输波特率设定:该卡的数据传输速率可设定为载波频率的 1/ 8, 1/ 16, 1/ 32, 1/ 40, 1/ 50, 1/ 64, 1/ 100, 1/ 128。同 bit12～bit14 确定。

② 编码及调制方式设定:调制方式由两部分组成,第一部分为二进制数的编码方式:直接编码,曼彻斯特码、双相位码、由 bit16～bit17 确定。第一部分为频率调制方式:相位键控、频率键控、直接编码、由 bit18～bit20 确定。这两部分以适当的方式组成最终的编码及调制过程。

③ 口令设定:bit28 = 1 时为口令工作模式,既用户对 block1～block6 中的数据读写需要一个与 block7 中的 4 字节数据相同的口令, bit28 = 0 时为非口令工作模式,可对以上 6 块数据直接进行读写,同时, block7 也可作为用户数据区使用。

④ 其他有:最大数据块 MAXBLX 设定 bit25～bit27(用户不一定使用到所有 6 个数据块时,可由 block 0 开始自行设定,其值为 1～6,但不可为 0,因 block 0 用户不能使用并且隐含不能读)、AOR 方式设定 bit23 及 STOP 方式设定 bit31,等。

4 . E5550 卡数据的读出与写入

① 数据读出。E5550 卡是利用线圈中产生的阻尼特性的载频信号向接收器传送数据的。如图 5 .12 - 3 所示,在卡接近读写器时,卡内接收到电源能量的信号后,首先产生上电复为过程,进入数据传送状态,此时根据 block 0 中已设定的工作模式装入模式寄存器,以便下一步按该模式发送数据,同时产生一个约 2 ms 的恒定磁场。此后产生一个约 320 μs 的同步头信号,接着便从第一块的第一位开始传送数据。每块 32 位,锁定位不传送,直到 MAXBLK 所设定的最大块的最后一位为止,数据传送时产生带有阻尼作用的磁场信息,读写器的线圈接收该信息既可读出数据。应当注意,当读某块时该块之前的所有块都要读出。

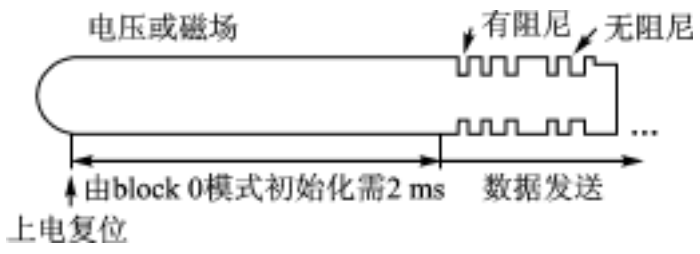


图 5 .12 - 3　E5550 卡上电后线圈中产生的磁场

② 数据写入。读写器通过对线圈中电磁场能量的间隔性中断将数据写入卡中,如图 5 .12 - 4所示。磁场间隔大约为 280 μs,“ 0 ”信号为 16～32 个磁场脉冲构成的段,“ 1 ”信号为 48～64 个磁场脉冲构成的段。在数据写入前应先写入操作码“ 10 ”,然后才是数据,数据的写入必需按块进行,各块可以独立与入,每块 33 位(含锁定位:“ 0 ”不锁定、“ 1 ”锁定),块后接着是 3 位块地址。无口令写时每块 38 位,有口令操作时在操作码之后加 32 位的口令,共需 70

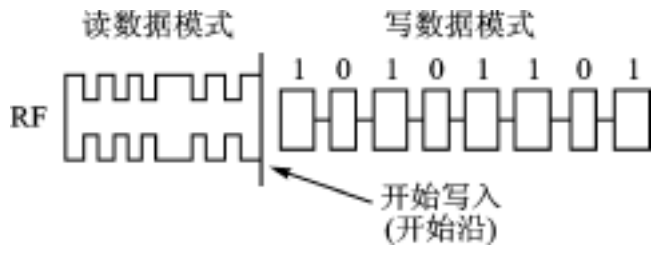


图 5 .12 - 4　E5550 卡的写入

位。为防止卡在一次接近读写器时产生不必要的多次写操作,可在写完数据后发停止信号,操作码为“11”,此后卡将不再接收或写入信号,直到下次上电复位。

二、读写器接口

1.硬件接口

该公司生产的IC芯片U2270B是与之配套的一种近距离非接触读写基站芯片,配备少量外围元件即可与单片机实现对接,它所产生的调制解调频率为100～150 kHz,工作电压为5～12 V,单电源供电,其结构如图5.12-5所示。

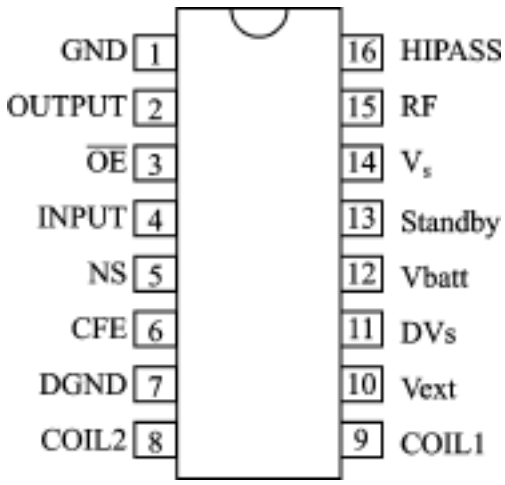


图5.12-5　U2270OB引脚图

主要引脚功能:

INPUT、OUTPUT分别为串行输入、输出端:COIL1、COIL2分别为天线线圈端;Vext、Vbatt、Standby分别为外部电源、备份电池及待机控制;OE、MS、CFE分别为输出允许、模式选择、载频控制;HAPISS、RF、DVs分别为直流退耦、射频控制、天线驱动;

GND、DGND、VS分别为模块地、数字地、电源;

由U2270B芯片与AT89C2051构成的E5550卡读写器电路如图5.12-6所示,$P_{3.3}$为输入端、$P_{3.4}$为控制端,C_1及天线线圈组成125 kHz的谐振电路。D_5、R_5、R_6、C_5构成解调器对天线信号进行解调然后经C_4耦合输入芯片,在片内进行滤波、放大、整形等送入单片机。电阻R_3,R_4用于调节发射频率,D_1～D_4构成输出反馈电路以稳定频率,C_2构成芯片退耦电路。

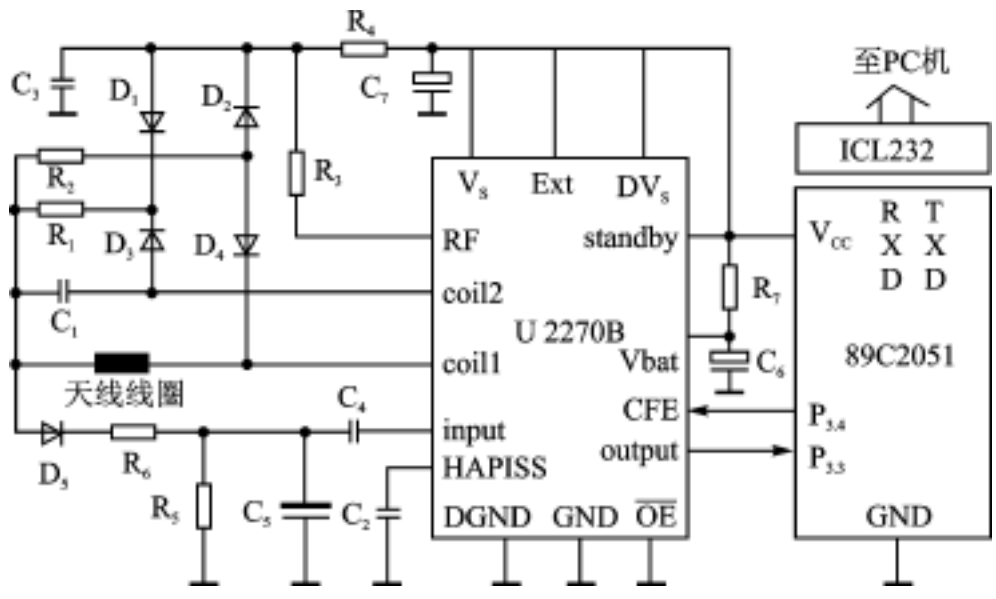


图5.12-6　读写器电路

2.软件编程

新卡中提供的默认参数为:曼彻斯特编码、RF/32、MAXBLK=2。因此本文以曼彻斯特编码方式介绍读卡部分的软件流程,图5.12-7给出曼彻斯特编码的波形图,由图中可以看到,在时钟的上升沿作用下数据“0”将产生一个下降沿,而数据“1”将产生一个上升沿,但是在两个连续的相同数之间同样也有上升沿或下降沿,这些变化是无效的,因此用曼彻斯特编码方式读取数据时,因看前一值以及信号变化的间隔时间从而决定该数据是否有效。程序开始时

先寻找一个 270～330 μs 的高电平同步信号,然后按上述编码原则开始逐个检测电平的变化并记录对应时间 T_1 或 T_2, T_1 = 90～180 μs, T_2 = 210～300 μs,如前一数据为‘1’的情况下,测的高电平时间为 T_1,对应的下降沿无效,应接着测下一上升沿并得‘1’;若测的高电平时间为 T_2 时,对应的下降沿则有效并得‘0’。如前一数据为‘0’时测的低电平时间为 T_1,对应的上升沿无效,应接着测下一下降沿并得‘0’,若测的低电平时间为 T_2 时,对应的上升沿有效并得“1”,以此即可逐位以串行方式读出卡内数据。流程图如图 5.12-8 所示。

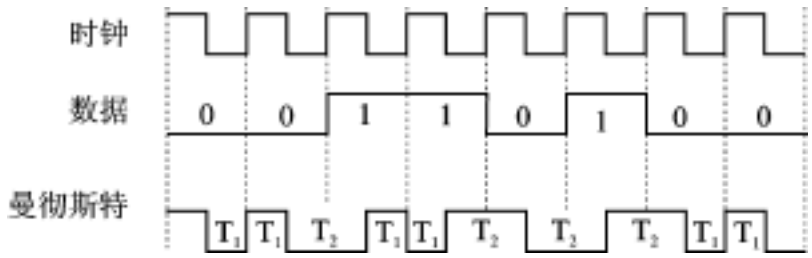


图 5.12-7 曼彻斯特码与数据之关系

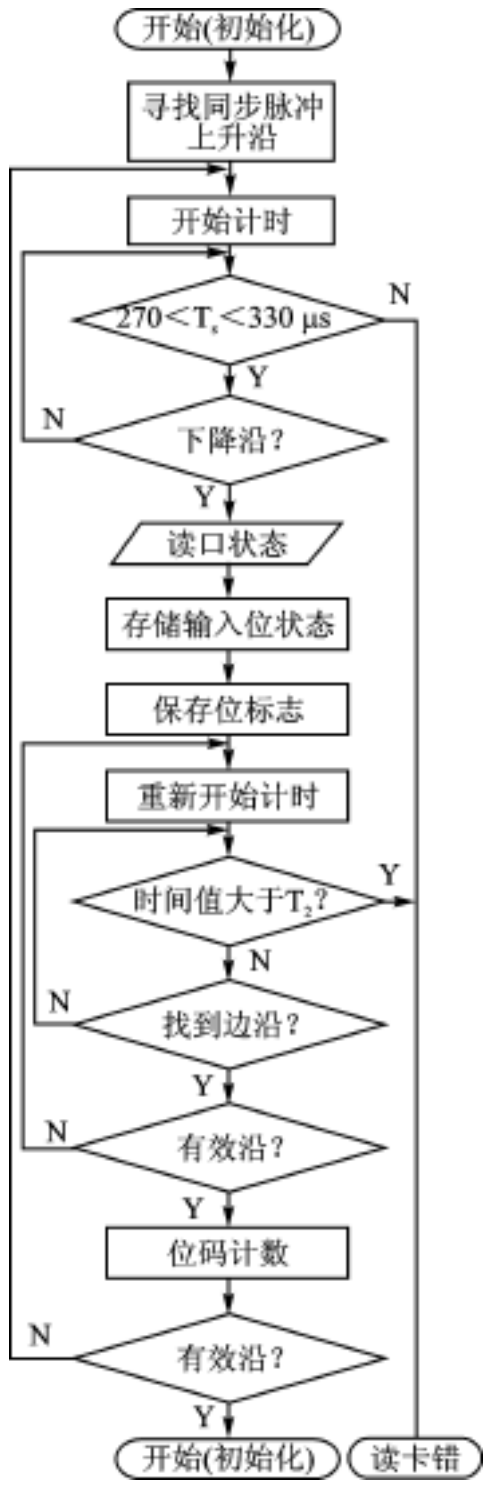


图 5.12-8 读卡流程图

写卡过程较简单:写“0”时为高电平 135 μs,然后低电平 280 μs;写“1”时为高电平 360 μs,然后低电平 280 μs;按前面叙述的操作格式逐位写即可。

三、应 用

E5550 卡是一种成本低廉的非接触卡,虽然容量较小,但也能适合许多应用场合。如考勤系统,食堂售饭系统,电子门票及要求存储容量不大的各种系统中取代当前的接触式 IC 卡,从而避免了接触式 IC 卡在一些易污染及损害等环境中的不足,如果设计合理其可靠性和安全性还是较高的,它的低成本、非接触性及读写电路的简单、廉价是主要优点,我们利用它设计了一种非接触式 IC 卡预付费煤气表及考勤系统,使用效果良好。

参 考 文 献

1 TEMIC Semiconductor e5550 Data Sheet, 1998(19)

2 TEMIC Semiconductor U2270B Data Sheet 1998(19)

3 ATMEL AT89C2051 Data Sheet

4 何立民 .单片机应用系统设计 .北京:北京航空航天大学出版社, 1990

选自《电脑开发与应用》月刊,2000 年第 9 期

5 .13　HD7279A 键盘显示驱动芯片及应用

山东东营石油大学自动化系(257062)　纪　钢

HD7279A 是一片具有串行接口的,可同时驱动 8 位共阴极数码管的智能显示驱动芯片。该芯片可连接多达 64 个键的键盘矩阵,并含有去抖动电路。HD7279A 芯片内部有译码器,可以直接接受 16 进制码,并且具有 2 种译码方式和多种控制指令,如:消隐、闪烁、左移、右移、段寻址等。可以广泛应用在仪器仪表,工业控制,条形显示器,控制面板等领域。

如图 5 .13 - 1 所示,HD7279A 是标准 28 引脚双列直插式芯片。引脚 1,2(V_{DD})为正电源;引脚 3,5(NC)不连接,使用时要求悬空;引脚 4(V_{SS})为接地端;引脚 6(CS)为片选输入端,此引脚为低电平时,可向芯片发送指令及读取键盘数据;引脚 7(CLK)为同步时钟输入端,向芯片发送数据及读取键盘数据时,该引脚电平上升沿表示数据有效;引脚 8(DATA)为串行数据输入/ 输出端,当芯片接收指令时,此引脚为输入端;当读取键盘数据时,此引脚在‘ 读 ’指令最后一个时钟的下降沿变为输出端;引脚 9(KEY)为按键有效输出端,平时为高电平,当检测到有效按键时,此引脚变为低电平;引脚 10～16(SG～SA)为段 g～段 a 驱动输出;引脚 17

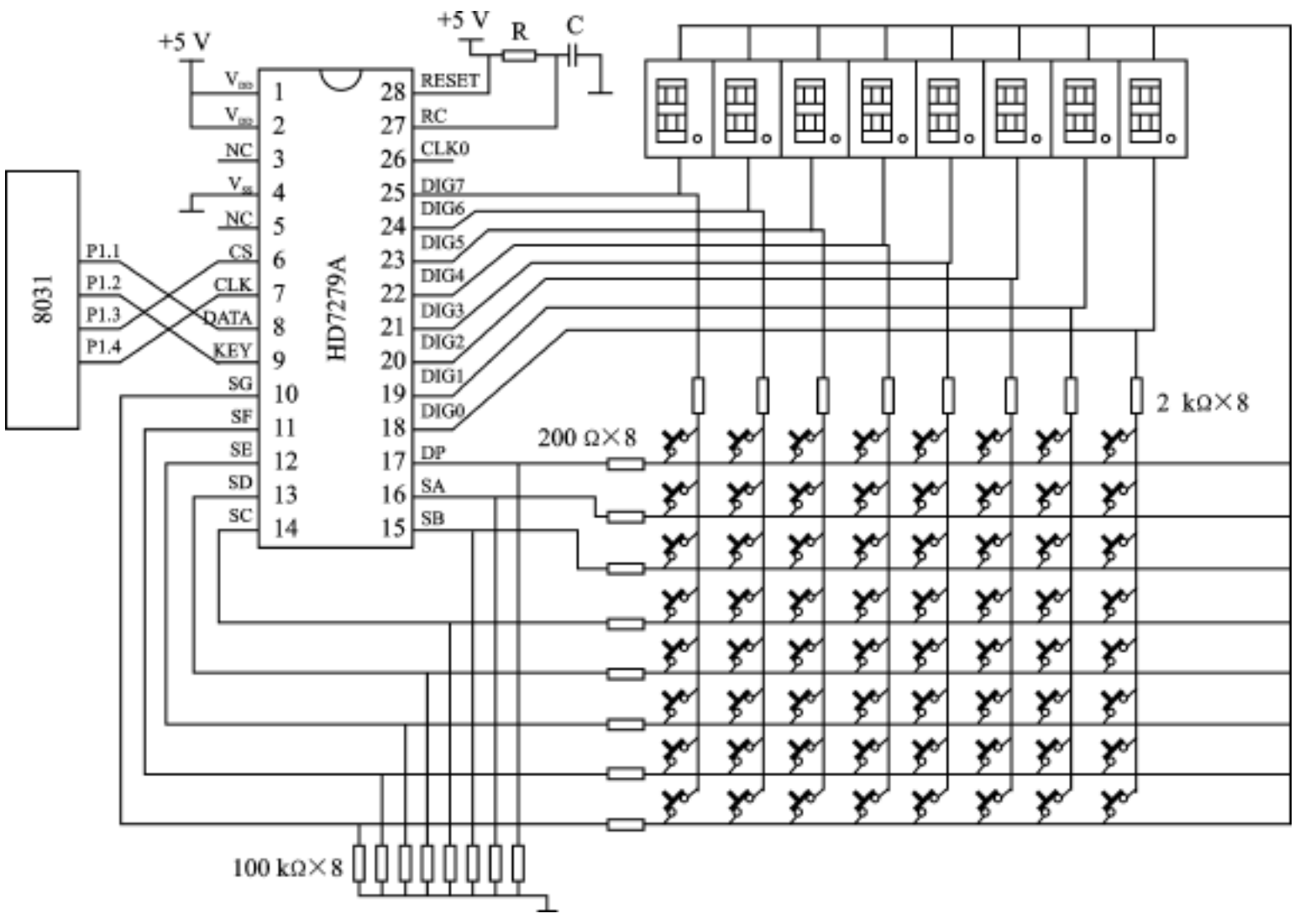


图 5 .13 - 1　HD7279A 典型应用电路图

(DP)为小数点驱动输出;引脚 18～25(DIG0～DIG7)为数字 0～7 位驱动输出;引脚 26(CLKO)为振荡输出端;引脚 27(RC)为 RC 振荡器连接端;引脚 28(RESET)为复位端。

HD7279A 采用串行方式与微处理器通信,串行数据从 DATA 引脚送入芯片,并由 CLK 端同步。当片选端 CS 信号变为低电平后,DATA 引脚上的数据在 CLK 引脚的上升沿被写入 HD7279A 的缓冲寄存器。HD7279A 的指令结构有三种类型(表 5.13-1):(1) 不带数据的纯指令,指令的宽度为 8 位,即微处理器需要发送 8 个 CLK 脉冲;(2) 带数据指令,宽度为 16 位,即微处理器需要发送 16 个 CLK 脉冲;(3) 读取键盘数据指令,宽度为 16 位,前 8 位为微处理器发送到 HD7279A 的指令,后 8 位为 HD7279A 返回的键盘代码。执行该指令时,HD7279A 的 DATA 端在第 9 个 CLK 脉冲的上升沿变为输出状态,并以第 16 个脉冲的下降沿恢复为输入状态,等待接收下一个指令。

表 5.13-1

序 号	指令名称	指令代码		指令功能
		代 码	数 据	
1	复位指令	10100100		清除所有的显示,包括清除所有设置的字符消隐、闪烁等属性
2	测试指令	10111111		使所有的 LED 全部点亮,并处于闪烁状态,主要用于测试
3	右移指令	10100000		使所有的显示自左向右(从第 8 位向第 1 位)移动一位
4	左移指令	10100001		使所有的显示自右向左(从第 1 位向第 8 位)移动一位
5	循环右移指令	10100010		与右移命令类似,但移动后最右边位(第 1 位)的内容显示于最左位(第 8 位)
6	循环左移指令	10100011		与循环右移指令类似,但移动方向相反
7	下载数据且按方式 0 译码	10000$a_2a_1a_0$	dp + + + $d_3d_2d_1d_0$	其中 $a_2a_1a_0$ 为数码管的地址,d_0～d_3 为数据,00H～09H 对应的七段显示码为 0～9,0AH～0FH 分别为 - EHLP 和空。dp 为 1 时小数点显示,dp 为 0 时小数点不显示。+ 为无影响
8	下载数据且按方式 1 译码	11001$a_2a_1a_0$	dp + + + $d_3d_2d_1d_0$	除 d_0～d_3 为 0AH～0FH 时分别为 ABCDEF 外,其他同上条指令
9	下载数据但不译码	10010$a_2a_1a_0$	dp a b c d e f g	$a_2a_1a_0$ 为数码管的地址;dp a b c d e f g 为显示数据,对应七段数码管的各段
10	闪烁控制	10001000	$d_8d_7d_6d_5d_4d_3d_2d_1$	d_1～d_8 分别对应数码管 1～8,1 闪烁,0 不闪烁
11	消隐控制	10011000	$d_8d_7d_6d_5d_4d_3d_2d_1$	d_1～d_8 分别对应数码管 1～8,1 显示,0 消隐
12	段点亮指令	11100000	+ + $d_5d_4d_3d_2d_1d_0$	点亮数码管中的某一指定段,+ 为无影响,d_0～d_5 为段地址
13	段关闭指令	11000000	+ + $d_5d_4d_3d_2d_1d_0$	关闭(熄灭)数码管中的某一指定段,+ 为无影响,d_0～d_5 为段地址
14	读键盘数据指令	00010101	$d_7d_6d_5d_4d_3d_2d_1d_0$	读出当前的按键代码。前字节为微控制器传送到 HD7279A 的命令,后字节 d_0～d_7 为返回按键代码,其范围是 0～3FH

现以图 5.13-1 所示的单片机系统介绍 HD7279A 智能控制芯片的应用方法。它利用 HD7279A 所带的 8 个显示数码管显示出时、分和秒。整个程序主要由主程序(void main( ))、发送子程序(void send(uchar x))、接收子程序(uchar receive(uchar x))、中断子程序(void

timer0(void)interrupt 1 using 1)和显示子程序(void display(uchar h, m, s))等组成。本应用程序由单片机的 C 程序设计语言编制(Franlin C－51)。

源程序如下:

```
#include< reg51.h>
sbit dat = P1^1;
sbit key = P1^2;
sbit cs = P1^3;
sbit clk = P1^4;
#define uchar unsigned char
#define uint unsigned int
uchar data dat_out;
uchar data dat_in;
uchar data hour, min, second, msec;

void delay(unsigned x)
    {for(;x>=2;)
        {x--;}
    }
                void send(uchar x)
{uint i;
  cs=0;                          /* 设 cs 为低电平 */
  delay(4);                      /* 100 μs 延时 */
  for(i=0;i<8;i++){
    if(x&0x80)                   /* 输出 1 位 */
      dat=1;
    else  dat=0;
    clk=1;                       /* 设 clk 为高电平 */
    x<<=1;                       /* 待发数据左移 */
    delay(1);                    /* 短延时 */
    clk=0;                       /* 设 clk 为低电平 */
    delay(1);                    /* 短延时 */
    }dat=0;
  }
uchar receive(uchar x)
  {uint i;  dat=1;
  delay(4);                      /* 100 μs 延时 */
  for(i=0;i<8;i++)
    {clk=1;                      /* 设 clk 为高电平 */
    delay(1);                    /* 短延时 */
    x=x<<1;                      /* 数据左移 */
    if(dat==1)                   /* 读取 1 位数据 */
      x=x|0x01;
```

```
      else   x = x&0xfe;
      clk = 0;                          /* 设 clk 为低电平 */
      delay(1);                         /* 短延时 */
      }dat = 0;
      return x;
   }
void timer0(void)interrupt 1 using 1{
   msec + + ;
   TH0 = - (5225/256);
   TL0 = - (5225%256);
   }
void display(uchar h,m,s)
   {uchar led[8],i,w;
   led[7] = h/10;                       /* 小时送缓冲区 */
   led[6] = h%10;
   led[5] = 0x0a;                       /* 分隔符 - 送缓冲区 */
   led[4] = m/10;                       /* 分时送缓冲区 */
   led[3] = m%10;
   led[2] = 0x0a;                       /* 分隔符 - 送缓冲区 */
   led[1] = s/10;                       /* 秒时送缓冲区 */
   led[0] = s%10;
   for(i = 0;i < 8;i + + )              /* 送 HD7279A 显示 */
      {w = i + 0x80;
      send(w);
      send(led[i]);
      cs = 1;}   }
void main( )
{uchar x;SP = 0x2f;
   TMOD = 0x01;
   while(key = = 0){;}                  /* 等待键按下 */
   while(key = = 1){;}                  /* 等待键释放 */
   dat_out = 0x15;
   send(dat_out);
   dat_in = receive(x);                 /* 取小时十位值 */
   cs = 1;
   hour = 10 * dat_in;                  /* 存小时十位值 */
   while(key = = 0){;}                  /* 等待键按下 */
   while(key = = 1){;}                  /* 等待键释放 */
   dat_out = 0x15;
   send(dat_out);
   dat_in = receive(x);                 /* 取小时个位值 */
   cs = 1;
   hour = hour + dat_in;                /* 加小时个位值 */
```

…

```
min = min + dat_in;                    /* 加分个位值 */
second = 0;msec = 0;
display(hour, min, second);            /* 显示 */
while(key = = ){;}                     /* 等待按任意键启动 */
TH0 = - (5100/ 256);
TL0 = - (5100%256);
EA = 1; ET0 = 1; TR0 = 1;
for( ; ; ){
while(msec > = 100)
  {msec = 0; second + + ;
  while(second > = 60)
    {second = 0; min + + ;
    while(min > = 60)
      {min = 0; hour + + ;
      while(hour > = 24)
        {hour = 0;}   }   }
    display(hour, min, second);
    }   }   }
```

参 考 文 献

1　马忠梅,等 . 单片机 C 语言应用程序设计[M] . 北京:北京航空航天大学出版社,1997

选自《仪表技术》双月刊,2001 年第 3 期

5.14 基于SPI接口的ISD4104系列语音录放芯片及其应用

湖南长沙市无线电厂(410003) 彭希南

ISD4104系列语音录放芯片是美国半导体信息存储器件公司采用先进的直接模拟数据存储(DAST TM)专利技术最新推出的语音录放电路,其最大特点是声音无须A/D转换和压缩即可直接存放。由于省去了A/D和D/A转换器,使得在其内部集成大容量的EEPROM成为可能,使用时不再需要扩展存储器。语音信息通过多级存储技术(Multi-Level Storage Technology),以接近原始模拟量的形式存储。片内有最大(380 KB)的非易失性RAM,封装为28脚,采用同步串行通信接口SPI(Serial Peripheral Interface),具有外部连线少、电路简单、应用方便、单片录放、不怕掉电、音色纯真、品种齐全、性价比高等特点,在众多的语音录放电路中独具特色。

一、电路的特点

ISD4104与普通的录音/重放芯片相比,功能上有较大的改进,其特点如下:

(1) 记录的声音没有早期产品ISD1400系列芯片段长度的限制;

(2) 声音的记录无须A/D转换和压缩,放音自然、完美,没有常见的背景噪音;

(3) 采用快速闪烁存储器作为存储介质。掉电可保存数据长达100年,语音重复记录100 000次以上,断电语音自动保护;

(4) 记录语音时间长达16 min(ISD4004-16 M);

(5) 解码器接口简单,单片录放仅需三线(多片录放需四线),其SPI接口提供全部数据和控制操作;

(6) 允许3 V低电压供电,静态电流放音耗电量30 mA,录音电流25 mA。

表5.14-1列出了常用ISD4104系列芯片的主要参数。

表5.14-1 常用的ISD4104芯片特点

ISD系列	型 号	存储时间/s	内存/KB	采样频率/Hz	频响/kHz
SD4104	ISD4004-08M	480	3840	8.0	3.4
	ISD4004-10M	600	3840	6.4	2.7
	ISD4004-12M	720	3840	5.3	2.3
	ISD4004-16M	960	3840	5.0	2.0

二、ISD4104的电路工作原理和引脚特性

ISD4104由时钟振荡器、语音存储单元、前置放大器、自动增益控制电路、抗干扰滤波器、输出放大器、静音电路等组成。其内部结构框如图5.14-1所示。输入放大器通过外部的隔直流电容与麦克风连接,隔直流电容用来去掉交流小信号中的直流成份。录音过程中,

ISD4104 器件首先把输入信号放大到存储电路动态范围的最佳电平,通过每个倍频程衰减 40 dB的 5 极点低通滤波器去掉背景噪声和取样频率 1/ 2 以上的输入频率分量,然后再进行存储操作。语音的数据采样遵循奈奎斯特取样定律,取样频率 8 kHz 时,连续时间 5 极点低通滤波器的高频频限选在 3 .4 kHz;取样频率为 6 .4 kHz 时,低通滤波器的高频频限选在 2 .7 kHz,保证有足够宽的频带可以得到高音质的语音,并满足奈奎斯特取样定律。经过模拟收发器再经过电平移位,产生不挥发写入 EEPROM 所需的高电压,在取样时钟对存储阵列的地址译码后,再将数据写入模拟存储器阵列中。放音时,录入的模拟电压在取样时钟的控制下顺序地从存储阵列中读出,恢复成原来的取样波形。输出通道上的平滑滤波器去掉取样频率分量并恢复原始波形。平滑滤波器的输出通过 1 个自动静音电路连接到输出功率放大器,2 个输出管脚直接驱动扬声器。一般来说,1 个由 ISD 组成的最小语音录放系统仅由 1 个麦克风、1 个喇叭、2 个按钮、1 个电源、少数电阻和电容组成。录音内容存入永久存储单元,可提供零功率信息存储。

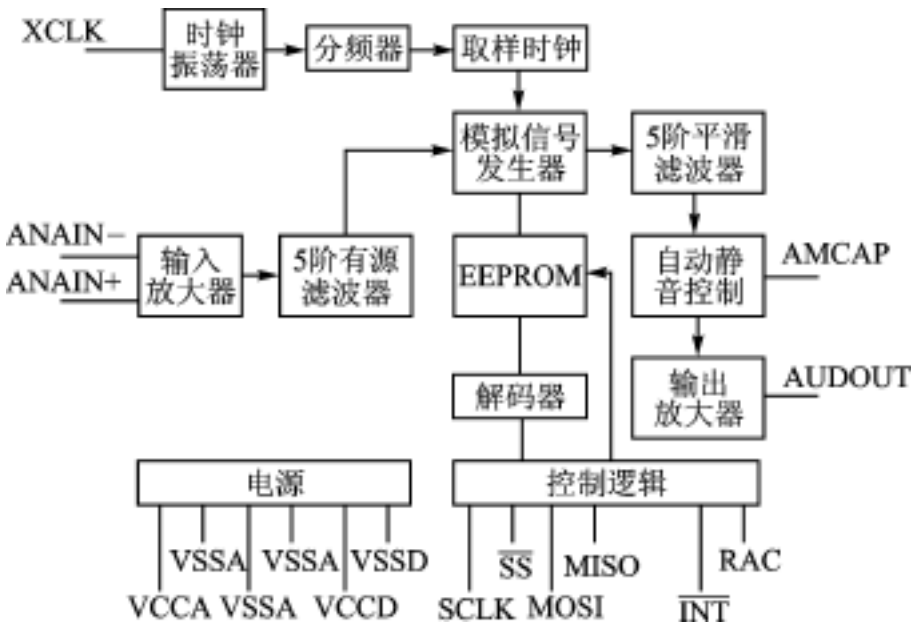


图 5 .14 - 1　ISD4104 内部电路结构框图

ISD4104 系列录音芯片为 28 脚芯片,有 TSOP、PDIP、SOIC 三种封装。有关引脚功能如下:

SCLK:为串行时钟,不用时必须接地;

XCLK:为外部时钟,大小决定取样频率;

MOSI:串行输入端;

MISO:串行输出端,未选中时为高阻态;

$\overline{\text{INT}}$:中断控制;

ANAIN - 和 ANAIN + :为录音信号输入端,单端和差动 2 种信号驱动方式;

AUDOUT:音频信号输出端,驱动 8～16 Ω 扬声器;

AMCAP:为自动静音端,在没有信号时清除噪音,静音时衰减 6 dB;

*V*CCA:模拟电源;

*V*SSD:数字地;

*V*SSA:模拟地;

$\overline{\text{SS}}$:片选信号,低电平选中;

RAC:行计数器;

NP:空脚,共有 11 个引脚未定义。

三、ISD4104 的 SPI 接口技术

SPI 是 Motorola 公司率先推出的在 68 系列单片机上使用的一种同步串行通信接口,用 3 根单独信号线主入从出(MISO)、主出从入(MOSI)、同步时钟(SCLK)传送数据,即可以实现与单片机全双工通信。通过 SS 片选线信号还可实现菊花链状的多路系统扩展。ISD4104 系列芯片采用的正是这种节省 I/O 引脚接口的串行数据传送方式,消除了对多条专用地址线的需求。内部存储器为新型结构的高性能 SPI 串行 EEPROM 器件,因此可直接利用微处理器的控制信号,无需微处理器的系统总线接口。具有低工作电压、节省时间、硬件写冲突保护功能、可防止发送时再次写入移位寄存器造成数据丢失、具有页写操作方式等优点;可通过程序控制串行时钟的极性和相位,实现与各种 SPI 芯片接口;当需延长语音录放时,可通过多块 ISD4104 芯片的级连构成全分布式系统,同时它还具有较高的数据传输速率(最高波特率可达总线速度的一半),不需要专用的串行总线。

1.工作时序

标准的 SPI 是 8 位结构,但 SPI 接口的位串行特性也适用于 16 位或 32 位环境。ISD4104 系列的 SPI 接口指令是 16 位的,在启动 1 次传送时由单片机产生 16 个脉冲传送给 ISD4104 系列芯片作为同步时钟,数据由 MISO 端移出,MOSI 端移入,典型的时序如图 5.14-2 所示。其中,采样数据线时的 SCLK 边沿与具体的芯片有关。在读取每一字节数据之后,芯片内的字节地址自动加 1,通过继续发出读周期,可以顺序读出存储在存储器下一地址中的数据。当达到最高地址时,地址计数器卷绕至起始地址 00000,读操作将无限地继续下去。发出暂停时序将中止读写时序,在任何时候这一时序可用来中断或结束顺序读或页装载操作。

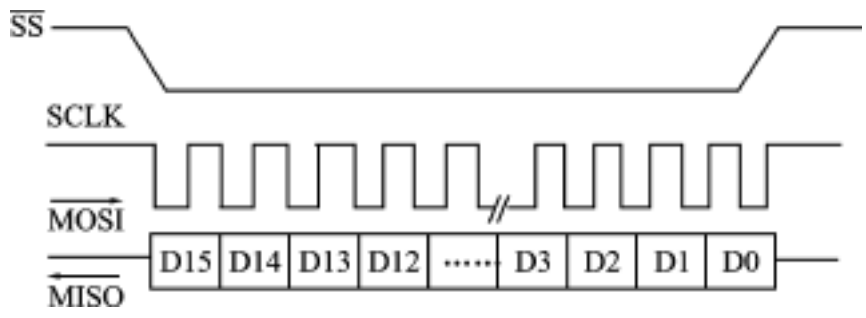


图 5.14-2 ISD4104 SPI 同步串行通信时序

2.ISD4104 系列的 SPI 接口指令系统

ISD4104 SPI 接口指令仅 10 条,采用 16 位操作码。其功能按照录放机上常用的英文标识符定义,其操作码由指令部分和地址码部分组成,高 5 位为命令,后 11 位为片内地址码或无关码(用 X 表示)。指令结构上分成数据出、数据入二大组,SPI 接口指令集如表 5.14-2 所列。

表 5.14-2 ISD4104 系列的 SPI 接口指令集

命 令	格 式	操 作
POWERUP	00100XXXXXXXXXXX	上电芯片处于准备状态
SETPLAY	11100(A10·····A0)	设置放音的起始地址
PLAY	11110XXXXXXXXXXX	从当前地址开始放音

续表 5.14-2

命　令	格　式	操　作
SETREC	10100(A10……A0)	设置录音的起始地址
REC	10101XXXXXXXXXXX	从当前地址开始录音
RINT	0X110XXXXXXXXXX	读当前中断状态字
SETMC	11101(A10……A0)	设信息跳转的起始地址
MC	11111XXXXXXXXXX	执行跳转命令
STOPPOWDN	0X01XXXXXXXXX	停止当前操作,进入待机
STOP	0X000XXXXXXXXX	停止当前操作

MOSI 指令部分的含义如表 5.14-3 所列。

表 5.14-3　MOSI 指令部分的含义

运行 (RUN)	放音/录音 (P/R)	电源 (PW)	忽略地址位 (IAB)	信息速查 (MIC)	地址 A10…A0

运行(RUN)位:放音/录音操作时为高电平,上电复位/停止时为低电平。

放音/录音(P/R)位:放音时为高电平,录音时为低电平。

电源(PW)位:工作时为高电平,停止时为低电平。

忽略地址(IAB)位:对记录的声音无段长度的限制。

信息速查(MIC)位:确定当前信息存储地址。

MISO 指令部分的含义如表 5.14-4 所列。

表 5.14-4　MISO 指令部分的含义

OVF	EOM	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11

OVF 位是溢出标志,EOM 位是信号结束标志,PC0～PC11 是 MISO 的地址计数器指针。由于 ISD 器件的地址不是通常意义上的字节地址单元,而是信息段的基本组成单位,因此 PC 每个地址单元指向其中的 1 行。

ISD 器件存储阵列中的每一行都可以独立寻址,且器件的每一行中都布置了 EOM 标记定位点。对 ISD 器件的寻址操作实际上就是给出某个信息段的起始地址,单片机或解码器指向该段的起点之后,ISD 器件将自动完成对该段的操作。由于 ISD 无段址的限制,录放操作可从任一行的行首开始,一直持续到行尾。内部的行地址计数器自动加 1,并指向下一行的起点地址。

应指出的是:ISD4104 器件没有现成的级连端子,需要更长的语音录放时间时,多片 ISD 的片选 SS 端可与解码器相连,通过地址计数器指针 PC0～PC10 和 OVF 位在解码器的控制下,确定哪片 ISD 的信息已经结束须禁止,并同时激活下一片 ISD。片间切换时,由于 ISD 的分辨率大约为 1 ms,因此有 1 ms 左右的间隙时间。

四、ISD4104 与 80C31 的模拟 SPI 接口编程

ATMEL 公司的 AT89C8252 和 Motorola 公司的 M68HC05 系列及 MC68HCII 全系列

芯片均内含 SPI 接口,可以很方便实现与 ISD4104 连接。而 8031 不具备与 SPI 接口功能,为实现与 ISD4104 的通信控制,可通过软件模拟的方式实现。80C31 与 ISD4104 硬件接口电路图如图 5 .14 - 3 所示,设定 P17 为 MOSI, P16 为 SCLK, P15 为片选/ SS, P10 为 MISO,利用上述 I/ O 线即可完成片选、同步串行时钟发生器和数据的串行传送/ 接收。其软件编程框图如图 5 .14 - 4 所示。

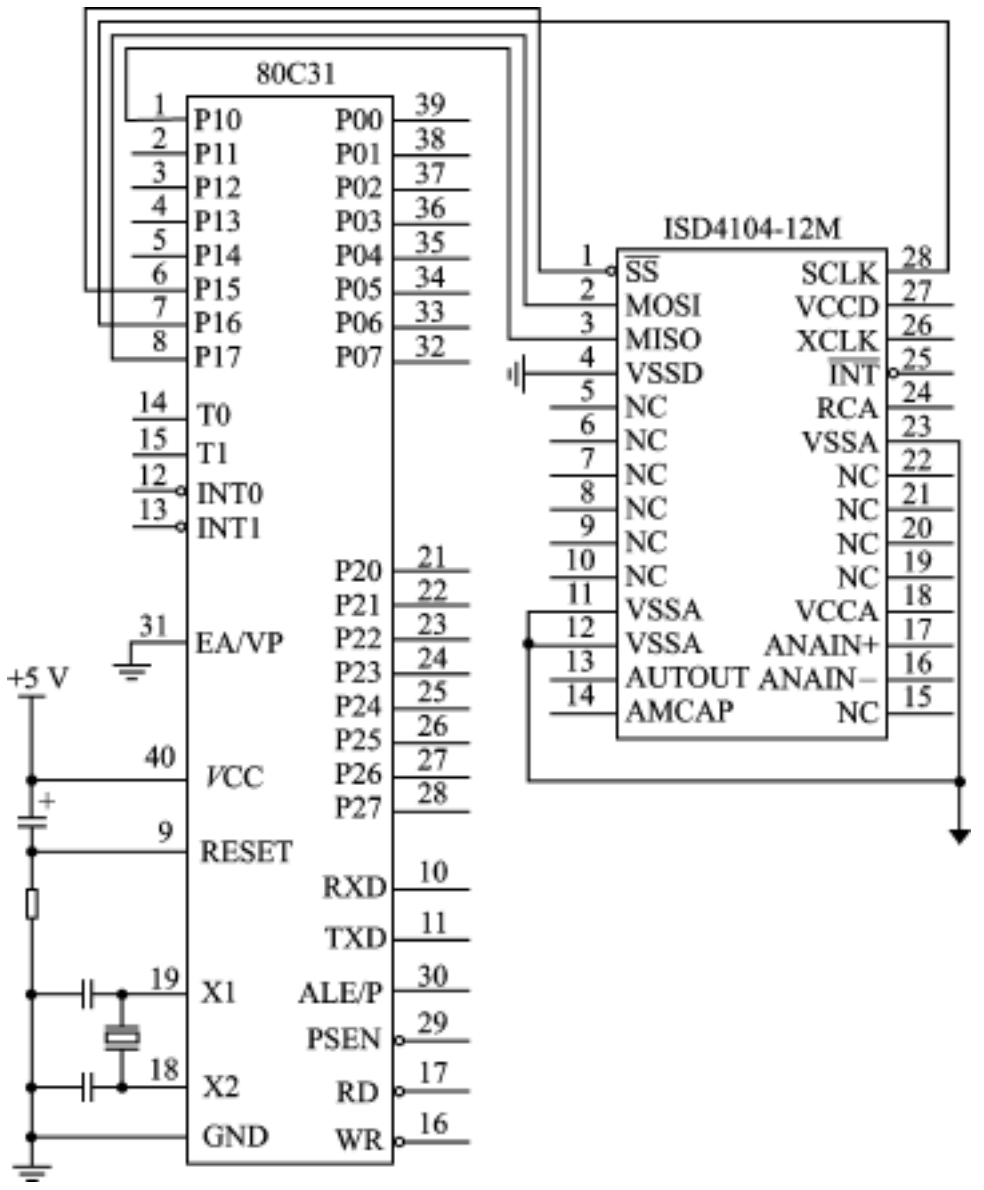


图 5 .14 - 3 ISD4004 - 12M 与 80C31 的模拟 SPI 串行接口控制电路

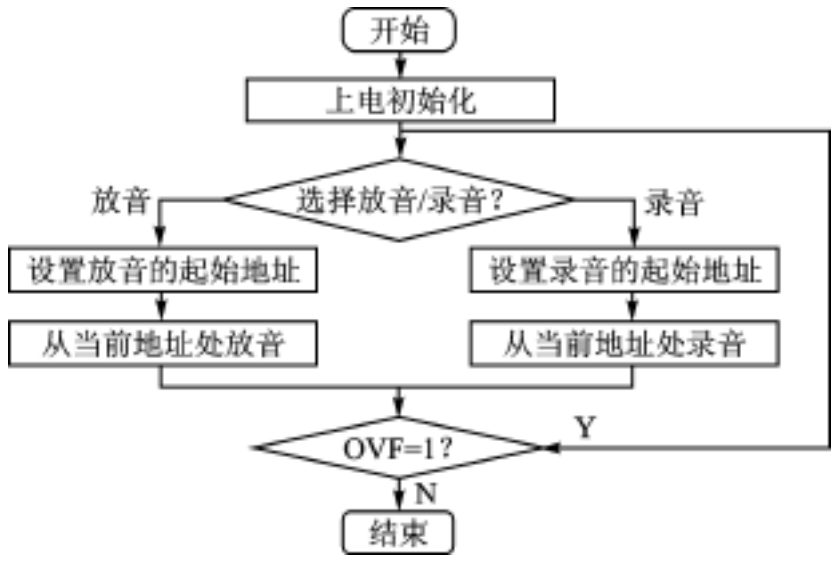


图 5 .14 - 4 ISD4104 录放程序框图

以下给出 80C31 模拟 SPI 接口的 8 位收发子程序清单的示例。ISD4104 的 2B 指令操作码分 2 次调用输入/输出子程序,从而完成 1 次命令操作。

输入子程序 INBIT:

功能:按 ISD4104 时序从 P10 读入 1B 到 R3 中。

入口参数:无。

出口参数:接收的数据在 R3 中。

```
INBIT:    LR     1.5          选中 ISD4104
          CLR    P1.7         ;
          MOV    R1,#8        ;计数器
INBIT1:   CLR    P1.6         ;产生负极性时钟下降沿,使 ISD4104 输出 1 位
          NOP                 ;延时
          MOV    C,P1.0       ;采样数据线
          RLC    A
          SETB   P1.6         ;时钟上升沿
          NOP
          DJNZ   R1,INBIT1    ;输入 1B
          MOV    R3,A
          SETB   P1.5         ;片选/SS 端为高电平
          RET
```

输出子程序 OUTBIT:

功能:按 ISD4104 时序从 P1.0 输出 A 中数据。

入口:传送的数据或命令在 A 中。

出口:无。

```
OUTBIT:   CLR    P1.7
          CLR    P1.6
          MOV    R1,#8
OUTBIT1:  CLR    P1.6         ;时钟下降
          RLC    A
          MOV    P1.7,C       ;输出 1 位
          NOP                 ;延时
          SETB   P1.6         ;时钟上升沿
          NOP
          DJNZ   R1,OUTBIT1   ;进行 8 次
          SETB   P1.5
          RET
```

选自《微型机与应用》月刊,2000 年第 9 期

5.15 解决 DS1820 通信误码问题的方法

武汉华中科技大学计算机科学与技术学院(430074) 张子蓬
武汉湖北工学院(430074) 王淑青

目前,生产和生活中测量温度的模式,基本上是以“温度传感器 + 变送器 + A/D”为主,这样,综合成本较高,调校繁琐,稳定性差。

DS1820 是一款数字式传感器,以其优良的性能、低廉的价格,越来越受到人们的欢迎。

一、DS1820 的特性

① 独特的单线接口,主机只需要一根 I/O 口线,即可与之通信;
② 多点接入能力使分布式测温得以简化;
③ 无需外围元件;
④ 可由数据线供电;
⑤ 测温范围 - 55～ + 125 ℃;
⑥ 静态时为零功耗;
⑦ 以 9 位数字值读出温度;
⑧ 温度转换成数字的典型时间为 1 s;
⑨ 用户可定义不挥发的上下报警限。

二、误 码

关于 DS1820 的介绍在许多杂志上已很常见,但人们好像都在回避一个非常重要的问题:误码!使用过 DS1820 集成温度传感器的同志都知道,在读出 DS1820 数据时,误码率很高,如不做任何处理,用户将无法确定读到是否是正确数据,尤其是在确定系列号和序列号时,如不能保证其正确性,多点接入将无法实现,所以必须对差错进行控制。

1.误码原因

误码原因主要来自四个方面:

① 总线时序与 DS1820 内部控制器的冲突;
② 总线受外部电磁场、静电、放电等干扰;
③ 长引线的分布电感、电容引起波形畸变和延迟增大;
④ 主机定时误差大或中断等影响了正常的命令和读出时序。

2.关于减少误码的方法

笔者认为上述原因①用户是无法控制的;针对原因②、③用户在设计安装时,尽量缩短导线,并认真考虑屏蔽和接地问题;关于原因④建议用户使用汇编语言编写程序,并测试指令执行时间,与 DS1820 通信时关闭中断等随机发生型事件。

三、如何解决误码问题

由上节所介绍的情况可见误码总是要发生的,那么,如何消除误码影响呢?主要有两种方法:其一为无校验法;其二为校验法。下面对这两种方法作些介绍。

1.无校验法

无校验法又可分为渐变法、中值法、加权平均法等。

(1) 渐变法

假定主机中温度值是放在一个变量 V 中,首先给变量 V 赋一初值,然后,每读到一个新的温度值 X,就比较 X 与 V 的大小,当新读到新的温度值 X 大于 V 时,V 就增加一个小步距 ΔV;反之,当新读到新的温度值 X 小于 V 时,V 就减少一个小步距 ΔV。这样,当出现误码时,只会造成 V 以 ΔV 为单位的跳动,如果 ΔV 取得较小,偶有误码,用户将感觉不到。

这种方法只适用主机与 DS1820 距离较近、误码率低的场合,但延迟较大。

(2) 中值法

主机每读到三个温度值时,做一次处理,去掉最大值,去掉最小值,剩下的即为正确值。这种方法延迟较小,但也只适用于误码率较低的用途。

上述两种方法,由于其编程简单,占机时少,效果也较理想,因而得到广泛使用。

其他方法从略。

2.校验法

(1) DS1820 的 BCH 编码方式

DS1820 采用 BCH 编码方式,监督码为 8 位,生成多项式为

$$G(x) = X^8 + X^5 + X^4 + 1$$

BCH 编码的原理和实现如下。

设信息原码为 P(x),余式为 R(x)即监督码 CRC,组码 F(x)即组合好并按要求发送到串行总线上去的信息码,表达式如下

$$F(x) = P(x) \cdot X^K + R(x)$$

式中　K——监督码的长度,即为 8。

DS1820 的 64 位 ROM 存储结构为

8 位 CRC	48 位序列号	8 位系列号
BYTE7		BYTE0

DS1820 的 SCRATCHPAD 数据存储器结构为

字　节	内　容
0	温度低字节
1	温度高字节
2	上限报警温度
3	下限报警温度
4	保留(FF)
5	保留(FF)
6	计数值 1
7	计数值 2
8	CRC

如何求出余式 R(x)呢?

我们首先计算出 $P(x) \cdot X^K$,即将信息原码 P(x)左移 K 位,后面补上 K 个“0”,然后除以一个生成多项式 G(x),这里的除是模 2 运算。

$$P(x) \cdot X^K / G(x) = Q(x) + R(x) / G(x)$$

式中 (x)——商;

R(x)——余式。

(2) CRC 校验法

校验法之原理法:即按上述 BCH 编码原理,进行解码校验。这种方法要求计算机足够快,在单片机上实现起来不太方便,因而少有人用,此处不做介绍。

校验法之查表法:由本机在上电初始化时,制做一张部分余式表存在数据存储器中或由其他计算机计算出这张表烧写在程序存储器中。

查表方法:首先把 P(x)当中的最高字节 B0 取出,查表对应 R0,将 P(x)的后继字节 B1 于 R0 相异或,并用异或后的数值再查表,又可以得到部分余式 R1,以此类推,最后用 P(x)的最低字节 Bn 异或 Rn - 1,用异或值再查表,便可得出 Rn,这就是我们所要的最终余式,即 CRC 监督码。

这种方法可以避免重复繁琐的运算,非常实用。

由于篇幅所限,上述部分余式表的制做算法这里不再赘述,笔者拟将部分余式表的算法及其详细源程序清单放在 http:// www .fenjin .com 网站上,届时,有兴趣的朋友可以去下载。下面仅给出已制做好的部分余式表。

00: 00H,5EH,BCH,E2H,61H,3FH,DDH,83H,C2H,9CH,7EH,20H,A3H,FDH,1FH,41H
10: 9DH,C3H,21H,7FH,FCH,A2H,40H,1EH,5FH,01H,E3H,BDH,3EH,60H,82H,DCH
20: 23H,7DH,9FH,C1H,42H,1CH,FEH,A0H,E1H,BFH,5DH,03H,80H,DEH,3CH,62H
30:BEH,E0H,02H,5CH,DFH,81H,63H,3DH,7CH,22H,C0H,9EH,1DH,43H,A1H,FFH
40:46H,18H,FAH,A4H,27H,79H,9BH,C5H,84H,DAH,38H,66H,E5H,BBH,59H,07H
50: DBH,85H,67H,39H,BAH,E4H,06H,58H,19H,47H,A5H,FBH,78H,26H,C4H,9AH
60: 65H,3BH,D9H,87H,04H,5AH,B8H,E6H,A7H,F9H,1BH,45H,C6H,98H,7AH,24H
70: F8H,A6H,44H,1AH,99H,C7H,25H,7BH,3AH,64H,86H,D8H,5BH,05H,E7H,B9H
80: 8CH,D2H,30H,6EH,EDH,B3H,51H,0FH,4EH,10H,F2H,ACH,2FH,71H,93H,CDH
90: 11H,4FH,ADH,F3H,70H,2EH,CCH,92H,D3H,8DH,6FH,31H,B2H,ECH,0EH,50H
A0: AFH,F1H,13H,4DH,CEH,90H,72H,2CH,6DH,33H,D1H,8FH,0CH,52H,B0H,EEH
B0: 32H,6CH,8EH,D0H,53H,0DH,EFH,B1H,F0H,AEH,4CH,12H,91H,CFH,2DH,73H
C0: CAH,94H,76H,28H,ABH,F5H,17H,49H,08H,56H,B4H,EAH,69H,37H,D5H,8BH
D0: 57H,09H,EBH,B5H,36H,68H,8AH,D4H,95H,CBH,29H,77H,F4H,AAH,48H,16H
E0: E9H,B7H,55H,0BH,88H,D6H,34H,6AH,2BH,75H,97H,C9H,4AH,14H,F6H,A8H
F0: 74H,2AH,C8H,96H,15H,4BH,A9H,F7H,B6H,E8H,0AH,54H,D7H,89H,6BH,35H

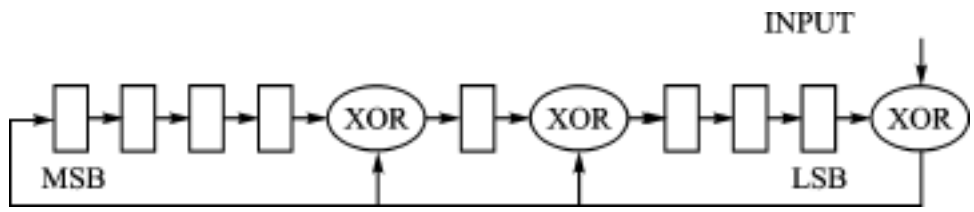
查表方法举例:

假若您从 DS1820 的 SCRATCHPAD 中读到了这样的一串数字:0x64,0x00,0x12,0x34,0xff,0xff,0x45,0x22,0x98。查表检验其正确性,过程如下。

序　号	输　入	输入与前部分余式异或	查表得部分余式
0	64	04	04
1	00	04	61
2	12	73	1A
3	34	2E	3C
4	ff	C3	28
5	ff	D7	D4
6	45	91	4F
7	22	6D	98
8	98	00	00

由 0 号输入字节查表得部分余式 04;1 号输入的 00 与 0 号的部分余式相异或得 04,04 查表得部分余式 61;以此类推,直到最后。由查表结果可知您读得的这串数据是正确的。

校验法之仿硬件 CRC 编码检测器法。我们知道,硬件 CRC 编码检测器是由异或运算器和移位寄存器组成。其中,异或运算器在两个输入 IN1,IN2 相同时,输出为 0,两个输入不同时输出为 1;移位寄存器则具有两个运算,即初始化和移位运算。初始化运算时,所有的寄存器比特位置 0,而在移位运算时,所有的比特位一次左移一位。用异或运算器和移位寄存器构成 DS1820 的硬件 CRC 编码检测器模型如下:



根据以上硬件模型,我们很容易编写出软件校验程序来。笔者拟将针对 8031、80C196、PIC 等单片机的汇编语言源程序清单发布在 http:// www .fenjin .com 网站上,供大家参考。

下面给出一个用 C 语言写成的小程序,我相信,读过后,您一定可以依此编写 CRC 校验程序了。

```
/ *  DS1820 crc test simulate hardware process  */
 # include  < stdio .h >
 # include  < conio .h >
 # include  < process .h >
void main( void)
{
      int i, j;
      unsigned char data0, data1, data2;
      while( 1)

   {      data0 = 0x00;
      printf( " Please enter 8 byte data for crc calculate" ) ;
      for  ( j = 0 ; j < 8 ; j + + )

      {      scanf( " %X" , &data2) ;
      for( i = 0 ; i < 8 ; i + + )
```

```
    {
      data1 = (data0&0x1)^(data2&0x1);
              data0 > > = 1;
        data2 > > = 1;
        if(data1) {data0^= 0xc; data0| = 0x80;}
        else data0& = 0x7f;
        printf("%X ", data0);                    / displaying process

}     printf("%X", data0);                      // part remainder

   }     printf("CRC = %X", data0);             // the last remainder
      printf("Enter = continue; any other key = exit");
      while( ! kbhit());
      if(getche() ! = 13) exit(1);
      }
}
```

参 考 文 献

1 Dallas Semiconductor data books . Dallas Semiconductor Corporation, 1995

2 FJ2000 电厂监控系统技术说明书 . 武汉奋进科技有限公司,2000(5)

3 串行通信 C 语言程序员指南 . 北京:清华大学出版社,1995(6)

选自《电测与仪表》月刊,2001 年第 2 期

5 .16　数字电位器在测量放大器中的应用

西安长安大学交通信息与控制工程系(710064)　居彩梅　汪贵平

所谓数字电位器其实就是用数字信号控制输出电阻值变化的元器件。它具有耐冲击、抗振动、噪音小、使用寿命长等优点。更重要的是其采用数字控制,可以方便地由计算机软件编程实现电阻的改变,不仅可使规模生产中的调节工作容易实现自动化,而且在操作使用上也容易实现智能化。

数字电位器的产品很多,其中应用较为典型的是美国 Xicor 公司推出的 X 系列非易失性数字电位器产品 E^2 POT。本文根据作者自己切身使用 X9221 的实践体验,介绍这种芯片的结构、原理及其在测量放大器中的应用。

一、数字电位器 X9221 基本结构及工作原理

X9221 把二个非易失性 E^2 POT 数控电位器集成在一个单片的 CMOS 微电路中。其引脚配置如图 5 .16 - 1 所示。

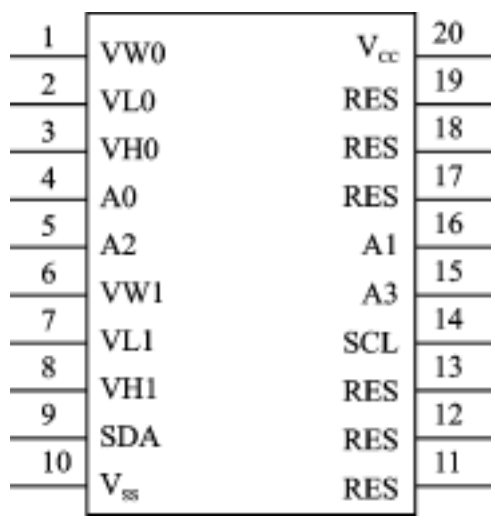


图 5 .16 - 1　X9221 引脚配置图

其中 VH0、VW0、VW1 分别为第一个电位器的高端、滑动端和低端(分别等效于机械电位器的固定端、中心抽头端、固定端);VH1、VW1、VL1 分别为第二个电位器的高端、滑动端和低端;A0、A1、A2 和 A3 为地址端,用来设置 8 位从属地址的低 4 位;SDA 和 SCL 分别为 I^2C 总线的串行数据线和串行时钟线,用来完成对电位器的控制;V_{SS} 和 V_{CC} 分别为地和电源;RES 端没有连接。

X9221 内部包括一个 I^2C 接口、二个数字电位器(每个数字电位器由电阻阵列及与之相应的滑动端计数寄存器 WCR)和四个 8 位数据寄存器 R0～R3 等部分。它的功能方框图如图 5 .16 - 2所示。

图 5 .16 - 2 X9221 功能方框图

X9221 串行接口支持双向总线的定向规约,在实际应用中,X9221 被考虑成一个从属器件,由主机启动数据传输并为发送和接收操作提供时钟,在 SDA 线的数据只有在 SCL 为低的期间才能改变状态,当 SCL 为高时 SDA 状态的改变被保留用作表示开始或终止的条件。

在开始条件后,主器件输出要访问的从器件地址,X9221 比较串行数据流与地址输入端的状态,若正确,发出响应信号。地址格式如图 5 .16 - 3 所示。X9221 识别出开始条件和它的从地址之后,给出一个应答作为响应,主器件接着发送一个字节包括指令及寄存器指针的信息,其格式如图 5 .16 - 4 所示。

0	1	0	1	A3	A2	A1	A0

图 5 .16 - 3 从器件的地址

13	12	11	10	0	P0/ P1	R1	R0

图 5 .16 - 4 指令字节的格式

当 X9221 在成功地接收命令字节后再一次应答,其滑动寄存器 WCR 的滑动端访问电阻阵列的抽头点,由多选一译码器来决定电子开关 FET 的通断,从而实现滑动端位置的改变,进一步实现了电阻的变化。WCR 的内容有四种方法来改变:(1) 由主机通过 Write WCR 指令来直接写入(串行加载);(2) 通过 XFR Data Register 指令把四个辅助数据寄存器之一的内容直接写入(并行加载);(3) 通过 Increment/ Decrement 指令一步一步地修改;(4) 可以在上电时装入数据寄存器 0(R0)的内容。

需要注意的是 WCR 是一个易失性存储器,即 X9221 断电时它的内容将丢失。

发送器件(不管是主器件还是从属器件)在发送完数据后将释放 SDA 总线,并为下一次操作做好准备。

需要说明的是,图 5 .16 - 3 中从器件的高 4 位地址是器件类型辨识符,对 X9221 来说,高 4 位为 0101[B],后 4 位是器件的地址,由 A0～A3 的状态来定义。图 5 .16 - 4 中最高 4 位是指令,第 6 位(P0/ P1)选择二个电位器中的哪一个,低 2 位(R0 和 R1)选择四个数据寄存器中的一个。

二、数字电位器 X9221 在测量放大电路中的应用

直流电压放大器在工业测控系统中应用非常广泛,其性能的好坏直接影响测控系统的准确度。而当放大器的输入信号有变化时,若使输出电压保持不变,就要求增益变化。改变增益的方法有多种,一种是使用机械电位器手动调整,其特点是线路简单,但不能自动改变量程;另

一种是采用多路模拟开关的通断来选择精密电阻网络中与所需增益相对应的电阻值,其特点是可由计算机自动改变量程,但量程变化范围有限且线路复杂。使用具有 I²C 总线数字电位器 X9221 不仅能实现量程多变,而且线路非常简单。

这里给出一例 X9221 直接替换机械电位器构成可程控增益放大器的接口电路,见图 5 .16 - 5。采用软件编程模式,使用户能应用数字量控制模拟开关的接通来选择电阻阵列中与所需要增益相应的电阻值。将 X9221 内部的二个电位器串联起来,再串联上电阻 R 作为增益电阻 R_G,从而实现由软件编程来调整放大器的增益。通过实现,规定 AD625 使用的 R_f 典型值为 20 kΩ,通过单片机控制数字电位器改变 R_G 实现增益调节。

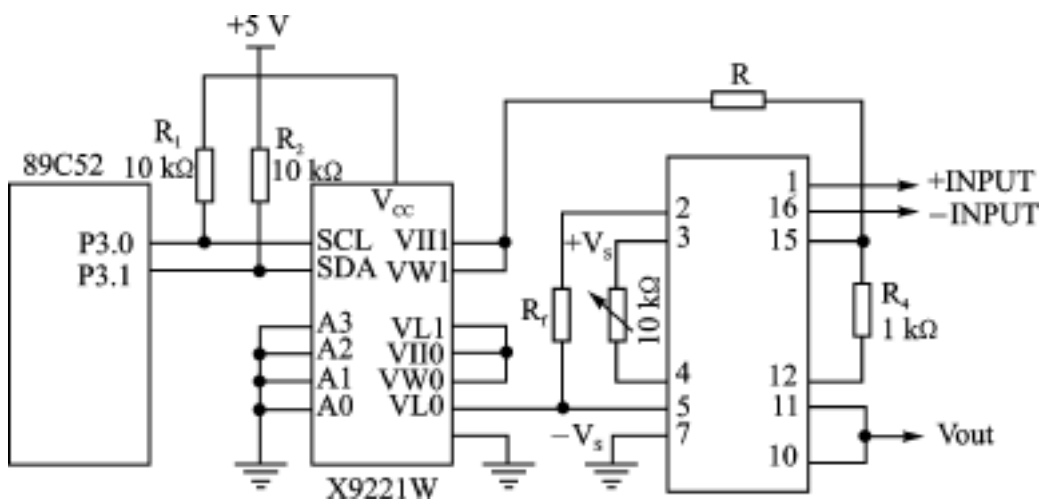


图 5 .16 - 5　应用电路连接图

由放大公式:$G = 1 + R_f / R_G$ 看出,若选用 $R_{f1} = R_{f2} = R_f = 20\ k\Omega$, $R = 10\ \Omega$(R_G 由 R 和数字电位器 X9221W 组成),则通过改变 R_G 从而实现增益 G 的改变。增益电阻可由如下公式确定:

$$R_G = 2\frac{R_f}{G - 1}$$

若 $R_G = 4\ k\Omega$ 时,则 $G = 11$

$$R_G = \frac{2\ k\Omega}{63\ \Omega} + 10\ \Omega \approx 40\ \Omega$$ 时,$G \approx 1000$

对于迭加在高共模电压上的微弱差分输入信号,需要采用一级放大,从而得到±10 V 的电压信号:

当 $V_1 = \pm 10$ mV,取 $G = 1000$,则 $R_G \approx 40\ \Omega$、$V_{OUT} = \pm 10$ V;

当 $V_1 = \pm 20$ mV,取 $G = 500$,则 $R_G \approx 80\ \Omega$、$V_{OUT} = \pm 10$ V;

当 $V_1 = \pm 50$ mV,取 $G = 200$,则 $R_G \approx 200\ \Omega$、$V_{OUT} = \pm 10$ V;

当 $V_1 = \pm 100$ mV,取 $G = 100$,则 $R_G \approx 404\ \Omega$、$V_{OUT} = \pm 10$ V;

当 $V_1 = \pm 200$ mV,取 $G = 50$,则 $R_G \approx 816\ \Omega$、$V_{OUT} = \pm 10$ V;

当 $V_1 = \pm 500$ mV,取 $G = 20$,则 $R_G \approx 2105\ \Omega$、$V_{OUT} = \pm 10$ V。

由以上电路分析可以看出,采用非易失性数字电位器 X9221,利用 $G = 1 + R_f / R_G$ 实现增益的改变,其优缺点是:能在全量程范围内自动切换,便于计算机控制,使用方便;数字电位器采用 I²C 总线和单片机通信,布线少且性能可靠;由于数字电位器阻值的限制,所以 $G_{min} = 1$,不能实现单位增益放大器。

由图 5 .16 - 5 可以看出,采用 I²C 总线数字电位器,使得硬件电路得到简化,却增加了软件的工作量。假如有了接口程序模块,对不需要掌握所有 X9221 指令的设计者来说,可以把

不需要的子程序去掉以缩短代码。当然,这需要重新汇编这组代码。

在上图的连接中,89C52 的时钟为 8 MHz, X9221 的器件地址固定为 01010000B,即 50H。命令字节放在 7CH 单元,程序执行结果将读出的数据放在 7DH 单元,要写入的数据放在 7EH 单元。

E^2 POT 驱动程序:

```
SETRF:          DA BIT      3.0
                SCL BIT     P3.1
                INCDEC BIT  02H
                DEVICEW     DATA 50H
                COMMAND     EQU 7CH
                RD-DATA     EQU 7DH
                WR-DATA     EQU 7EH
                ACALL       START-CONT
                MOV         A, #DEVICEW
                ACALL       WR-BYTE
                MOV         A, COMMAND
                SWAP        A
                ANL         A, #0FH
                CIJE        A, #09H,XRWA
                …
START-CONT:     SETB        SDA
                SETB SCL
                NOP
                NOP
                CLR         SDA
                NOP
                NOP
                CLR         SCL
                RET
STOP-CONT:      CLR         SDA
                SETB        SCL
                NOP
                NOP
                SETB        SDA
                RET
                …
CLOCK:          NOP
                NOP
                SETB        SCL
                NOP
                NOP
                MOV         C,SDA
```

```
CLR          SCL
RET
```

三、结束语

随着 I^2C 总线应用的日益广泛,兼容 I^2C 总线的器件品种也越来越多,其中数字电位器以其调节方便、准确,受物理环境的影响小以及性能稳定等特点,已被广大电子工程技术人员所认识。尤其随着仪器及设备的数字化程度不断提高,作为机械电位器的替代产品,数字电位器的应用将更为广泛。

参 考 文 献

1　Analog Device, Data Converter Reference Manual Volume II[Z],1992

2　Xicor 非易失性器件使用手册[Z] . P&S 武汉力源电子股份有限公司,1996

3　薛均义 . MCS－51/ 96 系列单片微型计算机及其应用[M] . 西安:西安交通大学出版社,1991

4　徐爱钧 . 智能化测量控制仪表原理与设计[M] . 北京:北京航空航天大学出版社,1995

5　宋大雷 . 数字电位器的工作原理及应用[J] . 电气自动化,1999(1)

6　汪贵平,等 . 增益可编程仪用放大器 AD525 工作原理及应用[J] . 电子技术应用,1993(3)

选自《电测与仪表》月刊,2001 年第 2 期

第六章

总线及其

应用技术

6.1 按平台模式设计的虚拟 I²C 总线软件包 VIIC

北京航空航天大学 何立民

一、I²C 总线及其虚拟应用

1. I²C 总线应用呼唤平台模式

目前,单片机应用系统的外围扩展已从并行方式为主过渡到以串行方式为主的时代。许多新型外围器件都带有串行扩展接口。通常的串行扩展接口和串行扩展总线有 UART 的移位寄存器方式、MOTOROLA 公司的 SPI、NS 公司的 Microwire、Dallas 公司的 1-Wire 和 Philips 公司的 I²C 总线等。其中,I²C 总线提供了较完善的总线协议、最简单的串行连接方式,并提供了总线操作的状态处理软件包,因而得到了广泛的应用。但厂家并未提供完善的平台模式应用软件包。因此,用户在扩展 I²C 总线外围器件时,还要在了解 I²C 总线协议、操作原理的基础上,采用直接方式进行 I²C 总线外围器件的应用程序设计。迄今为止,许多期刊的文章中还是以这种方式来介绍 I²C 总线的扩展应用。由于 I²C 总线协议的复杂性和操作管理的特殊性,从 I²C 总线结构原理到 I²C 总线应用的直接设计方式难度较大,使 I²C 总线推广应用较慢。因此,迫切呼唤推出 I²C 总线的应用软件平台,使人们不必了解 I²C 总线就能设计 I²C 总线应用程序。

2. 广泛使用的主工作方式

I²C 总线是 1 个十分完善的多主系统总线,总线上可以挂接多个 MCU,因此有 4 种工作方式,即主发送、主接收、从发送、从接收。但实际的单片机应用系统绝大多数都是单个 MCU 系统,只用到 I²C 总线的主方式,即主发送与主接收。

3. 虚拟 I²C 总线的广泛需求

目前,有许多外围器件带有 I²C 总线接口,然而,带有 I²C 总线接口的 MCU 只有少数厂家的个别型号,致使 I²C 总线难以推广。因此,人们便使用 MCU 通用的 I/O 口来虚拟 I²C 总线接口。早期东芝公司在彩电中就在自己的 MCU 中虚拟 I²C 总线接口,实现 I²C 总线外围器件扩展的虚拟应用。

采用虚拟 I²C 总线后,任何 1 个厂家的单片机都可以无障碍地使用 I²C 总线外围器件。同样,虚拟 I²C 总线的应用也呼唤平台模式。

二、VIIC1.0 软件包设计

VIIC1.0 是物化形式为软件包文档的广义平台,适合在 80C51 单片机系列单主系统中应用。

按照广义平台设计内容,VIIC 有最佳包容性设计、后归一化设计、前归一化设计、物化设计和应用界面设计。

1. 最佳包容性设计

最佳包容性设计是广义平台适用范围的最佳选择性设计。完整的 I²C 总线有 4 种操作方式,并且有指定的端口,有地址寄存器(S1ADR)来设定 MCU 的地址。在 VIIC 的包容性设计中规定为主方式下的通用 I/O 口虚拟。规定为主方式后,避免了多主方式下极大难度的总线冲突仲裁处理程序设计,又保证了能满足绝大多数 I²C 总线扩展应用。采用通用 I/O 口的虚拟,使用户可随意规定虚拟 I²C 总线端口,扩大了虚拟 I²C 总线应用的灵活性。

2. 后归一化设计

后归一化设计是确定应用范围的相关设计。VIIC的后归一化设计有:

(1) 应用范围指定为带 I²C 总线的外围器件扩展;

(2) 所有 I²C 总线外围器件的操作方式归一化为外围器件 N 个字节的读写操作;

(3) 总线节点通信方式归一化为 SLAW/ SLAR 节点寻址后的点对点的读写操作。

3. 前归一化设计

前归一化设计是从 I²C 总线协议原理、操作方式、时序规则出发,实现后归一要求的设计。VIIC 的前归一化设计有:

(1) 时序的指令模拟即模拟 I²C 总线操作时序。I²C 总线操作的典型时序信号有起始位(STAR)、停止位(STOP)、发送应答位(MACK)、发送非应答位(MNACK)。

(2) 数据传送操作虚拟即模拟 I²C 数据传送过程。例如,应答位检查(CACK)、发送 1 个字节数据(WRBYT)、接收 1 个字节数据(RDBYT)。

(3) 外围器件读写操作虚拟。要求虚拟 1 个 I²C 总线读写操作过程(RDNBYT、WRNBYT)。

4. VIIC1.0 的结构设计

(1) VIIC1.0 的组成

根据归一化设计,主方式下虚拟 I²C 总线由下列 9 个子程序组成:

- 时序模拟子程序 STAR, STOP, MACK, MNACK;
- 操作模拟子程序 CACK, WRBYT, RDBYT;
- 数据读写子程序 RDNBYT, WRNBYT。

由于篇幅所限,本文中省略了 VIIC1.0 软件包中这 9 个子程序的虚拟设计方法。需要详细了解的读者可参看文尾参考资料 2 中 7.3 节“I²C 总线的串行扩展技术”的有关部分。

(2) 软件包的出口界面

软件包 VIIC 实现非介入性操作,出口界面是软件包应用时惟一的触及面。VIIC1.0 中的出口界面为数据读写子程序 RDNBYT/ WRNBYT。

(3) 软件包的符号单元

VIIC 中的符号标记有发送数据缓冲区 MTD、接收数据缓冲区 MRD、传送字节数存放单元 NUMBYT 以及寻址字节 SLAW/ SLAR 存放单元 SLA。这些符号单元都采用了标准 I²C 总线状态处理软件包中规定的字符标记。

5. 应用界面设计

VIIC1.0 软件包规定了读写子程序 RDNBYT/ WRNBYT 的惟一出口界面,因此 RDNBYT/ WRNBYT 的调用操作命令,以及满足调用操作的初始化操作的三条命令为 VIIC 的应用界面,即:

```
MOV     LA, # SLAR/ SLAW          总线上节点寻址并确定传送方向
MOV     NUMBYT, # N               ;确定传送字节数 N
LCALL   RDNBYT/ WRNBYT            ;读/ 写操作调用
```

三、VIIC1 .0 软件包清单

VIIC1 .0 软件包清单如下。

```
① STAR:     ETB     SDA           启动 I²C 总线
            SETB    VSCL
            NOP
            NOP
            CLR     VSDA
            NOP
            NOP
            CLR     VSCL
            RET
② STOP:     CLR     VSDA          ;停止 I²C 总线数据传送
            SETB    VSCL
            NOP
            NOP
            SETB    VSDA
            NOP
            NOP
            CLR     VSDA
            CLR     VSCL
            RET
③ MACK:     CLR     VSDA          ;发送应答位
            SETB    VSCL
            NOP
            NOP
            CLR     VSCL
            SETB    VSDA
            RET
④ MNACK:    SETB    VSDA          ;发送非应答位
            SETB    VSCL
            NOP
            NOP
            CLR     VSCL
            CLR     VSDA
            RET
⑤ CACK:     SETB    VSDA          ;应答位检查
            SETB    VSCL
            CLR     F0
```

```
                MOV     C,VSDA
                JNC     CEND
                SETB    F0
CEND:           CLR     VSCL
                RET
⑥ WRBYT:        MOV     R0,#08H         ;向 VSDA 线上发送 1 个数据字节
   WLP:         RLC     A
                JC      WR1
                AJMP    WR0
   WLP1:        DJNZ    R0,WLP
                RET
   WR1:         SETB    VSDA
                SETB    VSCL
                NOP
                NOP
                CLR     VSCL
                CLR     VSDA
                AJMP    WLP1
   WR0:         CLR     VSDA
                SETB    VSCL
                NOP
                NOP
                CLR     VSCL
                AJMP    WLP1
⑦ RDBYT:        MOV     R0,#08H         ;从 VSDA 线上读取 1 个数据字节
   RLP:         SETB    VSDA
                SETB    VSCL
                MOV     C,VSDA
                MOV     A,R2
                RLC     A
                MOV     R2,A
                CLR     VSCL
                DJNZ    R0,RLP
                RET
⑧ WRNBYT:       MOV     R3,NUMBYT       ;虚拟 I2C 总线发送 N 个字节数据
                LCALL   STA
                MOV     A,SLA
                LCALL   WRBYT
                LCALL   CACK
                JB      F0,WRNBYT
                MOV     R1,#MTD
   WRDA:        MOV     A,@R1
                LCALL   WRBYT
```

```
                LCALL   CACK
                JB      F0,WRNBYT
                INC     R1
                DJNZ    R3,WRDA
                LCALL   STOP
                RET
⑨ RDNBYT:       MOV     R3, NUMBYT      ;模拟 I²C 总线接收 N 个字节数据
                LCALL   STA
                MOV     A,SLA
                LCALL   WRBYT
                LCALL   CACK
                JB      F0,RDNBTY
    RDN:        MOV     R1, # MRD
    RDN1:       LCALL   RDBYT
                MOV     @R1,A
                DJNZ    R3,ACK
                LCALL   MNACK
                LCALL   STOP
                RET
    ACK:        LCALL   MACK
                INC     R1
                SJMP    RDN1
```

四、VIIC1 .0 应用指南

1 . 适用范围

VIIC1 .0 适用于 80C51 系列单主系统中 I²C 总线外围器件扩展的应用程序设计。由于时序模拟基于 6 MHz 时钟设计,在高速时钟下,可适当增加时序模拟子程序中的空操作指令。

2 . 资源占用

VIIC 使用了 R0, R1, R2, R3, F0, C 等资源。

3 . 符号单元

VIIC 中有许多符号标记,这些符号标记有:

VSDA	虚拟 I²C 总线数据线;
VSCL	虚拟 I²C 总线时钟线;
SLA	寻址字节存放单元;
NUMBYT	传送字节数存放单元;
MTD	发送数据缓冲区;
MRD	接收数据缓冲区。

4 . 归一化操作命令

不论总线上扩展什么外围器件,都只须使用以下三条指令:

```
MOV     LA, # SLAW/ SLAR
MOV     NUMBYT, # N
```

LCALL　WRNBYT/ RDNBYT

5. VIIC 的装载

由于 WRNBYT/ RDNBYT 都使用长调用命令 LCALL,故 VIIC1.0 可放在程序存储器的任意空间。

6. 通用的应用界面

VIIC1.0 的应用界面如图 6.1-1 所示。与 VIIC1.0 有关的只是 3 条归一化操作命令;与硬件电路相关的是器件地址与引脚地址构成的寻址字节和由器件规定的数据操作格式。

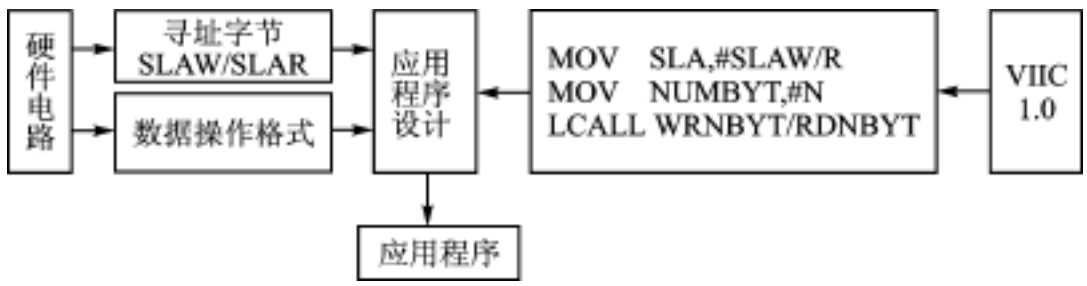


图 6.1-1

五、VIIC1.0 应用示例

在本文参考文献 2 中给出了一些基于 VIIC 软件包的外围串行扩展应用实例。现就带 I²C 总线接口 LED 显示驱动器 SAA1064 扩展 8 位 LED 显示电路来示范 VIIC1.0 的应用。

1. 硬件电路与寻址字节

用 2 片 SAA1064 扩展的 8 位 LED 显示电路如图 6.1-2(b)所示,图 6.1-2(a)为 SAA 的引脚排列图。设 80C51 用 P1.1/ P1.0 来虚拟 SDA/ SCL 口线。两片 SAA1064 的 ADR 引脚分别接地和 V_{CC}。SAA1064(1)、(2)的器件地址为 0111,引脚地址为 000 和 111,因此它们的寻址字节 SLAW/ SLAR 分别为 70H/ 71H 和 76H/ 77H。

2. 显示原理与数据操作格式

在 I²C 总线外围器件的数据手册中,都给出了实现器件功能的数据操作格式,以及实现功能的控制命令与寄存器的管理模式。

(1) SAA1064 的数据操作格式

80C51 只须对 SAA1064 进行写操作,便实现了 LED 的显示驱动控制。SAA1064 的写数据操作格式如下:

S	SLAW	A	SUBADR	A	COM	A	data1	A	data2	A	data3	A	data4	A	P

只须对 SAA1064 中首地址为 SUBADR 的 5 个寄存器单元中依次写入控制命令 COM 和 4 个 LED 共阴极段码 datal～data4,SAA1064 的 SUBADR = 00H。

控制命令 COM 格式及位功能规定如下:

D7							D0
——	C6	C5	C4	C3	C2	C1	C0

C0　静、动态显示选择。C0 = 1,动态显示。

C1　显示位 1,3 暗亮选择。C1 = 1,选择亮。

C2 显示位 2,4 暗亮选择。C2 = 1,选择亮。

C3 测试位。C3 = 1,所有段点亮。

C4,C5,C6 驱动电流控制位。C4,C5,C6 为“1”时,驱动电流分别为 3 mA,6 mA,12 mA;皆为 1 时输出驱动电流最大,达 21 mA。

3.应用程序设计

按下列步骤完成图 6.1-2(b)的 8 位 LED 显示程序设计。

(1) 将 VIIC1.0 装入程序存储器中。

(2) 根据硬件电路及资源分配,将 VIIC1.0 中的符号单元赋值如下:

```
VSDA    QU    1.1        用 P1.1 虚拟 SDA
VSCL    EQU   P1.0       ;用 P1.0 虚拟 SCL
SLA     EQU   50H        ;50H 为寻址字节存放单元
NUMBYT  EQU   51H        ;51H 为传送字节数据存放单元
MTD     EQU   30H        ;30H 为发送缓冲区首地址
```

(3) 8 位 LED 显示子程序设计

这里介绍 1 个在图 6.1-2(b)LED 显示器上显示“bUAA0706”固定字符的子程序。

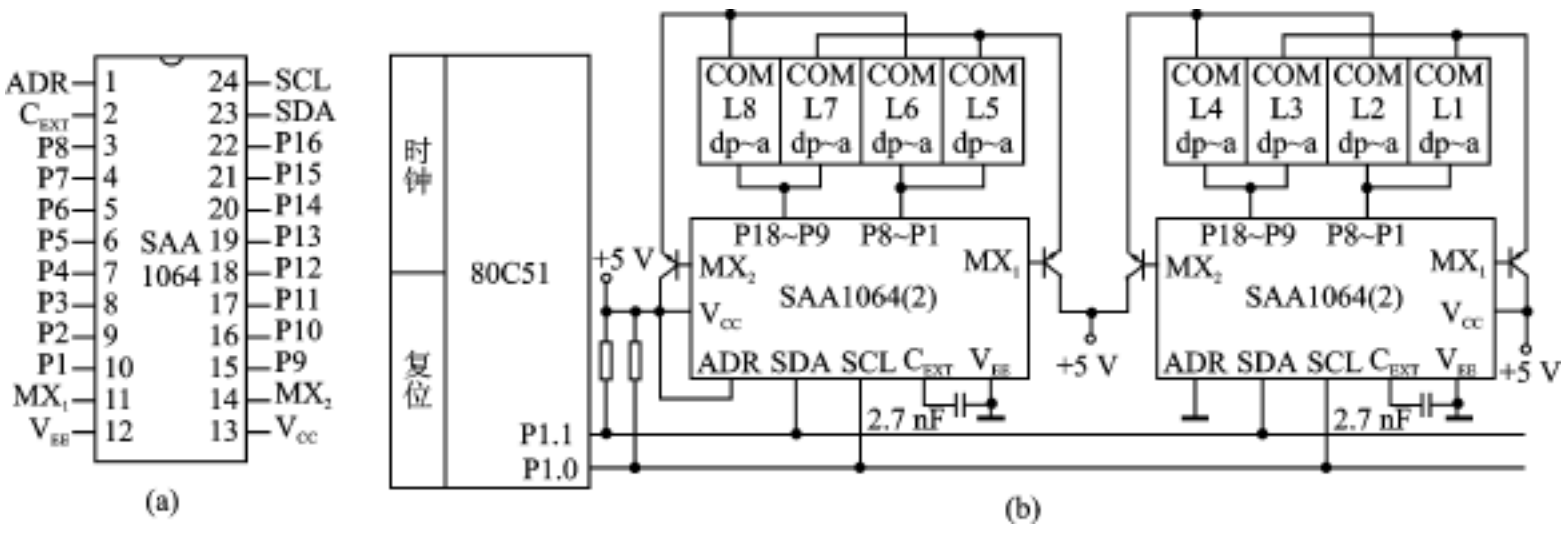


图 6.1-2

根据 SAA1064 的数据操作格式,点亮 4 个 LED 只须依首地址 SUBADR 顺序送入控制命令 COM 和 4 个 LED 共阴极段选码 data1～data4。

设 LED 驱动电流为 18 mA(C6,C5 为高电平;C4 为低电平)动态显示要求(C2,C1,C0 为高电平)故 COM = 67H。SAA1064(1)显示“0706”的段码依次是 7DH,3FH,07H,3FH;SAA1064(2)显示“bUAA”的段码依次为 77H,77H,3EH,7CH。因此,SAA1064(1),(2)的数据操作格式具体化为

SAA1064(1):

S	70H	A	00H	A	67H	A	7DH	A	3FH	A	07H	A	3FH	A	P

SAA1064(2):

S	76H	A	00H	A	67H	A	77H	A	77H	A	3EH	A	7CH	A	P

设显示“bUAA0706”的子程序名为 VSAA8。VSAA8 的程序清单如下:

```
SDA     QU    P1.0
```

```
        VSCL    EQU   P1.1
        SLA     EQU   50H
        NUMBYT  EQU   51H
        MTD     EQU   30H
VSAA8:  MOV     30H,#00H        ;将 SUBADR,COM,LED 共阴极段码 data1～data4 依次
                                ;装入发送缓冲区
        MOV     31H,#67H
        MOV     32H,#7DH
        MOV     33H,#3FH
        MOV     34H,#07H
        MOV     35H,#3FH
        MOV     SLA,#70H        ;寻址 SAA104(1)并为发送状态
        MOV     NUMBYT,#06H     ;确定发送字节数
        LCALL   WRNBYT          ;调用 VIIC1.0 中 N 个字节写入子程序
        MOV     30H,#00H        ;将 SUBADR,COM,data1～data4 依次装入发送缓冲区
        MOV     31H,#67H
        MOV     32H,#77H
        MOV     33H,#77H
        MOV     34H,#3EH
        MOV     35H,#7CH
        MOV     SLA,#76H        ;寻址 SAA1064(2)并为发送状态
        MOV     NUMBYT,#06H     ;确定发送字节数
        MOV     WRNBYT          ;调用 VIIC1.0 中 N 个字节写入子程序
        RET
```

如果要随意显示内存中的 8 个 BCD 码,可开辟显示缓冲区 DISRAM,设计 1 个子程序,将显示缓冲区中的 8 个 BCD 码转换成共阴极段码再和SUBADR和 COM 一道送入 MTD 中,然后使用 VIIC1.0 的三条操作命令即可。

从上述应用可看出,使用 VIIC 软件包后,编写应用程序不必了解 I²C 总线原理、协议和时序,只要了解 VIIC 的应用操作即可。

参 考 文 献

1 何立民.I²C 总线应用系统设计.北京:北京航空航天大学出版社,1995

2 何立民.单片机高级教程.北京:北京航空航天大学出版社,2000

选自《单片机与嵌入式系统应用》月刊,2001 年第 2 期

6.2 虚拟 I^2C 总线软件包的开发及其应用

北京航空航天大学机械工程及自动化学院 周正干 李和平

一、I^2C 总线简介

串行扩展总线在单片机系统中的应用是目前单片机技术发展的一种趋势。在目前比较流行的几种串行扩展总线中,I^2C 总线以其严格的规范和众多带 I^2C 接口的外围器件而获得广泛的应用。

I^2C 总线是 PHILIPS 公司推出的芯片间串行传输总线。它以 1 根串行数据线(SDA)和 1 根串行时钟线(SCL)实现了全双工的同步数据传输。随着 I^2C 总线研究的深入,它已经广泛应用于视/音频领域、IC 卡行业和一些家电产品中,在智能仪器、仪表和工业测控领域也越来越多地得到应用。

I^2C 总线的广泛应用是同它卓越的性能和简便的操作方法分不开的。I^2C 总线的特点主要表现在以下几个方面:

(1) 硬件结构上具有相同的硬件接口界面。I^2C 总线系统中,任何一个 I^2C 总线接口的外围器件,不论其功能差别有多大,都是通过串行数据线(SDA)和串行时钟线(SCL)连接到 I^2C 总线上。这一特点给用户在设计应用系统中带来了极大的便利性。用户不必理解每个 I^2C 总线接口器件的功能如何,只要将器件的 SDA 和 SCL 引脚连到 I^2C 总线上,然后对该器件模块进行独立的电路设计,从而简化了系统设计的复杂性,提高了系统抗干扰的能力,符合 EMC(Electromagnetic Compatibility)设计原则。

(2) 总线接口器件地址具有很大的独立性。在单主系统中,每个 I^2C 接口芯片具有惟一的器件地址,由于不能发出串行时钟信号而只能作为从器件使用。各器件之间互不干扰,相互之间不能进行通信,各个器件可以单独供电。MCU 与 I^2C 器件之间的通信是通过独一无二的器件地址来实现的。

(3) 软件操作的一致性。由于任何器件通过 I^2C 总线与 MCU 进行数据传送的方式是基本一样的,这就决定了 I^2C 总线软件编写的一致性。

(4) PHILIPS 公司在推出 I^2C 总线的同时,也为 I^2C 总线制定了严格的规范,如:接口的电气特性、信号时序、信号传输的定义等。规范的严密性,结构的独立性和硬、软件接口界面的一致性,极大地方便了 I^2C 总线设计的模块化和规范化,伴随而来的是用户在使用 I^2C 总线时的“傻瓜”化。

二、开发虚拟 I^2C 总线软件包的必要性

所谓的虚拟 I^2C 总线是指借用 MCU 的 2 根通用 I/O 口线来虚拟 I^2C 总线接口,而用软件来实现 I^2C 协议的一种总线虚拟方式。

虚拟 I^2C 总线软件包的产生及其应用是由虚拟 I^2C 总线的广泛需求和 I^2C 总线的特点来

决定的。其原因和应用现状主要表现在以下几个方面:

(1) 有大量的外围接口器件,如 PCF8574、PCF8591、CAT24WC256C 等都带有 I²C 接口。可是带 I²C 总线接口的 MCU 却不多,例如 AT89C52、8051、68HC05 和 68HC11 等都不带 I²C 总线接口。虚拟 I²C 总线的产生大大地扩展了 I²C 总线外围器件的应用范围。

(2) PHILIPS 公司为了将 I²C 总线器件应用到不带 I²C 接口的 MCU 系统中,推出了 PCF8584 作为不带 I²C 接口的 MCU 与 I²C 总线连接的转接器。但是,在一些小型的单片机应用系统和对时序要求不是很严格的系统中,这种做法不仅会增加硬件成本,而且会增加线路的复杂性,不利于实现系统设计的最简化和最优化。

(3) 如简介所述,I²C 总线在进行数据传输方面,对于不同的 I²C 器件具有相同的操作模式,遵循相同的规范,这就为 I²C 总线的软件虚拟提供了实现的可能性。

(4) 通过查阅近几年 I²C 总线应用方面的资料,发现虚拟 I²C 总线有了一定程度的应用。但是,在应用的过程中也有一些不尽人意的地方,如:大多数 I²C 总线操作程序都是针对具体的器件,按照 I²C 总线协议从最基本的时序编写的;I²C 协议的复杂性和不同器件的互异性决定了一个程序一个样,可移植性较差,给 I²C 总线的应用造成了不应有的障碍。因此,I²C 总线的用户只有转变观念,将 I²C 总线的应用开发由板凳模式变成平台模式,才能够为 I²C 总线应用范围的进一步扩大产生更大的推动作用。

基于此,本文将虚拟 I²C 总线软件包 VIICC1.0 公布出来,希望它能够给 I²C 总线应用人员提供一些帮助。

三、虚拟 I²C 总线软件包 VIICC1.0

I²C 总线是一种多主系统总线,总线上可有多个 MCU,因此有 4 种工作方式:主发送、主接收、从发送和从接收。虚拟 I²C 总线软件包 VIICC1.0 虚拟了单主系统的 I²C 总线,即主发送和主接收。不过,在实际应用中,单片机应用系统绝大多数是单主系统,因此它不会影响虚拟 I²C 总线软件包的广泛应用。

虚拟 I²C 总线软件包 VIICC1.0 是用 Franklin C51 编写的。它由 1 个头文件和 1 个源程序组成。头文件(VIICC.H)的清单如下:

```
/ * VIICC.H */
 #pragma db cd small
 #include <intrins.h>
 #include <reg52.h>
 #define NOP _nop_(); _nop_(); _nop_(); _nop_()
 #define SDA P1^7
 #define SCL P1^6

 void sta();
 void cack();
 void stop();
 void mnack();
 void mack();
 void wrbyt(uchar shu);
```

```
void wrnbyta(uchar slaw,uchar gg[],uchar n);
void wrnbyt(uchar slaw,uchar ff[],uchar number);
uchar rdbyt( );
void rdnbyt(uchar slar,uchar qq[],uchar number);
```

源程序由 9 个函数组成,程序清单如下:

```
void   sta( )                    * 启动 I²C 总线 */
{
   SDA = 1;
   SCL = 1;
   NOP;
   SDA = 0;
   NOP;
   SCL = 0;
}

void stop( )                    / * 停止 I²C 总线 */
{
   SDA = 0;
   SCL = 1;
   NOP;
   SDA = 1;
   NOP;
   SCL = 0;
}

void mack( )                    / * 发送应答位 */
{
   SDA = 0;
   SCL = 1;
   NOP;
   SCL = 0;
   SDA = 1;
}

void mnack( )                   / * 发送非应答位 */
{
   SDA = 1;
   SCL = 1;
   NOP;
   SCL = 0;
   SDA = 0;
}
```

```
void cack( )                    /*应答位检查*/
{
   SDA = 1;
   SCL = 1;
   F0 = 0;
   if(SDA = = 1)
       F0 = 1;
   SCL = 0;
   _nop_( );
}
/*向虚拟 I²C 总线上发送 1 个数据字节*/
void wrbyt(uchar shu)
{
   uchar i;
   for(i = 0;i < 8;i + + ){
     if((shu&0x80) > 0){
        SDA = 1;
        SCL = 1;
        NOP;
        SCL = 0;
        SDA = 0;
     }
     else{
        SDA = 0;
        SCL = 1;
        NOP;
        SCL = 0;
     }
     shu = shu < < 1;
   }
}
/*从 I²C 总线上读取 1 个数据字节*/
uchar rdbyt( )
{
   uchar nn = 0xff, mm = 0x80 ,uu = 0x7f;
   uchar j;
   for(j = 0;j < 8 ;j + + ){
     SDA = 1;
     SCL = 1;
     if(SDA = = 0)
        nn = (nn&uu);
     else
```

```
            nn = (nn|mm);
            nn = _crol_(nn,1);
        SCL = 0;
    }
    return(nn);
}

void wrnbyta(uchar slaw,uchar gg[],uchar n)
{
    do{
        sta( );
        wrbyt(slaw);
        cack( );
    }while(F0 = = 1);
    wrbyt(gg[n]);
    cack( );
}
/ * 向虚拟 I²C 总线上发送 n 个字节的数据 */
void wrnbyt(uchar slaw,uchar ff[],uchar number)
{
    uchar idata k;
    do{
        sta( );
        wrbyt(slaw);
        cack( );
    }while(F0 = = 1);
    for(k = 0;k < number;k + + ){
        wrbyt(ff[k]);
        cack( );
        while(F0 = = 1)
            wrnbyta(slaw,ff,k);
    }
    stop( );
}
/ * 从虚拟 I²C 总线上读取 n 个字节的数据 */
void rdnbyt(uchar slar,uchar qq[],uchar number)
{
    uchar idata data0,l;
    do{
        sta( );
        wrbyt(slar);
        cack( );
        }while(F0 = = 1);
```

```
    for(l = 0;l < number - 1;l + + ){
        data0 = rdbyt( );
        qq[l] = data0;
        mack( );
    }
    data0 = rdbyt( );
    qq[l] = data0;
    mnack( );
    stop( );
}
```

四、虚拟 I²C 总线软件包 VIICC1.0 应用界面

在应用 VIICC1.0 时,用户要做的工作是:

(1) 将虚拟 I²C 总线软件包的头文件复制到 Franklin C51 编译器的 ..\c51\inc 目录下;

(2) 将上面介绍的源程序复制到应用系统主程序的末尾;

(3) 在主程序的最上面添加如下 1 条指令:

```
# include < VIICC.H >
```

做完上面的工作后,用户就可以自如地使用这个软件包了。在使用时,用户不必关心 I²C 外围器件功能如何,都使用下面的几条指令:

```
rdnbyt(uchar slar,uchar qq[],uchar number);
wrnbyt(uchar slaw,uchar ff[],uchar number);
```

或

其中: lar/slaw 是从器件的地址(读/写);

number 是需要读写的数据字节的个数;

qq[]是从虚拟 I²C 总线上读出的数据数组;

ff[]是需要向虚拟 I²C 总线上写入的数据数组。

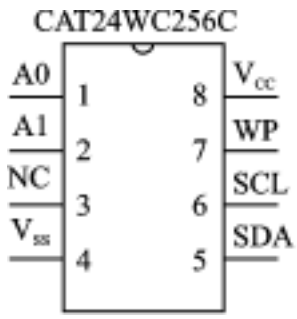
VIICC1.0 占用资源与 VIIC1.0(见《单片机与嵌入式系统应用》2001 年第 2 期上的相应文章)占用资源基本上一样。由于 VIICC1.0 的设计是基于 12 MHz 的时钟设计的,因此当晶振频率发生变化时,须要在头文件中适当修改_nop_()的个数。另外,当所使用的通用 I/O 口线发生变化时,须要在头文件中对 SDA 与 SCL 的宏定义进行修改。

五、虚拟 I²C 总线软件包在 CAT24WC256C 中的应用

CAT24WC256C 是 CATLYST 推出的可直接和 MCS-51 接口的 E²PROM。它遵循 I²C 总线协议,使用 SDA 和 SCL 双总线进行数据通信。

CAT24WC256C 是 256K 位(32 KB)的串行 CMOS E²PROM,有 1 个 64 B 的页写缓冲器。它能在 1.8～6.0 V 的宽电压范围内工作,实现了写保护功能,即当其 WP 引脚为高电平时进入写保护状态。CAT24WC256C 的引脚分配及其定义如图 6.2-1 所示。

从图 6.2-1 中可以看出,在单主系统中,由 A1 和 A0 地址译码最多可以接 4 片 CAT24WC256C 器件,即最多可扩展 128 KB 的 E²PROM。由于它不能发出串行时钟信号和启、停信号,因此,只能作为从器件被操作。



引脚	功 能
A0 A1	地址输入
SDA	串行数据/地址
SCL	串行时钟
WP	写保护
V_{cc}	+1.8 V~6.0 V电源
V_{ss}	地
NC	未连接

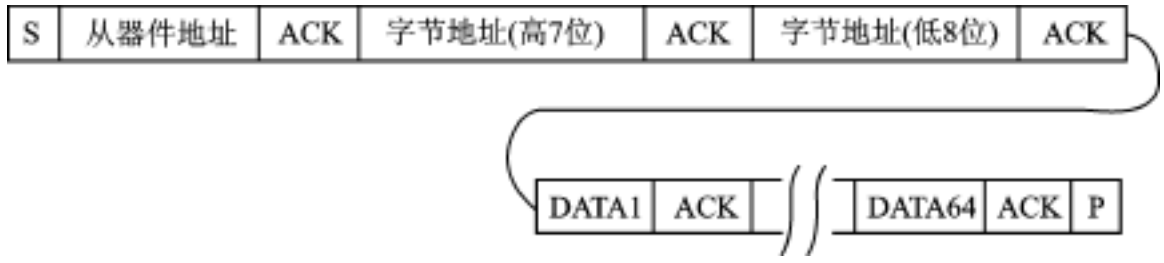
图 6.2-1 CAT24WC256C 的引脚分配及其定义

CAT24WC256C 的器件地址格式为:

1	0	1	0	0	A1	A0	R/瑏

这个器件地址的最高 5 位固定为 10100,A1 和 A0 为器件的地址,这些位必须与硬件连线输入脚 A1 和 A0 位相对应,最低位(R/瑏)为读写控制位,“1”表示对器件进行读操作;“0”表示对器件进行写操作。CAT24WC256C 收到起始信号和从器件地址字节后,比较从器件地址,如果发现与其自身地址相符合,则响应 1 个应答信号,然后根据(R/瑏)位来进行读/写操作。

CAT24WC256C 的写操作中有 2 种方式:字节写或页写,它们的本质是一样的,不同的是所传输的数据字节数不同,例如页写时序如下:



CAT24WC256C 的读操作有 3 种方式:立即/当前地址读、随机读和连续读。立即地址读与随机读基本上是一样的,不同的是立即读的地址是上次进行操作的地址 + 1;随机读可以由用户随意指定 1 个地址;连续读可以对 1 个存储区进行多个字节的连续读。以连续读为例,它的操作时序如下:

在操作中要注意页的自动翻转问题,解决页自动翻转问题的方法是在页操作中不要一次传输超过 64 个字节的数据。

虚拟 I^2C 总线软件包 VIICC1.0 在 CAT24WC256C 中应用的电路原理图如图 6.2-2 所示。

在这个应用实例中,首先向 CAT24WC256C 中起始地址为 0x0080 连续存储区中写入 64 个字节的数据,然后再将这 64 个字节读出来,放在输出数组中,下面是部分源代码。

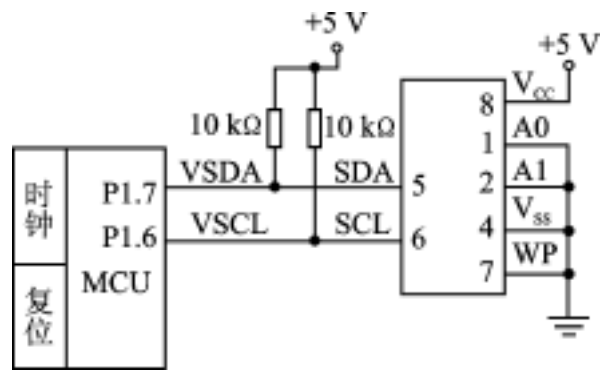


图 6.2-2 虚拟 I²C 总线软件包
在 CAT24WC256C 中的应用原理图

……

```
#include <VIICC.H>
#define uchar unsigned char
#define uint unsigned int
uchar code array[64] = { x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x4F,0x00,
                         0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x07,0x00,0x00,0x00,0x14,
                         0x7F,0x14,0x7F,0x14,0x00,0x00,0x00,0x24,0x2A,0x7F,0x2A,0x12,0x00,
                         0x00,0x00,0x23,0x13,0x08,0x64,0x62,0x00,0x00,0x00,0x36,0x49,0x55,
                         0x22,0x50,0x00,0x00,0x00,0x00,0x05,0x03,0x00,0x00,0x00,0x00};
uchar idata outarray[64];
uchar idata inarray[66];
……
main()
{
  uchar data slar,slaw,number;   * number 是传输的数据的个数,slar 是从器件读地址,slaw 是从器件
                                 写地址 */
  uchar data i;
  ……
  slaw = 0xa0; /* 向以 0x0080 为始地址的存储区中写入 64 字节的数据 */
  inarray[0] = 0x00;
  inarray[1] = 0x80;
  for (i=2;i<66;i++)
  inarray[i] = array[i-2];
  number = 66;
  wrnbyt(slaw,inarray,number);
  ……
  slaw = 0xa0; /* 从以 0x0080 为始地址的存储区中读出 64 字节的数据,放在 outarray 数组中 */
  inarray[0] = 0x00;
  inarray[1] = 0x80;
  number = 2;
  wrnbyt(slaw,inarray,number);
  number = 64;
  slar = 0xa1;
  wrnbyt(slar,outarray,number);
```

```
    ……
    /* 虚拟 I²C 总线软件包 VIICC1.0 的源程序代码直接放在这里就行了 */
}
```

虚拟 I^2C 总线软件包 VIICC1.0 在作者的多项科研工作中得到应用,运行效果良好,表明该软件包具有可靠的性能和广泛的适应性,具有一定的参考价值。

参考文献

1 何立民. I^2C 总线应用系统设计. 北京:北京航空航天大学出版社,1995

2 何立民. 单片机高级教程. 北京:北京航空航天大学出版社,2000

3 王晶晶,容太平. IC 卡中 I^2C BUS 设计及其在仪表中的应用. 仪表技术,1999(2):32～34

4 周 军,张瑞兰. PCF8584 I^2C 总线扩展器的应用. 电测与仪表,1998(10):36～38

选自《单片机与嵌入式系统应用》月刊,2001 年第 3 期

6.3 RS-485 总线的理论与实践

北京和利时系统工程公司开发部(100096) 虞日跃 史洪源

当前自动控制系统中常见的网络,如现场总线 CAN、Profibus、FF、INTERBUS-S 以及 ARCNet 的物理层都是基于 RS-485 的总线标准,所以从硬件开发角度很有必要对 RS-485 的总线进行总结和研究。

一、EIA RS-485 标准

在自动化领域,随着分布式控制系统的发展,迫切需要一种总线能适合远距离的数字通信。在 RS-422 标准的基础上,EIA 研究出了一种支持多节点、远距离和接收高灵敏度的 RS-485总线标准。

RS-485 标准采用平衡式发送,差分式接收的数据收发器来驱动总线,具体规格要求:

- 接收器的输入电阻 R_{IN} ≥12 kΩ;
- 驱动器能输出±7 V 的共模电压;
- 输入端的电容≤50 pF;
- 在节点数为 32 个,配置了 120 Ω 的终端电阻的情况下,驱动器至少还能输出电压 1.5 V(终端电阻的大小与所用双绞线的参数有关);
- 接收器的输入灵敏度为 200 mV(即 $V_+ - V_- \geq 0.2$ V,表示信号"0";$V_+ - V_- \leq -0.2$ V,表示信号"1")。

因为 RS-485 的远距离、多节点(32 个)以及传输线成本低的特性,使得 EIA RS-485 成为工业应用中数据传输的首选标准。

二、影响 RS-485 总线通信速度和通信可靠性的三个因素

1.在通信电缆中的信号反射

在通信过程中,有两种原因导致信号反射:阻抗不连续和阻抗不匹配。

阻抗不连续,信号在传输线末端突然遇到电缆阻抗很小甚至没有,信号在这个地方就会引起反射,如图 6.3-1 所示。这种信号反射的原理,与光从一种媒质进入另一种媒质要引起反射是相似的。消除这种反射的方法,就必须在电缆的末端跨接一个与电缆的特性阻抗同样大小的终端电阻,使电缆的阻抗连续。由于信号在电缆上的传输是双向的,因此,在通信电缆的另一端也要跨接一个同样大小的终端电阻,如图 6.3-2 所示。

从理论上分析,在传输电缆的末端只要跨接了与电缆特性阻抗相匹配的终端电阻,就再也不会出现信号反射现象。但是,在实际应用中,由于传输电缆的特性阻抗与通信波特率等应用环境有关,特性阻抗不可能与终端电阻完全相等,因此或多或少的信号反射还会存在。

引起信号反射的另一个原因是数据收发器与传输电缆之间的阻抗不匹配。这种原因引起的反射,主要表现在通信线路处在空闲方式时,整个网络数据混乱。

图 6 .3 - 1 由于阻抗不连续引起的信号反射

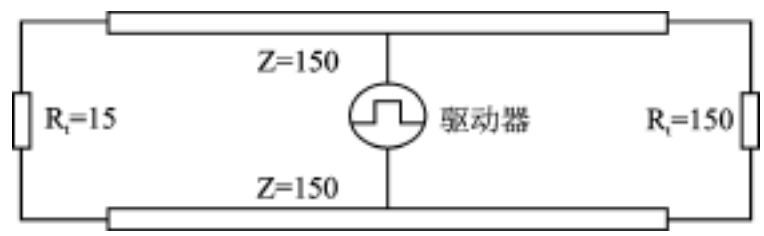


图 6 .3 - 2 终端电阻的正确连接

信号反射对数据传输的影响,归根结底是因为反射信号触发了接收器输入端的比较器,使接收器收到了错误的信号,导致 CRC 校验错误或整个数据帧错误。

在信号分析中,衡量反射信号强度的参数是 RAF(Refection Attenuation Factor 反射衰减因子)。它的计算公式为

$$RAF = 20 \lg(V_{ref} / V_{inc}) \quad (1)$$

式中 V_{ref} ——反射信号的电压大小;

V_{inc} ——在电缆与收发器或终端电阻连接点的入射信号的电压大小。

具体的测量方法如图 6 .3 - 3 所示。例如,由实验测得 2 .5 MHz 的入射信号正弦波的峰-峰值为 + 5 V,反射信号的峰-峰值为 + 0 .297 V,则该通信电缆在 2 .5 MHz 的通信速率时,它的反射衰减因子为:

$$RAF = 20 \lg(0.297 / 2.5) = -24.52 \text{ dB}$$

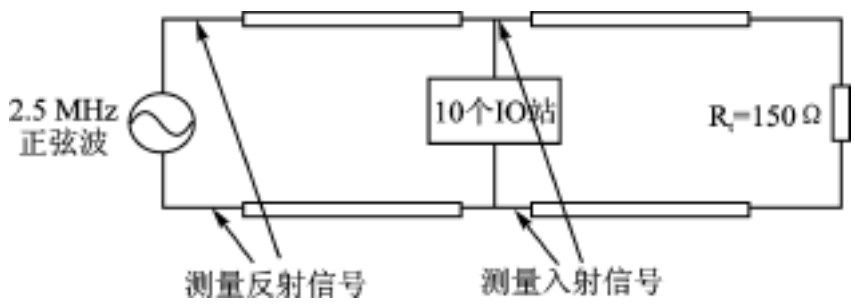


图 6 .3 - 3 测量反射信号的大小

要减弱反射信号对通信线路的影响,通常采用噪声抑制和加偏置电阻的方法。在实际应用中,对于比较小的反射信号,为简单方便,经常采用加偏置电阻的方法。在通信线路中,如何通过加偏置电阻提高通信可靠性的原理,后面将做详细介绍。

2 .在通信电缆中的信号衰减

第二个影响信号传输的因素是信号在电缆的传输过程中衰减。一条传输电缆可以把它看作由分布电容、分布电感和电阻联合组成的等效电路,如图 6 .3 - 4 所示。

电缆的分布电容 C 主要是由双绞线的两条平行导线产生。导线的电阻在这里对信号的影响很小,可以忽略不计。信号的损失主要是由于电缆的分布电容和分布电感组成的 LC 低

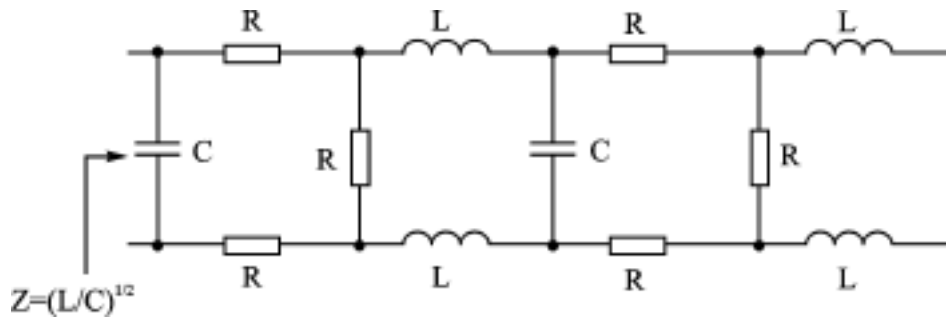


图6.3-4 传输电缆等效电路图

通滤波器。PROFIBUS用的LAN标准型二芯电缆(西门子为DP总线选用的标准电缆),在不同波特率时的衰减系数如表6.3-1所列。

表6.3-1 电缆的衰减系数

通信波特率	16 MHz	4 MHz	38.4 kHz	9.6 kHz
衰减系数(1 km)	≤42 dB	≤22 dB	≤4 dB	≤2.5 dB

3.在通信电缆中的纯阻性负载

影响通信性能的第三个因素是纯阻性负载(也叫直流负载)的大小。这里指的纯阻性负载主要是由终端电阻、偏置电阻和RS-485收发器三者构成。

在叙述EIA RS-485规范时曾提到过RS-485驱动器在带了32个节点,配置了150 Ω终端电阻的情况下,至少能输出1.5 V的差分电压。一个接收器的输入电阻为12 kΩ,整个网络的等效电路如图6.3-5所示。按这样计算,RS-485驱动器的负载能力为:

R_L = 32个输入电阻并联||2个终端电阻

= ((12 000/32)×(150/2))/((12 000/32)+(150/2))≈51.7 Ω

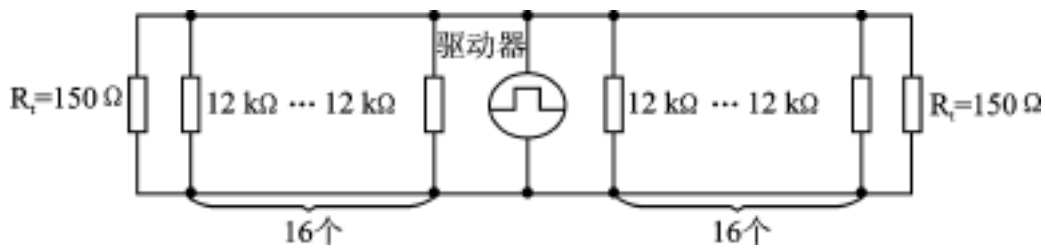


图6.3-5 带32个节点的通信网络等效电路图

现在比较常用的RS-485驱动器有MAX485、DS3695、MAX1488/1489以及和利时公司使用的SN75176A/B等,其中有的RS-485驱动器负载能力可以达到20 Ω。在不考虑其他诸多因素的情况下,按照驱动能力和负载的关系计算,一个驱动器可带节点的最大数量将远远大于32个。

在通信波特率比较高的时候,在线路上加偏置电阻是很有心要的。偏置电阻的连接方法如图6.3-6。它的作用是在线路进入空闲状态后,把总线上没有数据时(空闲方式)的电平拉离0电平,如图6.3-7。这样一来,即使线路中出现了比较小的反射信号或干扰,挂接在总线上的数据接收器也不会由于这些信号的到来而产生误动作。

通过下面的例子,可以计算出偏置电阻的大小:

终端电阻 $R_{t1} = R_{t2} = 120\ \Omega$;

假设反射信号最大的峰-峰值 $V_{ref} \leq 0.3\ V_{p-p}$,则负半周的电压 $V_{ref} \leq 0.15$ V;

终端电阻上由反射信号引起的反射电流 $I_{ref} \leq 0.15/(120||120) = 2.5$ mA。

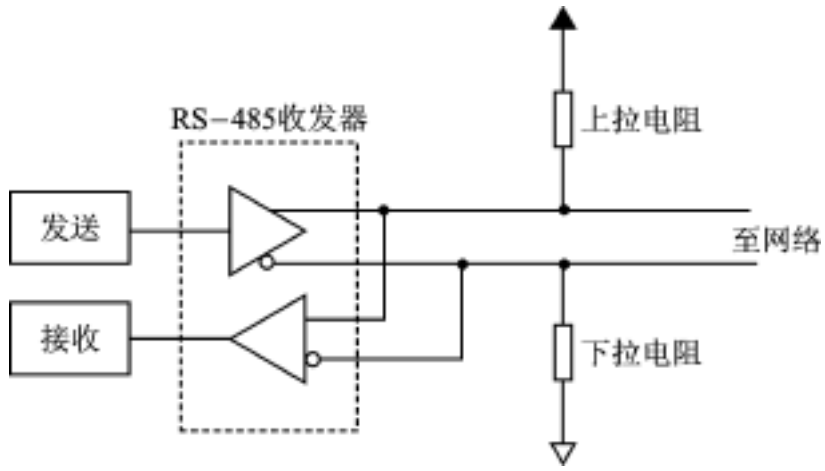


图 6.3-6 偏置电阻配置图

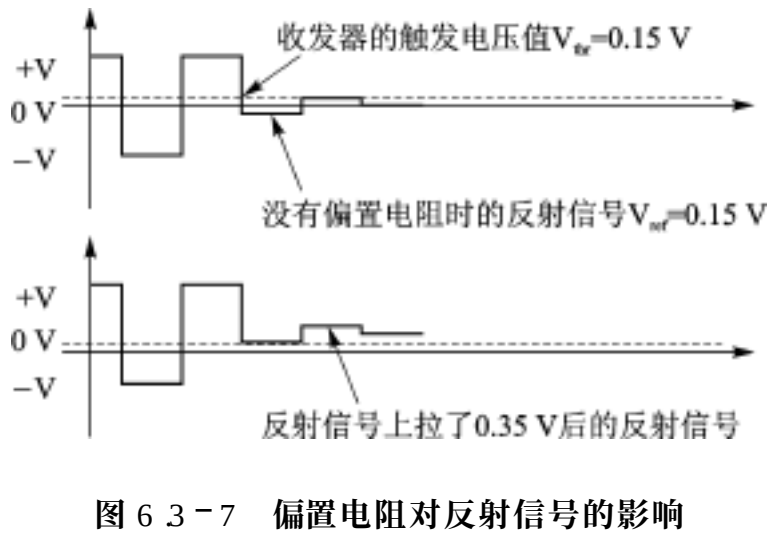


图 6.3-7 偏置电阻对反射信号的影响

一般 RS-485 收发器(包括 SN75176)的滞后电压值(hysteresis value)为 50 mV,即:

$$(I_{bias} - I_{ref}) \times (R_{t1} \,||\, R_{t2}) \geq 50 \text{ mV}$$

于是可以计算出由偏置电阻产生的偏置电流 $I_{bias} \geq 3.33$ mA

$$+5 \text{ V} = I_{bias}(R_{上拉} + R_{下拉} + (R_{t1} \,||\, R_{t2})) \tag{2}$$

通过式(2)可以计算出 $R_{上拉} = R_{下拉} = 720\ \Omega$

在实际应用中,RS-485 总线加偏置电阻有两种方法:

(1) 把偏置电阻平均分配给总线上的每一个收发器。这种方法给挂接在 RS-485 总线上的每一个收发器加了偏置电阻,给每一个收发器都加了一个偏置电压。

(2) 在一段总线上只用一对偏置电阻。这种方法对总线上存在大的反射信号或干扰信号时比较有效。值得注意的是偏置电阻的加入,增加了总线的负载。

三、RS-485 总线的负载能力和通信电缆长度之间的关系

在设计 RS-485 总线组成的网络配置(总线长度和带负载个数)时,应该考虑到三个参数:纯阻性负载、信号衰减和噪声容限。纯阻性负载、信号衰减这两个参数,在前面已经讨论过,现在要讨论的是噪声容限(Noise Margin)。RS-485 总线接收器的噪声容限至少应该大于 200 mV。前面的论述都是在假设噪声容限为 0 的情况下进行的。在实际应用中,为了提高总线的抗干扰能力,总希望系统的噪声容限比 EIA RS-485 标准中规定的好一些。从下面的公式能看出总线带负载的多少和通信电缆长度之间的关系:

$$V_{end} = 0.8(V_{driver} - V_{loss} - V_{noise} - V_{bias}) \quad (3)$$

其中：V_{end} 为总线末端的信号电压，在标准测定时规定为 0.2 V；V_{driver} 为驱动器的输出电压（与负载数有关。负载数在 5～35 个之间，V_{driver} = 2.4 V；当负载数小于 5，V_{driver} = 2.5 V；当负载数大于 35，$V_{driver} \leq 2.3$ V）；V_{loss} 为信号在总线中的传输过程中的损耗（与通信电缆的规格和长度有关），由表 6.3－1 提供的标准电缆的衰减系数，根据公式衰减系数 $\delta = 20\lg(V_{out}/V_{in})$ 可以计算出 $V_{loss} = V_{in} - V_{out} = 0.6$ V（注：通信波特率为 9.6 kbps，电缆长度为 1 km，如果波特率增加，V_{loss} 会相应增大）；V_{noise} 为噪声容限，在标准测定时规定为 0.1 V；V_{bias} 是由偏置电阻提供的偏置电压（典型值为 0.4 V）。

式(3)中乘以 0.8 是为了使通信电缆不进入满载状态。从式(3)可以看出，V_{driver} 的大小和总线上带负载数的多少成反比，V_{loss} 的大小和总线长度成反比，其他几个参数只和用的驱动器类型有关。因此，在选定了驱动器的 RS－485 总线上，在通信波特率一定的情况下，带负载数的多少，与信号能传输的最大距离是直接相关的。具体关系是：在总线允许的范围内，带负载数越多，信号能传输的距离就越小；带负载数越少，信号能传输的距离就越远。

四、分布电容对 RS－485 总线传输性能的影响

电缆的分布电容主要是由双绞线的两条平行导线产生。另外，导线和地之间也存在分布电容，虽然很小，但在分析时也不能忽视。分布电容对总线传输性能的影响，主要是因为总线上传输的是基波信号，信号的表达方式只有“1”和“0”。在特殊的字节中，例如 0x01，信号“0”使得分布电容有足够的充电时间，而信号“1”到来时，由于分布电容中的电荷来不及放电，$V_{in}+ - V_{in}-$ 还大于 200 mV，结果使接受器误认为是“0”，而最终导致 CRC 校验错误，整个数据帧传输错误。具体过程如图 6.3－8 所示。

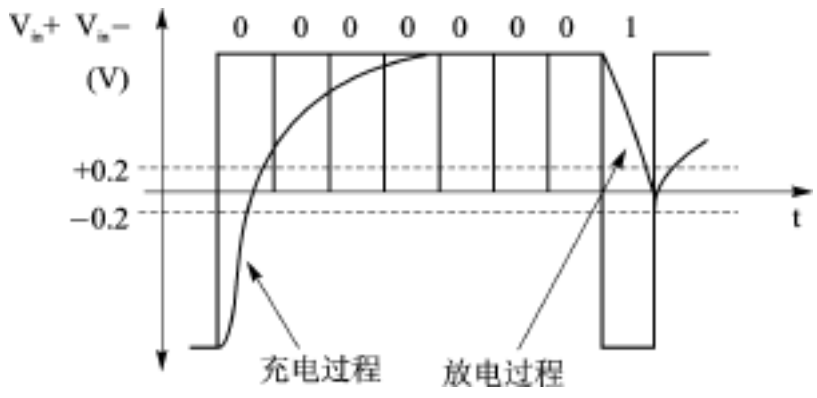


图 6.3－8 放电不及时导致数据接收错误

由于总线上分布电容的影响，导致数据传输错误，从而使整个网络性能降低。解决这个问题有两种方法：

(1) 降低数据传输的波特率；

(2) 使用分布电容小的电缆，提高传输线的质量。

参 考 文 献

1 TI 公司. SN75176 User s Guide. 1996

2 阳宪惠. 现场总线技术及其应用. 北京：清华大学出版社，1999

选自《电子技术应用》月刊，2001 年第 11 期

6.4 RS-232至RS-485/RS-422接口的智能转换器

MAXIM北京办事处北京8328信箱(100083) 刘武光

随着计算机在工业的广泛应用,控制局域网络也深入应用到各行各业之中。现行的诸多控制系统,若采用单机控制方式已越来越难以满足设备控制的要求,因为往往我们所控制的设备只是整个系统的一个基本单元,它既需要外部输入一些必要的信息,同时,也需要向外部输出自身的运行参数和状态。所有这些,都要求我们采用控制网络技术,将众多设备有机地连成一体,以保证整个系统安全可靠地运行。

目前,在我国应用的现场总线中,RS-485/RS-422使用最为普遍。当用户要将基于标准的RS-232接口设备,如PC机,连接至由RS-485/RS-422构成的通信网络时,则必须作RS-232和RS-485/RS-422之间的电平转换。传统的做法是在设备内扩展一个通信适配卡,由通信适配卡实现电平转换,内部主机再通过并行总线读出或写入数据。显然,这种设计方法存在下列缺点:

· 由于适配卡是基于某一种总线标准扩展的,而不是基于RS-232电平标准,所以其应用范围受到限制,只能一种适配卡适用一种总线(如ISA适配卡不可能插入STD总线或用户自定义的总线),其通用性较差。

· 虽然实现的仅仅是电平转换,但是由于需要考虑与扩展总线的接口和增加一个标准的UART,并且需要占用系统的其他宝贵资源,使硬件和软件变得过于复杂。

· 复杂的硬件设计大大增加了元器件的数目和电路板面,使适配卡的成本过高。

· 由于采用内置插卡方式,使变更通信方式比较麻烦,如将半双工通信方式设置为全双工方式等。另外,维修和测试也比较麻烦。

· 对于现有的基于RS-232的设备,在无法变动系统软件和硬件的情况下,显然适配卡无法将这些设备连成基于RS-485或RS-422通信网络的分布式系统。

为了克服上述缺点,同时考虑到RS-232接口的自身特点,我们设计了一种小巧的、无须外部供电的智能收发转换器,实现RS-232和RS-485/RS-422之间的电平转换。

一、功能描述及结构框图

本智能转换器作为一个独立的电平转换控制器,涉及线上取电、发送和接收状态的智能切换、通信方式设置、RS-232电平与RS-485/RS-422电平之间的转换等方面。具体描述如下:

· 从RS-232接口上取电

由于不采用外部供电方式,则必须从RS-232接口线取电,为内部元器件供电。我们知道,标准的RS-232接口定义中,TXD、RTS和DTR是RS-232电平输出。设计一个DC-DC转换器,从这些信号上,能够为系统提供一定的电源功率。

· 低功耗微处理器

微处理器通过监测TXD信号的变化,决定是否允许数据发送和数据接收。另外,有关通信方式、波特率和半/双工工作方式选择也是通过TXD信号,或I/O口来设定的。

· RS-232电平与TTL电平之间的转换

· RS-485/RS-422电平与TTL电平之间的转换

其内部电路结构示意图如图6.4-1所示。

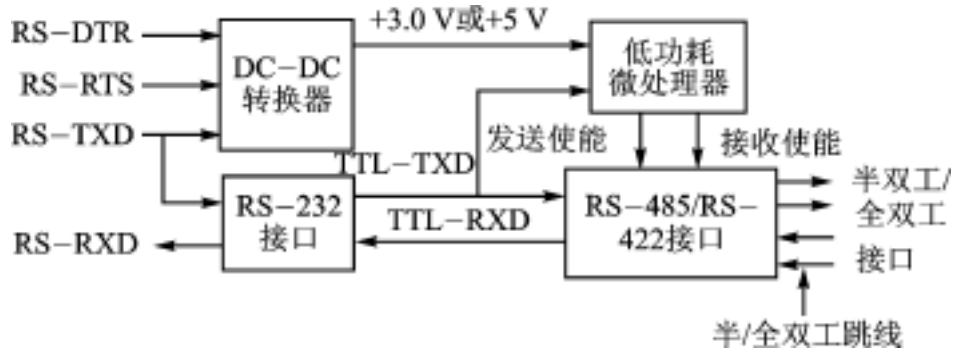


图6.4-1 智能转换器的内部电路结构示意图

二、工作原理

该智能转换器必须解决两个关键问题,即如何从RS-232线上获得电源和RS-485/RS-422接口驱动所需的功率,和如何智能控制RS-485/RS-422的收发使能。

1.电源方案

标准的RS-232定义中,有3个发送信号:TXD,RTS和DTR。每根线上的典型输出电流为±8 mA/±12 V,考虑到TXD为负电平(处于停止发关,或发送数字"1"时)的时间较多,因而电源转换决定采用负电源输入,以最大限度地增加电源输入功率,升压至所需的工作电源。从RTS和DTR上输入功率=2×8×12 mW=192 mW。另外,由于通信为间歇工作方式,所以输入电源端的储能电容和TXD(为负电平时)能够补充一定的功率。假设,我们设计一个效率为85%、输出电压为3 V的DC-DC转换器,则输出电流可达54.4 mA。

2.智能控制收发使能

RS-232通信接口采用电平方式传输,适用于点-点通信,无须专门的收发使能控制;而对于RS-485/RS-422通信接口则不同,由于采用差分电平方式传输,且允许在一条通信总线上挂接多个节点,必然要求各个节点能够独立地控制总线驱动器关断或打开,保证不会影响到其他节点的正常通信。为了简化与转换器RS-232接口端相连的软件工作,更重要的是为了提高本转换器的通用性和灵活性(即插即用,无须要求用户更改任何相关软件和硬件),本转换器内置微处理器,实现收发使能的智能控制。具体方法:微处理器在检测到UART的通信起始位后,打开发送使能,允许串行数据发送至RS-485/RS-422通信网络。微处理器根据所设定的波特率延时至UART停止位发送一半时(例如11位格式时,延时10.5T,$T=1/f_{BAUD}$),开始检测是否有下一个起始位到来。在时间T内,若有下一个起始位到来,则保持发送状态,否则将关闭发送使能,结束数据发送。

三、硬件设计

由于本转换器供电来自RS-232信号线,其输入功率受到限制,因而在本设计中将尽可能地采用+3 V供电的低功耗器件,保证总电流小于54.4 mA。主要包括4个部分:DC-DC

转换器、RS－232 接口、RS－485/ RS－422 接口和微处理器。分别介绍如下:

1 . DC－DC 转换器

显然,还没有一个 DC－DC 转换器能够直接实现 - 12 V 输入, + 3 V 输出的 IC。但是,如果我们利用现有的 IC,稍作改动,即可实现该功能。图 6 .4－2 所示的 DC－DC 转换电路,就是利用 MAX761 实现的 - 12 V 输入, + 3 V 输出、效率高于 85%的升压 DC－DC 转换器。该转换器实际输入电压范围为 - 2 .5～ - 13 .5 V,静态工作电流仅 I1 = 120 μA,具有输出电流大于 54 .4 mA 的能力(如果前端输入功率未受到限制,则输出电流可达 300 mA 以上)。由于 MAX761 采用高效率的 PFM 控制方式,而且在本电路中,开关损耗较小(因为开关电流小于负载电流),所以能够达到比 MAX761 典型应用更高的效率(MAX761 典型应用效率为 86%)。输出电压由下列方程确定:

$$V_{OUT} = V_{REF} \times R1/R2 + 0.7(V)$$

其中 V_{REF} = 1 .5 V。

选取 R2 = 100 kΩ,根据所需要的输出电压,计算 R1。

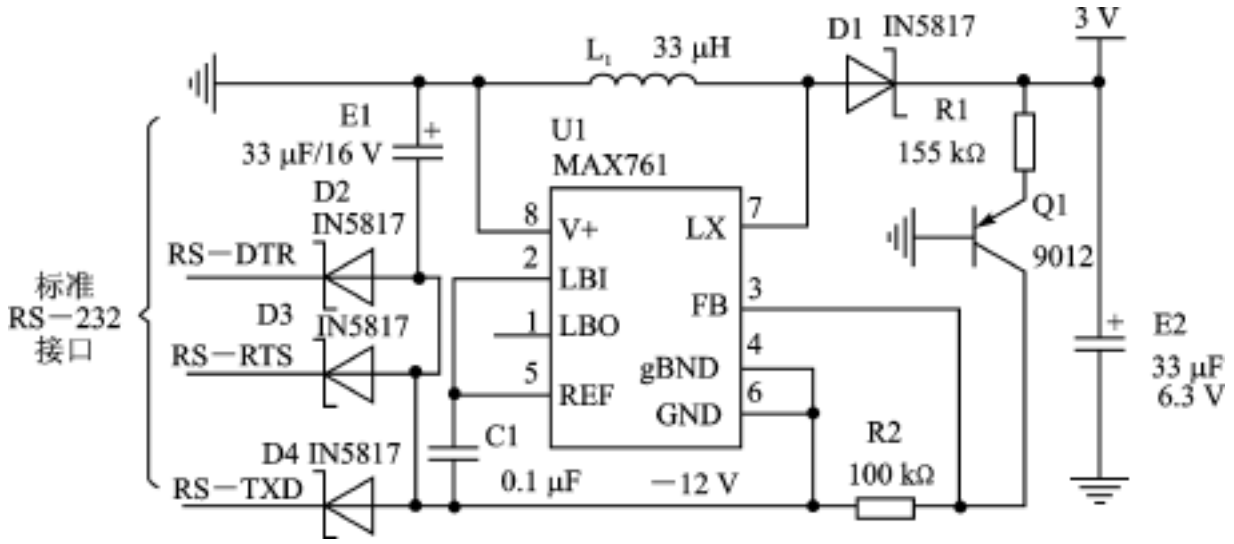


图 6 .4－2　- 12 V 至 3 V、效率高于 85%的 DC－DC 转换器

2 . RS－232 接口

本转换器只需要一片单发/ 单收 RS－232 接口就可以满足要求,但必须要求 + 3 V 单电源工作、工作电流尽可能地小的接口电路。MAX3221/ MAX3221E(带±15 k VESD 保护)刚好能够满足上述要求,具有 1TX/ 1RX,其工作电压 + 3～ + 5 .5 V,仅 1 μA 的静态电流,负载电流小于 I2 = 2 mA。

3 . RS－485/ RS－422 接口

为兼顾 RS－485/ RS－422 接口中半双工和全双工的要求,本转换器采用 MAX3491 作为 RS－485/ RS－422 接口电路,其主要指标为: + 3～ + 3 .6 V 单电源工作、工作电流 1 mA,驱动 60 Ω 负载时(半双工时,两个 120 Ω 终端匹配电阻的并联值),峰值电流可达 I3 = 3 V/ 60 Ω = 50 mA。半双工和全双工工作方式是通过跳线器来设置的,见图 6 .4－3。

4 . 微处理器

在本转换器中,微处理器所要完成的任务很简单,仅需要几根 I/ O 线即可实现参数的设置和发送使能的自动控制。实际选择中,采用 Microchip 公司的 PIC12C508A,其主要指标为:工作电流 I_4 < 1 .0 mA(工作电压 3 V,频率 4 MHz),6 条 I/ O 线,512 KB 的 ROM。其中,GP0、GP1、GP4 和 GP5 四个引脚设定对应于 16 种常用波特率(300、600、1 200 至 38 .4 Kbps

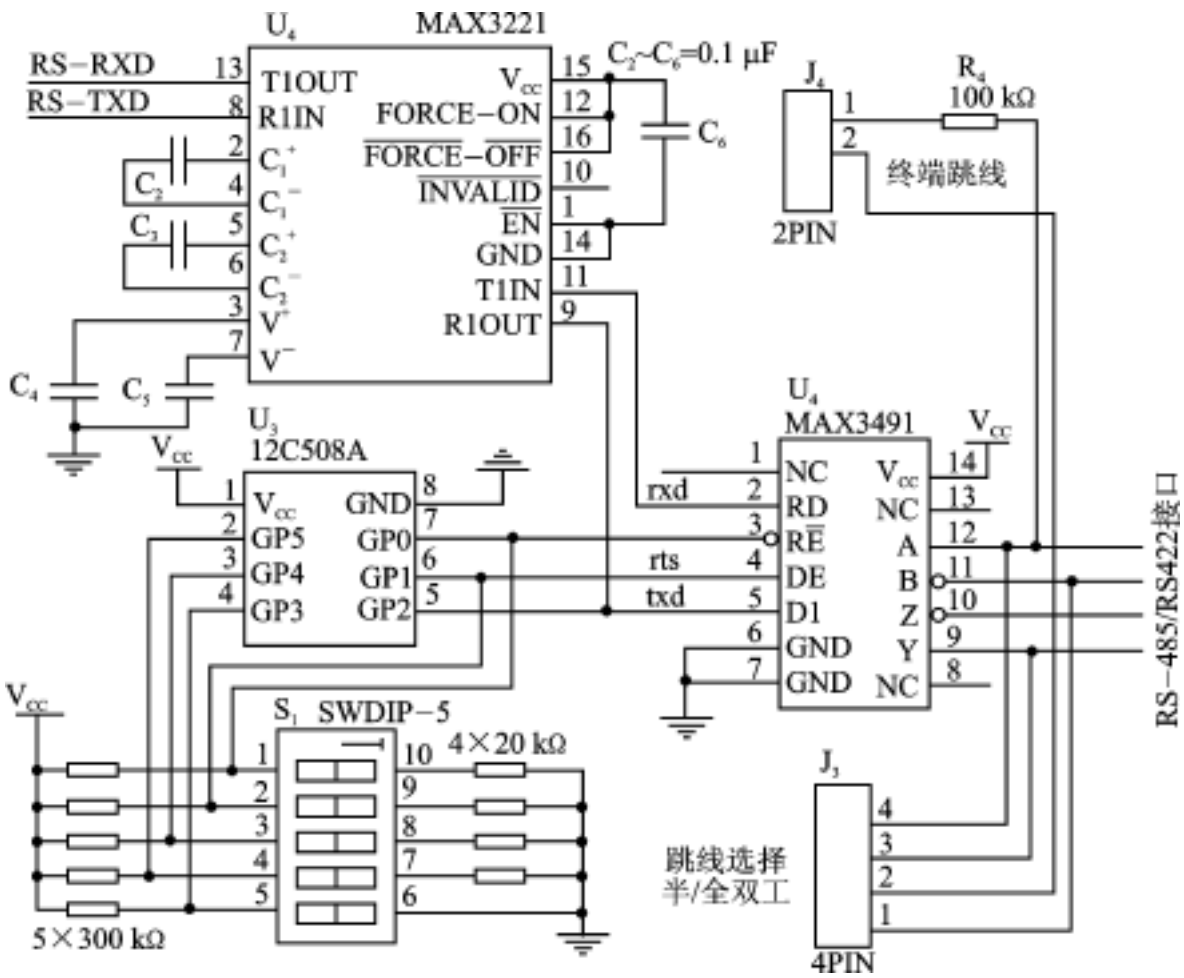


图 6.4-3 RS-232到RS-485/RS-422接口的智能转换

等8种,以及900、1 800至115.4 Kbps等8种)的延时时间;GP3对应于10位或11位串行数据格式;GP2为TXD输入,用来检测UART何时发送和停止数据;GP1为复用输出引脚,用来控制MAX3491的发送使能控制端;GP0也为复用输出引脚,用来控制MAX3491的接收使能。详见图6.4-3。

本转换器的最大电流总和 $< I_1 + I_2 + I_3 + I_4 = 0.12 + 2.0 + 50.0 + 1.0 = 53.12$ mA,小于DC-DC转换器的最小输出电流54.4 mA,因而通过RS-232信号线为本电路供电是完全可行的。实际上,由于输入电源端的储能电容E1和TXD(为负电平时)能够为电路补充一定的功率,所以设计上留有较大的电源功率裕量。

四、软件设计

本转换器的软件设计较为简单,微处理器复位后,将所有的I/O口设为输入,并读入所有的I/O状态,保存到寄存器;将GP2和GP3改设为输出状态,并输出低电平,使RS-485/RS-422接口处于禁止发送、允许接收的状态。CPU根据GPIO的初始状态,确定出用户设定的通信波特率和串行数据格式,从而预置内部的延时设定。CPU检测到UART开始通信后,打开发送使能,经内部预置延时后,开始在一个位宽时间内检测是否有下一个起始位到来,如检测到,则重新延时等待;否则,关闭发送使能,结束当前通信,重新检测UART的起始位。对于半双工通信方式,允许发送使能前应该关闭接收使能,而在发送使能关闭后才打开接收使能。对于全双工通信方式,其接收使能可以不受此信号控制,而可以直接通过跳线接地,始终允许接收。

总之,在本RS-232到RS-485/RS-422接口的智能转换器设计中,除了本身这个产品

具有较高的应用价值外,文中所涉及的 RS－232 信号线供电方案,由于其高效率、大电流输出能力,在许多基于 RS－232 接口的应用中都能够很好地满足应用;另外,这种智能控制 RS－485/ RS－422 接口的收发使能的思想,在扩展基于 RS－485/ RS－422 接口的网络分支及延伸通信距离都能够得到很好的应用。

参 考 文 献

1 MAX761 datasheet, Rev1 .0 . Maxim Integrated Products, INC

2 MAX3221 datasheet, Rev1 .0 . Maxim Integrated Products, INC

3 MAX3491 datasheet, Rev1 .0 . Maxim Integrated Products, INC

4 PIC12C508A datasheet Microchip Technology INC . 1998

选自《电子技术应用》月刊,2000 年第 5 期

6.5 实用隔离型 RS-485 通信接口的设计

西安西北工业大学(710072) 李玉忍 谢利理

计算机技术的发展,使得不论是厂矿、机关,还是工业控制及测量领域,计算机之间的信息交换、资源共享,受到人们的极大重视,已取得了很大发展。网络将各个分散的计算机连在一起,形成一个网络系统,实现了信息和资源共享。网络技术已广泛地应用于国民经济的各个部门,网络的形式和结构也千变万化。而在工业控制及测量领域较为常用的网络之一就是物理层采用 RS-485 通信接口。因为 RS-485 具有网络连接方便、抗干扰性能好、传输距离远等特点。一般情况下计算机均有 RS-232 通信接口,而 RS-232 通信接口除数据传输距离短、抗干扰能力差等缺点外,它只适合两机之间数据通信,无法将更多的计算机组成一个网络。这样,就提出了将 RS-232 转换成 RS-485 的要求。本文所介绍的将 RS-232 转换成实用隔离型 RS-485 通信接口的方法是利用美国 Maxium 公司单电源隔离型 RS-485 接口芯片 MAX1480B 设计而成的。

一、MAX1480B 简介

MAX1480B 将光电耦合器、隔离型 DC-DC 变换器和 RS-485 驱动器集成在一个芯片内,是一个完整的输入/输出在电气上隔离的 RS-485 数据接口芯片。其主要特性如下:

① 隔离的数据接口。典型隔离电压的有效值大于 1 600 V;

② 具有用于无差错数据传送的转换速率限制特性;

③ 最高传输速率达 250 Kb/s;

④ 相对于隔离地具有 -7～+12 V 的共模输入电压范围;

⑤ 单一 +5 V 电源供电;

⑥ 1 μW 低功率关断模式;

⑦ 具有电流限制和热关断特性的驱动器过载保护功能。

MAX1480B 更详细内容请查阅参考文献[1]。

二、转换电路设计

MAX1480B 其输入端为 TTL 电平或 CMOS 电平,而标准的 RS-232 通信接口都是经过 MC1488、MC1489 将 TTL 电平转换成 +12 V 标准的 RS-232 电平,所以转换电路中首先应将 +12 V 通信电平转换成 TTL 电平,再送 MAX1480B 将其转换成 RS-485 差动传输。其电路原理如图 6.5-1 所示。

图中 RS-232 电平与 TTL 电平之间的转换由 ICL232 芯片完成。该芯片是由两个 RS-232 接收器和两个发送器组成,且由单一 +5 V 电源供电。其接口电路原理如图 6.5-2 所示。

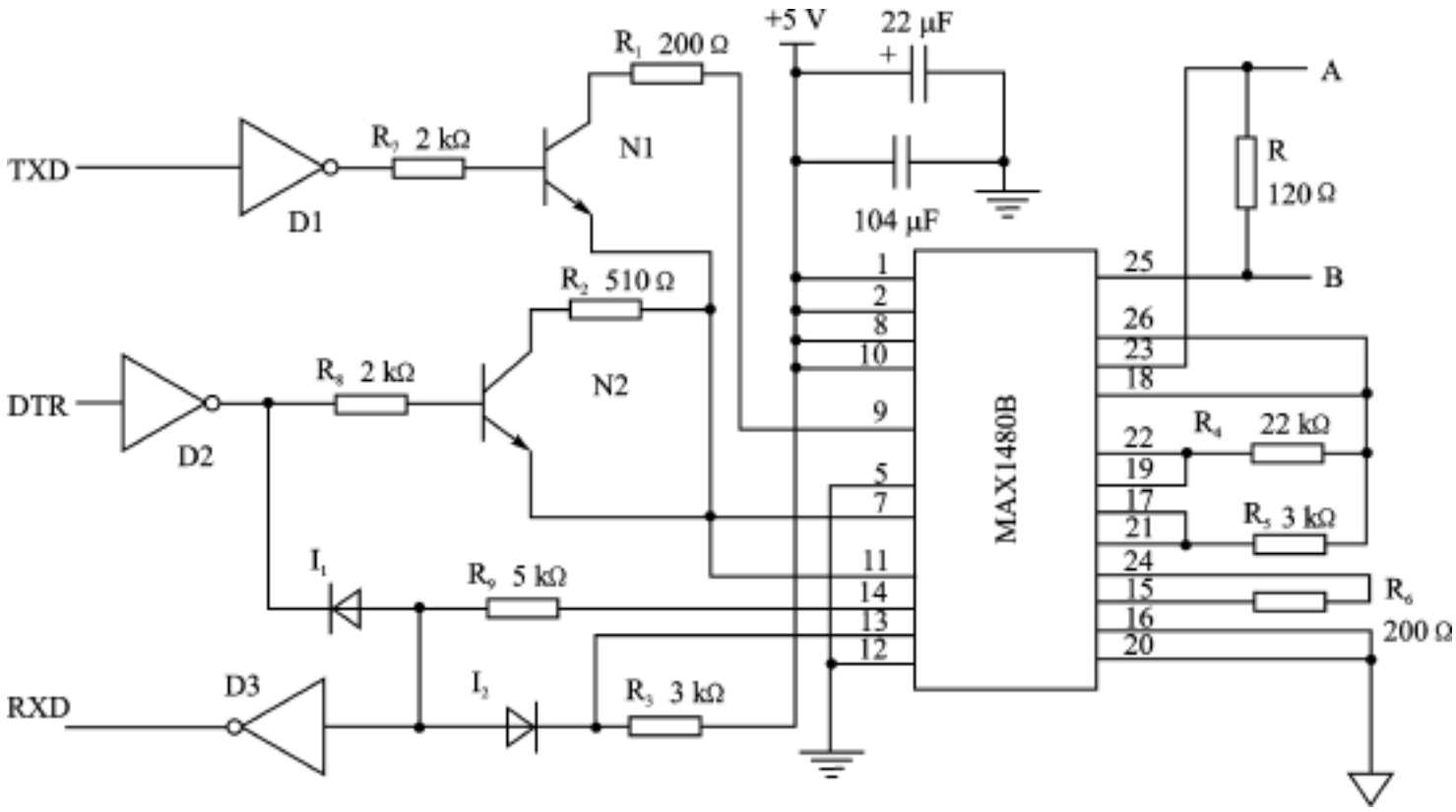


图 6.5-1 RS-232 与 RS-485 转换接口电路原理图

图 6.5-2 ICL232 接口电路原理图

数据发送由 RS-232 接收器 D1、R_7、R_1、N1 及 MAX1480B 芯片内光耦器件构成的电流环回路组成。“数据终端准备好”DTR(data terminal ready)使能线由 RS-232 接收器 D2、R_8、R_2、N2 及 MAX1480B 片内光耦器件构成的电流环回路组成。

该输出除具有 DTR 功能外,由于 ICL232 芯片只有两个 RS-232 接收器,所以用 D2 的输出和二极管 I_1 组成“请求发送”RTS(req to send)使能。当 DTR 有效允许发送时,RTS 无效,此时只能发送不能接收。而当 DTR 无效 RTS 有效时,只能接收数据而不能发送数据,从而保证 RS-485 通信线上数据的可靠性,避免因失控或干扰造成的收发数据混乱。数据接收由

RS－232 发送器 D3、R_3、I_2 及 MAX1480B 芯片内集电极开路的光耦器件组成。而 MAX1480B 隔离副边的 A 端为正相驱动器输出和正相接收器输入;B 端为反相驱动器输出和反相接收器输入。A、B 两端构成标准的 RS－485 数据通信接口。图 6.5－1 中 R 是网络匹配电阻,一般在网络两端的通信接口中使用。其余通信接口不使用,其阻值在 120 Ω 左右。

三、通信软件设计要点

图 6.5－1 所示电路是采用两线制差动传输,适合于半双工通信方式。既可应用到双机系统的点对点通信,又可应用于多机系统的网络通信。对于双机系统的点对点通信,其通信软件设计方法类似 RS－232,比较简单。但是对于多机系统的通信,在进行 RS－485 通信软件设计过程中,有几点须注意,如多机通信的地址识别问题、半双工方式通信的时延问题等都是很重要的。下面将讨论在多机系统通信中,如何解决 RS－485 通信中存在的这些主要问题。

1.从机地址识别

半双工通信时,从机地址的识别是非常关键和重要的。一般解决此问题的方法是:由主机向从机发送不同的控制命令来识别各从机,但是在这样的通信方式中,每个从机要接收主机和其他从机所有命令与数据,从而花费很多宝贵时间去接收处理不必要的信息,造成该从机的时间浪费。因此提出了一种简单而又实用的解决从机地址识别问题,那就是利用通信线路的奇偶校验位来识别从机的地址。

该方法为:主机发出从机地址时,主要允许奇偶校验位的奇校验,而所有从机只允许奇偶校验的偶校验。一开始主机发地址,所有从机处于听命令状态,从机奇偶校验和主机不一致,从机允许接收中断,收到主机发来的地址,所有从机进入中断服务例程,读取数据并和本机预定地址比较,当确认是本机地址时,设接收数据标志,允许接收数据。当主机发完从机地址时,其奇偶校验转换成偶校验,和从机奇偶校验一致,这样主机就和某一从机建立了数据通信关系。数据交换完毕,主机将奇偶校验位又变为奇校验。发另一从机地址,实现和另一从机的数据交换。从而实现了多从机地址识别。

2.半双工通信的延时

在图 6.5－1 所示的电路中,当采用半双工通信方式时,通信方向的改变是通过 DTR 来控制的,根据 RS－485 通信规则,当主机(或从机)发送数据(或命令)时,DTR 应置为高电平,这时 MAX1480B 的驱动输出端(A 和 B)处于有效状态,即发送状态。当主机(或从机)发送完数据后,DTR 置成低电平,这时 MAX1480B 的驱动输出端(A 和 B)处于接收状态。这样,通过 DTR 的控制就可以容易地实现主机和从机的数据通信。

但是在数据发送到接收的转换过程中,DTR 由高电平变成低电平时,必须经过一段延时,否则会造成数据发送不完整或接收数据受阻。这主要是因为主机或从机发送的最后一个数据要经过图 6.5－1 的 RS－232/ RS－485 转换接口电路中反相器和 MAX1480B 内部传输电路的延迟才能到达 RS－485 总线上去。如果主机或从机发送最后一个数据时,DTR 立即由有效转换成无效,该数据还未传输到 RS－485 总线上就将总线置成接收状态,从而造成最后一个数据发送受阻。因此,在数据发送到接收的转换过程中,DTR 的有效到无效转变必须经过一段时间的延迟。该延时可以通过软件编程来实现。在程序设计中,通过一段延迟程序就可容易地解决。

3．软件实现方法

主从机工作方式都采用中断方式，程序中开两个缓冲区，即数据接收缓冲区和数据发送缓冲区，它们的最大长度为 BUFsize。当要发送数据时，只要向发送缓冲区写入待发数据，然后允许发送中断，在中断服务程序中自动将缓冲区的数据发出去。接收也采用中断方式，所以只要判断接收缓冲区不空，则说明已接收到数据。所附程序以主、从机初始化串行口程序、中断通信子程序(以 COM1)为例。程序实现的是两线制半双工数据交换，数据方向控制开关由 3FC 寄存器的 D_0 控制，发送数据时，3FC 的 D_0 为 1，接收时 D_0 为 0，此位直接控制线路 DTR(见图 6.5－1)。必须特别注意的是 DTR 开关换向的时候，发送完数据不能马上将 DTR 转向接收，如果那样会切断数据发送，必须等到发送寄存器空。程序如下：

```
#include<dos.h>
#include<bios.h>
#include<conio.h>
#include<stdlib.h>
#include<mem.h>
#define BUFsize 2048
#define Address 1
typedef struct {
                short insertHere, removeHere;
                unsigned short length;
                unsigned char buf[BUFsize];
                }Buffers;
void interrupt NewInt0 CH(void);
static void interrupt (*OldInt0CH(void) = NULL);
static unsigned char OldPort21H = 0;
static unsigned int Com1Buf = 0;
static unsigned char SendFlag;
static void restoreCom(void);
Buffers bufCom1;                  /Com 口通信接收缓冲区
Buffers bufcoms1;                 //Com 口通信发送缓冲区
void initCom(void)                //COM1 初始化
{
  bufCom1.insertHere = 0;
  com1Buf = 0;
  bufCom1.removeHere = 0;
  bufCom1.length = 0;
  OldInt0CH = getvect(0x0c);
  OldPort21H = inport(0x21);
  Outportb(0x21,Oldport21H 0x18);
  setvect(0x0B,NewInt0BH);
  setvect(0x0C,NewInt0BH);
  outportb(0x3FB,0x80);
```

```
    outportb(0x3F8,0x70);
    outportb(0x3F9,0x00);
    outportb(0x3FB,0x1F);
    outportb(0x3FC,0x0b);
    outportb(0x3F9,0x04);
    inport(0x3F8);
    outportb(0x21,OldPort21H&0xE7);
}
// 下段程序功能为从 COM1 硬中断服务子程序
void interrupt NewInt0CH(void)
{
    char byte;
    while(((byte=inport(0x3fa))&0x01)==0){
      swich(byte){
        case 0x06:
          if(inport(0x3fd)&0x04)
            if(inport(0x3f8)==(Address))
              outport(0x3f9,0x03);
            else outport(0x3f9,0x04);
          break;
        case 0x4:
          bufCom1.buf[bufCom1.insertHere]=import(0x3f8);
          bufCom1.length++;
          if(++bufCom1.insertHere==Bufsize)
            bufCom1.insertHere=0;
          break;
        case 0x02:
          if(bufComs1.length==0)
            { utport(0x3fc,0x09);
              outport(0x3f9,0x01);
              break;
            };
          outport(0x3fc,0x0b);
          outport(0x3fb,SendFlag);
          byte=bufComS1.buf[bufComS1,removeHere];
          if(++bufComS1.removeHere==Bufsize)
            bufComS1.removeHere=0;
          outport(0x3f8,byte);
          --bufComS1.length;
          break;
          };
    }
enable();
```

```
        outport(0x20,0x20);
}
// 下段程序功能为恢复原 COM 中断号子程序
void restorecom(void)
{
   outporb(0x21,inportb(0x21)|0x18);
   outportb(0x3FB,0x03);
   outportb(0x3FC,0);
   setvect(0x0C,OldInt0CH);
   outportb(0x21,Oldport21H);
}
```

四、结 论

在工业控制、测量及远距通信中,网络通信的可靠性常常会受干扰及传输距离的影响。有时由于计算机与通信网络无法隔离或者因为隔离要增加电源等设备使系统使用不便,造成设备的维护性差、可靠性下降。采用以上 RS-485 转换电路,在不需外加电源的情况下,很容易实现 RS-232 到 RS-485 的转换。转换后的总线符合 RS-485 标准。除具有 RS-485 总线的特点外,还具有电路设计简单、电气上完全隔离、抗干扰能力强、可靠性高等优点。该电路已成功地应用于某广播电台发射台计算机监控系统,经长期运行实践证明,该转换电路稳定可靠。

参 考 文 献

1 胡戎 . 单片机电源隔离型 RS-485/ RS-422 数据接口芯片 . 电子技术应用,1995(5)

2 方建淳,夏东培 . ICL232 单电源双 RS-232 发送/ 接收器及其应用 . 电子技术应用,1993(7)

选自《测控技术》月刊,2000 年第 4 期

6 .6　几种 RS485 接口收发方向转换方法

河南安阳大学计算机系(455000)　赵重明

RS485 总线传送距离远、速度快、抗干扰能力强,是工业现场广泛使用的数字通信标准。虽然 RS485 总线是半双工通信标准,但它支持总线方式多站点互连,这使其成为在集散控制系统和现场总线控制系统中采用最多的通信和组网方法。采用 1 条 RS485 通信线连接多个站点时,任一时间最多只能有 1 个站点“说”,而其他站点只能处于“听”的状态;如果有 2 个站点同时“说”,数据将在线路上碰撞,需接收数据的站点不能收到正确的数据。在这种通信网络系统中,必须控制好每一个站点的听说状态即 RS485 接口的收发状态转换,以保证数据能及时、正确地传输。

图 6 .6 - 1 是一种最常用的 RS485 接口收发方向转换法。这种方法有 2 个缺点:

(1) 占用 1 位 I/ O 口来控制 RS485 发送器是否有效。当 I/ O 口置“0”时 RS485 发送器被关闭, TXD 不能往总线上发送数据; I/ O 口置“1”时 TXD 可将数据发送到总线上。

(2) 由于在 RS485 总线通信系统中,挂在总线上的站点可能有很多个,使用图 6 .6 - 1 所示的收发电路时,如果系统中的某一站点死机或出了问题,可能使其单片机的 P1 .0 口恒置为“1”。产生这种情况时,其发送器将长期占用总线,这将导致整个通信系统的崩溃。

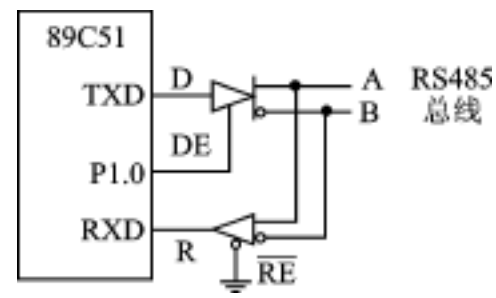


图 6 .6 - 1　最常用的 RS485 接口收发方向转换法

笔者在实际工作中曾使用过其他几种改进的 RS485 接口收发方向转换方法,在此介绍给大家。

一、用 RC 电路自动转换 RS485 接口收发方向

图 6 .6 - 2 是在图 6 .6 - 1 的基础上改进的电路。图中使用了 1 个电容来传递打开发送器的控制信号,接在 P1 .0 的上接电阻是为了增大对电容的高电平驱动能力,该电阻最小可取 250 Ω;接在发送器控制端上的下拉电阻一般可取 4 .7 kΩ,它与电容形成 RC 回路。这种电路在接口不发送时, P1 .0 应置为“0”。发送数据之前,先将 P1 .0 口置为“1”,电容的另一端马上也变成高电平,发送器打开,数据被发送出去,过一段时间后,由于下拉电阻的作用发送控制端将变为低电平,接口由发送态自动转变为接收态。从发送器打开到发送控制端的电平降到门限

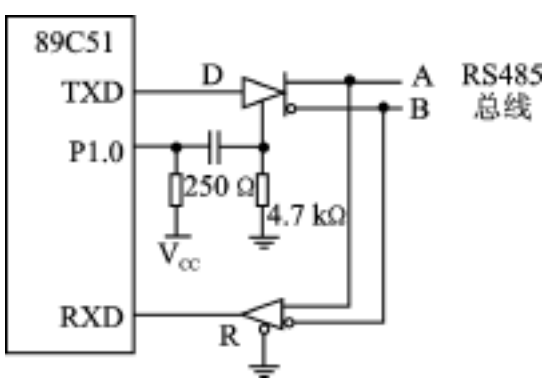


图 6 .6 - 2　用 RC 电路自动转换 RS485 接口收发方向

以下的这段时间可以用 RC 电路的时间常数来估算。一般地,可以让单片机发送一包数据之后再将接口转变为接收态,再根据发送一包数据所需要的时间来选取电容的容量。由于普通电容的容量易受温度、湿度和时间等因素的影响,电路中的电容应采用稳定性较好的钽电容,不过,这种电路自动转换方向的时间仍不易精确控制。

这种电路的优点是:RS485 总线通信系统中的 1 个站点死机后,其 RS485 接口自动变为接收态,不影响其他站点的通信。

二、总线无高阻状态的 RS485 接口

图 6.6-3 是一种巧妙的 RS485 总线收发接口。它没有控制收发方向,靠 2 个 4.7 kΩ 的电阻将平时 RS485 总线上本应呈现的高阻态设置为"1"态,其数据发送端 TXD 不是接在发送器的数据输入端,而是反相后接到发送器的控制端,而发送器的数据输入端却接为固定低电平。这种电路,当发送"1"时,发送器不打开,"1"是靠图中的 1 个上拉电阻和 1 个下拉电阻来实现的;当发送"0"时,发送器打开,接在 TXD 上的低电平"0"被发出去。

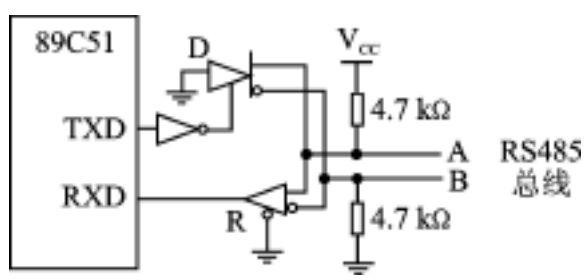


图 6.6-3　总线无高阻状态的 RS485 接口

三、用精确单稳态定时器控制 RS485 接口收发方向

图 6.6-4 是一种每发送 1 个字符自动转换 1 次收发方向的接口。它使用了一种电路来感测每 1 帧数据起始位下降沿的到来,并及时将发送器的状态由平时的关闭自动变为打开,等到 1 帧数据发完后,发送器又自动关闭。图 6.6-4 所示的电路是用 1 个精密单稳态定时器和 1 个"或门"构成的。图中 E 是单稳态定时器的低电平有效的触发端,F 是单稳态定时器的输出端,"或门"的作用是:当 1 个帧数据起始位下降沿到来并触发单稳态定时器使其进入暂态,进而在 F 输出高电平并打开发送器的同时封锁单稳态定时器触发端,使其在 1 帧数据的发送过程中不再重新触发。图 6.6-5 是单片机串行口数据发送端 TXD 发送 1 帧数据时,TXD 上的信号与单稳态定时器输出的时序关系。需要注意的是:单稳态定时器的定时时间常数不能大于 1 帧数据的发送时间,只能略小于 1 帧数据的发送时间,只要不足以引起接收方的帧格式错即可。

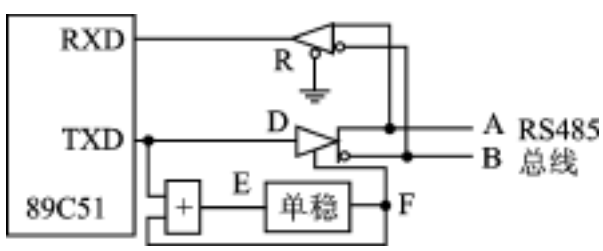


图 6.6-4　发送 1 个字符自动转换 1 次收发方向的接口

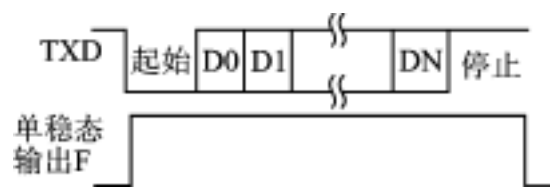


图 6.6-5　单片机串行口发送端 TXD 上的信号与单稳态定时器输出时序关系

四、结束语

本文介绍的3种电路中,第1种接口方向转换占用处理器1位I/O口,而且增加通信程序的软件开销;后2种真正实现了收发方向的自动转换,这使得原来在其他机器上使用的全双工通信程序几乎可不加改动地移植到使用半双工RS485接口的系统中。第一种方法在站点死机或不正常后自动关闭本地的RS485接口,使其不影响其他站点的通信,后2种方法稍加改进也可以做到这一点。实际应用的接口还需要加入光电隔离电路和保护电路。3种接口方式略加变动还可以用于将PC机上的RS232串口变换成RS485串口。对于第一种方式一般需占用RS232串口的1条调制解调器控制线(如$\overline{\text{DTR}}$),而后2种方式则不需要。

参 考 文 献

1 邬宽明.CAN总线原理和应用系统设计.北京:北京航空航天大学出版社,1996

2 邱公伟.实时控制与智能仪表多微机系统的通信技术.北京:清华大学出版社,1996

3 阳宪惠.现场总线技术及其应用.北京:清华大学出版社,1999

选自《微型机与应用》月刊,2000年第10期

6.7 LonWorks 总线技术及发展

南京解放军理工大学工程兵学院(210007) 王双庆 邢建春 王 平

一、前 言

到目前为止,工业控制自动化技术经历了两次革命。一次是 20 世纪五六十年代从气动传动控制到模拟信号为主的自动化仪表系统;第二次是 20 世纪七八十年代集散控制系统(Distributed Control System,DCS)的出现。DCS 对工业控制技术的发展功不可没。它将计算机应用至控制系统,用分散 I/O 模板替代控制室仪表,将分散的测量点通过计算机进行集中监控。然而,DCS 系统本身有着克服不了的问题:一对一接线,不仅增加安装维护费用,也降低了系统可靠性;采用"操作站——控制站——现场仪表"三层主从结构模式,仍存在集中的可靠性隐患;各个 DCS 系统采用封闭的协议不利于系统的互操作;模拟信号传递,限制了传输媒体和拓扑结构的选择。现在,工业控制领域正进行着第三次革命,即基于现场总线的控制系统(FCS)的出现。

现场总线系统采用数字信号传递信息,可以实现一对电线上传递多个信号,同时可为多个设备提供电源,现场设备之外不再需要 A/D、D/A 转换器,简化系统结构,节约硬件设备和连接电缆。采用公开一致的通信标准,各个厂家设备可以交互信息,实现互操作,是真正意义的开放系统。它将测量传感、补偿计算、数据处理和控制实现分散至现场设备,实现系统的彻底分散。这种 FCS 控制系统代表了工业自动化的发展方向。

由于利益驱使,各个厂商推出自己的总线标准,并不遗余力地扩大其应用市场。世界范围内形成了许多较有影响的总线标准:HART、CAN、LonWorks、FF、PROFIBUS、WorldFIP、ControlNET、DeviceNet 等等。其中 LonWorks 总线以其支持 OSI/RM 七层模型、支持多种通信媒介及易于实现等诸多卓越特点,受到了广泛的关注。

二、LonWorks 的技术优势及应用

LonWorks 总线技术由美国 Echelon 公司推出并由 Motorola 和 Toshiba 公司共同倡导,于 20 世纪 90 年代初正式公布。其通信协议 LONTALK 支持 OSI 的所有七层模型,这是 LON 总线最杰出的特点。LONTALK 协议通过神经元芯片(Neuron Chip)上的硬件和固件(firmware)实现,提供介质存取、事务确认和点对点(peer to peer)通信服务;还有一些先进服务如接收认证,优先级传输,单一/广播/组播消息发送等。网络拓扑结构可以是总线型、星型、环型和混合型,可实现真正的自由组合。它采用面向对象的设计方法,通过网络变量把网络通信的设计简化为参数设置,通信速率从 300 bps 到 1.5 Mbps,直接通信距离可达 2 700 m(78 Kbps,双绞线)。通信介质支持双绞线、同轴电缆、光纤、射频、红外线及电力线等,并开发了相应的本质安全防爆产品。编址方法提供了巨大的网络寻址能力(系统支持 32385 个网络设备)。

高可靠性、安全性、易于实现和互操作性,使得 LonWorks 产品应用非常广泛:灌溉管理、电路板诊断、分散和过程控制、电梯控制、能源管理、环境监视、污水处理、火灾报警、采暖通风和空调控制、轨道机器控制、交通灯管理、家庭网络自动化等等。除了 Motorola 和 Toshiba 公司外,Cypress 半导体公司也在今年开始了 Neuron Chip 的生产。目前全世界有 4 000 多个 Lonworks 开发商提供收发器、网络管理工具、接口、张端用户产品和系统,这个数字还在不断扩大。超过 4 000 个公司正在使用 LonWorks 系统,其中包括 ABB、British Telecom、Honeywell、AT&T、Olivetti 和 NASA 等。世界范围内有几千个 LON 方案的几百万节点设备正在运行。

为便于各个厂商不同设备的互操作,Echelon 公司和一些 LonWorks 用户在 1994 年 5 月成立了 LonMark 互操作协会。到目前已经超过了来自 30 多个国家的 230 多个成员,其中包括 ABB、Ahlstrom、Ameritech、Echelon、Honeywell、HP、IBM、Leviton、Microsoft、Philips、Olivetti、Toshiba 和 Yokogawa 这些国际知名大公司。协会的主要任务是制定并推广 LonWorks 技术的设计指导标准,并帮助生产厂商和最终用户制造和使用可互操作的 LonWorks 产品。

LonWorks 网络协议已经成为诸多组织、行业的标准。消费电子制造商协会(CEMA)将 LonWorks 协议作为家庭网络自动化的标准(EIA—709)。1999 年 10 月,ANSI 接纳 LonWorks 网络的基础协议作为一个开放工业标准,包含在了 ANSI/ EIA 709 .1 中。LonWorks 协议也被美国供暖、空调和制冷工程师学会(ASHRAE)接受,成为建筑物 BACnet 控制标准的一部分,即现在的 ANSI/ ASHRAE－135。同时 LonWorks 也是国际 Forecourt 联盟(欧洲所有加油站)标准。美国铁路组织也选择了 LonWorks 作为其气动刹车系统标准。1999 年 8 月,LonWorks 协议成为 IEEE 火车通信方面新标准(IEEE1473－1999)的一部分,这巩固了 LonWorks 在铁路应用方面的地位。国际半导体仪器原料协会(SEMI)明确采纳 LonWorks 网络技术作为其行业标准(SEMI E61－0697)。

三、神经元芯片(Neuron Chip)

Neuron Chip 是 LonWorks 系统设备的核心器件,是由 Echelon 公司研制的集通信、控制、调度和 I/ O 支持为一体的 VLSI 器件。包括 3 个 8 位 CPU,时钟频率从 625 kHz 到 20 MHz;三类存储器(RAM,ROM 和 EEPROM),两个 16 位定时器/ 计数器,15 个软件定时器。3 个 8 位 CPU,两个用于网络通信(其中一个完成 OSI 模型中 1、2 层功能,叫做 MAC 处理器;另一个完成 3 到 6 层的功能,称为网络处理器),一个用于应用,这使得复杂的应用不会影响网络的反应能力;3 个 CPU 通过共享内存方式工作(见图 6 .7－1)。把通信和应用集成在一个芯片上,节省了设计和生产费用,同时提高了系统的可靠性。LonWorks 7 层网络通信协议集成在神经元芯片上,使它成了一个 LonWorks 协议的硬件运行环境。协议由硬件和固件来实现,保证足够的处理能力。允许定制芯片的制造,允许便于节点设计的额外功能加入芯片中。

神经元芯片的设计目的是大范围应用于工业和建筑业,而且由两个最大的半导体生产商(Toshiba 和 Motorola)生产,这保证了芯片的低价位,同时也证明了使用具有 8 位处理能力的 LonWorks 处理器是既节省又能收到最好的使用效果。当 8 位处理能力满足不了需要时,LonWorks 通信协议可以集成到其他的微处理器中。10M 的最大时钟频率一直是神经元芯片

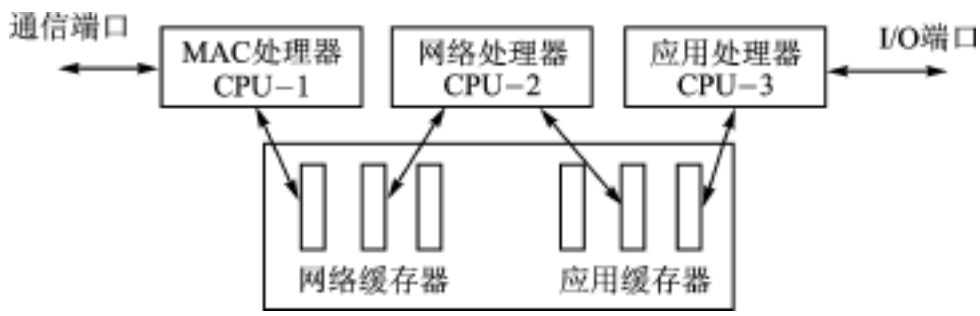


图 6.7-1 神经元芯片处理器结构及存储器分配

向高速应用迈进的障碍;1997 年 9 月,20M 芯片(Motorola 的 MC143120FE2DW 和 ToshibaTMP3120-FE1M)的出现使网络的反应时间降至 3～4 ms,使 I/O 性能提高了一倍。新出现的低能量芯片 MC143120LE2FB 采用了 3 V 的运行电压,减小了电力消耗。适合于传感器、执行器、原料消耗跟踪器、安全检测器和自动调温器,尤其是设计建筑物内嵌的分散控制网络应用。

四、通信协议 LonTalk

LonTalk 协议是 LonWorks 系统的灵魂,它固化于神经元芯片中,是直接面向对象的网络协议。支持 OSI/RM 模型的 7 层协议(见表 6.7-1),支持多种传输介质和多种传输速度。地址设置方法提供了巨大的寻址能力,提供了可靠的通信服务,保证了数据的可靠传输。

表 6.7-1 LonTalk 的 OSI 七层协议

OSI 层	目 的	提供的服务	所用 CPU
1.物理层	电气连结	媒介接口,信号调制方案	MAC CPU
2.链路层	介质访问和组帧	组帧和解帧;数据编码,CRC 容错检查,可预测 P—坚持 CSMA 冲突检测,冲突避免,优先级	MAC CPU
3.网络层	寻址	点对点寻址,广播寻址,组播寻址,信息包路由	网络 CPU
4.传输层	端对端可靠通信	包序列化,重复检查,端对端确认服务,服务类型	网络 CPU
5.会晤层	控制	请求/响应,鉴别服务	网络 CPU
6.表示层	数据解释	网络变量解释,应用消息,外来帧传送	网络 CPU
7.应用层	应用兼容性	LonMark 对象,标准网络变量类型,配置属性,文件转换	应用 CPU

1.介质访问控制 MAC 子层协议

LonWorks 网络的 MAC 子层协议为可预测 P—坚持载波监听多路访问冲突检测(Predictive P-Persistence CSMA/CD)协议。CSMA 协议要求一个节点在发送报文之前进行网络监听,当信道空闲时发送数据,否则延长一个时隙再监听发送。LON 采用可变的 P 来加入时隙。实验证明,在网络通信量较小和较大的情况下,MAC 协议保证了网络运行的高效可靠。

2.网络地址分配

地址用来标识通信中的一个或一组对象。LonTalk 协议采用一种分层的编址方法。它使用域(domain)、子网(subnet)、节点和组的概念,可以方便地实现整个域、某个子网、某个节点、某个组的通信。一个域中节点最多可达 255×127=32 385 个,最多有 256 个组,一个组最多 64 个节点,一个节点最多可从属于 15 个组。这样的编址方法有利于系统的各种通信。

3. 网络消息服务

LON 网提供了 4 种消息服务类型:应答服务、请求/ 响应、重发服务和非应答服务。这样可以满足网络的高效率、快响应、安全性和高可靠性。LonTalk 协议支持冲突检测和消息重发。这样保证了节点的反应时间,提高了媒介的利用率。LonTalk 提供优先级机制,提高了重要消息包的响应时间。LonTalk 协议支持消息的鉴别服务,允许消息的接收方确认发送方的发送权限,可防止节点和应用的侵权访问。

4. 网络接口和通信的实现

LonTalk 中有个网络接口协议,可以实现 LonWorks 在其他处理器上的应用。神经元芯片作为通信处理器,负责 LonTalk 协议的 1 到 5 层,而主处理器实现第 6、7 层。网络接口的实现可以是交钥匙(turn—key)设备,如 Echelon 的串行 LonTalk 适配器(SLTA),或者是基于微处理器接口程序(MIP)的定做设备。节点通信连接一般是通过建立节点间网络变量连接或显式消息的消息标签连接来实现。

五、NEURON C 编程语言

NEURON C 是神经元芯片的编程语言,便于开发人员高效开发分布式 LonWorks。它由 ANSI C 中派生而来,对 ANSI C 进行了补充和删减。不再支持 C 的浮点运算,但提供了浮点库以使用浮点数。它不再需要 MAIN 函数,而采用 WHEN 语句,引入了任务调度概念;使用优先级 WHEN 语句,可以方便实现紧急事务的处理。附加了许多 ANSI 中没有的保留字和语法。毫秒级和秒级定时器,可以干预任务调度,激活用户定制任务。

建立和绑定网络变量,可以方便实现网络消息传递,不需要用户编程进行管理。显式消息可以实现变长度数据传递,从 1 到 228 个数据字节,可以不受应用场合的限制。

六、互操作性的实现

LonWorks 系统的互操作性是它的一个杰出之处。项目集成商可选择最合适的系统;生产商可拓宽产品市场;用户可自由选择网络产品;工厂的管理人员可使用标准工具监视设备运行。

LonWorks 系统的互操作标准由 LonMark 互操作协会负责制定。通过 LonMark 认证的 LonWorks 设备可冠以 LonMark 标志。

LonMark 互操作协会的工作集中在两个方面:一是标准收发器及相关的物理通道规范,另一个是设备应用程序的建立及文档化的标准化定义。

LonMark 行规详细描述了应用层接口,包括网络变量、配置属性、一般和特殊的控制所用的缺省和上电行为。

每个 LonWorks 设备有一个惟一的设备应用标识符,即程序 ID。LonMark 设备的程序 ID 称为标准程序 ID。

为提供多语言支持,节点必须使用数字的索引量而不是明确的报文串来标识命名数据的每一部分。而安装维护工具必须把索引值映射成合适的字符串。这些通过资源文件实现。资源文件包括 4 个种类:类型文件、行规模板、格式文件和语言文件。

LonWorks 网络的互操作性还体现在 LonWorks 网络操作系统 LNS 上。

七、LonWorks 网络

LonWorks 系统包括了单一的网络操作系统 LNS,它是 Echelon 公司吸收 C/S 模型、OLE 对象和基于部件(component)的软件开发等先进思想后推出的 NOS。它保证了网络工具和应用之间的互操作性,提供了一个支持 LonWorks 网络互操作应用的标准平台。它是设计、组态、安装和维护 LonWorks 系统的互操作工具及应用的基础。LNS 使用 C/S 结构,保证多个应用在网上同时激活,实现多用户同时安装设备、操作系统、诊断问题和维护修理。LNS 的内建标准鼓励 LonWorks 设备制造商通过连接单一产品的软组件,为用户提供多个选择。网络集成商并不需要为一个领域的每个工程开发定制应用,相反他采用内插件。设备的内插件包括内建的问题诊断和排除工具,辅助并确认配置选项的用户对话,还有为设备数据监视和绘图的定制用户接口。使用 LNS,制造商的设备内插件可以不做修改就运行在各种 PC 上。LNS 内插件屏蔽了软组件和设备间的通信,简化了网络的管理。许多现有的设备可以通过简单地写一个插件实现互操作,降低了用户的使用成本。

LNS 在 Windows NT 或 Windows 95 环境下的应用程序接口由 LonWorks 组件架构(LCA)定义。LCA 使用多个协作软件组件实现 LonWorks 网络工具,提供一个标准的网络工具内核作为开放标准,这样网络工具可由多个厂商的多个软组件建立。LCA 定义了标准的 Windows OLE 服务接口用以触发网络服务,标准的应用接口用以触发 LCA 软组件。

网络工具是建立在 NOS 上的软件。现有的产品有:网络集成工具、网络诊断工具、HMI 开发工具、I/O 服务器等。基于 LNS 的网络工具可以实现互操作。Echelon 公司的网络工具包括:LanMaker for Windows 集成工具、LonManager 协议分析工具和 LNS DDE 服务器。

值得一提的是 Gesytec 公司的 Easylon OPC 服务器。OPC(OLE for Process Control)是现代工业控制软件的标准,拥有 OPC 服务器的硬件设备可以实现互操作,使用 OPC 接口,不同总线域的数据(如 LonWorks 和 Profibus),可以集成在单一应用中。Easylon OPC 服务器可以跟踪 LonWorks 网上的网络变量,因此 HMI 和 SCADA 可以通过 OPC 服务器来监控 LonWorks 网络。可以用 OLE 兼容的应用程序如 Excel, Visual Basic, Access 等实现 LonWorks 网络的过程可视化和监视控制。

LonWorks 系统(见图 6.7-2)中,不同域间的设备不能相互通信,但可通过上层的应用程序来弥补。上层应用通过 LNS 和各自的域接口分别与各个域通信,并在应用中实现两个域的相互通信。

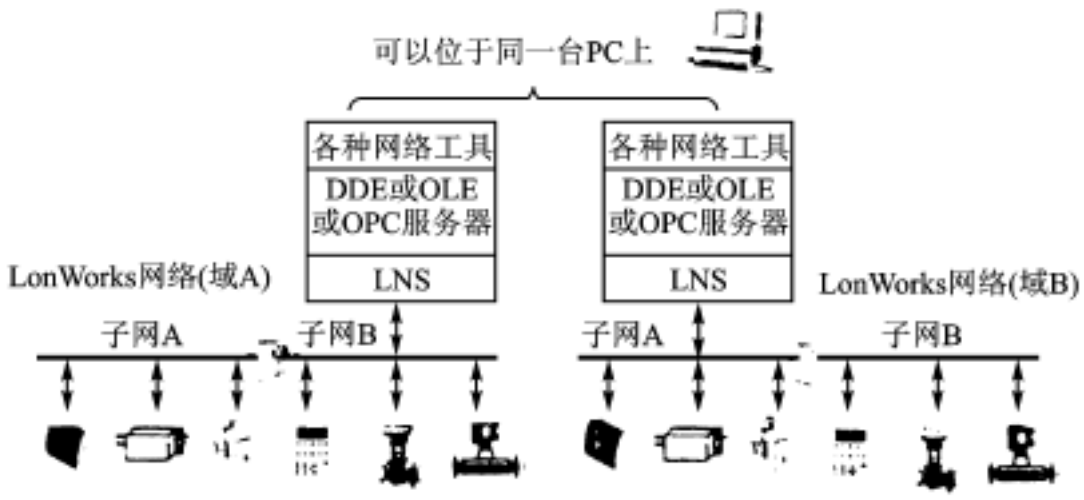


图 6.7-2 LonWorks 网络系统

八、LonWorks 的安全性和可靠性

LonWorks 网络的可靠性和安全性体现在多个方面。

LonTalk 协议提供了两个重要的可靠性技术。真正的端对端(end-to-end)确认服务保证了数据的可靠传输,数据完整性由循环冗余编码(CRC)校验保障。另外,为一些在恶劣环境使用的收发器(例如,低带宽、高噪声和高衰减)额外地融合了前向纠错方法,可以检测并纠错而无需消息重发。

LonWorks 网络可以实现网络设备和功能的彻底分散。网络采用对等结构,任何部分出现问题,不会影响其他设备。LonWorks 网络的节点地址分配中,子网不能跨越智能路由器,不同子网的错误可通过智能路由器物理分开。LonWorks 网络易于实现设备和通信的双重冗余,也提高了可靠性。通信设计面向对象,通过节点网络变量的设置和绑定,即可实现网络通信。

LonWorks 网络中使用的网络操作,包括网络管理操作,都可使用 OSI 模型中第 4 层的服务项目——发送者鉴别服务,这可以保证发送者的正确性。

九、LonWorks 的发展及中国市场

LonWorks 网络性能卓越,但其自身仍在不断发展。今年 3 月,Echelon 公司开始出售 i.LON[™] 1000 Internet 服务器。这是一个突破性产品,可使应用在家庭、建筑、工厂、运输系统中的百万计的 LonWorks 设备,变成 WEB 的一部分。它拓宽了工业领域中的基于 Internet 的市场和应用。1999 年 6 月,Sun 公司展示了 LonWorks 设备在 Jini 网络作为共享服务的一幕。Jini 的 LonWorks 代理基于 LNS 网络操作系统,不需要另外的硬软件和设备驱动。Echelon 公司的总裁兼首席执行官 Ken Oshman 先生这样评价 Jini 技术与 LonWorks 网络的结合:“通过结合,LonWorks 成为 Jini 服务提供者;我们可以想象,任何一个设备都能在任何时间任何地点参与共享其他设备的服务。”

LonWorks 系统的卓越性能也早已引起中国市场的注意。北京航天金穗高技术有限公司和航空工业总公司 618 研究所已经成为 Lonmark 互操作协会会员。中国计算机协会工控专委会成立了 LonWorks 控制网络协作网,成员已达 40 多个。LonWorks 技术在我国的推广力度还是较大的。

国际现场总线技术的发展虽才十几年的时间,但国外的工控基础比较好,而且各种总线技术的标准都是由他们提出,我国同他们的差距还是很大。在短时间内,我们的任务只能是消化和应用各种总线标准,使我国的工业自动化能跟得上国际先进水平。

参考文献

1 chelon Corp . LonWorks Engineering Bulletin [J] . 1995

2 Echelon Corp . Echelon s LonWorks products [W] . 1999, version A

3 Echelon Corp . Introduce to LonWorks System [W] . 1999, version 1 .0

4 Echelon Corp . LonMark Application Layer Interoperability Guideline [W], version 3 .1

5 杨育红 . LON 网络控制技术及应用[M] . 西安:西安电子科技大学出版社,1999

6 王俊杰,等 . LonWorks 技术及其应用讲座:第一讲现场总线的发展与 LonWorks 技术[J] . 自动化仪表,1999,20(7):40～43

7 胡艳军,等 . LonWorks 现场总线技术及应用[J] . 自动化与仪器仪表,1999,5(85):6～10

8 路建萍,等 . 分散式智能化控制网络系统及应用[J] . 自动化仪表,1999,20(12):32～33

9 张云贵 . LonWorks 现场总线控制系统的软硬件全面解决方案[J] . 测控技术,1999,18(5):18～20

10 张金雄,等 . LonWorks 在生产监控系统的应用[J] . 工业控制计算机,2000,2

11 秦立军 . LonWorks 现场总线与以太网的互联研究[J] . 工业控制计算机,1999 .5

12 Pradip Madan . Overview of Control Networking Technology [R] . Echelon Corp

13 Pradip Madan . Device Bus ? Field Bus ? Or Sensor Bus ? Is this Segmentation Obsolete [R] . Echelon Corp

14 吕丽萍,等 . 开创自控技术新纪元的现场总线[J] . 计算机自动测量与控制,1999,7(3):1～4

15 http:// www .echelon .com .[DB/ OL]

16 http:// www .lonmark .org .[DB/ OL]

17 http:// www .gesytec .de .[DB/ OL]

选自《计算机自动测量与控制》双月刊,2000 年第 5 期

6.8 LonWorks 网络监控的简单实现

上海同济大学电气工程系机器人研究室(200092)
赵振功 胡 东 陈辉堂

近年来,现场总线技术迅猛发展,取代传统的集中式控制系统已成必然趋势。在众多现场总线中,LonWorks 控制网络以其优秀的分布处理能力、开放性、互操作性、多媒介适应能力以及多网络拓扑结构等特性适应了未来发展对测控网络的要求,成为其中的佼佼者。利用 LonWorks 技术设计网络监控,实现起来非常方便。然而,由于对于网络节点的监控需要专用的网卡和计算机,使得整个网络成本大大增加,而且不利于现场操作。我们针对以前开发的小规模网络控制系统,设计了一种专用的低成本网络监控器,方便了现场操作,极大地降低了整个 LonWorks 局域控制网络的成本。经试用效果良好,可以部分取代原来的网络监控计算机。

一、LonWorks 控制网络的网络结构[2]

LON 网是一种分布智能网,它以 Neuron 芯片为主构成的节点作为分布设备控制器,节点之间通过网络总线进行通信,现场网可以通过上位机接入局域网,进行必要的集中监控。网络结构如图 6.8-1 所示。

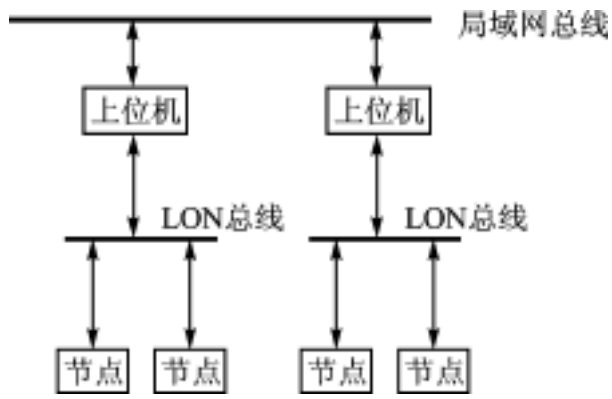


图 6.8-1 LON 网络结构

LonWorks 控制网络由许多节点组成,节点通过收发器接入网络总线。上位机通过 LonWorks 网卡与 LON 总线相连。典型的 LonWorks 网络节点如图 6.8-2 所示(主处理器可选),它是以 Neuron 芯片为主的控制器[3],能独立地处理任务并可以接入网络接受上层控制器的控制,可用类似 C 语言的 Neuron C 语言编程,并采用事件驱动程序结构。Neuron 芯片提供丰富的 I/O 对象,可与多种外部设备相连。另外 Neuron 3150 芯片固件内含 LonTalk 通信协议和任务调度程序(Scheduler),可方便地实现网络通信和程序控制。

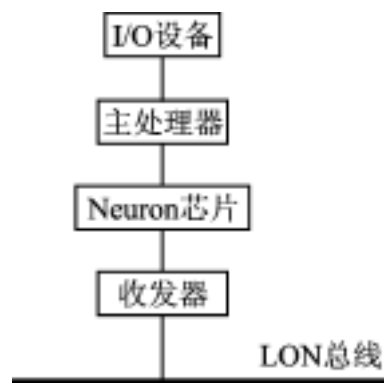


图 6.8-2 节点结构

二、网络监控节点的设计

目前的网络监控一般采用上位机,功能比较齐全,但存在体积大、成本高等缺点。其实,对于单纯的监控功能,可以由价格低廉的单片机外加一些外围设备来实现,操作起来非常简单。我们设计了一个监控节点,可方便地实现网络监控。选用的元器件主要有 LonWorks 控制模块(含有 1 片 Neuron 3150, AT29c256 E^2PROM, FTT-10A 收发器)、1 片 AT89C52 单片机芯片,1 个 LCD 显示模块以及键盘单元。电路原理图如图 6.8-3 所示。单片机 P1 口作为并行数据口,P3.2 作为 LonWorks 控制模块选通控制位,P3.1 作为读写控制位,P3.0 作为握手信号位,P0 口作为键盘接口,P2 口作为 LCD 模块数据口,P3.3～P3.5 分别作为选通控制位,读写控制位和数据写入控制写入选择位。需要注意,为了实现单片机芯片和 Neuron 芯片之间的同步,单片机芯片的复位电路应该也能触发 Neuron 芯片的复位。另外,工作时单片机需要监视 Neuron 芯片的 HS 位的状态以保证数据传送的同步,但是有可能 Neuron 芯片未及时设置好 HS 的状态而单片机已开始轮询 HS,在 HS 引脚上加上上拉电阻(10 kΩ)可避免单片机读取 HS 的无效状态。实验中我们将 LonWorks 控制模块、单片机芯片、LCD 模块、键盘模块等焊接在一块小电路板上,做成一个 LON 网监测节点,对外只留出电源线和网络线作为接口,键盘模块,LCD 显示模块采用插线型。

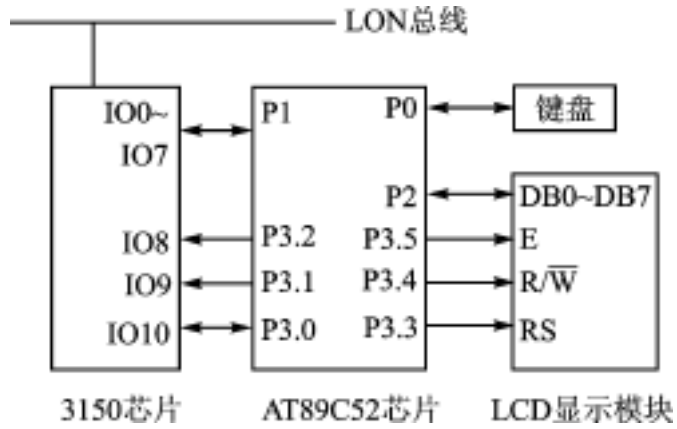


图 6.8-3 监控节点电路原理图

Neuron 3150 神经元芯片具有功能丰富的 I/O 接口,可定义 34 种不同的 I/O 对象。单片机和 Neuron 3150 芯片之间的通信可以由简单的握手/令牌传递协议来实现。单片机工作在主机方式,3150 芯片工作在并行从 A(Slave A)方式。这种方式下 Neuron 3150 芯片驱动 IO10 产生握手应答信号(HS),接收 IO8、IO9 的片选信号($\overline{CS}$)、读写控制信号(R/$\overline{W}$),IO0～IO7 作为双向数据口。对于主频为 10 MHz 的 Neuron 芯片并行口的最大数据传输速率为 3.3 Mb/s。每个字节读/写完毕,主机监视握手信号 HS 用以确定从 A 是否准备好下一个字节的传送。

Neuron 3150 神经元芯片内部固件包含 LonTalk 通信协议,该协议遵循 ISO 的 OSI 协议,支持灵活编址,并且单个网络可存在多种类型的通信媒体构成的多种通道。网上任一节点使用该协议可以与同一网上的其他节点互相通信。对设计者来说,LonTalk 协议就是一个服务集,设计者可按需选择使用。LonWorks 控制网络技术采用网络变量(network variables)和显式消息(explicit messages)的方法进行网络通信,设计者可在自己的节点程序里定义自己的输入输出网络变量并对其直接读取或改写。LonWorks 技术利用数据绑定(binding)方法实现输入变量和输出变量的连接,数据绑定可由 LonMaker 工具来完成。设计者也可将所要传输

的数据放在构造的消息中,以显式消息的方式进行通信。数据和消息的传送由 Neuron 芯片的固件完成。

该监测节点的工作流程是:首先由键盘输入所需监测网络变量的编号,由单片机传给 LonWorks 控制模块,然后由 Neuron 3150 通过 LON 网查询相应网络变量,并将其回传给单片机,由单片机送到 LCD 显示。其他控制功能可编程完成。编程框图如图 6.8-4 所示。

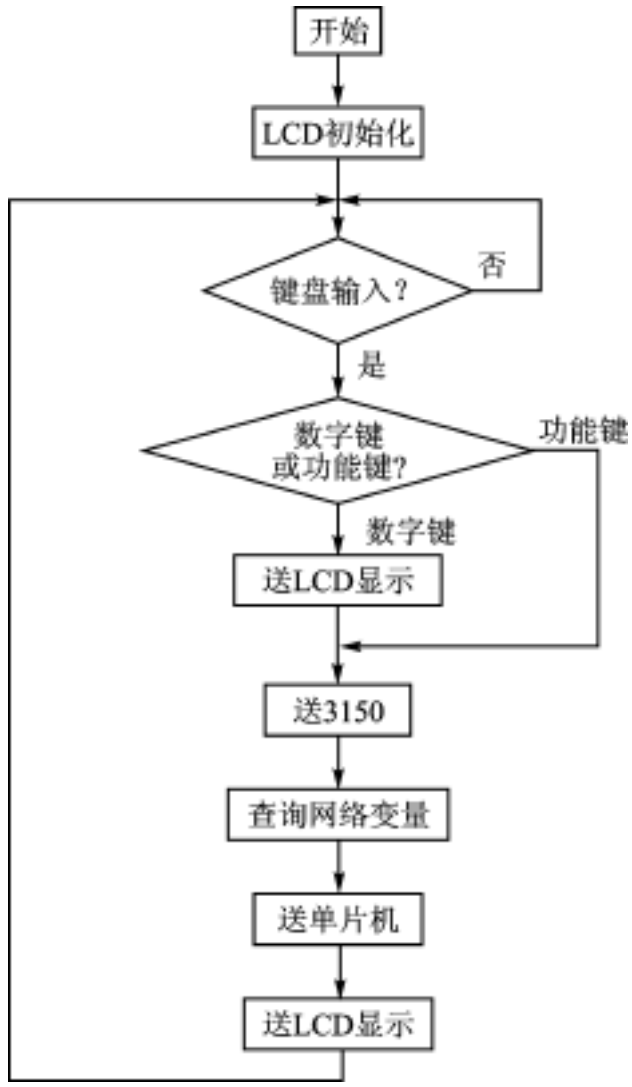


图 6.8-4 软件工作流程图

三、软件编程要点

(1) 单片机芯片和神经元芯片通过令牌传递/握手协议进行通信,Neuron 芯片的握手及令牌传递的实现是自动地由 Neuron C[4] 编程语言提供的几个函数和事件来完成的。但是对于单片机芯片,编程人员必须通过编程使其能执行 Neuron 芯片的握手/令牌传递算法,也就是说在单片机这一侧要复制神经元芯片的主工作方式(master mode),从而控制 Neuron 芯片工作在从 A 方式。

(2) 程序由两部分组成:单片机程序由汇编语言编写,神经元芯片程序用 Neuron C 语言编写。对交互部分,要合理分载,数据的处理工作可由单片机完成。

(3) Neuron C 对 ANSI C 做了扩展,包括:一个内部多任务调度程序、网络变量类型、软件定时器、I/O 数据读写函数、事件驱动编程结构等,编程时应安排好程序结构。需要注意,一个事件处理程序不要过长,否则一方面影响其他事件的触发,另一方面可能引起节点的复位。

(4) 输入、输出网络变量的绑定要在同类型之间进行,注意多利用 Neuron C 自身提供的标准网络变量类型。另外要注意节点能连接的网络变量数是有限的。

四、结束语

本文论述的设计网络监控器的方法易于实现,成本低廉,实用性很强。对照国外同类产品,该监控器成本低,体积小,操作简单,可在许多场合取代 PC 机,完成网络监测任务。另外,LonWorks 控制模块包含一片 E^2 PROM,应用程序的修改比较方便,非常利于功能的扩展。我们设计的这种网络简易监控器已经在基于 LonWorks 的局域网中得到了实际应用,效果良好,可以部分取代 PC 机的功能。

参 考 文 献

1 Echelon . Developing a Hand-Held Network Tool with LNS,1998

2 杨育红 . LON 网络控制技术及应用 . 西安:西安电子科技大学出版社,1999

3 Toshiba . User s Guide for TMPN3150 Neuron Chip,1996

4 Echelon Corporation Inc . Neuron C Programmer s Guide, 1995

5 齐秋群,刚砺韬 . Motorola & Intel 单片机程序设计与应用 . 北京:机械工业出版社,1998

选自《测控技术》月刊,2000 年第 8 期

6.9 现场总线 CANbus 与 RS-485 之间透明转换的实现

南京国家电力总公司电力自动化研究院大坝所(210003) 邓检华

现场总线技术以其独有的技术优势和特点,在现代分布式测量与控制技术领域中的应用已愈来愈广泛。各种现场总线的主控制器一般都内嵌有相当完善的、开放式的互联通信协议,它具有通信速度快、误码率低、开发设计简单及网络使用维护方便等诸多特点,是实现网络化现场测量与控制技术的一个发展方向。但目前,在许多现场已投入使用的测量与控制系统中,各仪器设备或装置之间通信所使用的仍是传统的 RS-485 或 RS-422 总线。在不断投入新型现场总线系统的同时,要在短期内改造或淘汰那些旧系统是不现实的。况且,在许多应用场合,新老系统中主机的控制算法及功能是相似或兼容的,所以在一定时期内,新老总线系统同时并存是客观的现实需要。对此,若能将新老仪器设备或装置通过一种透明转换装置而有机地糅合在一起,去掉老系统中重复的部分,是一种很好的选择。

一、CANbus 简述

CAN 现场总线技术是德国 BOSCH 公司于 20 世纪 80 年代初为解决现代汽车业中众多的控制与测试仪器之间的数据交换而开发,目前已逐步应用到其他领域的一种符合国际标准的串行数据通信协议。CAN 的主要技术特点叙述如下。

(1) 对等网络结构,网络上任意节点可在任意时刻向网络其他节点发送信息,不分主从,通信方式灵活。

(2) 采用非破坏性总线仲裁技术,网络中的节点可以分成不同的优先级,当多个节点同时向网络上传送信息时,优先级低的节点主动暂停数据发送,而优先级高的节点可不受影响地继续传送数据。之后,按优先级高低,其他节点在总线空闲时依次进行被暂停的数据传送,有效地避免了总线冲突。

(3) 每一帧的有效字节数最多为 8 个,帧传送时间短,受干扰概率小,重发时间短。每一帧信息都有 CRC 校验及其他检错措施,通信误码率低。

(4) 网络节点在错误严重的情况下,具有自动关闭总线接口的功能,避免影响总线上的其他节点的正常操作。

(5) 通信距离最远达 10 km(5 kb/s),通信速率最高可达到 1 Mb/s(40 M),节点数目实际可达 110 个。通信介质采用双绞线,也可用光纤。

二、实现方法

CAN 现场总线与 RS-485 总线的主要区别是:CAN 总线是以帧为单位进行数据通信的,且每帧均携带对应的 ID 标示符,而 RS-485 是以字节为单位进行数据通信,不带任何其他附属信息。如果不考虑帧中的 ID 标示符,那么这二种总线传输的信息就可以认为完全相同。考虑到绝大多数应用 CAN 总线的场合都不可能分配完系统中的 ID 标示符资源,因而在系统中

可以另分配一至多个 ID 标示符给 RS－485 总线数据,即给总线透明转换电路所用,收、发数据的 ID 标识符可以不相同。CAN 总线系统中的公用主控设备一般被设置成直通状态,它可通过发送、接收具有不同 ID 标示符的不同数据帧,方便地对网络中的各种设备进行管理和控制。对那些纯粹靠帧数据本身而忽略 ID 标示符的系统来说,这种不同总线之间的转换更加简单。

实现二种总线之间的透明转换的基本过程是这样的:电路加电进入正常状态后,首先以电路中保存的参数(如未初始化则以缺省参数)对二个总线通信端口分别进行初始化,并将所有通信端口设置成中断接收工作方式,在启动内部看门狗(Watchdog)后,电路等待外来数据的中断。一旦某一总线端口有有效数据进入,电路首先将这些数据读入内部数据缓冲区,置相关内部标志,关闭其他端口的中断,以使接收端数据能及时被接收到。在下一数据到来之前,电路将及时通过另一端口将缓冲区中的数据转发出去,直到缓冲区空。在这种转换过程中,如果数据是自 CANBUS 一端流入,则电路只将帧内数据读入到缓冲区。相反,则电路在转发数据之前自动按设置将设定 ID 标示符加到发送信息中,一旦转换过程结束,所有端口又被允许中断接收。电路中缓冲区一般可配置 8～32 KB 的静态 RAM。

在未进入转换工作时,电路允许参数设置端口的设置中断进入。参数设置包括:设置 RS－485的通信波特率(当然端口数据格式也可设);设置 CANBUS 的通信波特率、接收码(accept－code)、屏蔽字(mask－code)、发送标示符。参数设置好后将立即生效,通过该端口还可查询当前电路中的各运行参数值。电路工作的主要流程图如图 6.9－1 所示。

三、实现电路

图 6.9－2 是一种可选的线路原理示意图。电路中使用的单片机是 ATMEL 公司的 AT89S8252,它是一种内带 8 K Flash RAM、2 K EEPROM、同时内嵌独立的硬件 Watchdog 电路,最高工作主频为 24 MHz 的新型单片机。所配数据缓冲器 62256 为 32 KB 的静态 RAM。配置较大缓冲区的明显好处是提高了对二种不同速率总线的适应能力。图中 RS232 口是专用来设置二总线端口的工作参数,所设参数均保存在 CPU 的 2 K EEPROM 中。另外,为提高线路的可靠性,对 CANBUS 端采用了电隔离措施,RS－485 端所使用的 MAX1480 也是一种内部电隔离的芯片。图中 V_{CC} 与 V_{CC1} 为相互隔离的二组 5 V 电源。

对于那些对总线响应速度要求比较苛刻的场合,可采用双 CPU 控制电路,即每个 CPU 分别负责一端总线的通信事务。数据缓冲区仍采用公用的单口或双口 RAM。相应的控制算法必须增加一些有关对公共数据区的管理操作,在此不做详述。

以上电路多适用于二种总线并存于同一系统的场合,以实现二种总线之间的有机结合,使新老设备能很好地同时运行,节省系统开支。这种透明转换电路在已研制的分布式数据采集系统中应用以后,效果良好。

对以上电路稍做修改,即将 MAX1480 换成 MAX232 芯片就可设计出一种能在 CAN 现场总线与 RS－232 之间实现透明转换的电路。这种电路可用于那些需要用便携机与现场设备之间实现通信(如现场调试等)的场合,因为目前便携机一般只配 RS－232 接口。另外,如将电路的二个总线端口设计成相同的接口,那么它还可以用在使用同一种总线,但不同的区域却有不同通信速率的应用场合。对于那些首次接触 CANBUS 技术的开发人员来说,以上电路还是一种很好的端点开发辅助设备,即开发人员只要对该电路设置合适的端口参数,并将用

户电路与之连接好后,开发人员即可在一相对熟悉的环境下专心开发自己的应用电路。

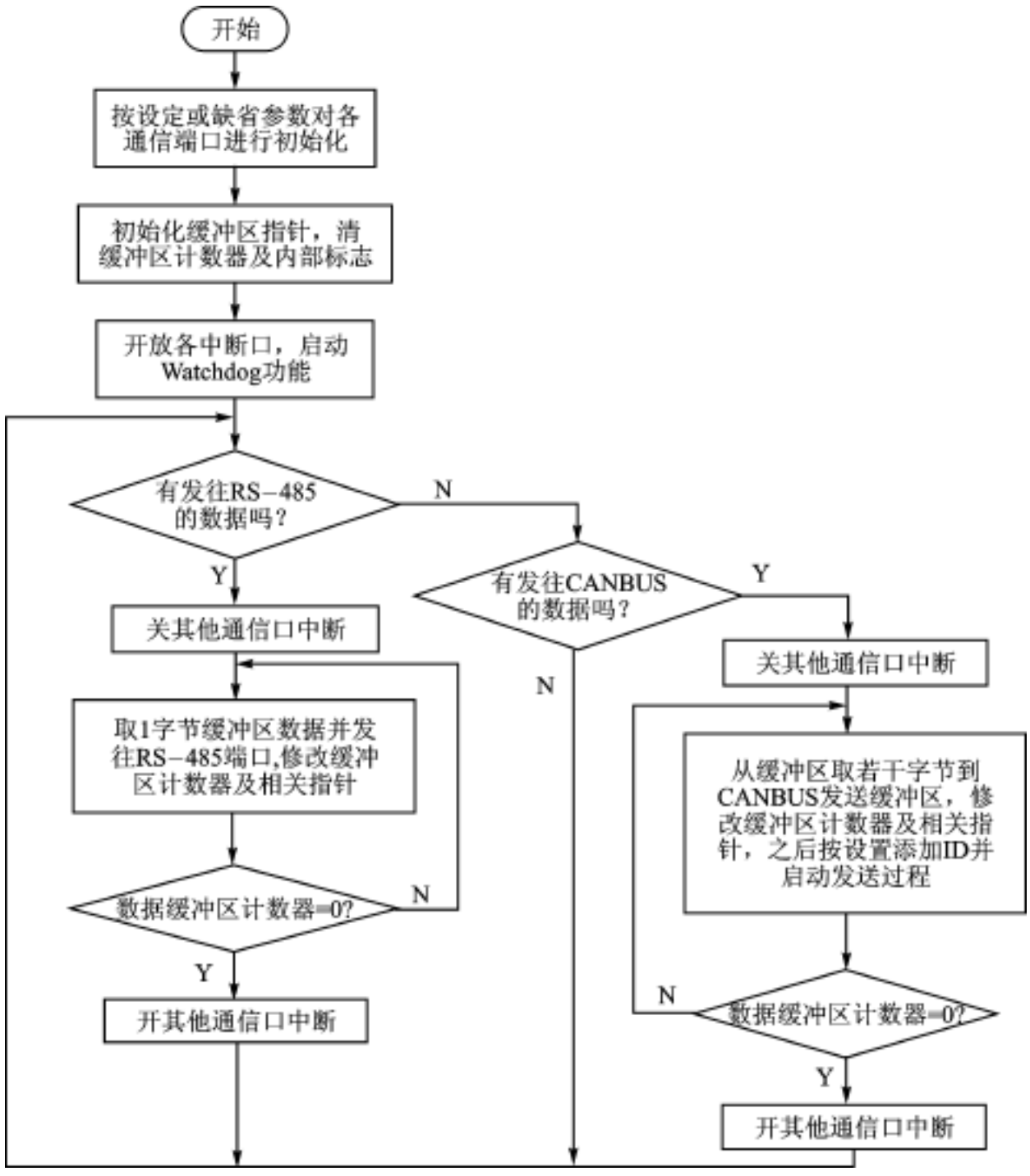


图 6.9-1 主程序工作流程

参 考 文 献

1 SJA 1000 Stand-alone CAN controller. Philips Semiconductors, 1997; (04)

2 8-Bit Microcontroller with 8 K Bytes Flash, AT89S8252. ATMEL, 0401 D-A-12/97

3 邬宽明. CAN 总线原理和应用系统设计. 北京:北京航空航天大学出版社, 1996

选自《电子技术应用》月刊,2001 年第 5 期

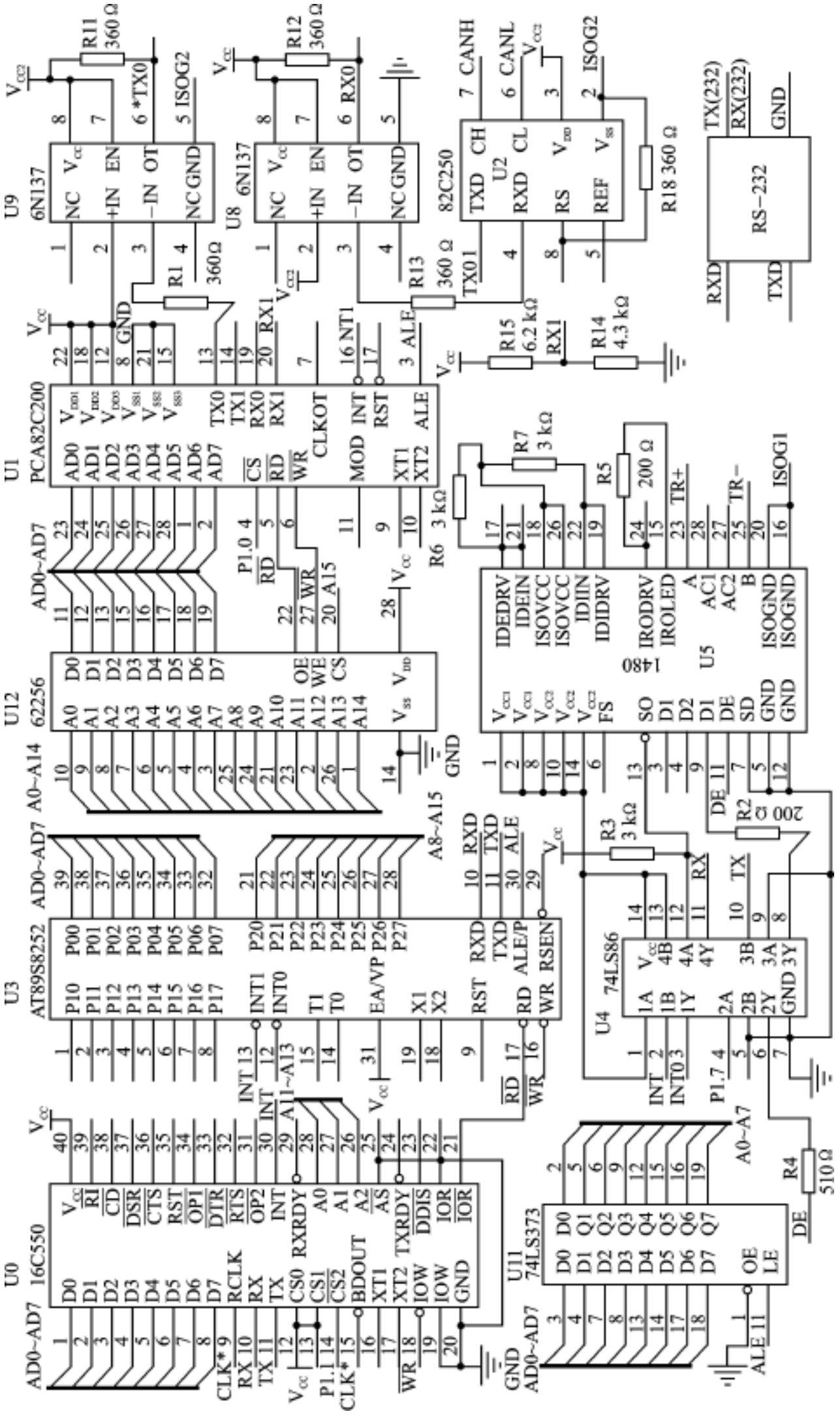

U0
16C550
U1
PCA82C200
U2
82C250
U3
AT89S8252
U4
74LS86
U5
1480
U8
6N137
U9
6N137
U11
74LS373
U12
62256
RS-232
TX(232)
RX(232)
GND
R1 360Ω
R2 200 Ω
R3 3 kΩ
R4 510 Ω
R5 200 Ω
R6 3 kΩ
R7 3 kΩ
R11 360 Ω
R12 360 Ω
R13 360 Ω
R14 4.3 kΩ
R15 6.2 kΩ
R18 360 Ω
AD0~AD7
A0~A14
A8~A15
A0~A7
CANH
CANL
ISOG1
ISOG2
ISOGND
TR+
TR−
TXD
RXD
ALE
INT
RST
CLKOT
MOD
GND

图 6.9-2 线路原理图

6 .10　居室自动化系统中的 X－10 和 CE 总线

哈尔滨工业大学(150001)　刘思久　江　军
哈尔滨电工仪表研究所(150040)　冯玉贵　张礼勇

居室自动化始于人们对未来生活的想象,人们希望能够对家用电器进行远程控制,也希望能够在居室内部房间之间实现对电器的直接操作,进而建立具有集中管理模式的自动化系统,用以控制电、热、水、家用电器和防盗安保等设备。电力线作为居室的基础设施,以其作为通信媒介自然成为人们的首选目标。通常这一目标的实现需要若干工业部门和若干公司的合作,而最终需要各个电器生产厂家遵循统一的标准,使其产品在系统中可以相互兼容。X－10 和 CEBus 总线在这一方面的不同时期担任了人们所期望的角色,熟悉和掌握这些标准,对于建立居室自动化系统来说,往往可以起到事半功倍的作用。

一、X－10 协议及其特点

X－10 协议是美国家用电器自动化的先驱,它极为广泛地用在家电领域的各种产品中。在 X－10 标准中,一个持续时间为 1 ms 的 120 kHz 的载波信号被加载到交流电过零点处,如图 6 .10－1 所示。

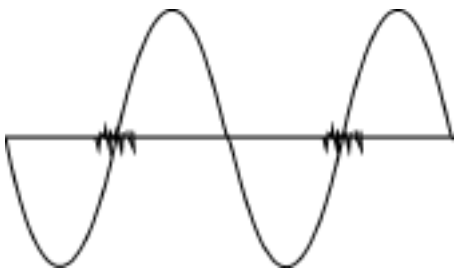


图 6 .10－1　X－10 总线中信号的传输

这里信号“1”是用“10”模式来表示的,即在第一个过零点处先发送一个 120 kHz 的载波信号,然后在下一个过零点插入一个空闲状态(不发载波信号)。信号“0”是用“01”模式来表示的,即在过零点处先插入一个空闲状态,再在下一个过零点处发送一个 120 kHz 的载波信号。这样,如果使用频率为 60 Hz 的电力线来传输,一秒钟内能够发送 60 个“0”或“1”的数字。

X－10 总线严格按照时间顺序进行数据传输,并使用起始代码对数据进行同步。起始代码由开始的三个连续过零点处发送的三个 1 ms 的 120 kHz 的脉冲串,和在第四个过零点处插入的一个空闲状态组成。然后接着传送 4 位的房间代码和 5 位的键代码,代码包的格式如图 6 .10－2 所示。这里房间代码用来选择名为 A 至 P 中的哪一个组单元将接收命令;而在 32 个键代码中,16 个用于表示通信单元的地址,另外 16 个表示 X－10 的操作命令。在 X－10 系统通信过程中,首先需要寻址,通过在键代码中指定单元地址,并与房间代码结合起来选择 256 个 X－10 单元,告诉此单元需要等待接收命令。寻址后就应接着发送命令。X－10 还可以通过连续发送多个单元地址来对多个单元寻址,并对它们进行统一的操作;也可以对同一单元连续发送一连串的命令。X－10 总线是单工工作方式,因此没有固定的反馈确认机制。尽管 X－10 本身也允许双向通信,但是很少有 X－10 设备支持这一功能。

开始代码 (2 位)	房间代码 (4 位)				键代码 (5 位)	开始代码 (2 位)	房间代码 (4 位)				键代码 (5 位)	
房间代码	H1	H2	H4	H8		键代码	D1	D2	D4	D8	D16	
A	1	0	0	1		1	0	1	1	0	0	
B	1	0	1	1		2	1	1	1	0	0	
C	1	0	0	0		3	0	0	1	0	0	
D	1	0	1	0		4	1	0	1	0	0	
E	0	1	0	0		5	0	0	0	1	0	
F	0	1	1	0		6	1	0	0	1	0	
G	0	1	0	1		7	0	1	0	1	0	
H	0	1	1	1		8	1	1	0	1	0	
I	1	1	0	1		9	0	1	1	1	0	
J	1	1	1	1		10	1	1	1	1	0	
K	1	1	0	0		11	0	0	1	1	0	
L	1	1	1	0		12	1	0	1	1	0	
M	0	0	0	0		13	0	0	0	0	0	
N	0	0	1	0		14	1	0	0	0	0	
O	1	1	0	1		15	0	1	0	0	0	
P	0	0	1	1		16	1	1	0	0	0	
						所有单元 off	0	0	0	0	1	
						所有灯 on	0	0	0	1	1	
						on	0	0	1	0	1	
						off	0	0	1	1	1	
						Dim	0	1	0	0	1	
						Bright	0	1	0	1	1	
						所有灯 off	0	1	1	0	1	
						扩展代码	0	1	1	1	1	
						Hail 请求	1	0	0	0	1	
						Hail 认可	1	0	0	1	1	
						预设 Dim	1	0	1	×	1	
						扩展数据	1	1	0	0	1	
						状态 = on	1	1	0	1	1	
						状态 = off	1	1	1	0	1	
						状态请求	1	1	1	1	1	

图 6.10-2 X-10 命令结构和代码

二、CBSus 总线

CEBus(用户电子总线)是由 EIA(美国电子工业协会)制定并颁布的一种用于居室自动化的标准,许多从事家用电器、电话、计算机和报警系统的中小型公司的代表参与了标准的起草。1992 年正式推出命名为 CEBus IS-60 标准,在 1998 年升级为 EIA-600。标准给出了 CEBus 使用媒介(电力线 PL、红外线 IR、双绞线 TP、同轴电缆 CX、无线电 RF 和光纤 FO)的接口描述及信令方式;标准还规定了与所用媒介无关的网络协议和控制消息格式;同时标准也规定了用于设备间相互通信用的面向对象语言 CAL(公共应用语言),以实现对设备资源的分配和控制。CEBus 标准是一种应用于网络的开放式的通信协议,它采用的是节点到节点的通信方式,所以电器设备间不需要中心控制器。同时所有媒介以同一速率(10 kbps)传输数据,作为

一种有效、低成本的通信协议,它能够保证各种电器设备间实现自动控制的通信要求。

CEBus 网络模型是参考 ISO 的 OSI 七层模式设计的,但 CEBus 协议仅有四层:物理层、数据链路层、网络层和应用层(见图 6.10-3)。每一个 CEBus 信息是由报头和数据包组成。报头是 CSMA(冲突检测载波侦听)协议的一部分,发送方只是用它来监听传输介质中是否有其他发送方占用信道,以获取传输通道的控制权限。一旦获取了传输通道,该节点就可以发送有用的数据包了。

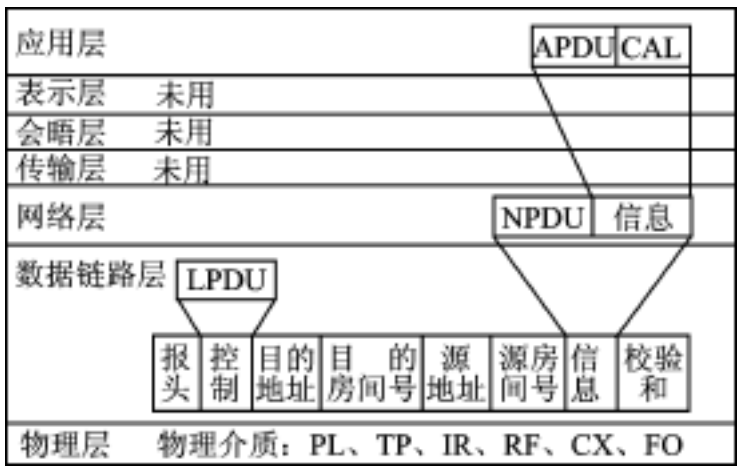


图 6.10-3 基于 ISO/OSI 七层网络模式的 CEBus 结构图

CEBus 中物理层的主要功能是负责传输信号的接收和发送。该层分为编码子层(SE)和媒介物理子层(MDP)。前者完成信号编码和解码、错误检测工作;后者完成电信号的收发和连接到物理媒介的接口。与其他的标准不同,CEBus 标准并不使用特殊的状态来表示不同数字位,而是采用交替变化的两个状态中同一状态所持续的时间来表示,即脉冲宽度编码。所有的信息包都由以下 4 个信号所组成:“1”,“0”,“EOF”(帧结束符)和“EOP”(包结束符)。其脉冲宽度:信号“1”持续一个 UST(单元信号时间),信号“0”持续 2 个 UST,信号“EOF”持续 3 个 UST,信号“EOP”持续 4 个 UST。这里,UST 持续时间及两个状态的定义均视所用通信媒介的类型来确定。

通常,两个编码状态被分别称作 Superior 状态和 Inferior 状态。对电力线(PL)载波来所说,UST 为 100 μs。同时,为了将报头和数据部分区分开来,又将 Superior 状态分为 Superiorθ1 和 Superiorθ2 两种情况,并为不同的阶段采用。编码采用的两个交替状态在报头阶段是 Superiorθ1 和 Inferior;而对一般数据包,采用的是 Superiorθ1 和 Superiorθ2。在 Inferior 状态中,发送方实际并不发送信号,只是静静地监听其他的传输信号。而 Superiorqθ1 和 Superiorqθ2 则如图 6.10-4 所示,为相位相反、持续时间为 100 μs、频率从 203 kHz 到 400 kHz,回到 100 kHz 后再升到 203 kHz 的正弦波扫频信号。

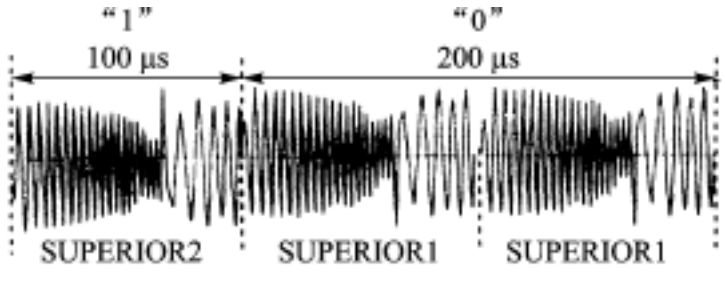


图 6.10-4 SUPERIORθ1 和 SUPERIORθ2 状态信号

数据链路层分为逻辑链路子层(LLC)和媒介接入控制子层(MAC)。与物理层相连的MAC子层几乎完成了数据链路层的所有工作,而LLC子层是个空壳,只转发命令,无实质性工作。数据链路层所发送的数据顺序如图6.10-3中所示。在报头之后的数据链路层协议数据单元(LPDU)包含了包类型、包的优先级、权限、所属的服务类型(见图6.10-5)。CEBus的32位地址分为两个部分,系统地址和MAC地址。系统地址对应于X-10中的房间代码,MAC地址对应于X-10中的单元代码。目的地址和源地址都被发送出去,这样接收方就能将响应信息返回给发送方。在包地址后面,紧跟的是来自网络层的包信息,其最大长度为32字节。最后是除报头以外的其他所有数据的校验和,用于错误检测。

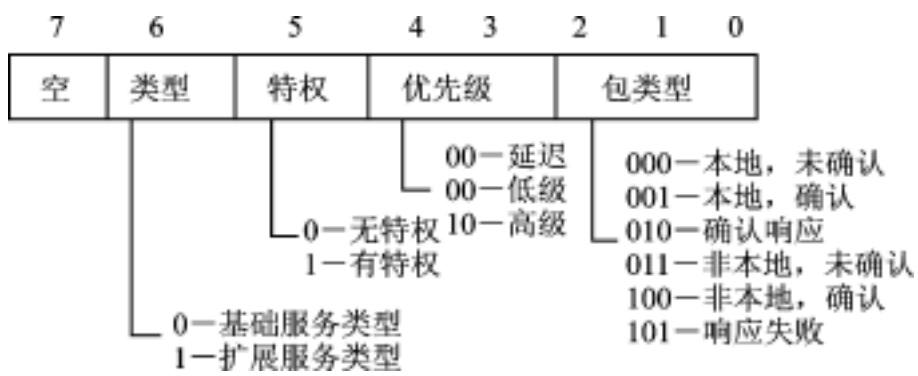


图6.10-5 LPDU的组成结构

网络层主要负责路由控制、确定网址、流量控制、数据包的分割和重复包的弃用。由于网络层对所有媒介都是相同的,所以要考虑信息的跨媒介传输问题。网络层在将应用层下传的信息向数据链路层传输时,将网络层协议数据单元(NPDU)添加到该信息的前面。NPDU(见图6.10-6)的长度为8位,其中6位用于表示接收数据的通信媒介类型。另外两位决定路由的类型,其中扩散路由是将发送包传输到在NPDU后面的6位中所指定的所有的传输媒介中;而且录路由下的发送包则仅被位于传输路径上的路由器转发。任何一个设备,当它上电或改动网址时,都要发出一个ID包以声明自己的新网址。该ID包将传遍网络,帮助路由器更新各自的路由表。如果路由器在转发一个包时,发现自己的路由表中没有该包的目的网址,它就通过设置一个标志位,把该包变为ID需求包然后转发。目的网址收到ID需求包后,要发出一个ID包。所以路由器的设置是全自动的。

图6.10-6 NPDU的组成结构

应用层是CEBus的最高层。作为一般的ISO/OSI网络组成部分,应用层负责终端用户所看到的内容。而在CEBus中,它只被程序员所见。应用层的信息包括两部分:应用层协议数据单元(APDU)和CAL命令信息。美国电子工业联合会专门定义了公共应用语言(CAL),用于设备间的智能通信。应用层不仅用CAL语言发布命令实现对电器设备操作的控制,而且还用CAL向网络层指明了优先级、是否需要应答、是否有网址和是否使用流量控制。APDU可长达3810个字节,但是到目前为止,只对前面两个字节进行了定义(见图6.10-7)。第

一个字节包括模式信息和类型识别符,模式指定了服务级别、头类型和后面的数据长度。服务级别包括基础类和优先类(目前所用的大多数是基础类)。头类型的数据长度分为固定和可变两种,一般情况下头长度是固定的。数据长度分为短(32 字节)、长(3808 个字节)和超长(1638375 字节)三种情况。类型识别符用来确定命令是隐式还是显式命令,并为显式命令规定相应代码。显式命令需要目的节点返回响应,隐式命令则不需要应答信号,因此易于编程,发送速度比较快,但容易出错。APDU 的第二个字节用来确定传输是同步还是异步,在显示命令的情况下所需要的相应的类型是什么。

图 6.10－7 APDU 的组成结构

CEBus 规定 CAL 依次由 CONTEXT、SASE 和 CASE 三个部分组成。CONTEXT 规定了设备环境和设备所属类型,比如声音处理类型(放大器、衡音器)、图像显示器、通信控制系统(电话,无线电)、时间服务器件(时钟,定时器)、环境管理系统和照明系统等。特定应用服务单元 SASE 则用来确定命令序列的主要功能。每个 CONTEXT 都有一个已定义好的 SASE 列表,但 SASE 列表对于各种 CONTEXT 都是相似或相同的。如声音处理主要有主模式切换(电源)、源切换(无线、CD、录音机),特性切换(噪音、环境)和水平控制(音量、低音、高音、平衡)。公共应用服务单元 CASE 用于定义最后的操作,包括真、假、加、减和加载等。使用 CONTEXT、SASE 和 CASE 来构成 CAL 命令信息,就可以实现所期待的功能。EIA 制定了预设的 CONTEXT、SASE 和 CASE 表,通常是通过查表来使用这些命令的,而且厂方也是遵循 CAL 命名规则来开发新命令的。CAL 命令中 CONTEXT、SASE 和 CASE 三部分都是用 16 进制数表示的。如我们想用 CAL 命令来打开电视,查表知:在 CONTEXT 表中 38h 表示图像显示器(适用与电视和电脑显示器),在 SASE 表中 D0h 表示主模式切换(用于上电等主要功能),在 CASE 表中 81h 表示打开。这样最后只需将 38h、D0h、81h 作为 CAL 命令信息发出就可以了。

三、CEBus 和 X－10 总线的比较

X－10 总线自从引入市场后,很快就在电力线载波通信中已经变成了实际上的通信标准。在 20 世纪 80 和 90 年代,所有在市场上销售较好的家用控制系统都在某种程度上与 X－10 兼容。20 世纪 90 年代后出现的 CEBus 总线提出了通过相同的电源线进行通信的方法。然而总的来说,CEBus 和 X－10 是完全不兼容的系统。由于 X－10 是一种单工方式工作,又要依赖于交流电的过零点,每个数位必须传输两遍,因此尽管具有成本低、用户多等明显特点,但其

被 CEBus 取代则仍将是大势所趋。CEBus 总线接口技术比较复杂,成本也较高,因此实际应用的 CEBus 产品相对还较少。但随着通信技术、网络和微控制器的发展与相互结合,CEBus 总线的应用已经开始出现高潮。尤其在电力线载波通信方面取得了很大时展,诸如美国 Intellon 公司的 SCC－P300 等与 CEBus 兼容的通信芯片已陆续投放市场。同时,由于一些较大的公司也参与到该领域的研究上来,极大地推动了居室自动化的发展。出现了基于 CEBus 总线标准的灯光控制器,无线保安系统,集中供热通风系统,手持射频遥控器,音频视频设备控制器等家用电器自动控制设备。一些设备提供商还向用户承诺,可以很容易地在现有的产品上安装控制系统,并执行读表,负载控制及保安监视等服务。目前 CEBus 的业界团体(CEBus Industry Council)一致同意采用微软公司的 P&P 和 SCP(Simple Control Protocol)作为家电控制产品的协议。这样将使 CEBus 居室网络总线的功能更强,更灵活性、更可靠,并使更多的家电设备更方便地接入网络,进而实现居室信息的因特网化。

四、结束语

CEBus 总线标准作为一种居室自动化系统的通信标准,已经取得了一定的应用成果。作为 21 世纪被看好的智能家电或网络家电市场,这也是一项重要的基础技术。在 CEBus 总线的研究和应用上投入更多的力量,必能得到相应的回报。

参 考 文 献

1 Jeff Bachioch . Digital and Analog Output Control [J] . Circuit Cellar, May 2000

2 Rick Zarr . Add a Serial X－10 Interface to Your PC[J] . Circuit Cellar, Jan 1994

3 Christopher Yasko . Put a CEBus Power Line Interface in Your Next Design[J] . Circuit Cellar, Jan 1994

选自《电测与仪表》月刊,2001 年第 12 期

6 .11　通用串行总线 USB

清华大学微电子学研究所(100084)　黄维柱　许　军

一、引　言

如今,随着计算机应用的日益普及,其外设也越来越多,如:打印机、鼠标、键盘、扫描仪、游戏杆、音箱、MIC、Modem……。每个外设都需要通过一个接口与计算机相连,当外设多了以后,会产生一些问题。

首先,外设增多,计算机的接口也必须跟着增多,但计算机的接口总是有限的,虽然可以通过给计算机添加板卡的方式来扩展出一些接口,但这终究不是解决问题的根本方法。

其次,随着技术的不断发展,大量新的外设不断出现,这些外设对计算机接口提出了更高的要求,如高速度、双向传输数据等。传统的计算机接口,如并行打印机接口(LPT)、串行 RS232 接口已经不能满足用户的需要。

还有,计算机越来越向简单、实用、方便、廉价方向发展。传统计算机外设安装过程中,在加、减设备时,必须关掉电源,加、减设备完成之后再重新启动机器。对于板卡式的外设就更加不方便了,您不得不打开机箱,既麻烦,又容易出问题,而且有些板卡价格还很昂贵。

由于传统计算机接口有以上一些缺点,不能满足当前计算机的发展需要。于是,Compaq, Intel, Microsoft, NEC 等公司联合制定了一种新的计算机串行通信协议 USB(Universe Serial Bus),该标准的 Ver 0 .7 版本最早于 1994 年 11 月推出,以后进行了多次修改和补充,现在流行的版本为 1998 年 9 月 23 日公布的 Ver1 .1。USB 协议出台后得到各计算机生产商、芯片制造商和计算机外设厂商的广泛支持。如今,计算机主板都带有 USB 接口, Windows 98 也全面支持 USB 标准,很多计算机外设都采用 USB 接口,各种带 USB 接口的芯片也在市场上不断涌现。

二、USB 的优点

USB 与传统的外围接口相比,主要有以下一些优点。

1 . 速度快

USB 有高速和低速两种模式。主模式为高速模式,速率为 12 Mbps,从而使一些要求高速数据的外设,如高速硬盘、摄像头等,都能统一到同一个总线框架下。另外为了适应一些不需要很大吞吐量,但有限高实时性要求的设备,如鼠标、键盘、游戏杆等, USB 还提供低速方式,速率为 1 .5 Mbps。不管是高速还是低速模式,速度都比 RS232 接口快得多。

2 . 易扩展

USB 采用的是一种易于扩展的树状结构,通过使用 USB Hub 扩展,可连接多达 127 个外设。标准 USB 的电缆长度为 3 m(5 m,低速)。通过 Hub 或中继器可以使外设距离达到 30 m。

3．支持热插拔和即插即用

在 USB 系统中,所有的 USB 设备可以随时接入和拔离系统,USB 主机能够动态地识别设备的状态,并自动给接入的设备分配地址和配置参数。这样一来,安装 USB 设备不必再打开机箱,加、减已安装过的设备完全不用关闭计算机,也不必像过去那样,需要手动跳线或拨码开关来设置新的外设。

4．USB 提供总线供电和自供电两种供电形式

当采用总线供电时,不需要额外的电源。USB 主机和 USB Hub 有电源管理系统,对系统的电源进行管理。

5．使用灵活

USB 共有 4 种传输模式:控制传输(Control)、同步传输(Synchronization)、中断传输(Interrupt)、批量传输(Bulk),以适应不同设备的需要。

6．支持多个外设同时工作

在主机和外设之间可以同时传输多个数据和信息流。

总之,USB 是一种方便、灵活、简单、高速的总线结构。

三、USB 的拓扑结构

USB 系统由主控制器(Host Controller),USB Hub 和 USB 器件(Device)组成。系统的拓扑结构如图 6.11-1 所示。

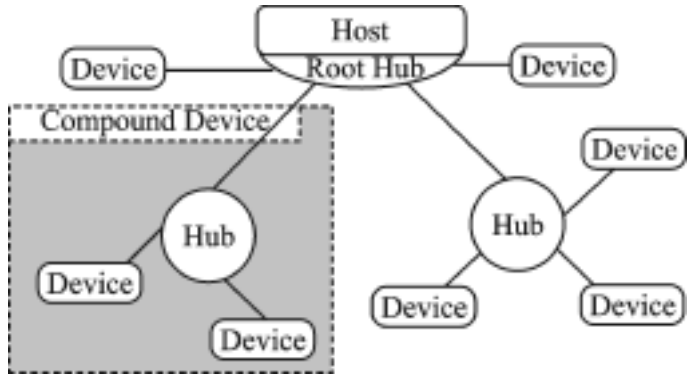


图 6.11-1 USB 的拓扑结构

主控制器由硬件、系统软件、应用软件构成。主控制器提供一个根结点(Root Hub),它可以直接与 USB 设备相连,也可以连接 USB Hub,通过 USB Hub 来扩展接口。主控制器的功能主要有:动态检测外设的接入和拔除,给新接入的设备分配地址和配置参数,管理系统中的数据通信,对系统的电源进行管理等。

USB Hub 用来扩展接口,以使系统连接更多的外设(不超过 127 个)。Hub 也能动态识别 USB 外设的接入,处理属于自己的信号,并将其他的信号放大传输给外设或主机,它还能进行电源的管理和分配。每一个 USB Hub 接入时,主机分配其一个独立的地址。USB Hub 下端可以接 USB 设备,也可以继续接 USB Hub。

USB 设备(Device)是指带有 USB 接口的外部设备,如鼠标、扫描仪、MIC、Speaker 等。它们使用标准的 USB 数据结构与主机进行通信,能识别主机发出的各种命令,并对其作出响应。

四、USB 的逻辑结构

虽然说 USB 系统的拓扑结构是树状结构,一个主机可能要通过几个 USB Hub 才和一个 USB 设备相连接。但对主机来说,它对每个外设的管理和通信都是一样的,好像主机同外设直接连接起来一样。主机将 USB Hub 也当作一个 USB 设备来进行管理。USB 系统的逻辑结构如图 6.11-2 所示。

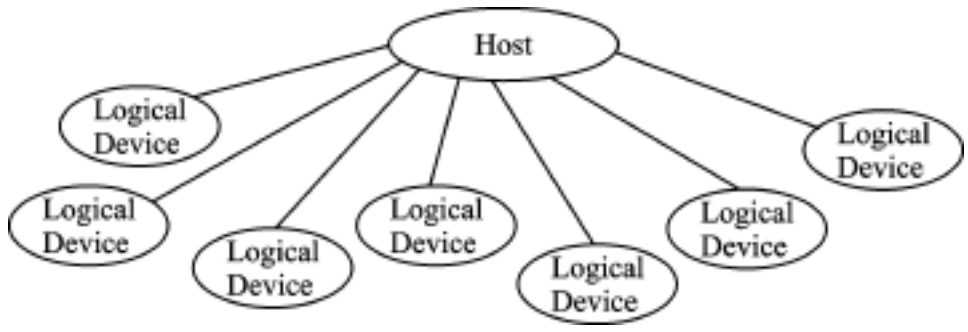


图 6.11-2 USB 系统的逻辑结构

从图 6.11-2 可以看出,一个完整的 USB 系统,在逻辑上由 USB 主机和 USB 设备两部分组成。它们之间的层次划分可以由图 6.11-3 来说明。

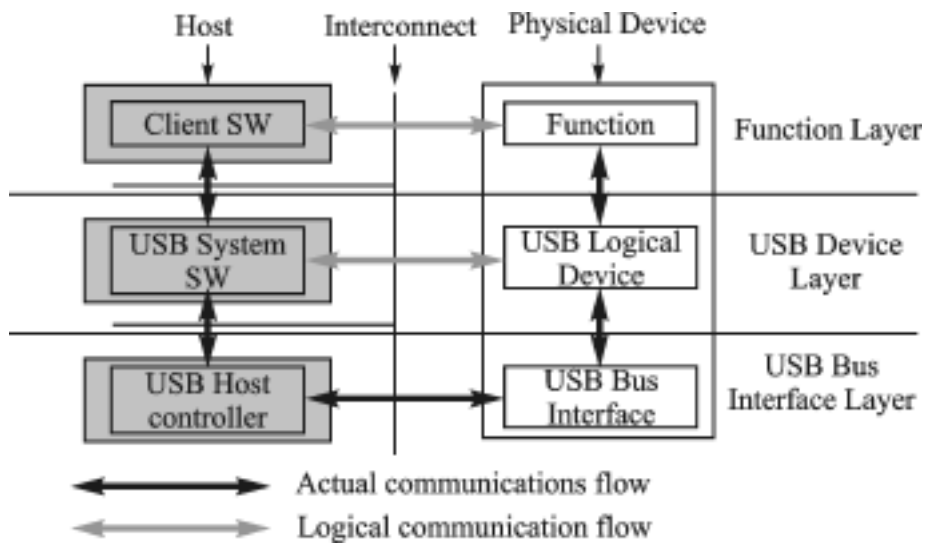


图 6.11-3 USB 系统的层次划分

其中,客户软件通过调用 USB 系统软件中的相应功能,与 USB 设备完成功能级数据交换;USB 系统软件通过与 USB 主机控制器硬件之间的寄存器接口和共用存储器接口,与 USB 设备完成逻辑级数据交换;USB 控制器通过物理连线与 USB 设备完成物理级的信号交换。各层次之间遵守相应的总线规范。

USB 设备是通过 USB 地址与主机来进行通信的。系统支持在 USB 设备和主机之间有一个或多个通道来传输数据和信息,而且它们之间存在一个缺少的通道,通过它,USB 设备和主机就可以建立起通信联系,进行命令和应答的传送。

五、USB 的物理连接

USB 设备通过四线电缆与主机或 USB Hub 相连接。这四根线分别是:Vbus,GND,D+,D-,其中 Vbus 为总线的电源线,GND 为地线,D+和 D-为数据线。USB 利用 D+和 D-线,采用差分信号的传输方式传输串行数据。

USB 设备与主机的连接有两种方式。图 6 .11 - 4 是高速设备的连接。

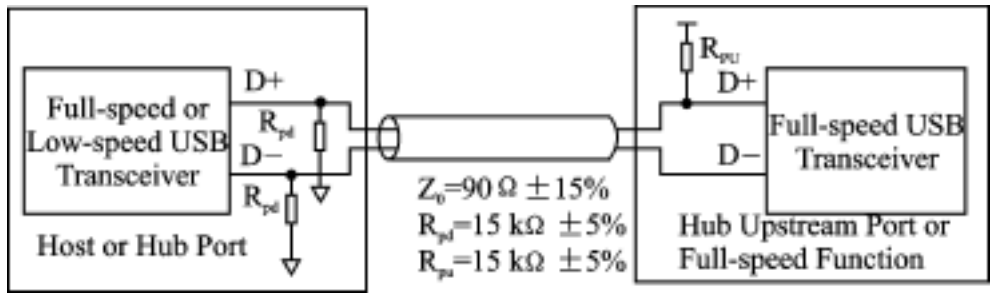


图 6 .11 - 4 USB 的物理连接(高速设备)

USB 主机和 USB Hub 同时支持高速和低速两种传输模式,但 USB 设备只支持其中的一种传输模式。所以在总线的连接上,USB 高速和低速设备是有区别的,以让其系统能够识别其设备的类型。若系统接入的是 USB 高速设备,则在设备的上行端口处,总线的 D + 接上拉电阻(Rpu),如图 6 .11 - 4:若是 USB 低速设备,上拉电阻接在 D - 上。在 USB 主机或 USB Hub 的下行端口,D + 和 D - 都接有下拉电阻(Rpd),而且 Rpd 是 Rpu 的 10 倍。

当 USB 设备没有接入时,总线上 D + 和 D - 都为低电平(此时主机或 Hub 的输出为高阻),称之为 SE0 状态(Single-ended 0)。当 USB 设备接入时,由于 Rpd > Rpu,所以 D + (高速)或 D - (低速)被拉成高电平,这时总线的状态为 J 状态。主机若检测到总线状态从 SE0 到 J 的变化,并且 J 状态持续一定的时间,就认为有 USB 设备接入系统。在 USB 设备与主机不通信时,总线处于 J 状态,所以 J 状态也叫空闲态(Idle State)。当 USB 设备和主机要进行通信时,D + 和 D - 的电平反向,总线的状态变为 K 状态,然后系统以 J,K 态分别代表 0 和 1,传输数据。系统将总线从 J 到 K 的改变定义为一包(Package)数据的开始 SOP(Start Of Package),将持续两个周期的 SE0,加一个周期的 J 看作是一包数据的结束 EOP(End Of Package)。需要注意的是,高速设备和低速设备的 J,K 状态刚好是相反的。

六、USB 的数据结构

USB 系统中,串行数据的传输采用 NRZI(Non Return to Zero Invert)编码。具体说就是,当数据是 1 时,电平不变;当数据是 0 时,电平反向。而且在传输过程中,若遇到 6 个连续的 1,则在这 6 个连续的 1 后插入一个 0。数据的传送是先传低位,再传高位。每包数据的开始是一个同步场,同步场固定为 80H。USB 的数据传输方式如图 6 .11 - 5 所示。

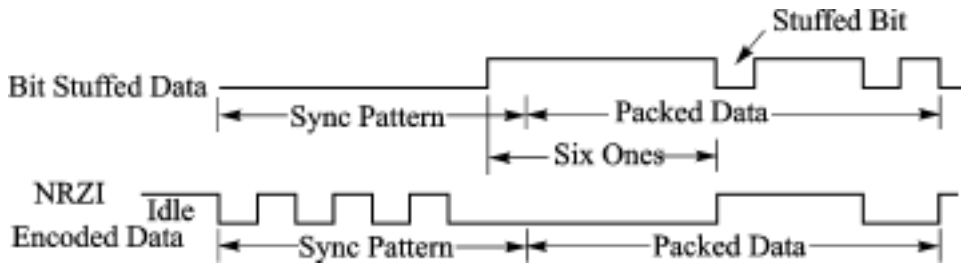


图 6 .11 - 5 USB 的数据传输方式

由于 USB 是通过两根差分信号线来传输数据,因此所有的数据必须按一定的格式来组织。具体来讲,在 USB 系统中,是通过以下 3 个层次来组织数据的:

(1) 帧(Frame)。USB 标准规定每 1 ms 信号为一帧,每一帧数据可以包括不同传输模式的数据包,而每种数据包所占的比例,以及每帧的附加信息,则由 USB 主机中的帧管理器

完成。

(2) 包(Package)。包是构成帧的单位,每一包数据由包开始信息(SOP)起始,直至出现包结束信息(EOP)。包可以分为:① 帧开始包(SOF),用于标志一帧数据的开始;② 信令包,用于完成主-从机之间的信令及应答信号的传输;③ 数据包,用于传输数据;④ 特殊用途的包。每一种包又由不同的场数据、场结构构成。

(3) 场(Field)。场是构成包的单位,每一个包都是由若干个数据场构成的。主要的数据场有以下几种:① 同步场(SYNC),为固定的数据 80H,编码为 0101010100,可为接收方提供同步;② 标识场(PID),用于标识包的类型;③ 地址场(ADDR),用于传送数据的目的地址;④ 端点场(ENDP),用于标明传送数据的目的端点;⑤ 帧序号场,用于标明当前帧的序号;⑥ 校验场(CRC),用于传送 CRC 校验码;⑦ 数据场,用于传输数据。

七、USB 的前景展望

USB 标准推出后,得到了业界的普遍认可,并很快走进市场,特别是 1998 年 9 月 23 日推出 Ver1 .1 版本后,USB 接口得到广泛的应用。如今所有的计算机主板都带有一个或两个 USB 接口,Windows 98 全面支持 USB 标准,各厂商都纷纷将其产品的接口改为 USB 接口;有关 USB 的芯片也层出不穷,如带 USB 接口的单片机,485 - USB 转换器,232 - USB 转换器等等。

由于 USB 的巨大成功,一些著名的大公司纷纷加入 USB 标准制定俱乐部。在 1999 年 10 月,由 Compaq, HP, Intel, Lucent, Microsoft, NEC, Philips 等著名公司参加的规划会议上,推出了 USB 2 .0 标准的草案,2000 年 4 月正式推出了 USB 2 .0 版本。USB 2 .0 完全与 USB 1 .1兼容,而速度是 USB 1 .1 的 40 倍,即可以达到 480 Mbps。接口速度的大幅度提高,将使未来的计算机的功能大大增强,使得计算机可以通过 USB 接口接入宽带 Internet,连接高清晰度的数码相机,连接下一代的高速打印机、扫描仪及高速的外存单元等。使用 USB 2 .0 接口,下载一幅画将从现在的几分钟缩短为几秒钟,USB 的数据从硬盘备份只需几分钟,而今天干同样的事需要花几个小时。

USB 2 .0 标准在 2000 年第一季度刚刚发布,主要系统和外设预期在 2000 年下半年上市,到时,会掀起应用 USB 接口的热潮。

八、结束语

本文介绍了一种得到广泛应用的接口总线 USB。USB 由于速度快、使用方便灵活、易于扩展、支持即插即用、成本较低等一系列优良的特性,正逐步取代传统的接口总线而应用于计算机的各种外设中。特别是 2000 年第一季度推出 USB 2 .0 标准后,USB 接口的应用将会更加广泛。

选自《计算机应用研究》月刊,2001 年第 2 期

6.12 USB2.0 技术概述

中共广州市委办公厅自动化中心(510046) 汪 胜
广州工程总承包集团有限公司(510620) 时亚弘

随着世界范围内更多的外设制造商采用 USB 标准开发产品,这个标准在市场上取得了巨大的成功。事实上所有新的 PC 机在机箱上都有几个 USB 端口,USB 已经成为 Intel 和 Microsoft 所倡导的,使 PC 机易于操作的关键技术之一。1995 年制定 USB 规范的 Compaq, HP, Intel, Microsoft, Lucent 等几家主要的公司,在 2000 年继续制定了 USB2.0,将数据传输速度提高到 480 Mb/s。

一、USB2.0 的动机和目标

USB 的产生源于三个方面的因素:① 将 PC 机与电话相连,计算机和通信技术的出现是信息化时代的基础。不幸的是计算机和通信产业却在各自的领域独自地发展;② 易用性,传统的接口缺乏重新配置的灵活性,这将是其未来使用的致命弱点;③ 端口扩充,外设的增加继续受 PC 机端口的可用数量限制。

制定 USB2.0 的动机主要源于 PC 机和外设的性能日益提高并已可以处理大量数据的事实,需要在 PC 和外设之间进行高性能的连接。为适应这种需要,USB2.0 在原来 USB1.1 定义的 12 Mb/s 和 1.5 Mb/s 的基础上增加了第三个传输速度 480 Mb/s。UB2.0 是 USB1.1 的自然演化,同时保持了对已有外设的所有兼容性。

这个规范定义一个工业标准 USB,描述了总线特性、协议定义、多类型事务、总线管理,以及为适应这个标准而设计的系统和外设的程序接口。USB2.0 的目标是使不同厂家的设备在一个开放的体系结构中互相操作,并给予了系统 OEM 和外设设计者足够空间开发产品。

二、USB2.0 新增特性

图 6.12-1 是一个支持更高速度的 USB2.0 系统。480 Mb/s 高速连接是在根 Hub 和外部 USB2.0 Hub 之间以及外部 USB2.0 Hub 和电视会议摄像机(一个 USB2.0 外设)之间协商完成。所有的其他连接都按 USB1.1 数据速度进行,即 12 Mb/s 和 1.5 Mb/s。任何 USB2.0 Hub 下行端口都支持连接任何速度的 USB 设备。

从用户角度来看,USB2.0 系统和 USB1.1 系统并无差别。但是,USB2.0 具有更高的带宽,可以更多地选择有吸引力且高性能的设备。

从 PC 机制造商的角度来看,USB2.0 将为系统制造商提供最经济的方式连接高性能设备。增加 USB2.0,将不会过多地提高整个系统的成本。在许多未来的 PC 机上仅 USB 连接器是必须的。

从外设制造商的角度来看,今天的 USB 设备将完全兼容地在 USB2.0 系统上运行。另外,可以设计更高数据传输速度的 USB2.0 外设,例如,下一代高速、高清晰度打印机和扫描仪

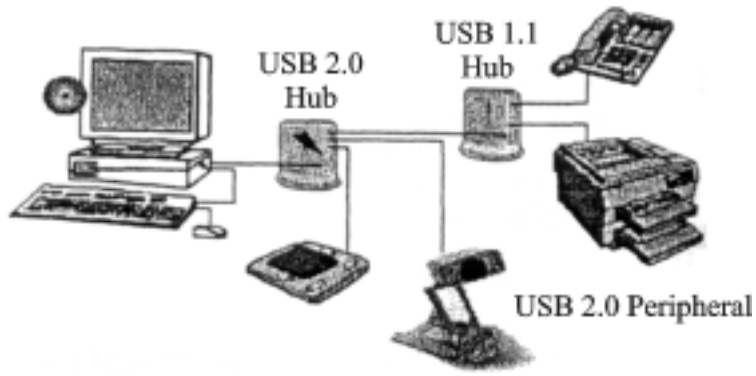


图 6.12-1 USB2.0 系统配置的实例

设备,高密度存储设备如 R/WDVD 和高容量 CDROM 自动点唱机。USB2.0 的附加性能将扩大 USB 外设的市场份额。

三、USB 体系结构

USB 是一个快速的、双向的、同步的、低成本的、动态的串行连接接口,支持主机和并发存取外设之间的数据交换。所连的外设通过主机调度和基于标识(Token)的协议共享 USB 带宽。总线允许外设在主机与其他外设操作时连接、配置、使用和拆除。

USB 系统从三个方面进行描述:(1) USB 互连;(2) USB 设备;(3) USB 主机。其中,USB 互连(Interconnect)指的是 USB 设备和主机之间的连接和通信方式。包括① 总线拓扑;② 层间的关系;③ 数据流动模式;④ USB 调度。

1.总线拓扑

USB 将 USB 主机和设备进行连接。USB 物理互连是一个层次的星型拓扑结构。Hub 是每一个星型结构的中心。每一个线段是主机和 Hub(或 Function),Hub 和另一个 Hub(或 Function)之间的点到点连接。图 6.12-2 描述了这种 USB 的拓扑。

由于 Hub 和电缆的传播时间限制,系统最大只允许七层(含根层),仅 Function 可以在第七层上。复合设备(Compound Device,见图 6.12-2)占用两层,因此它不能在第七层和系统连接。

2.USB 主机

在任何 USB 系统中仅有一台主机(Host)。主机系统中 USB 接口称为主机控制器(Host Controller)。主机控制器可以由硬件、固件或软件结合实现。根 Hub 集成在主机系统中,以提供一个和多个连接点。

USB 主机通过主机控制器与 USB 设备进行交互。主机负责:① 检测 USB 设备的连接和拆除;② 管理主机和 USB 设备之间的控制流;③ 管理主机和 USB 设备之间的数据流;④ 收集状态和活动的统计;⑤ 为连接的 USB 设备提供电源。

主机上的 USB 系统软件用于管理 USB 设备和基于主机的设备软件之间的交互操作。USB 系统软件和设备软件之间的交互操作包括设备的枚举(Enumeration)和配置、同步数据传送、异步数据传送、电源管理以及设备和总线的管理信息等五个方面。

3.USB 设备

所有的 USB 设备都是通过 USB 地址来存取的,这个地址在连接或枚举时分配。每一个设备都支持多个管道,主机通过这些管道与设备进行通信。其中 USB 控制管道(Control

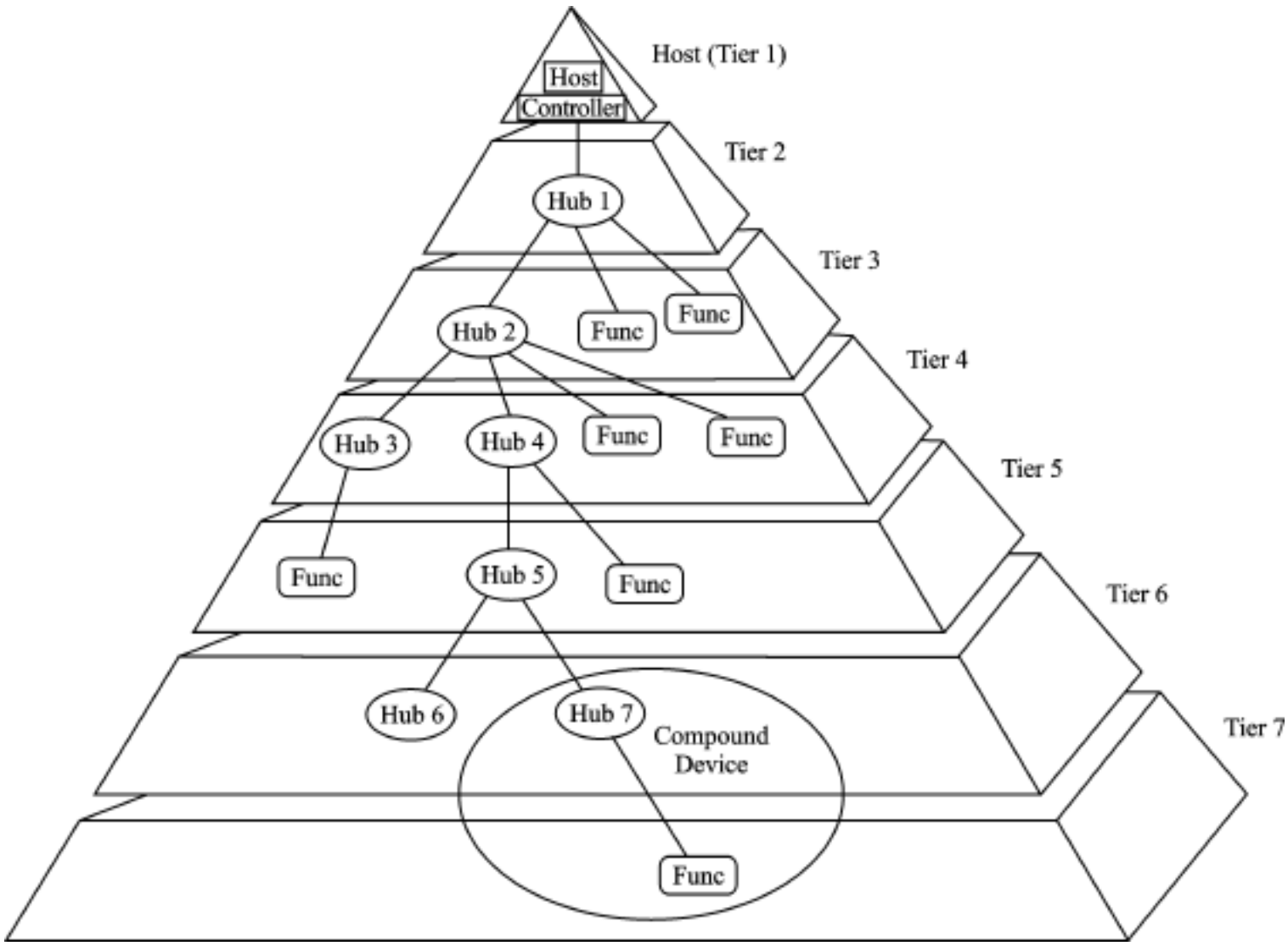


图 6.12-2 USB2.0 总线拓扑

(即一个连接到端点 0 上特定设计的管道)提供一个共同存取信息的机制。

存取的信息包括:① 所有 USB 设备通用的信息,包括制造商标志、电源管理能力等;②设备类别信息,信息的内容取决于 USB 设备所属的类别(USB 设备可以被分为 Hub、人性化接口、成像设备和大容量存储设备等类别);③ USB 制造商自定义信息。

每一个 USB 设备都包括控制和状态信息。USB 设备分为两大类: Hub 和 Function。Hub 为 USB 提供附加的 USB 连接点,Function 为主机提供附加的功能,即扩充主机的功能。

(1) Hub

Hub 是 USB 即插即用体系结构中的关键部分。图 6.12-3 显示了一个常见的 Hub 结构。从用户角度来说,Hub 简化了 USB 的连接,并提供了相对低成本和低复杂度的健壮性。Hub 是配线的连接中心,实现 USB 的多连接特性。连接点称为端口(Port)。每个 Hub 将信息从一个连接点转换到多个连接点。这种体系结构支持多 Hub 的级联。Hub 的上行端口连

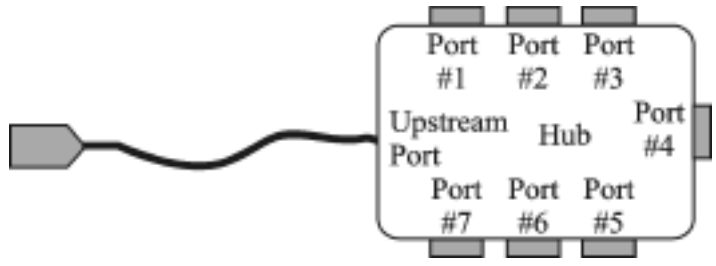


图 6.12-3 一个典型的 Hub

接朝向主机的 Hub。每个 Hub 的下行端口可以连接另一个 Hub 或 Function。Hub 可以检测每个下行端口的连接和拆除,并分配电源给下行设备。每一个下行端口可以独立连接到高速、全速或低速设备上。

一个 USB2.0 Hub 由三个部分组成: Hub 控制器, Hub 转发器,事务解释器。Hub 转发器(Repeater)在上行端口和下行端口之间进行协议控制交换,硬件上支持复位和挂起/恢复信号。Hub 控制器(Controller)提供与主机的通信。Hub 特有的状态和控制命令允许主机配置 Hub 并监视控制它的端口。Hub 中的事务解释器(Transaction Translator)提供支持全速/低速设备的机制,即在主机和高速 Hub 之间传送所有设备的数据。

(2) Function

Function 是一个 USB 设备,它可以在总线上传送或接收数据或控制信息。一个 Function 通常由单独的外设实现,并由一根电缆线连接到 Hub 的端口。但是,一个物理包可以通过一根 USB 电缆实现多个 Function 和一个内嵌的 Hub,这称为复合设备(Compound Device)。一个复合设备在主机看来是一个 Hub,包括一个或多个不可移动的 USB 设备。

每一个 Function 包含一些配置信息,用于描述它的性能和资源需求。在使用 Function 之前,主机必须对它进行配置。配置包括分配 USB 带宽和选择 Function 特有的配置选项。Function 的例子包括:① 人性化接口设备,如鼠标,键盘,写字板或游戏操纵杆;② 成像设备,如扫描仪,打印机或摄像机;③ 大容量存储设备,如 CDROM 驱动器,软盘驱动器或 DVD 驱动器。

4. 物理接口

USB 的物理接口包括电气和机械两方面规范。USB2.0 有三种数据传送速率:① USB 高速(High Speed),480 Mb/s;② USB 全速(Full Speed),12 Mb/s;③ USB 低速(Low Speed),1.5 Mb/s。

电气方面,USB 是通过一条含 4 根导线(一对信号线和一对电源线)的电缆来传输信号和电源的,如图 6.12-4 所示。在每个点到点段中,通过两根导线(D+和 D-)传送信号。在主机控制器和 Hub 之间可以高速传送全速和低速设备的数据,而在 Hub 和设备之间全速和低速传送数据。这种性能减少了全速或低速设备对高速设备带宽的影响。定义低速模式是为了支持少量的低带宽设备如鼠标。USB 采用位填充 NRZI 编码方案,每个数据包之前是 SYNC 域,用于同步位时钟。电缆也有 VBUS 和 GND 线,用于向设备传送电源。VBUS 通常是 +5 V电压。

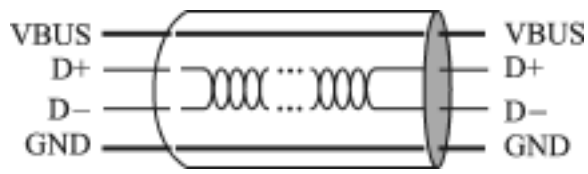


图 6.12-4 USB 电缆

机械方面,所有的设备都有上行连接。上行和下行连接器在机械上不可以互换使用,这样消除了在 Hub 上非法的回路连接。

5. 电 源

电源分配,主机或 Hub 通过电缆向那些直接连接到它的设备提供电源。每一个 USB 设备也可以由它自己的电源供应。完全依赖于电缆供应电源的 USB 设备被称为总线电源设备

(Bus-powered Device)。相应的,那些本身有电源供应的设备称为自供电源设备(Selfpowered Device)。

电源管理,USB 主机可以有一个独立于 USB 的电源管理系统。USB 系统软件与主机电源管理系统进行交互,以处理系统电源事件,例如,挂起或恢复。另外,USB 设备通常也实现附加的电源管理能力。

6.总线协议

大部分总线事务最多传送三个数据包。当基于调度基础的主机控制器发送描述事务类型和方向、USB 设备地址和端点号的数据包时,表明每个事务的开始。这个数据包被称为标识包(Token Packet)。USB 设备通过解释相应的地址域来决定是否接受。对于某个给定事务来说,数据既可以从主机传送到设备,也可以从设备传送到主机,数据的传送方向在标识包中说明,事务通常用握手包(Handshake Packet)响应以表明传送是否成功。有些主机控制器和 Hub 之间的总线事务包括 4 个包的传送,这类事务用于管理主机和全/低速设备之间的传送。

主机与设备上的端点之间的 USB 数据传送模式称为管道(Pipe)。有两种类型的管道:流(Stream)和消息(Message)。流数据是无结构的,而消息数据是有结构的。另外,管道是数据带宽、传输服务类型和端点特性(如方向性和缓冲器大小)的结合。大部分管道在 USB 设备配置时即存在。一个消息管道(即默认的控制管道)通常在一个设备供电时即存在,这是为了可以存取设备配置状态和控制信息。

事务调度允许为某些流管道进行流量控制。在硬件方面,通过 NAK 握手控制数据传输速度,防止缓冲区出现过速和过缓的情况。

7.健壮性

有许多 USB 的特性用于实现健壮性:① 不同驱动程序、接收者的信号集成;② 控制和数据域上独立的 CRC 保护;③ 连接和拆除的检测,资源的系统级配置;④ 协议中的自恢复,使用超时机制确认丢失或被破坏包;⑤ 流数据的流量控制,确保同步和硬件缓冲管理;⑥ 数据和控制管道,确保 Function 之间不利操作的隔离。

8.系统配置

由于 USB 设备在任何时候都可能连接和拆除,USB 系统必须支持在物理总线拓扑上的动态变化。

USB 设备的连接。所有的 USB 设备是通过 Hub 上的端口连接到 USB 的。Hub 的状态位用于向主机报告某一个端口上的 USB 设备的连接和拆除状态。在连接的情况下,主机使端口有效,并通过设备的控制管道访问 USB 设备,主机分配一个惟一的 USB 地址给这台设备,主机用分配的 USB 地址和端点号 0 来构建 USB 设备控制管道的端,然而判断这个新连接的 USB 设备是一个 Hub 还是一个 Function。如果所连的 USB 设备是一个 Hub,并且 USB 设备已连到它的端口上,那么每一个所连的 USB 设备也执行上述过程。如果所连的设备是一个 Function,那么相应 Function 的主机软件将处理连接的通知信息。

USB 设备的拆除。当一个 USB 设备已从 Hub 端口拆除时,Hub 将使此端口无效,并向主机报告,主机相应的 USB 系统软件就进行处理。如果被拆除的 USB 设备是一个 Hub,USB 系统软件必须处理所有的先前连接到这个 Hub 上的 USB 设备的拆除。

总线枚举(Bus Enumeration)是标识和分配惟一地址给连接总线的设备的活动。由于 USB 允许 USB 设备在任何时候连接或从 USB 上拆除,USB 枚举是 USB 系统软件中实时的

活动。另外,USB 的总线枚举也包括拆除的检测和处理。

9 . 数据流类型

管道之间的数据流动是相互独立的,一个给定的 USB 设备可以有多个管道,例如,一个管道用于数据传送,另一个端点管道用于数据接收。

USB 体系结构支持 4 种最基本类型的数据传送(一个管道仅支持一种类型):① 控制传送　主要用于在连接时刻配置一个端口;② 批量数据传送　用于相对大量的数据量的产生和消耗,批量数据传送是有序的、可靠的(通过错误检测和少量的重发)、占用的带宽可变,例如用于打印机和扫描仪;③ 中断数据传送　用于实时但可靠的数据传送,例如回显或反馈响应;④ 同步数据传送　占用大量的 USB 带宽(也称为流实时传送),同步数据的实时传送通过数据流上牺牲潜在的短暂的丢失来确保,同步传送是持续的、实时的、速率稳定的,同步数据的典型事例是声音。

四、总　结

USB2 .0 规范是 1 .1 版本自然演化,并支持范围更广的产品。许多制造商正转向 USB,是因为被其包含的多 PC 平台和易使用性所吸引。越来越多有创新的外设将充分利用 USB,这将进一步提高 USB 产品的效用。

参 考 文 献

1　USB 2 .0 Specification[Z] . Compaq, HP, Intel, Lucent, Microsoft, NEC, Philips,2000

2　USB 1 .0 How Many Ports and How To Get There ? [Z] Intel Corp, 1999

选自《计算机应用研究》月刊,2001 年第 3 期

6.13 带通用串行总线 USB 接口的单片机 EZ-USB

南京东南大学电气工程系(210096) 戴红梅 胡仁杰

美国 Cypress 公司是一家从事 USB 接口芯片和 USB 单片机开发和生产的公司。Cypress 最新推出的带智能 USB 接口的单片机 EZ-USB,极大地降低了 USB 外设的开发难度,为 PC 外设的制造商提供了一个性能优良,价格较低的设计方案。以下简要介绍该单片机的结构、特点及使用方法。

一、芯片结构组成

EZ-USB 的内部结构如图 6.13-1 所示。它集成了一个加强的 8051、一个智能 USB 引擎、一个 USB 收发模块、RAM、一个锁相环,对外有 24 个 I/O、16 位的地址线、8 位的数据线、一个 I^2C 口和一对 USB 口(D+、D-)。EZ-USB 遵从 USB 1.0 规范(12 Mbps),支持远程唤醒。

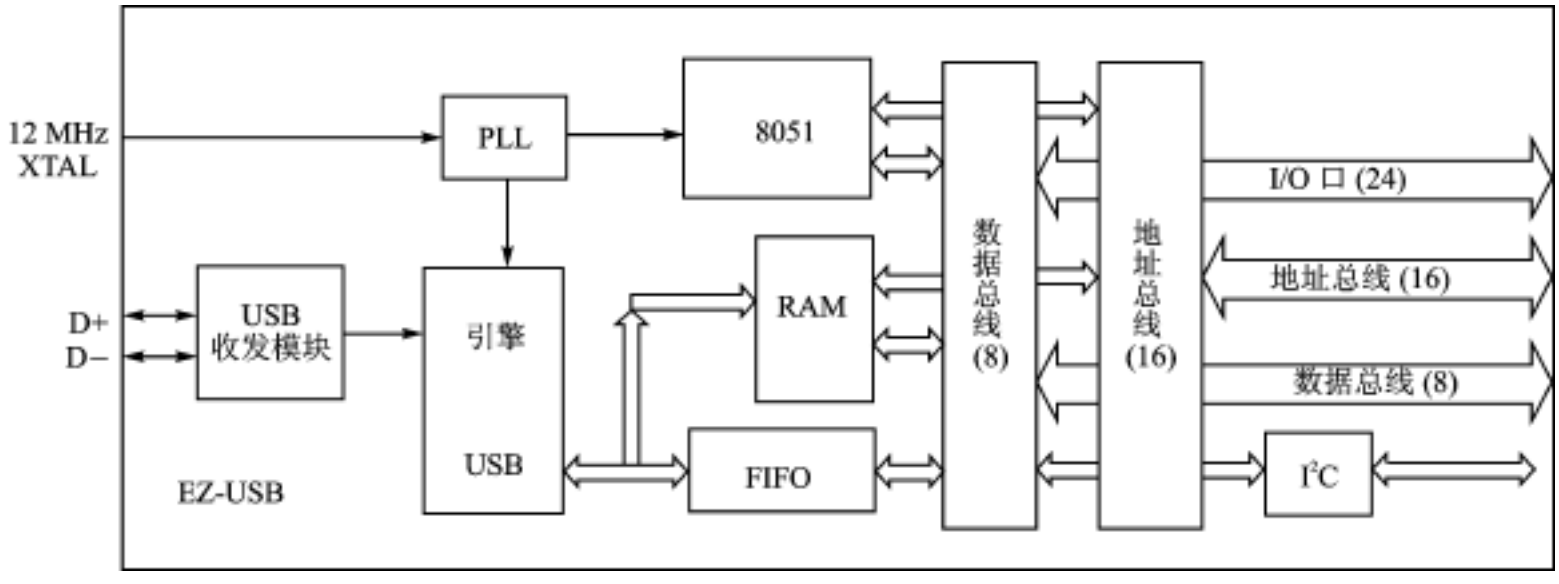


图 6.13-1 EZ-USB 的结构

二、特 性

1. 加强 8051 内核

性能可达到标准 8051 的 5～10 倍,与标准 8051 的指令完全兼容。

2. 高度集成

传统的 USB 外设的硬件设计通常包括非易失性存储器(如 EPROM、EEPROM、FLASH ROM)、微处理器、RAM、SIE(串行接口引擎)、DMA,EZ-USB 将上述多个模块集成在一粒芯片中,从而减少了各芯片接口部分时序配合的麻烦。

3. 智能 USB 接口引擎

智能的接口引擎可以代替 USB 外设开发者完成 USB 协议中规定的 80%～90%的通信

工作,这使得开发者不需要深入了解 USB 的低级协议即可顺利的开发出所需要的 USB 外设。

4. 软配置

如图 6.13-2,外设未通过 USB 接口接到 PC 机之前,外设上的固件存储在 PC 上;一旦外设接到 PC 机上,PC 先询问该外设是"谁"(即读设备描述符),然后将该外设的固件下载到 EZ-USB的 RAM 中,这个过程叫做再枚举。

这个特性给 USB 外设开发者带来很多方便,比如开发过程中当固件需要修改时,可以在 PC 机上修改好后,下载到 EZ-USB,从而省去了烧片子的麻烦。当开发好的外设已经批量售出,开发商需要向用户提供固件的升级版本时,就可以把代码在网上发布,各地的用户自行下载到本地 PC 机上,即完成了升级,而无需用户有硬件开发的专业技能。

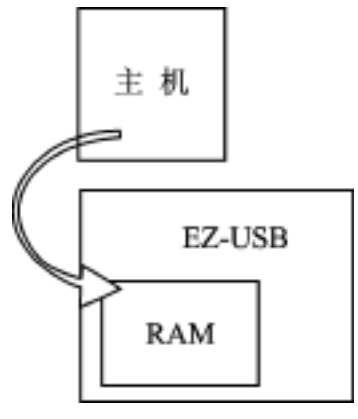


图 6.13-2 从 Host PC 下载 firmware

5. 易用的软件开发工具

固件可独立于驱动程序被测试,驱动程序和固件的开发和调试相互独立,可加快开发的速度(详见第三部分)。

三、软件设计

EZ-USB 的软件包中包括一个通用驱动程序(General Purpose Driver 简称 GPD),一个固件代码框架及一些例子代码。以下将分别介绍 GPD 和代码框架。

1. PC 上的应用程序

EZ-USB 的开发工具提供了通用设备驱动程序,PC 机上的应用程序以不同的参数调用 GPD 的函数,实现对外设的控制,其中,最重要的函数是 DeviceIoControl(),它的功能是应用程序与驱动程序之间数据的交换。

```
BOOL DeviceIoControl(
  HANDLE hDevice,
  DWORD dwIoControlCode,
  LPVOID lpInBuffer,
  DWORD nInBufferSize,
  LPVOID lpOutBuffer,
  DWORD nOutBufferSize,
  LPDWORD lpBytesReturned,
  LPOVERLAPPED lpOverlapped
};
```

(1) hDevice: 调用 CreateFile()函数打开外设驱动程序时,返回的外设句柄。

(2) dwIoControlCode: I/O 控制代码,可以为 IOCTL_Ezusb_GET_DEVICE_DESCRIPTOR(读设备描述符)、IOCTL_EZUSB_BULK_READ(批量读操作)、IOCTL_EZUSB_BULK_WRITE(批量写操作)、IOCTL_EZUSB_ISO_READ(同步读操作)、IOCTL_EZUSB_ISO_WRITE(同步写操作)等,限于篇幅,不再赘述。

(3) IpInBuffer: 指向输入缓冲区的指针,存放 PC 机传给外设的数据。

(4) nInBufferSize: 输入缓冲区的大小,以字节记。

(5) pOutBuffer: 指向输出缓冲区的指针,存放 PC 机从外设接收的数据。

(6) nOutBufferSize: 输出缓冲区的大小,以字节记。

(7) IpBytesReturned: 该操作实际返回的字节数。

(8) IpOverlapped: 指向用于异步操作的结构的指针。

下面是一段 PC 机读外设的设备描述符的程序,其中 struct_USB_DEVICE_DESCRIPTOR 定义了设备描述符的数据结构,DumpDeviceDescriptor( )函数可将读到的设备描述符显示在屏幕上。

```
#include< windows .h >
#include< stdio .h >
#include< stdlib .h >
#include< winioctl .h >
#include" ..\ ..\ ..\drivers\ezusbdrv\ezusbsys .h"
// define data structure of device discriptor
typedef struct_USB_DEVICE_DESCRIPTOR {
   UCHAR bLength;
   UCHAR bDescriptorType;
   …
   UCHAR bNumConfigurations;
} USB_DEVICE_DESCRIPTOR, * PUSB_DEVICE_DESCRIPTOR;
void DumpDeviceDescriptor  (PUSB_DEVICE_DESCRIPTOR deviceDescriptor);
void_cdecl main()
{
   HANDLE handle = NULL;
   BOOLEAN success;
   ULONG nBytes;
   PUCHAR buffer = NULL;
   buffer = (PUCHAR)malloc(sizeof(USB_DEVICE_DESCRIPTOR));
  // get a handle to our device
handle =  reateFile("\\\\ .\\ezusb-0",
          GENERIC_WRITF,
          FILE_SHARE_WRITE,
          NULL,
          OPEN_EXISTING,
          0,
          NULL);
  if(handle = = INVALID_HANDLE_VALUE)
  {
     printf("Failed to open driver\n");
     return;
     }
  else
```

```
{
   printf("Opened successfully .\n");
}

// fetch device descriptor and put it in buffer
   success =  eviceIoControl(handle,
              IOCTL_Ezusb_GET_DEVICE_DESCRIPTOR,
              NULL,
              0,
              buffer,
              sizeof(USB_DEVICE_DESCRIPTOR),
              &nBytes,
              NULL);
   if(success)
   {// display descriptor
      DumpDeviceDescriptor((PUSB_DEVICE_DESCRIPTOR) buffer);
   }
   else
   {
      printf("IOCTL Failed\n");
      }
      CloseHandle(handle);
}
```

2. USB 外设上的固件设计

开发工具中的框架是自动生成的,USB 外设开发者根据外设功能具体要求,选择感兴趣的函数逐一填写函数体。固件的框架如图 6.13-3 所示。开始是一段初始化程序,当外设检测到一个 Setup Packet,框架启动调度程序,由调度程序按顺序反复执行以下三个任务。

(1) 在 TD_Poll( )中调用实现外设主要功能的用户子程序(开发者必须填写的部分)。

(2) 决定设备是否有未处理完的任务。如果有,则解析收到的命令,然后给主机相应的回复。

(3) 决定是否 USB 核心报告了 USB 悬挂事件。如果是,调度程序调用用户子程序 TD_Suspend( ),如返回 TRUE,处理器挂起,当检测到唤醒事件程序继续运行,如返回 FALSE,程序继续运行。

下面是一段从 EP2 接收数据,然后从 EP2 转发回主机的程序中的片断,说明了如何在 TD_Init( )中配置各端点,如何在 TD_Poll( )中实现外设对 USB 口数据的收发及处理。

```
void TD_Init(void)
{
     //  Enable endpoint 2 in, and endpoint 2 out
      IN07VAL = bmEP2;
OUT07VAL = bmEP2;
//  Enable double buffering on endpoint 2 in, and endpoint 2 out
```

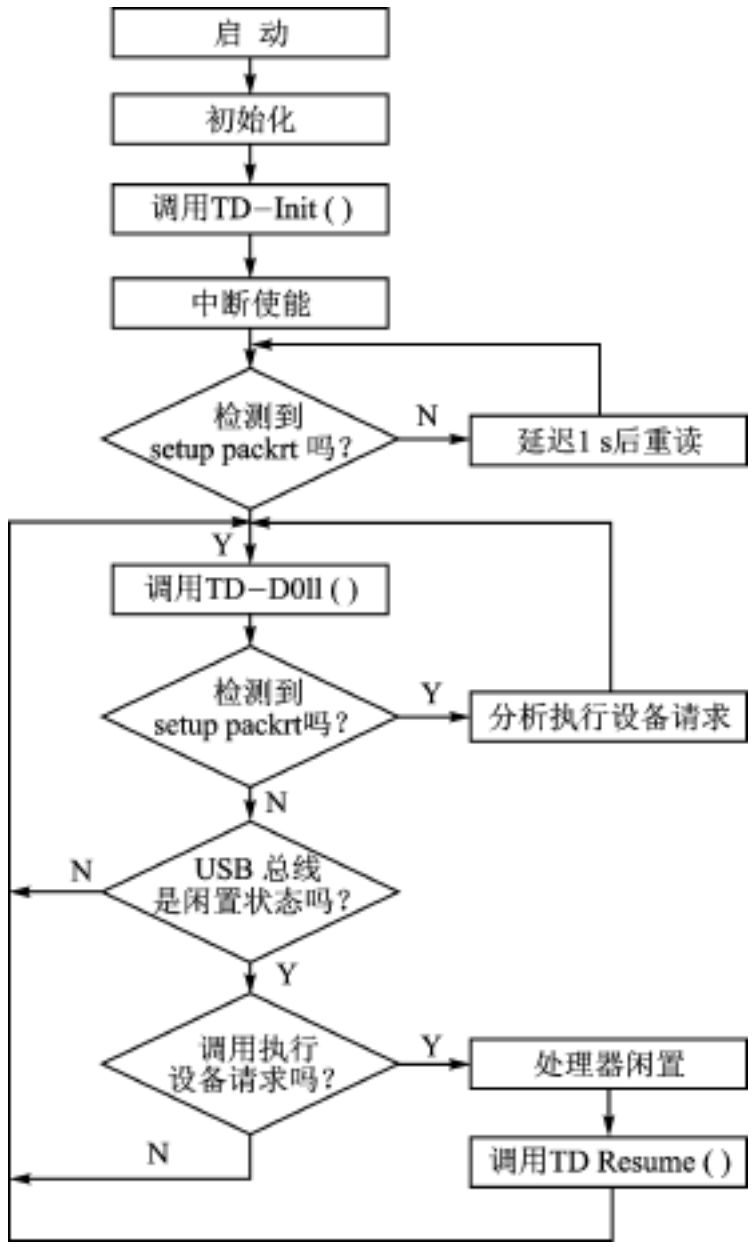


图 6 .13 - 3 固件框架

```
    USBPAIR = 0x09;
    // Arm Endpoint 2 out to recieve data
    EPIO[OUT2BUF_ID] .bytes = 0;
    // Setup breakpoint to trigger on TD_Poll( )
    BPADDR = (WORD)TD_Poll;
    USBBAV| = bmBPEN;          // Enable the breakpoint
    USBBAV& = ~bmBPPULSE;
}
void TD_Poll(void)      // Called repeatedly while the device is idle
{
    BYTE count, i;
    if(     (EPIO[OUT2BUF_ID] .cntrl & bmEPBUSY)
                // Is there something in the OUT2BUF buffer,
        if( !(EPIO[IN2BUF_ID] .cntrl&bmEPBUSY)
                            // Is the IN2BUF available,
        {
                count = EPIO [OUT2BUF_ID] .bytes;          /
Then looback the data
```

```
        for(i=0;i<count;++i)
            IN2BUF[i]=OUT2BUF[i];
        EPIO[OUT2BUF_ID].bytes=0;
        EPIO[IN2BUF_ID].bytes=count;
    }
}
```

四、典型应用

可以用 EZ-USB 开发一个带 USB 接口的数据采集、控制器,如图 6.13-4 所示。

图 6.13-4 带 USB 接口的数据采集控制器

五、结束语

开发智能设备时,单片机的选择至关重要不但要考虑单片机本身的性能,还要考虑其开发系统的易用性如何,固件是否容易维护等。EZ-USB 具有高集成度,Windows 界面的开发系统,C 风格的代码,自动生成的程序调度机制等特点,大大减轻了开发人员的工作量。其软配置功能使开发灵活,产品风险小。可见,EZ-USB 为 USB 外设开发人员提供了一个好方案。

参 考 文 献

1 Universal Serial Bus Specification, Revision 1.0, USB Implementers Forum, January 15, 1996

2 http://www.cypress.com

3 http://www.anchor.com

4 王云飞.用 MC68H05JB4 开发 USB 外设[J].电子技术应用,2000(5)

选自《电测与仪表》月刊,2001 年第 3 期

6.14 嵌入式处理器中的慢总线技术应用

上海交通大学金桥网络工程中心(200030)
梁阿磊 赵玉源 李 琪 白英彩

传统的总线结构成了整个系统的“瓶颈”,在一些系统设计中已采用 PCI、VME 等先进的总线技术,但在小系统的应用中由于设计复杂度和成本考虑,仍然要面对这个瓶颈。或许正是这个原因,出现了许多嵌入式的微控制器,这些微控制器的内部嵌入了存储器及 I/O 控制单元,不再提供传统的外部总线接口。但应用中就出现扩展性问题:无法外接总线接口的外围单元(例如:无法外部扩展存储器芯片)。

一、接口问题

对于不提供这些总线接口的嵌入式处理器,根本无法从硬件上解决扩展外接的问题。惟一的选择就是更换提供这种接口的处理器(传统的或嵌入的),而具有总线接口的嵌入式处理器一般是为了兼容传统接口设计的,成本较高、引脚增多(失去了嵌入式的意义)。

1.软件解决思路

软件方法可以解决一般嵌入式的接口问题。嵌入式处理器一般提供很多 I/O 接口,如果 I/O 引脚能产生总线的操作时序,那么功能上就和传统的硬件总线接口一样了。

本文以 PIC16C54(以下简称 P)与 MT8880(以下简称 MT)的接口设计阐述这个想法:MT8880 的硬件访问接口时序是符合 Motorola 处理器的总线协议的,这与 Intel 处理器和大多数的嵌入式处理器都不兼容,所以慢总线技术也适合于传统处理器面临的一些接口问题(例如 Intel 处理器与 Motorola 接口标准的器件的兼容问题)。

2.双方的接口描述

P 提供了两组普通 I/O 接口:RA0,RB0～RB6。

MT 提供的是 Motorola 总线接口:系统时钟 Φ,读写信号 R/W,片选信号 CS,地址线 A 和数据总线 D0～D3。

物理上双方的引脚连接如图 6.14-1 所示。逻辑上通过 P 上的软件控制 P 的 I/O 引脚的输入/输出状态,模拟 Motorola 处理器的相应引脚的时序状态。

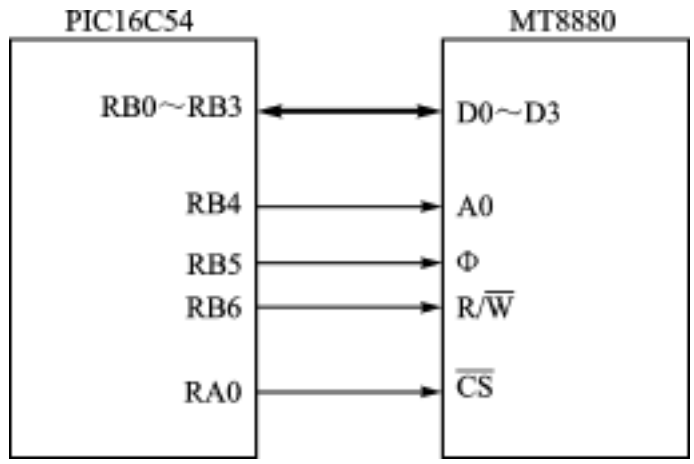


图 6.14-1 PIC16C54 和 MT8880 的接口

二、状态机的时序描述

总线的时序从根本上讲就是总线的状态转移,通过总线连接的双方遵从共同的协议完成各自的状态转移,对方提供的信号是自己状态转移的条件。

图 6 .14 - 2 是 P 读访问 MT 时的接口时序中定义了 6 个状态,每个状态表示的是接口引脚当前的电平状态的总和。状态之间的转移有一定的延时规定,转移条件(又称事件)包括启动、延时和对方的应答信号(但在本例中不要求对方应答)。

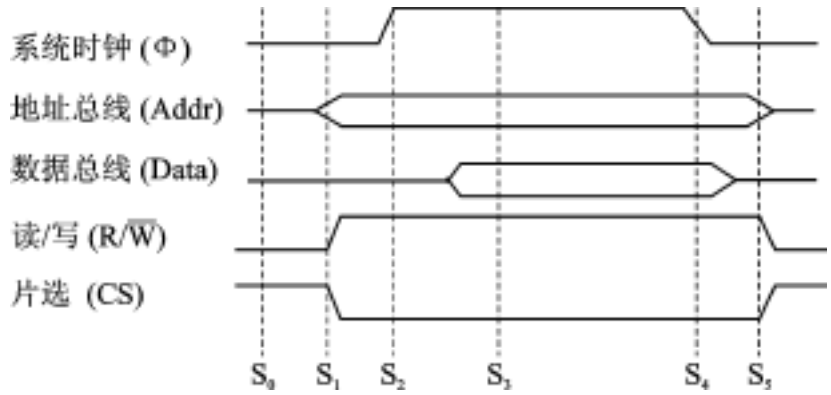


图 6 .14 - 2　MT8880 读时序

P(PIC16C54)的接口时序可以用状态机 M 来描述:

$$M = (S, \Sigma, f, s_0, Z)$$

其中:① S 是一个有限集,是所有状态的集合,这里 $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$;② Σ 是事件的集合,这里 $\Sigma = \{d, s\}$, d 表示延时, s 表示启动;③ f 是从 SxΣ 到 S 的映射, $f(s_i, e) = s_k, i, k = 0 \ldots 5, e = d, s$;④ $s_0 \in S$,是惟一的初始态;⑤ Z 是终止态集合,这里 $Z = \{s_0\}$。

其中 f 为:

$f(s_0, s) = s_1$　$f(s_1, d) = s_2$　$f(s_2, d) = s_3$

$f(s_3, d) = s_4$　$f(s_4, d) = s_5$　$f(s_5, d) = s_0$

对应的状态矩阵表如表 6 .14 - 1 所列。

表 6 .14 - 1　状态转换矩阵

状态 s\事件 e	s	d
S_0	S_1	S_0
S_1	/	S_2
S_2	/	S_3
S_3	/	S_4
S_4	/	S_5
S_5	/	S_0

三、软件实现机制

实现状态机的方法很多:in - line 直接写代码、API 接口、组态等,为了编写一个可移植的通用代码,采用组态的状态栈机制实现上面所描述的状态机 M。状态栈的内容是根据状态机 M 编制的,针对不同的外围接口可以组态地添加或修改状态参数表(M 的数据结构)。当开始

对特定接口进行特定操作时,将参数表的对应项压入栈。栈内容结构的状态号对应了 M 中的编号,其他部分表示了当前引脚的输入/ 输出值以及要延时的时间长度。栈内容的出栈顺序表示了状态的转变顺序,与编号无关(本例中恰巧是顺序的),根据应用需求存放的顺序可以是任意的或重复的。栈顶的内容反映了当前的执行状况。

1 .状态栈的结构

状态号	RA0(CS)	RB6/ 5/ 4(控制信号)	RB3～0(数据)	延时值

2 .P 读访问 MT 数据寄存器时的栈编排情况

栈的内容实际上是对应特定操作的 M 的数据结构。

3 .不同操作栈也不同

P 对 MT 的操作不同时,栈的顺序、个数或内容是不同的。例如读数据寄存器和写数据寄存器时的 RB5(R/ 瑜)的值是不同的,访问数据寄存器和控制寄存器时 RA0(A)的值是不同的。具体应用中,根据被访问外围器件的具体特性(访问时序、访问速度)可以调整栈的结构和内容。

4 .后台执行和批处理

为了尽可能减少有关栈操作的占用时间(由于弹栈的连续两次操作之间有一定的延时要求),提高处理器的响应速度,栈内容的执行过程被设计为后台执行方式。同时,栈机制还可以实现多个操作的批处理。多个读写操作的状态机内容可以同时放在一个栈中(见表 6 .14 - 2),操作被执行的顺序是后进先出(LIFO)的,注意上一次操作的最后一项的延时值应该足够大(栈底一般为 0)。

表 6 .14 - 2 栈的内容

状态号	RA0	RB6/ 5/ 4	RB3～0	延时值
本次操作的内容				
S_1	0	101	X	100
S_2	0	111	X	500
S_3	0	111	读数据	50
S_4	0	100	X	50
S_5	1	000	X	1000
下一次操作的内容				
状态号	RA0	RB6/ 5/ 4	RB3～0	延时值

其中:状态号表示状态的编号(S_0 到 S_5);RA0,RB6/ 5/ 4 表示这些引脚当前的状态值;RB3～0 作为数据总线表示当前读入或写出的数据内容;延时值表示本栈和下一个栈内容被弹出栈之前的时间间隔。

与栈操作有关的线程包括栈编排线程(Tw)、栈执行线程(Ts)和定时器(Tt),见图 6 .14 - 3。

(1) 栈编排线程(Tw)负责将对应状态机 M 的参数压入栈。根据应用线程的要求查询相应接口的 M 参数表项,寻找对应操作的参数表项。

(2) 栈执行线程(Ts)负责输出/ 输入栈顶的相关内容,如将 RA0 的内容赋值给 RA0 引脚,并且待所有当前操作完成后设定定时器(根据延时值),然后挂起。

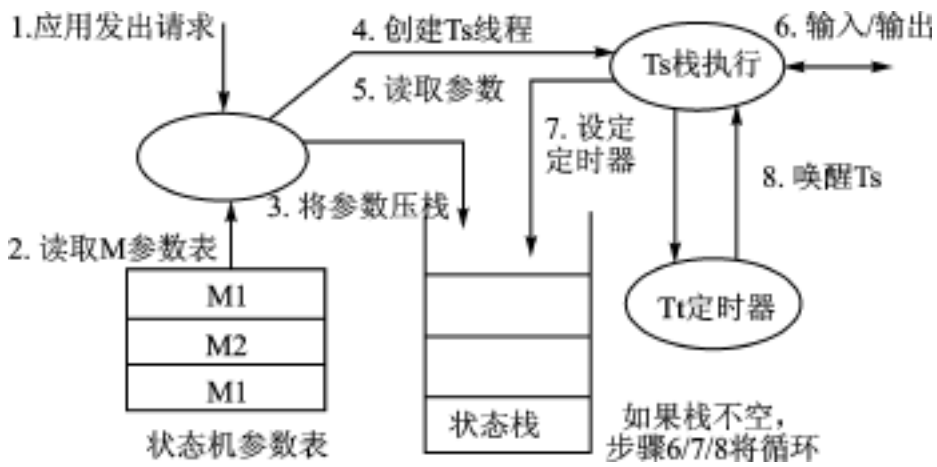


图 6 .14 - 3　有关栈操作的线程和步骤

(3) 定时器(Tt)实际上相当于一个调度程序,每当定时器溢出时,负责弹栈并激活“ 栈执行线程 ”。在栈底时,通常定时器的值被赋予 0,此时不再激活“ 栈执行线程 ”。

四、结束语

本文用状态机描述并解决了嵌入式处理器中常遇到的接口扩展问题,针对总线接口提出了“ 慢总线 ”的想法。状态机是描述时序问题的有力工具,可以应用于各种规模的现实问题的仿真。本文是在微控制器 PIC16C54 上实现的,同样适用于规模更大的嵌入式系统中的传统总线扩展问题,在有文件管理的系统中还可以考虑采用组态方式配置状态机的参数,这样会方便状态机参数的配置、保存以及扩充。

参 考 文 献

1　陈火旺 .编译原理 . 北京:国防工业出版社,1984

2　H wang K, Advanced Computer Architecture Parallelism Scalability Programmability, McGraw - Hill, 1995

3　Mitel Corp . Data Book .1991

选自《计算机工程》月刊,2000 年第 8 期

6.15 SPI 串行总线在单片机 8031 应用系统中的设计与实现

铜陵有色金属设计研究院自动化研究所(244000) 卜玉明

一、SPI 总线概述

SPI(Serial Peripheral Interface 串行外设接口)总线系统是一个同步串行外设接口,允许 CPU 与各种外围接口器件以串行方式进行通信,交换信息。外围接口器件包括简单的 TTL 移位寄存器(用作并行输入或输出口)、A/D、D/A 转换器、实时时钟(RTO)、存储器至 LCD、LED 显示驱动器等。SPI 系统可直接与各个厂家生产的多种标准 SPI 外围器件直接接口,它使用四条线:串行时钟线(SCK),主机输入/从机输出数据线 MISO,主要输出/从机输入数据线 MOSI 和低电平有效的从机选择线/SS。由于 SPI 系统总线只需 3～4 数据和控制线即可扩展具有 SPI 各种 I/O 器件,而并行总线扩展方法需 8 根数据线、8～16 位地址线、2～3 控制线,因而 SPI 总线的使用可以简化电路设计,提高设计的可靠性。在 8031 等不具有 SPI 接口单片机组成的智能化仪表和测控系统中,对于速度要求不高,使用 SPI 总线,无疑会增加应用系统接口器件的种类,增强应用系统的性能。

二、SPI 总线的组成

SPI 总线可在软件的控制下构成各种简单的或复杂的系统(见图 6.15-1),如:一个主 CPU 和几个从 CPU;几个从 CPU 相互连接构成多主机系统(分布式系统)一个主 CPU 和一个或几个从 I/O 设备。在大多数应用场合中,使用一个 CPU 作为主机,它控制数据向一个或多个从外围器件的传送。从器件只能在主机发命令时才能接收或向主机传送数据。SPI 总线数据的传输格式是高位(MSB)在前,低位(LSB)在后。SPI 典型的时序图如图 6.15-2 所示。

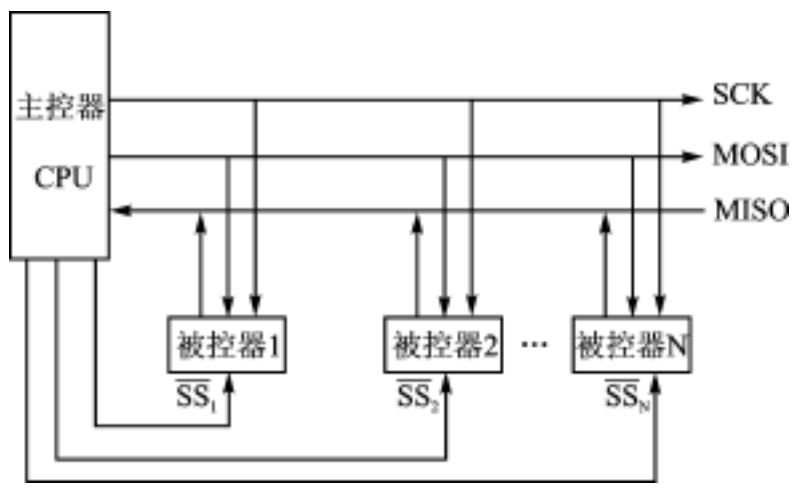


图 6.15-1 SPI 总线的组成

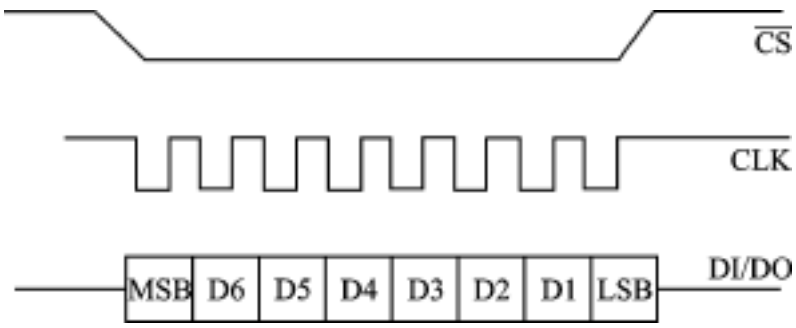


图 6 .15 - 2　SPI 总线时序图

在把 SPI 与几种不同的串行 I/ O 芯片相连时,必须用每片的允许控制端,可用 CPU 的 I/ O 端口输出线来实现。此时应特别注意这些串行 I/ O 芯片的输入输出特性。

(1) 输出芯片的串行数据输出是否有三态控制端。平时未选中芯片的输出端应处于高阻态。若没有三态控制端,应外加三态门。否则 CPU 的 MISO 端只能连接一个输入芯片。

(2) 输出芯片的串行数据输入是否有允许控制端。即应该只有在这片芯片允许时,SCK 脉冲才把串行数据移入该芯片;芯片禁止时 SCK 对芯片无影响。若没有允许控制端,应在外部用门电路对 SCK 进行控制后,再加到芯片的时钟输入端,或者 SPI 只连接一个芯片,不能再连接其他输入或输出芯片。

三、SPI 总线在单片机 8031 中的实现方法

对于 MOTOROLA 公司的 68HC05 系列的一些 CPU 和 68HC11 系列全 CPU 均具有专用的 SPI 接口,它可直接同具有 SPI 接口器件相连。但对于无 SPI 接口 8031 的 CPU 来说,当其串行口置为方式 0,即移位寄存器方式,在这种方式下,波特率固定为 CPU 时钟频率的 1/ 12,若 CPU 时钟频率为 6 MHz,则波特率为 50 000 bps,位周期为 2 μs,显然这种连接方式,只能适应于高速数据传输,如对于串行 A/ D 转换器 MAX176 适合,而对于转换时间为 10 μs 的 ADC0832 等 A/ D 转换器则不能匹配,因而,用这种移位寄存器方式的 SPI 接口在使用上具有一定的局限性。为了适用各种器件的 SPI 接口,可使用软件来模拟 SPI 的操作,包括串行时钟、数据输入和输出。对于不同的串行接口外围芯片,它们的时钟时序是不同的。对于在 SCK 的上升沿输入(接收)数据和在下降沿输出(发送)数据的器件,一般应取串行时钟输出(如图 6 .15 - 3 中的 P1 .1)的初始状态为 1,在允许接口芯片后,先置 P1 .1 为 0。因此,CPU 输出一位 SCK 时钟,同时,使接口芯片串行左移,从而输出一位数据至 8031 的 P1 .3(模拟 CPU 的 MISO 线),再置 P1 .1 为 1,使 8031 从 P1 .0 输出一位数据(先为高)至串行接口芯片,到此模拟一位数据输入输出完成,以后再置 P1 .1 为 0,模拟下一位的输入输出……,依此循环 8 次,可完成一次 SPI 一个字节的操作。对于在 SCK 的下降沿输入数据和上升沿输出数据的器件,则应取串行时钟输出的初始状态为 0,在接口芯片允许时,先置 P1 .1 为 1,此时,外围接口芯片输出一位数据(CPU 接收一位数据),再置时钟为 0,外围接口芯片接收一位数据(CPU 发送一位数据),可完成一位数据的传送。

图 6 .15 - 3 为 8031(CPU)与美国 Xicor 公司芯片 X25045(集 uP 监控、看门狗定时器、EEPROM 于一体)的硬件连接图,有关 X25045 的详细情况请参见本文所附参考文献[2]。图中 P1 .0 模拟 CPU 的数据输出端(MOSI),P1 .1 模拟 SPI 的 SCK 输出端,P1 .2 模拟 SPI 的从机选择端,P1 .3 模拟 SPI 的数据输入端(MISO)。下面介绍用 8031 汇编语言模拟 SPI 串行输

入、串行输出二个子程序。这些子程序也适用于在串行时钟的上升沿输入和下降沿输出的各种串行外围接口芯片,如 8 位或 10 位 A/ D 芯片, 74LS 系列输出芯片等,对于下降沿输入、上升沿输出的各种串行外围接口芯片,只要改变 P1 .1 的输出顺序,即输出 0 再输入 1,再输出 0……,则这些子程序也同样适用。

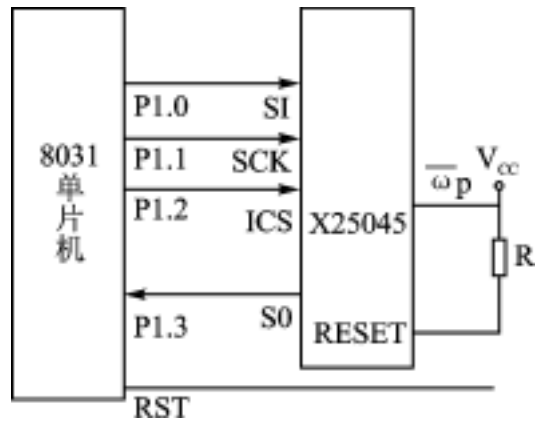


图 6 .15 - 3 SPI 总线接口原理图

1 . CPU 串行输入子程序 SPIIN

功能:按 X25045 时序从 P1 .3 读入 1 字节数据至 A。

入口参数:无

出口参数: A = 数据

```
SPIIN:    ETB P1 .1                ;使 P1 .1(时钟)输出为 1
          CLR P1 .2                ;选择从机
          MOV R1 , # 08H           ;置循环次数
SPIIN1:   CLR P1 .1                ;使 P1 .1(时钟)输出为 0
          NOP                      ;延时
          NOP                      ;
          MOV C, P1 .3             ;从机输出至进位 C
          RLC A                    ;左移至累加器 ACC
          SETB P1 .1               ;使 P1 .1(时钟)输出为 1
          DJNZ R1 , SPIN1          ;判是否循环 8 次(1 字节数据)
          RET                      ;返回
```

2 . CPU 串行输出子程序 SPIOUT

功能:按 X25045 时序从 P1 .0 输出 A 中数据。

入口参数: S = 数据

出口参数:无

```
SPIOUT:   SETB P1 .1               ;使 P1 .1(时钟)输出为 1
          CLR P1 .2                ;选择从机
          MOV R1 , # 08H           ;置循环次数
SPIOUT1:  CLR P1 .1                ;使 P1 .1(时钟)输出为 0
          NOP                      ;延时
          NOP
          RLC A                    ;左移至累加器 ACC 最高位至 C
```

```
MOV P1 .0,C          ;进位 C 送从机输入线上
SETB P1 .1           ;使 P1 .1(时钟)输出为 1
DJNZ Ri ,SPIOUT1     ;判是否循环 8 次(1 字节数据)
RET                  ;返回
```

四、结束语

本文给出了用 8031 汇编语言模拟 SPI 总线的输入、输出传送 1 字节的子程序,读者也可根据 SPI 总线的操作时序在单片机的外设接口如 8255, 8155 和 74LS377 等 I/O 接口来实现 SPI 总线的操作,因而具有一定的通用性。

参 考 文 献

1　涂时亮主编 . MC68HC05 单片机原理、应用及技术手册 . 上海:复旦大学出版社,1991

2　涂时亮,张有德,陈章龙编著 . 单片微机软件设计技术 北京:科学技术文献出版社,1988

选自《工业控制计算机》月刊,2000 年第 1 期

第七章

可靠性及

安全性技术

7.1 软件可靠性及其评估

北京计算机应用与仿真技术研究所 3 室(100854) 石 柱
北京系统工程研究所 4 室(100101) 何新贵
北京航空航天大学计算机系(100083) 武 庄

一、引 言

Bendell 指出,“软件系统是分立状态系统,没有计算机电路中那种重复结构。很少有理由把软件构造成高度重复性的结构。软件系统的状态要比计算机非重复部分的状态数大许多数量级。描述这些系统特性的数学函数不是连续函数,不能用传统的工程数学来验证它们。很清楚,这一差别是软件系统相对不可靠的一个原因,也是软件工程师似乎无能为力的一个原因。这是不会随着技术进步而消失的一个根本区别”[1]。正是软件的这些特点,使得软件日益变得不可控制,由于软件不可靠而造成的灾难触目惊心,屡见不鲜,其主要原因如下。

(1) 软件正成为许多关键系统的核心。由于计算机的使用具有提高效率、能取代人进行某些工作等优点,因此,计算机正日益广泛地应用于监视和控制复杂的、时间关键的物理过程和机械设备。其中软件所起的作用非常关键,它是控制的中枢和灵魂。一旦软件因质量问题出错或失效,会造成系统危险,乃至造成灾难性的损失。

(2) 软件是由人开发的,而人又不可避免地会犯错误。Bailey 指出人们在编制计算机程序和数据中出现的人为错误率如表 7.1-1 所列[2]。

表 7.1-1 在编制计算机程序和数据中出现的人为错误率

错误类别	所占百分比
系统级错误	10%～25%
数据编写错误	10%
抄写错误	2.5%
数据输入错误	0.5%

(3) 软件是由没有容错能力的机器执行的,计算机从不考虑在其上运行的软件是否存在错误,只是按部就班地执行它的命令。

(4) 在当前的软件开发和维护中,主要考虑因素是费用和进度,而不是可靠性。美国“阿波罗”宇航员 Gus Grissom 曾经指出:“每当我想到所有的火箭和宇航员舱都是由要价最低的投标者制造的这一事实时,就使我思索再三”[2]。

因此,软件可靠性问题应引起我们的重视和关注。

二、软件可靠性的定义

1.定 义

软件可靠性指:软件在规定的条件下和规定的时间内,完成规定功能的能力。从软件可靠性的定义可以看出,软件可靠性与“规定的条件”、“规定的时间”和“完成规定功能”有关[3]。

“规定的条件”指软件的用法,一个软件的可靠性随着用法的不同而不同。有些用法可以揭露软件的故障,另外一些则不能揭露软件的故障。因此,如何定义软件的用法以及如何度量软件用法对软件失效的影响,是软件可靠性研究中的一个主要难题。

“规定的时间”指软件的工作周期,软件的可靠性是时间的函数,失效的概率随着系统工作周期的长度而增加。常用的时间概念有:软件执行时间、日历时间和计算机使用时间。

“完成规定功能”指软件不出现失效。如果一个系统不能完成其功能,就说明它已经失效。为了识别一个失效,必须明确要求它完成的功能是什么。规定的功能通常在软件需求规格说明书中定义。

对于软件可靠性这一概念,只做到定性的定义是远远不够的,必须把软件可靠性这一软件质量指标数量化,即用定量的数学方法来研究软件产品的可靠性。关于可靠性度量的重要性,Musa 早在 1975 年就指出:“可靠性是一个极端重要的性能参数,迫切需要进行量化”。此外,在美国空军管理学院的系统工程教材中,有一段话很精辟地论述了可靠性及可靠性度量的重要性:“如果对产品没有可靠性要求,或是有了可靠性要求而不要求验证,则空军只能依靠希望与信心了”。目前,在我国计算机已成为复杂系统的重要组成部分,有些系统甚至包含大大小小一百多台计算机,其对系统可靠性的影响越来越显著,然而在进行可靠性分配时,通常把计算机软件的可靠性看成 1,这是不切实际的。何国伟同志指出,“抓软件的可靠性同硬件一样,也必须既有要求,又有验证。有要求才有动力,有验证才有压力”[4]。

2.软件失效机理

由于软件内部逻辑较为复杂,运行环境变化较大,而且软件不同,其差异可能很大,因而软件失效机理可能有不同的表现形式。例如,有的失效过程比较简单,易于跟踪分析;而有的失效过程可能非常复杂,难于甚至不可能进行详尽的描述和分析。为了避免在使用术语时造成不必要的混乱,在具体阐述软件的失效机理之前,首先应明确几个有关的概念。

人为错误(human error) 是指在软件生存期内出现的不期望或不可接受的人为差错,其结果会导致产生软件缺陷。人为错误是人们在软件开发活动中不可避免的一种行为过失。

软件缺陷(software defect) 是存在于软件中的、不期望的或不可接受的偏差,其结果是当软件运行于某一特定条件时将出现软件故障(即,软件缺陷被激活)。软件缺陷以一种静态的形式存在于软件的内部,是软件开发过程中人为错误的结果。

软件故障(software fault) 是指在软件运行过程中出现的一种不期望的或不可接受的内部状态,此时若无适当措施加以处理(例如,容错)就会产生软件失效。软件故障是一种动态行为,是软件缺陷被激活后的表现形式。

软件失效(software failure) 是指软件对要求行为的偏离,是软件运行时产生的一种不期望的或不可接受的外部行为结果。

在某种意义上讲,软件缺陷和软件失效代表了软件的两个不同层次的问题。软件缺陷是软件的内部问题,是软件开发人员能够察觉到的问题;软件失效是软件的外部问题,是用户能

够察觉到的问题。

软件的失效机理可以形象地描述为:人为错误→软件缺陷→软件故障→软件失效(如图7.1-1所示)[4]。

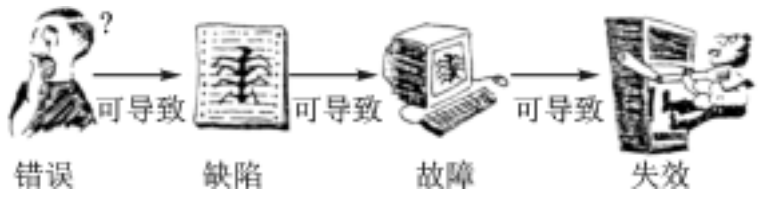


图7.1-1 软件失效机理

3.软件可靠性与硬件可靠性的差异

软件作为一种产品与硬件有许多不同,但从可靠性的角度来看,它们间的主要不同点如表7.1-2所列[4]。总的说来,软件可靠性与硬件可靠性的本质差别是智力失效与物理失效之间的差别。

表7.1-2 软件与硬件的不同点

软 件	硬 件
软件是逻辑实体,始终不会自然变化,只是其载体可变	硬件是物理实体,每件同规格产品的质量特性之间有散布,会随时间和使用而老化、磨损以至失效
软件的接口是不可见的	硬件的接口是可见的
软件尚没有标准件	硬件有标准的零部件
软件的研制过程主要是紧张的脑力劳动过程,在本质上是无形的,不可见,难控制	硬件的研制过程不只是脑力劳动过程,还有体力劳动过程,其过程有形,便于测控
其不可靠问题基本是由于开发过程中的人为差错所造成的缺陷而引起的	其不可靠问题不只是设计问题,在生产和使用过程中也会产生新的故障
程序是指令序列,即使每条指令都正确,但由于在执行时其逻辑组合状态千变万化,不一定完全正确	硬件失效总是由其零部件或其结合的故障所引起
系统数学模型是离散型的,其输入在合理范围内的微小变化可能引起输出的巨大变化,故障的形成无物理原因,失效的发展取决于输入值和运行状态的组合,无前兆	系统在正常工作条件下其行为是渐变的,故障的形成和失效的发生一般都有物理原因,有前兆
应在开发的全过程采取措施防错、检错、纠错和容错,而在批量复制过程中,软件本身不会变化	除了开发过程外,生产过程对产品的影响也很大,均需加强控制
精心设计测试实例,执行严格测试,查出错误并排除	建立适当的环境应力条件,进行筛选,排除缺陷
采用冗余设计时,应确保冗余软件间的相异性,否则,相同的冗余软件不仅不能提高可靠性,反而增加了复杂性,降低了可靠性	相同的部件之间自然是独立的,适当的冗余可以提高可靠性
在出现故障后必须修改原产品以解决问题;若在修改时未引入新问题,那么其可靠性就会增长	在出现故障后不需修改原产品,只需要更换或修复失效的部件,使产品恢复良好状态,其可靠性一般不会提高
某处的修改会影响它处,维护时必须考虑这种影响	维修某处一般不会影响它处
失效率随故障的排除而下降	失效率的变化类似于溶盆曲线
可靠性参数估计无物理基础	可靠性参数估计有物理基础

三、软件可靠性评估研究现状

到目前为止已经发表了近百种软件可靠性模型,人们将其形象地称为“模型战争”。基于软件失效历史,按软件失效过程的性质,可将软件可靠性模型分为如下四类。

(1) 失效时间间隔模型:这类模型最常用的方法是假定失效间隔时间服从于某一分布,分布的参数依赖于各间隔时间内程序中的残留错误数。模型参数的估值通过实测的失效间隔时间来获得,由获得的模型可以估算软件的可靠度以及各失效间的平均工作时间等;也可以把失效间隔时间看作随机过程,然后通过适当的时间序列模型来分析和描述软件的失效过程。这类模型有 Jelinski-Moranda 的非增长模型、Schick-Wolverton 模型、Littlewood-Verrall 的贝叶斯模型和 Goel-Okumoto 的不完善模型等等。

(2) 缺陷计数模型:这类模型特别关心在特定的时间间隔内软件的错误数或失效数,根据在给定的测试时间间隔发现的错误数或失效数来建模。随着错误的不断排除,在单位时间内发现的失效数将不断减少。所采用的时间既可以是 CPU 时间,也可以是日历时间或实测的运行次数。当时间间隔事先确定时,在每个间隔时间内的失效次数就是随机变量。这类模型有 Shooman 模型、Musa 的执行时间模型、Goel-Okumoto 的 NHPP 模型、Goel 的 NHPP 模型和 Musa-Okumoto 的对数泊淞执行时间模型等等。

(3) 错误植入模型:这类模型的基本思路是通过将一组已知的错误人为地植入到一个固有错误总数尚不清楚的程序中,然后在程序的测试中观察并统计发现的植入错误数和程序的错误数,从而通过估计程序的固有错误总数来评价软件可靠度及其有关指标。这是由 Mills 首先提出并用于估计程序中残留错误而采用的超几何分布模型。使用超几何模型可以估计出程序中的固有错误数。后经 Lipow 的改进并经过 Basin 的修改构成了 Mills-Basin 模型。这种模型已从理论上证明可以提供程序正确性概率的独立估计,并已实际应用于 EPRI(电子研究所)的工程中。

(4) 基于输入域的模型:这类模型以 Nelson 模型为代表,该模型假定 E 为全体输入数据的集合:$E = \{E_i \mid i = 1, 2, \cdots, n\}$,N 为集合 E 中元素的个数,$E_i$ 是 E 的子集,N_i 为子集 E_i 中元素的个数。当用 E_i 中的元素作为输入数据将导致系统失效时,Nelson 定义程序运行一次的失效概率为:$P = N_i / N$,因而,程序运行一次无失效的概率,即可靠度为:$R = 1 - N_i / N$。于是系统运行 n 次无失效的概率为:$R(n) = (1 - P)^n$。其中,P 是从 E 中随机选取 E_i 运行导致一次失效的概率,实际上,输入有可能根据某一特定要求从输入集 E 中加以选择,这一要求可用概率分布 P_i 来描述。P_i 是 E_i 从 E 中被选取的概率,通常称 P_i 的集合$\{P_i\}$为运行剖面。为了计算 P,定义执行变量 Y_i:

$$Y_i = \begin{cases} 0, & \text{若 P 在 E 上一次运行可以接受} \\ 1, & \text{若 P 在 E 上发生一次失效} \end{cases}$$

于是 $P = \sum_{i=1}^{n} P_i Y_i$,因此 $R(n) = (1 - P)^n = \left[\sum_{i=1}^{n} P_i (1 - Y_i)\right]^n$。为采用本模型来估计程序的可靠性,可抽取 n 组数据进行试验,若有 n_e 次失效,则可得到可靠度 R 的估计值为:$\hat{R} = 1 - n_e / n$。这类模型还有 Brown-Lipow 的基于输入域模型和 Ramamoorthy-Bastani 的基于输入域模型。

四、软件可靠性的模糊评估方法

1 . 基本概念

根据 GB/ T11457 的定义,软件可靠性评估与软件可靠性估计同义,均指“确定现有系统或系统部件可靠性所达到的水平的过程”。

目前,大多数软件可靠性模型都是基于概率和统计技术的,由于软件可靠性评估方法还远远不像硬件那样成熟,尚没有一种方法得到普遍的承认。现有方法同硬件相比其适用的局限性要大得多。Harris 指出,“评定软件可靠性时是否适用可靠性概率的概念,这个问题还是悬而未决的。尽管软件企业界清楚地了解可靠性问题,也知道需要评定可靠性并使他们自己相信确已满足可靠性要求,却很少有人认为可靠性的概率定义是适用的。作者认为,这个情况并不是由于对有关软件的问题了解不够,而是由于所了解的只是情况已经发生。软件企业界的大多数人宁愿把可靠性看作是正确性的表示,但是像概率一样,正确性也是一个直觉的观念,只有在正式的数学框架之内才能做到准确、客观和严谨”[1]。因此,不论将软件可靠性视为概率的定义,还是将软件可靠性视为正确性,软件可靠性都是一种直觉的观念。模糊技术为处理软件可靠性这类需要人们进行主观判断的直觉观念提供了一个正式的数学方法。

模糊集合论可以用来表示在可靠性分析中固有的、专家判断的主观性和模糊性。例如,在估计失效概率和差错概率时,这些概率应当从大量的数据中进行估计。但是,在实际上不可能收集到大量的第一手数据,因此,不可能估计出一个有意义的概率。通常,这些失效概率和差错概率是通过其他领域的参考数据,根据专家的工程判断和经验来估计。这些判断和经验与主观性和模糊性密切相关。又如,在考虑环境条件对可靠性的影响时,硬件可靠性和人的可靠性都受到环境条件的影响,例如,机器运行的条件和操作人员的心理压力。因此,估计的失效概率和差错概率并非物理常量,因而必须评价硬件使用条件和任务的环境条件,而这种估计又是由专家根据工程判断和经验进行的,存在着主观性和模糊性。在可靠性领域,存在模糊性和主观性的地方还有许多,从而为模糊集合理论提供广阔的应用背景。

传统的软件可靠性定义是由四个要素构成的:规定条件、规定时间、规定功能和能力。原则上讲,对这四个要素均可进行模糊化处理。但为简单起见,我们仅对“能力”进行模糊化处理,即:将“能力”用模糊概率来表示。在此基础上,我们将提出两种基于模糊数算术运算的软件可靠性估计方法[5]。

限于篇幅,本文将不对有关模糊数的定义及运算等内容展开论述,有兴趣者可参见相关文献。

2 . 基于三角形模糊数的并串联系统可靠性估计

(1) 并联系统可靠性估计

设软件系统 sys 是一个由 n 个组成部件组成的并联系统,且这 n 个组成部件的失效相互独立,系统 sys 只有“正常”和“失效”两种状态,F_1 , F_2 , …, F_n 分别为 n 个组成部件的模糊失效概率,则系统的模糊失效概率为:

$$F_{sys} = F_1 \times F_2 \times \cdots \times F_n$$

若这 n 个组成部件的失效并不相互独立,则系统的模糊失效概率为:

$$F_{sys} = \min(F_1 , F_2 , \cdots , F_n)$$

例如,系统 sys 是由两个组成部件 1 和 2 构成的并联系统,组成部件 1 和 2 的失效相互独

立,且根据以往的经验或专家的判断得出:组成部件 1 的失效概率 F_1“大约是 0.1”(即 F_1 =“大约是 0.1”),组成部件 2 的失效概率 F_2“大约是 0.15”即(F_2 =“大约是 0.15”)。若分别采用三角形模糊数 F_1 = (0.05,0.1,0.15)和 F_2 = (0.1,0.15,0.2)来表示组成部件 1 和 2 的失效概率,则系统 sys 的失效概率 F_{sys} 为:

$$\begin{aligned} F_{sys} &= F_1 \times F_2 = \text{“大约是 0.1”} \times \text{“大约是 0.15”} \\ &= (0.05,0.1,0.15) \times (0.1,0.15,0.2) \\ &= (0.005,0.015,0.03) \\ &= \text{“大约是 0.015”} \end{aligned}$$

假设系统 sys 是由两个组成部件 1 和 2 构成的并联系统,组成部件 1 和 2 的失效并不相互独立,且组成部件 1 的失效概率 F_1“大约是 0.1”,组成部件 2 的失效概率 F_2“大约是 0.15”。则系统 sys 的失效概率 F_{sys} 为:

$$\begin{aligned} F_{sys} &= \min(F_1, F_2) \\ &= \min(\text{“大约是 0.1”}, \text{“大约是 0.15”}) \\ &= \min((0.05,0.1,0.15)(0.1,0.15,0.2)) \\ &= (0.05,0.1,0.15) \\ &= \text{“大约是 0.1”} \end{aligned}$$

(2) 串联系统可靠性估计

设软件系统 sys 是一个由 n 个组成部件组成的串联系统,且这 n 个组成部件的失效相互独立,系统 sys 只有‘正常’和‘失效’两种状态,$F_1, F_2, \cdots, F_n$ 分别为 n 个组成部件的模糊失效概率,则系统的模糊失效概率为:

$$F_{sys} = (1,1,1) - [((1,1,1) - F_1)((1,1,1) - F_2)\cdots((1,1,1) - F_n)]$$

若这 n 个组成部件的失效并不相互独立,则系统的模糊失效概率为:

$$F_{sys} = \max(F_1, F_2, \cdots, F_n)$$

例如,系统 sys 是由两个组成部件 1 和 2 构成的串联系统,组成部件 1 和 2 的失效相互独立,且根据以往的经验或专家的判断得出:组成部件 1 的失效概率 F_1“大约是 0.1”,组成部件 2 的失效概率 F_2“大约是 0.15”。若分别采用三角形模糊数 F_1 = (0.05,0.1,0.15)和 F_2 = (0.1,0.15,0.2)来表示组成部件 1 和 2 的失效概率,则系统 sys 的失效概率 F_{sys} 为:

$$\begin{aligned} F_{sys} &= (1,1,1) - [((1,1,1) - F_1)((1,1,1) - F_2)] \\ &= (1,1,1) - [((1,1,1) - \text{“大约是 0.1”}) \\ &\quad ((1,1,1) - \text{“大约是 0.15”})] \\ &= (1,1,1) - [((1,1,1) - (0.05,0.1,0.15)) \\ &\quad ((1,1,1) - (0.1,0.15,0.2))] \\ &= (1,1,1) - [(0.85,0.9,0.95)(0.8,0.85,0.9)] \\ &= (1,1,1) - (0.68,0.765,0.855) \\ &= (0.145,0.235,0.32) \\ &= \text{“大约是 0.235”} \end{aligned}$$

假设系统 sys 是由两个组成部件 1 和 2 构成的串联系统,组成部件 1 和 2 的失效并不相互独立,且组成部件 1 的失效概率 F_1“大约是 0.1”,组成部件 2 的失效概率 F_2“大约是 0.15”,

则系统 sys 的失效概率 F_{sys} 为:

$$\begin{aligned} F_{sys} &= \max(F_1, F_2) \\ &= \max(\text{“大约是 0.1”}, \text{“大约是 0.15”}) \\ &= \max((0.05, 0.1, 0.15), (0.1, 0.15, 0.2)) \\ &= (0.1, 0.15, 0.2) \\ &= \text{“大约是 0.15”} \end{aligned}$$

3. 基于梯形模糊数的并串联系统可靠性估计

(1) 并联系统可靠性估计

传统的并联系统可靠性使用的“或”门算子为:

$$q_{OR} = 1 - \sum_{i=1}^{n} (1 - q_i)$$

式中 q_i 为事件 i 发生的概率,为一精确值。

如果事件 i 发生的概率为一模糊数 $\tilde{q}_i$,则“或”门模糊算子可记为:

$$\begin{aligned} \tilde{q}_{OR} &= (a_{OR}, b_{OR}, c_{OR}, d_{OR}) = 1 \ominus \prod_{i=1}^{n} (1 \ominus q_i) \\ &= 1 \ominus \prod_{i=1}^{n} (1 \ominus (a_i, b_i, c_i, d_i)) \\ &= \left((1 - \prod_{i=1}^{n} (1 - a_i)), (1 - \prod_{i=1}^{n} (1 - b_i)), \right. \\ &\quad \left. (1 - \prod_{i=1}^{n} (1 - c_i)), (1 - \prod_{i=1}^{n} (1 - d_i))\right) \end{aligned}$$

式中 $\prod$ 为模糊乘法运算。

(2) 串联系统可靠性估计

传统串联系统可靠性分析使用的“与”门算子为:

$$q_{AND} = \prod_{i=1}^{n} q_i$$

式中 q_i 为事件 i 发生的概率为一精确值。

如果事件 i 发生的概率为一模糊数 $\tilde{q}_i$,则“与”门模糊算子可记为:

$$\begin{aligned} \tilde{q}_{AND} &= (a_{AND}, b_{AND}, c_{AND}, d_{AND}) \\ &= \prod_{i=1}^{a} q_i = q_1 \quad q_2 \quad \cdots \quad q_n \\ &= \left[\prod_{i=1}^{n} a_i, \prod_{i=1}^{n} b_i, \prod_{i=1}^{n} c_i, \prod_{i=1}^{n} d_i \right] \end{aligned}$$

式中 $\prod$ 为模糊乘法运算。

参 考 文 献

1 Bendell,等.软件可靠性:当前技术发展现状报告.质量与可靠性资料汇编之10[Z].胡昌寿,沈家楠译.《质量与可靠性》编辑部,1990

2 Paul Rook.软件可靠性手册[M].关锡镗,等译.北京:航空工业出版社,1996

3 石柱.基于模糊技术的软件质量评价及可靠性评估[D].北京:北京航空航天大学出版社,2000

4 何国伟.软件可靠性[M].北京:国防工业出版社,1998

5 石柱,何新贵,遇今.一种基于模糊数算术运算的可靠性分析方法[J].系统工程与电子技术,1999,(9):74～77.

选自《计算机应用》月刊,2000年第11期

7.2 网络通信中的基本安全技术

北京化工大学自动化系(100029) 潘立登 盛乃军

Internet 已经成为当今全球数据通信的有效工具,它的迅猛发展对全球经济和社会生活都产生了巨大影响。Internet 网的应用领域极其广阔,如许多高等学校都已建立自己校园网并与 Internet 相连。作为远程教学的工具和获得信息的重要途径,商业界也在积极地建立企业内部网络并通过 Internet 向公众提供种类繁多的信息服务,其中最引人注目的当属电子商务,电子商务正是在 Internet 快速发展的浪潮下应运而生的,它是信息时代社会生产与社会消费之间发生的一次革命。

Internet 在为人们带来无限商机的同时,也给人们提出一个十分严峻的课题,即如何保证各种网络应用的安全性。例如在电子商务中网上购物是在线付款,用户的信用卡号等许多信息都是敏感信息,而这些网上传输的敏感数据和存放敏感信息的站点正是网络黑客的重点攻击对象。因此,人们在开展各种网络业务时,首先考虑的是这种网络业务是否能够保证安全,如果不能保证安全,人们也就不会接受这种业务。

网络通信的数据安全包括以下几个方面。

(1) 数据传输的安全性 数据传输的安全性即是要保证在公网上传输的数据不被第三方窃取。

(2) 数据的完整性 对数据的完整性需求是指数据在传输过程中不被篡改。

(3) 身份验证 由于网上的通信双方互不见面,必须在相互通信时(交换敏感信息时)确认对方的真实身份。

(4) 不可抵赖性 在网上开展业务的各方在进行数据传输时,必须带有自身特有的、无法被别人复制的信息,以保证发生纠纷时有所对证。

通常情况下,网络通信中所采用的安全技术主要有防火墙技术、数据加密技术和身份认证技术等。本文讨论的重点是数据加密技术和身份认证技术以及它们在网络通信安全策略中的应用。

一、加密技术

计算机网络中保护数据安全性最有效的方法就是数据加密技术。数据加密算法有很多种,每种加密算法的加密强度各不相同。目前存在两种基本的加密体制:对称密钥加密和非对称密钥加密。

1. 对称密钥加密

对称密钥加密体制又被称为私钥加密体制,它使用同一组钥匙对消息进行加密和解密。因此,消息的接收者和发送者必须拥有一组相同的密钥。在私钥加密体制中,比较有名的加密算法是 DES(Data Encryption Standard)(数据加密标准)。

美国国家标准局于 1977 年宣布数据加密标准 DES 用于非国家保密机关。该加密算法是

由 IBM 公司研究提出的,使用 64 位的密钥对 64 位的数据进行加密和脱密。

DES 可以采取多种操作方式,下面介绍两种最为通用的操作方式,即 ECB、CBC。

(1) 电子密码本型(ECB) 这种操作模式是用同一把钥匙独立地加密每个 64 位的明文组,其操作特点如下:

· 可加密 64 位。

· 加密与代码组的顺序无关。

· 对同一组密钥,相同明文组将产生相同密文组,因此易受 字典攻击 '的破译。

· 错误只影响当前的密文组,不会扩散传播。

(2) 密码分组链接型(CBC) 每组明文在加密之前先与前一个密文组进行异或运算,然后再进行加密,其操作特点如下:

· 可加密 64 位的整数倍。

· 对相同的密钥和初始向量,相同的明文将生成相同的密文。

· 链接操作使密文组依赖于当前及其前面所有的明文组,因此密文组的顺序不能被打乱。

· 可用不同的初始向量来防止相同的明文产生相同的密文。

· 错误将影响从当前开始的两个密文组。

DES 在密码学发展历史上具有重要的地位。在 DES 加密标准公布以前,密码设计者出于安全性考虑,总是掩盖算法的实现细节,而 DES 开历史之先河,首次公开了全部算法。同时,DES 作为一种数据加密标准,推动了保密通信在各种领域的广泛应用。

2.非对称密钥加密

非对称密钥加密又被称为公开密钥加密体制,是由 Whitfield Diffie 和 Martin Hellman 在 1976 年提出。其加密机制是,每个人拥有一对密钥,一个称为公开密钥,另一个称为秘密密钥,这两个密钥是数学相关的。公开密钥是公开信息,秘密密钥由用户自己保存。在这种体制中,加密和解密使用不同的密钥,因此,发送者和接收者不再需要共享一个秘密(对称密钥加密体制中,发送者和接收者必需共享一个密钥),即在通信的全部过程中不需要传送秘密密钥。公开密钥算法的主要特点如下。

(1) 用加密密钥 PK 对明文 A 加密后得到密文,再用解密密钥 SK 对密文解密,即可恢复出明文 A。

$$D_{SK}(E_{PK}(A)) = A$$

(2) 加密密钥不能用来解密,即:

$$D_{PK}(E_{PK}(A)) \neq A$$

$$D_{SK}(E_{SK}(A)) \neq A$$

(3) 用 SK 加密的信息只能用 PK 解密;用 PK 加密的信息只能用 SK 解密。

(4) 从已知的 PK 不可能推导出 SK。或者说,由 PK 推导出 SK 在计算上是不可能的。

(5) 加密和解密的运算可以对调,即:

$$E_{PK}(D_{SK}(A)) = A$$

公开密钥算法在运算速度较对称密钥加密算法慢一些,因此在实际应用中,对称密钥算法主要用于产生数字签名、数字信封,而并不直接对大量的应用数据进行加密。

在公开密钥体制中,最为通用的是 RSA 公钥加密体制,它已被推荐为公开密钥数据加密

标准。RSA 是由 Rivet、Shamir 和 Adleman 提出的,它的安全性是基于大数因子分解,由于大数因子分解在数学上没有行之有效的算法,因此该加密技术的破译是相当困难的。

3 . 数字指纹技术

在继续介绍其他的安全技术之前,我们还要先讨论一下数字指纹技术。

数字指纹是一种形象的说法,在密码学上又被称为“信息摘要”(message digest)。它是通过安全的单向散列函数(Secure Hash)作用于将要发送的信息(message)上产生的:

message digest = ecure Hash message)

单向散列函数有三个主要特点:

(1) 它能处理任意大小的信息,并将其按信息摘要(message digest)方法生成固定大小的数据块,对同一个源数据反复执行 Secure Hash 函数将总是得到同样的结果。

(2) 它是不可预见的。产生的数据块的大小与原始信息的大小没有任何联系,同时源数据和产生的数据块看起来也没有明显关系,源信息的一个微小变化都会对小数据块产生很大的影响。

(3) 它是完全不可逆的,没有办法通过生成的数据块直接恢复源数据。

数字指纹技术并不是一种加密机制,但却能产生信息的数字“指纹”,通过验证信息的“指纹”来确保数据没有被修改或变化,保证信息的完整性不被破坏。

常用的信息摘要算法有 MD2、MD5 和 SHA - 1 等。

二、身份认证技术

1 . 数字签名

数字签名是用来保证信息传输过程中信息的完整和提供信息发送者的身份认证和不可抵赖性的一种安全技术。首先,接收者能够验证发送者对报文的签名,以确保数据的完整性。同时,由于第三方公证机构可以通过数字签名进行公证,因此发送者事后不能抵赖对报文的签名。另外,数据签名还具有不可伪造性,同现实世界中手工签名具有相同的效果。

公开密钥算法是实现数字签名的主要技术。使用公开密钥算法实现数字签名技术,类似于公开密钥加密技术。它有两个密钥:一个是签名密钥,它是对外保密的,因此称为私有密钥或秘密密钥,简称私钥;另一个是验证密钥,它是对外公开的,因此称为公开密钥,简称公钥。

由于公开密钥算法的运算速度比较慢,因此可使用安全的单向散列函数对要签名的信息进行摘要处理,减小使用公开密钥算法的运算量。实现数字签名的过程如图 7 .2 - 1 所示。

(1) 信息发送者 A 使用一单向散列函数对信息生成信息摘要。

(2) 信息发送者 A 使用自己的私钥签名信息摘要(用私钥对摘要加密)。

(3) 信息发送者 A 把信息本身和已签名的信息摘要一起发送出去。

(4) 任何接收者 B 通过使用与信息发送者 A 使用的同一个单向散列函数对接收的信息生成新的信息摘要,再使用信息发送者 A 的公钥对数字签名解密,并与新生成的信息摘要比较,以确认信息发送者的身份和信息是否被修改过。

在数字签名的基础上,还可以实现双重签名技术。双重签名技术是为了保证在事务处理过程中三方安全地传输信息的一种技术,实现了三方通信时的身份认证和信息完整性、防抵赖的保护。例如在网上购物的业务中,客户和商家之间要完成在线付款,那么客户、商家和银行之间将面临以下问题:客户(甲)需要给商家(乙)发送购买信息和客户的付款账户信息;乙作为

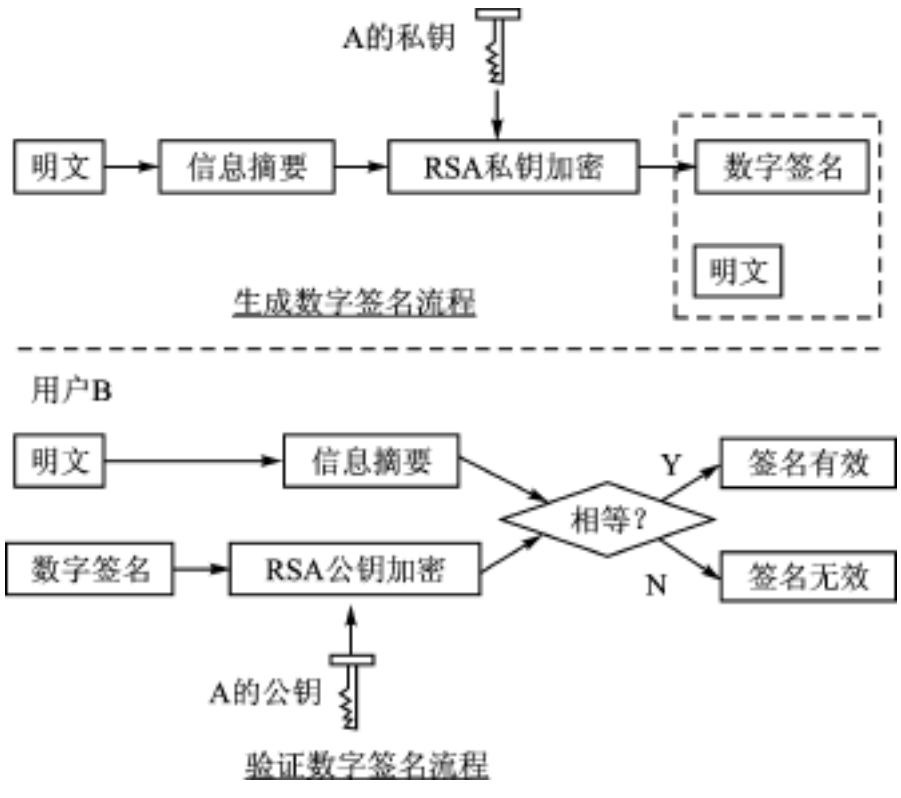


图 7.2-1

商家,接受购买信息后,还要同银行(丙)交互,以实现资金转账。但甲不愿让乙看到自己的付款账户信息,也不愿让处理甲付款信息的丙看到订购信息。此时甲使用双重签名技术对两种信息作数字签名,来完成以上功能。

双重数字签名的实现步骤如下:

(1) 甲对发给乙的信息 1 生成信息摘要 1;

(2) 甲对发给丙的信息 2 生成信息摘要 2;

(3) 甲把信息摘要 1 和信息摘要 2 合在一起,对其生成信息摘要 3,并使用自己的私钥签名信息摘要 3;

(4) 甲把信息 1、信息摘要 2 和信息摘要 3 的签名发给乙;

(5) 甲把信息 2、信息摘要 1 和信息摘要 3 的签名发给丙;

(6) 乙接收信息后,对信息 1 生成信息摘要,把这信息摘要和收到的信息摘要 2 合在一起,并对其生成新的信息摘要,同时使用甲的公钥对信息摘要 3 的签名进行验证,以确认信息发送者的身份和信息是否被修改过;

(7) 丙接收信息后,对信息 2 生成信息摘要,把这信息摘要和收到的信息摘要 1 合在一起,并对其生成新的信息摘要,同时使用甲的公钥对信息摘要 3 的签名进行验证,以确认信息发送者的身份和信息是否被修改过。

2. 数字信封

数字信封技术结合了秘密密钥加密技术和公开密钥加密技术的优点。克服了秘密密钥加密中秘密密钥分发困难和公开密钥加密中加密时间长的问题,使用两个层次的加密来获得公开密钥技术的灵活性和秘密密钥技术的高效性,保证信息的安全性。由于数字信封技术是把要发送的报文用收信方的公钥进行加密,只有收信方的私钥才能解开被加密的报文,而其他人不能解开被加密的报文,这样就确保了只有收信方才能准确地接收到报文,该技术也因此被形象地称为“数字信封”。

数字信封的具体实现步骤如图 7.2-2 所示。

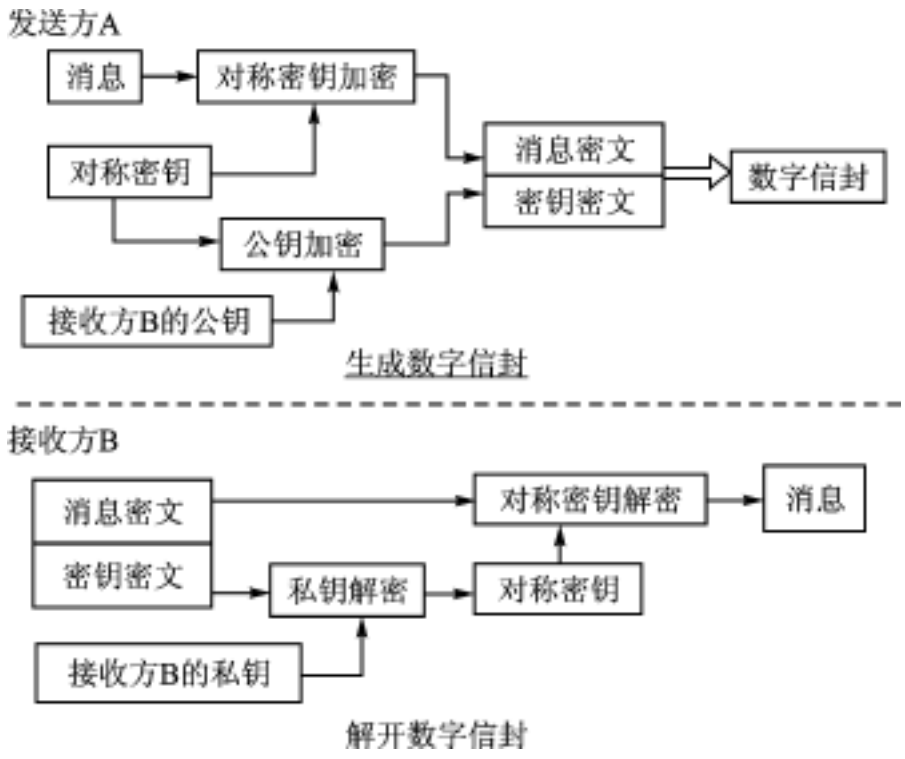


图 7.2-2

(1) 发送方 A 首先生成一个对称密钥,用该对称密钥加密要发送的报文;

(2) 发送方 A 用接收方 B 的公钥加密上述对称密钥;

(3) 发送方 A 将第一步和第二步的结果(即数字信封)传给接收方 B;

(4) 接收方 B 使用自己的私钥解密被加密的对称密钥;

(5) 接收方 B 用得到的对称密钥解密被发信方加密的报文,得到真正的报文。

数字信封技术在外层使用公开密钥加密技术,因此可以获得公开密钥技术的灵活性。由于内层的对称密钥长度通常较短,从而使得公开密钥加密的相对低效率被限制在最低限度,而且由于可以在每次传送中使用不同的对称密钥,系统有了额外的安全保证。

3.密钥管理和身份认证技术

在许多网络攻击事件中,密钥的安全管理是黑客攻击的一个主要环节,因此网络的安全性另一个方面就是密钥的安全保护上,密钥管理包括密钥的设置、产生、分配、存储、注销、验证和使用等一系列过程。

密钥分配是密钥管理中最大的问题,也是网络的安全的重中之重。密钥必须通过最为安全的方式进行分配。数字证书是一种安全分发公钥的方式。在这种方式中,必须设立一个密钥管理中心,它全权负责密钥的发放、注销及验证。RSA 公开密钥体制就是采用这种方式进行密钥管理的。在 RSA 公开密钥体制中存在一个或多个密钥管理中心,又称为证书授权(CA——Certificate Authority)中心。证书授权中心为每个申请公开密钥的用户发放一个证书,该证书证明了该用户拥有证书中列出的公开密钥。数字证书中基本包括证书持有人的个人信息、公钥以及证书签发者的对这些信息的数字签名和证书签发者的数字证书(私钥由用户秘密保存)。CA 的数字签名使得攻击者不能伪造和篡改该证书,因此,数字证书既起到分配公钥的作用,同时又实现了身份认证的功能。

CA 体系有单级和多级之分。在单级 CA 中,用户可以根据证书签发者(即根 CA,顶级 CA)证书中的公钥来验证用户证书的数字签名。在多级 CA 中,各级 CA 形成一个至上而下的链级体系,每一级 CA 证书的合法性都要由其上级 CA 证书来验证,直至根 CA。如果用户

要验证某一证书的合法性,就要由该证书的签发者一直验证根 CA 证书为止,才能完全信任该证书。那么,如果根 CA 中心被攻破,则整个 CA 体系彻底崩溃。

本文主要分析了网络安全中的基本加密算法和安全技术,在具体的网络业务中,还要根据本业务的特点,来制定相应的安全协议和安全策略。例如在电子商务中,越来越多的商业活动是在互未谋面的实体之间进行的,因此客户和商家之间就存在着相互的身份认证问题。此外,在这类业务中,最为敏感的信息是网上传送的用户口令和信用卡号码等,一旦被窃听,将产生灾难性的后果。为了解决电子商务中这类安全问题,国际信用卡集团 VISA 和 MasterCard 联合制定了“安全电子交易”(SET)协议,用来保证因特网中在线持卡交易的安全性。从网络七层协议的角度来看,SET 协议处在应用层上,制定了以信用卡为基础的电子付款系统规范,定义了使用信用卡购物的全部支付流程。SET 协议中所使用的安全技术也是上面介绍的基本加密算法和身份认证技术。

网络安全问题涉及到许多方面,确保网络安全将是各种技术的大融和。随着网络的飞速发展,人们对安全问题的探索将不断深入,各种安全技术也必将在这种探索中得到进一步的完善与发展。

参 考 文 献

1 卢开澄.计算机密码学.北京:清华大学出版社,1990

2 SET Secure Electronic Transaction Specification. Visa and MasterCard. 1997

3 Warwick Ford Nichael S. Baum. SECURE electronic commerce. 1997

4 RSA LaboratoriesPKCS # 10: Certification Request Syntax Standard. Version 1.0, November 1993

5 Common Security: CDSA and CSSM. The Open Group. 1997

6 Kaliski B. RFC 1424: Privacy Enhancement for Internet Electrornic Mail: Part IV: key Certification and Related Services. February 1993

7 Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatrues and public-key cryptosystems Communications of the ACM, 21(2): 120 - 126, February 1998

8 National Bureau of Standards. FIPS PUB 46 - 1: Data Encryption Standard. January 1998

9 Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatrues and public-key cryptosystems. Communications of the ACM, 21(2):120 - 126, February 1978

选自《电子技术应用》月刊,2000 年第 3 期

7.3 数字语音混沌保密通信系统及硬件实现

南宁广西大学计算机与信息工程学院(530004)
唐秋玲 覃团发 姚海涛 林 硒

混沌系统具有对初条件极端敏感的特性,它可以提供大量非相关、类随机而又确定可再生的混沌序列。近几年来,研究混沌和应用混沌已经成为国际电子工业界前沿最活跃的一个研究热点,其中在保密通信方面的应用研究也越来越得到人们的重视[1,2]。

混沌序列在密码学方面的应用起源于 20 世纪 80 年代末期,由英国数字家 Matthews 首先提出[1],其后得到了一定的发展。国内南京大学声学研究所的倪皖荪、中国科学院的张洪均等也正在进行这方面的研究工作。

基于混沌系统之间能够达到自同步[2],发展了多种同步技术,如:混沌掩埋技术[3]、混沌调制技术[4]、混沌开关技术[5]以及数字混沌通信技术[6]等,分别运用于连续混沌通信系统和数字混沌通信系统。众所周知,数字通信系统以其抗干扰能力强,易于加密,易于大规模集成等特点,在通信行业中将取代模拟通信而占主要地位。而且,数字混沌系统比较模拟混沌系统具有结构简单,易于实现,保密性能高等优势。因而,混沌技术在数字保密通信中的应用研究也就更具有现实意义。

本文对离散时间动力学系统逻辑影射进行变换,使其在一定精度下产生数字混沌序列,采用该数字混沌序列作为密码,构造了语音保密通信系统,并运用单片机实现了该系统的硬件实验。

一、数字混沌序列的产生

研究证明逻辑映射

$$x_{n+1} = 1 - 2x_n^2, x_n \in (-1,1) \tag{1}$$

可以产生大量具有均值为零、自相关为δ函数、互相关为零统计特性的优良混沌序列,因而可作为理想的密码序列,应用于语音信号的保密传输。

要实现逻辑映射的数字化,一种方法是采用浮点运算。实际运算表明,浮点单精度(32位)的运算结果脱离了混沌态,浮点双精度(64 位)的运算结果与理论接近。但在实际应用中,64 位浮点双精度运算需要内存空间大,运算速度慢,而且不利于数字硬件实现。下面我们把逻辑映射的迭代过程由浮点运算变换为定点运算。

日常生活中普遍使用的十进制小数同计算机中使用的二进制存在如下关系:

$$(0.x_1 x_2 x_3 \cdots x_L)_2 = \left[\sum_{i=1}^{L} x_i 2^{-i}\right]_{10} \tag{2}$$

其中 $x_i = 0$ 或 1。对式(2)右边进行变换:

$$\sum_{i=1}^{L} x_i 2^{-i} = \sum_{i=1}^{L} x_i 2^{L-i-L} = 2^{-L}\sum_{i=1}^{L} x_i 2^{L-i} = 2^{-L}X \tag{3}$$

其中 $$X = 2^{L}\sum_{i=1}^{L} x_i 2^{-i} = \sum_{i=1}^{L} x_i 2^{L-i}$$

从式(3)可知,X 为一十进制整数,它是由一个十进制小数映射而来,而十进制整数在计算机中可用定点整数形式来表示。

将式(3)代入逻辑映射式(1)可得:

$$2^{-L} X_{n+1} = 1 - 2(2^{-L} X_n)^2 \tag{4a}$$

$$X_0 = 2^{L} x_0 \tag{4b}$$

对式(4a)作进一步简化得:

$$X_{n+1} = (2^{2L} - 2X_n^2)/2^{L} \tag{5}$$

这就是逻辑映射的整数表达式。在作者即将发表的另一篇文章中,对 L 为 64 位、32 位和 16 位分别进行了计算机编程模拟,证明当 L = 32 时,式(5)产生的序列仍然处于混沌态(而同样 32 位条件下,采用浮点运算得不到混沌序列)。当 L = 16 时,式(5)产生的序列已经脱离了混沌态,但经过一定的非线性变换仍可产生混沌序列,对式(5)进行微小的改动(即非线性变换)为:

$$X_{n+1} = (2^{2L} - 2X_n \times X'_n)/2^{L} \tag{6}$$

其中 $X_n = (X_nH)(X_nL)$, $X'_n = (X_nL)(X_nH)$, 即 X'_n 为 X_n 的高低字节互换后的 16 位二进制数。取 L = 16 比特根据式(6)产生数字混沌序列的流程图如图 7.3-1 所示。因此整数运算优于浮点运算,它降低了对计算精度的要求。产生 L 比特输出,只需运算 L × L 比特定点运算,加快了计算速度,从而减低了对硬件电路的要求。

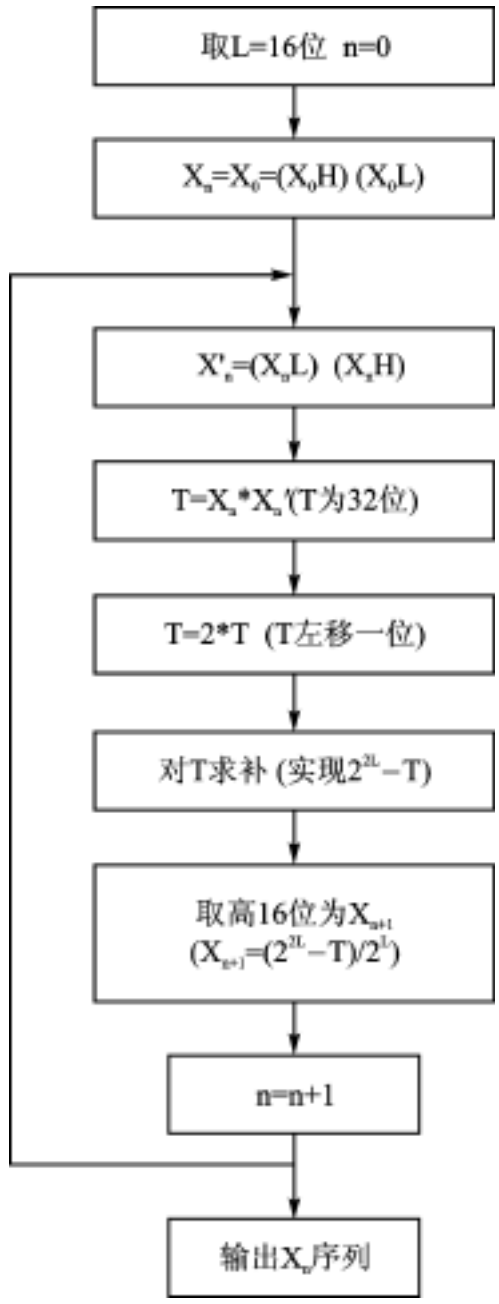


图 7.3-1 产生数字混沌序列流程图

二、数字语音混沌通信系统

我们利用上述数字混沌序列作为密码构建了一个有混沌加密装置的语音数字通信系统,系统框图如图 7.3-2 所示。

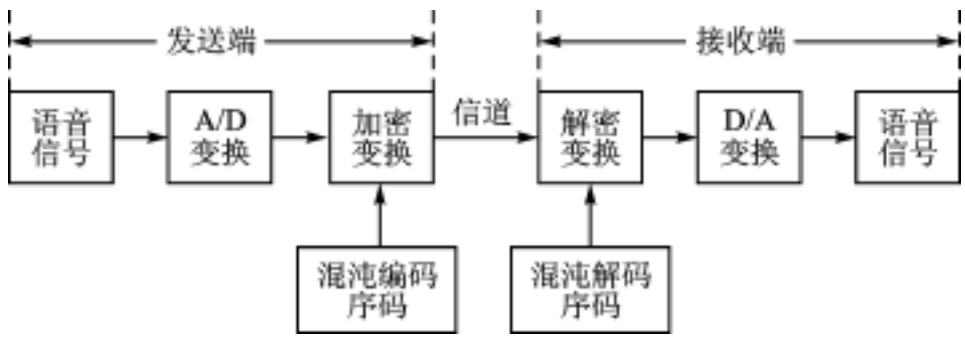


图 7.3-2 数字语音混沌通信系统框图

发送端加密过程为:

$$Y_{sn} = X_{sn} \quad U_{sn} \tag{7}$$

其中 U_{sn} 为发送端的语音信号，X_{sn} 为发送端的混沌编码序列，Y_{sn} 为发送端的混沌加密信号，为逐位模二加计算符。

接受端解密过程为加密过程的逆运算：

$$\begin{cases} U_{rn} = Y_{rn} \quad X_{rn} \\ Y_{rn} = Y_{sn} + S_n \end{cases} \tag{8}$$

其中，Y_{yn} 为接收到的加密信号，S_n 为信道的接收噪声，U_{rn} 为解密输出。显然，当通道噪声为零，收发双方采用相同的混沌系统(相同初条件的逻辑映射整数表达式(6))产生的编码序列进行加解密，即 $S_n = 0$，$X_{rn} = X_{sn}$ 时，就有 $U_{rn} = U_{sn}$。从而实现误码率为零的数字语音混沌保密通信。

三、硬件实现

我们运用单片机实现了上述数字语音混沌通信系统。单片微型计算机是微型计算机发展中的一个重要分支，它具有体积小、功能扩展性强、环境适应性强等独特结构和性能。用单片机实现数字语音混沌保密通信系统可以满足保密通信的隐蔽性、灵活性、保密性等要求。

我们运用 8031 单片机设计的数字语音混沌通信系统硬件结构图如图 7.3－3 所示。在发送系统中，从话筒输入的语音信号经过 4066 芯片采样保持，根据人类语音的频谱成分一般在 10 kHz 以下，而从一定可懂度考虑，只需保留 3.5 kHz 以下的频谱，故采样频谱选取 8 kHz；在 0809 芯片进行数模转换；在 8031 芯片编程产生混沌密码序列，与输入的数字语音信号进行加密运算，然后把密文发送到通信线路上(采用基带通信)。接收端的单片机 8031 芯片用于接收密文信号并产生与发送端相同的混沌编码序列，然后两者进行解密运算并把解密信号输出到 0832 芯片；在 0832 芯片对解密信号进行数模转换变成语音模拟信号并经过放大和低通滤波后推动扬声器播放，整个通信过程为实时通信。从扬声器中能够清晰地听到从发送端话筒输入的语音信号，但不可避免伴有一定的噪声，这是由量化误差和通道干扰引起的，其中主要原因是量化误差。为了说明该系统保密通信效果，在输入端，我们用信号发生器产生一 1 kHz 方波信号输入该系统，并编写了两套程序。一套程序在发送端对方波信号加密但在接收端不解密。另一套程序在发送端对方波信号加密且在接收端解密，从示波器可观察到两种输出结果，加密信号的时间波形杂乱无章，完全覆盖了原始信号，从扬声器听到的是一片噪声。信号的加密、传输、解密过程产生的时延很小，不影响实时通信；解密后输出的方波信号与输入方波信号比较出现了一定的失真，但主要是高频失真，因而该系统成功地实现了语音保密通信。

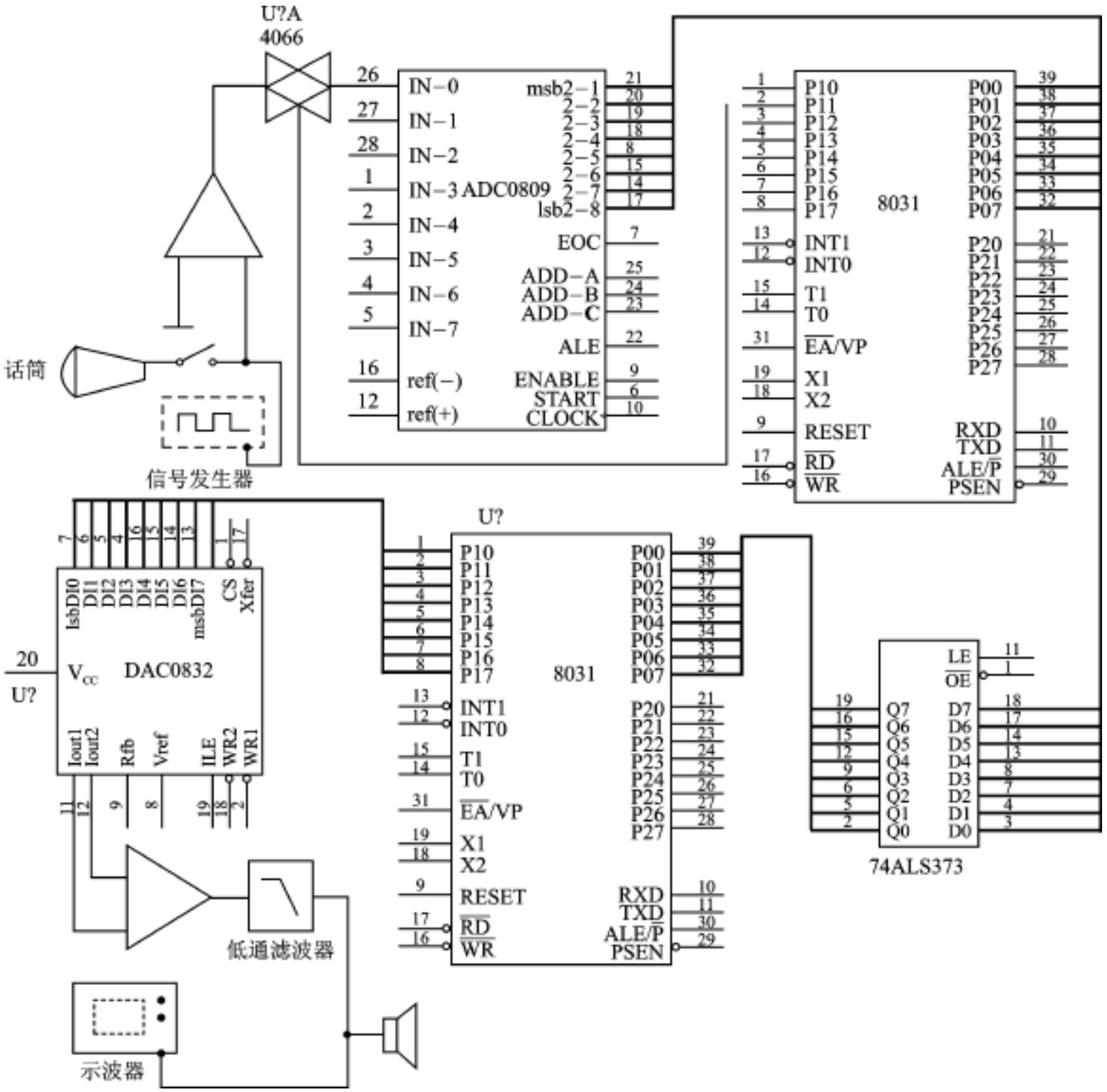


图 7.3-3 数字语音混沌通信系统硬件结构图

四、讨 论

在数字语音混沌通信系统的设计和实现中,数字混沌序列的产生是很重要的一个方面,它产生的方法直接影响到通信系统硬件的复杂程度。采用本文所述的由浮点运算变换到定点运算的逻辑运算的整数表达式来产生数字混沌序列,可以极大地加快计算速度,降低对计算精度的要求,因而可用广泛使用的 8031 单片机实现数字语音混沌通信系统。当然,要提高通信的效果,还要提高语音信号的采集速度,减少量化噪声,这就要求单片机具有较高 CPU 运行速度,可考虑使用高位单片机来实现该语音保密通信的硬件系统。同时,在选用高速 CPU 单片机的前提下,也可尝试利用上述语音数字通信系统对大数据量的图形、图象和多媒体信号实现保密通信或进一步改进系统为无线传输,以满足广泛的实际应用的要求。

参 考 文 献

1 atthews . On the derivation of a chaotic encryption angorithm .Cryptonogia, 1989,(4): 29～42

2 Pecora L M, Carroll T L . Synchronized chaotic signal and systems . IEEE ICASSP, in Proc . 1992

3 Cuomo K M, Oppenheim A V, Strogatz S H . Dynchronization of Lorenz-based chaotic circuits with applications to communications . IEEE Trans . on CAS, 1993,CAS - 40 (10):626～633

4 Halle K S, Wu C W, Itoh M, et al . Spread spectrum communication through modulation of chaos . Int . J . Bifurcation & Chaos, 1993,3(2):469～477

5 Dedieu H, Kennedy M P, Hasler M . Chaos shift keying: Modulation and demodulation of a chaotic carrier using self-synchronizing Chua's circuits . IEEE Trans . on CAS, 1993,CAS - 40(10):634～643

6 Frey D R . Chaotic digital encoding: An approach to secure communication . IEEE Trans . on CAS, 1993, CAS - 40(10):660～666

选自《电子技术应用》月刊,2000 年第 2 期

7.4 伪随机序列及 PLD 实现在程序和系统加密中的应用

西安交通大学工程与科学研究院生物医学工程研究所(710049)
张登福 林 刚 蒋大宗
西安空军工程学院四系信息处理实验室(710038) 毕笃彦

可编程逻辑器件(PLD)经历了 PAL、GAL、CPLD 和 FPGA 几个发展阶段。使用 PLD 具有设计灵活、调试方便、系统可靠性高等众多优点,并有利于硬件设计的保护,防止他人对电路的分析、仿照,使其成为科研实验、样机试制和小批量产品的首选方案。

随着计算机、单片机技术的发展和广泛应用,软件加密成为知识产权保护的重要手段。目前微机软件加密的方法可分为两大类:软加密和硬加密。软加密主要有密码方式、软件自校验方式、钥匙盘方式等多种。随着软加密的发展,解密软件也大量出现。硬加密由于具有加密强度大、可靠性高等特点,已广泛用于微机软件保护。硬加密将硬件和软件相结合来实现软件的加密,软件在运行时需与硬件正确交换数据,否则程序不运行,或不能执行主要功能,典型的产品有:插在计算机总线上的加密卡,接在计算机并口或 USB 口的软件狗(加密锁)、微狗等。软件狗大多用 E^2 PROM 存储密码数据,电路简单,成本低,但用 SOFT-ICE 等软件进行软件狗的解密和复制并不很困难。在这种情况下,软件狗内部增加了一个单片机称为微狗,通过对数据的处理来提高软件的加密强度;少数专业的硬加密生产商则采用独自的 ASIC 芯片。加密卡的原理与软件狗和微狗的相似,不同的是通过总线操作,使得设计更灵活、功能更强,只是安装不方便。

随着某一加密产品加密操作方式的公开,其解密也就为期不远。新出现的解密软件能模拟绝大部分软件与加密狗间的数据交换过程,从而达到解密;国产的加密卡和微狗,大多外用 E^2 PROM 存储配置数据和用户密码,获取这些数据就可能解密、甚至复制微狗,因此加密方法的独特性、手法的反常规性在加密应用中非常重要。单片机/ DSP 系统的控制及外围电路都相对简单, 对软件的跟踪比较容易分析硬件的设计思想和实现功能,而目前对其软件的保护并不受重视,并且单片机/ DSP 系统没有现成的加密产品,设计者应结合系统软、硬件的要求灵活决定。我们利用 PLD 器件本身的加密保护特性,由其产生的伪随机序列实现单片机/ DSP 系统和计算机应用程序的加密,具有简单方便,解密难度大的特点,同样可用 PLD 器件实现更复杂的加密方法如 EDS。

一、硬件加密的 PLD 实现原理

硬件加密必须综合考虑加密方法的可行性、有效性、硬件复杂度等因素,由逻辑电路产生大量密码的一种简单有效的方法是使用线性反馈移位寄存器,其产生的伪随机数据已广泛用于数据通信中的加扰、扩频、跳频和数据加密。图 7.4-1 所示是具有防跟踪、产生 $2^N - 1$ 个 N 位伪随机数据的加密电路原理框图,密码生成所用触发器和门电路少,并且密码是加电后动态产生的,不同的预置产生的数据不同,因此密码的强度、隐蔽性优于加密狗和微狗(卡)。

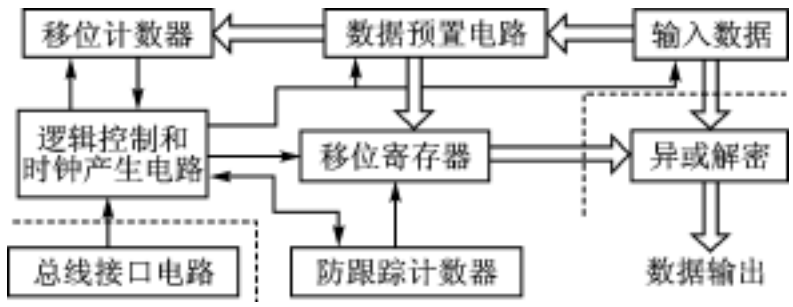


图 7.4-1 硬件加密原理框图

N 位移位寄存器产生的伪随机码作为读取的密码或输入数据解密的密钥,在移位时钟的作用下,可输出 2^N - 1 个 N 位的有效密码或密钥。对某一具体电路,移位寄存器初值不同时,这 2^N - 1 个随机数有 2^N - 1 种排序,可预置伪随机数产生器的原理电路如图 7.4-2 所示。要使移位寄存器产生一确定的值,首先置其初值,然后置移位计数器初值并允许时钟电路产生移位时钟;当移位计数器计满时,产生一个数据准备好状态可供软件读取,该状态同时阻断时钟电路,停止移位操作,并且启动防跟踪记数器的时钟产生电路;若在规定的时间内读取密码字或者写入待解密数将清零防跟踪记数器,否则超时使防跟踪记数器满导致触发器翻转,打开三态门,扰乱输出数据,虽然这时读/写数据能清零防跟踪记数器,阻断其时钟,但必须重加电或系统复位才能断开三态门。

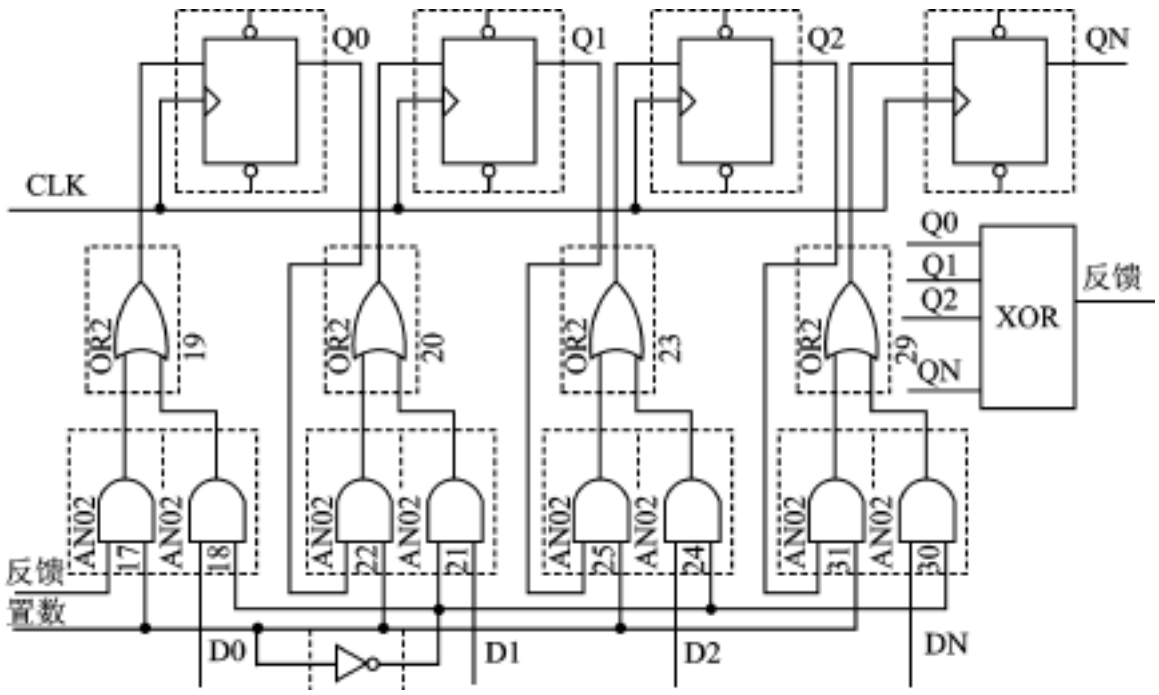


图 7.4-2 可预置伪随机数产生器原理电路

对加密电路的正确操作步骤是:在主程序中预置移位计数器和移位寄存器初值,在子程序的一处判断移位是否完成,另一处读密码或写待解密数,在另一子程序判断或取解密数据,以防止跟踪。

为在使用较少触发器情况下保证程序正常运行并能有效地防止解密跟踪,应恰当地选择防跟踪计数器的时钟,若用移位寄存器时钟源,则需要较高的分频才能满足高速的数据产生、适当的延时时间;因此在实际使用中,应根据需要,选择系统可能提供的低频持续脉冲信号作为防跟踪计数器的时钟。在微机系统中,尤其在 WINDOWS 操作系统的分时事件驱动运行模式下,由于系统固有的存储器刷新、时钟中断,以及运行中硬件中断、DMA 操作、任务切换,必须有较长的防跟踪延时才能保证合法程序正常运行,要注意在软硬盘操作时对密码数据操作的影响。

二、应用举例

1. 并行口加密电路

在 PC 机系统中,一般打印机并行接口包括单向输出的 8 条数据线 D0～D7 和四条控制线、5 条状态输入线,因此每次读操作只读取 4 位密码,其加密电路原理框图如图 7.4-3 所示。由于并行口不提供电源,将联机控制信号 SLCT 置高提供的电流很小,因此直接挂在并行口上的电路必须选用规模不大的低功耗器件。

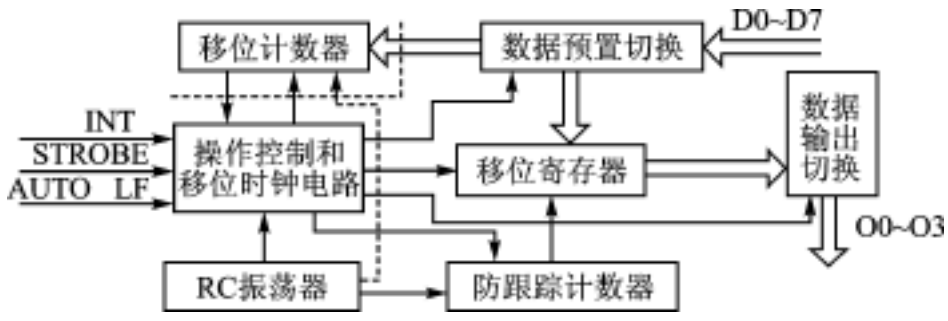


图 7.4-3 并行口加密电路原理框图

经并行口控制移位寄存器产生密码的基本操作步骤是:(1) SLCT 置高加电;(2) STROBE 触发经 D0～D8 写入 16 位控制字,选通加密电路,否则打印口正常;(3) 初始化信号 INIT 置低,STROBE 选择预置移位寄存器和移位计数器;(4) INIT 置高选通移位时钟;(5) 检测数据是否准备好;(6) 换行控制 AUTO LF 置高,在 STROBE 脉冲作用下数据按 4 位输出供 CPU 读取;(7) SLCT 置低断电。

防跟踪记数器由移位计数器满状态启动,其溢出脉冲使触发器翻转选通三态门,扰乱输出数据。防跟踪记数器和移位寄存器时钟可由门电路构成的 RC 振荡器产生。

2. ISA 总线加密电路

与并行口方式相比,通过总线方式对加密电路的移位寄存器和移位记数器的预置、输入数据的异或解密等操作更灵活、方便,并可与其他电路结合,其原理框图与图 7.4-1 相似。总线接口电路对端口地址和控制信号译码,产生移位寄存器和移位记数器输入写、移位寄存器及异或解密输出读信号,读信号清零防跟踪记数器并阻断其时钟。在 ISA 总线接口中,移位寄存器时钟用 OSC 或总线时钟 BCLK,而防跟踪记数器时钟可采用移位时钟或状态持续变化的总线控制信号及状态信号,如:地址锁存信号 BALE、刷新指示信号 REFRESH、DMA 操作允许信号 AEN(因用于 I/O 地址译码,隐蔽性好)等,用 REFRESH 信号时计数器规模小、运行可靠,但是切断该信号防跟踪功能不起作用,密码数据仍正常产生。

3. 单片机/DSP 系统程序加密电路

早期的单片机系统大都用扩展 EPROM 作为程序存储器,这种方式目前在高速单片机/DSP 中仍然很常用。对于程序量小,不需外部程序寻址的简单系统(如微狗),采用内置 EPROM/FLASH RAM、带加密控制字的 MCU(如 GMS97C2051)本身就能可靠地保护程序;因此我们的讨论只限于用扩展程序存储器的系统。

对扩展程序存储器的加密保护可通过对其数据和地址线的异或/取反扰乱来实现,其目的都是不能直接获取程序存储器内部保存的数据。由于 X=X K K,X=X,因此在系统工作时由硬件实现代码和密钥的异或/取反操作可得到正确的程序(文献[1]中用二级异或提高加

密强度的观点不正确,因为 Y = X K1 K2 = X K),其原理如图 7.4-4 所示。通常单片机加密的方式是密钥固定不变,或 CPU 读取程序代码的同时,从另一片 EPROM 读取密码,使每一代码对应一密钥。这两种方式的解密只需用简单的组合逻辑电路,对前一种方式,用逻辑分析法很容易求解逻辑关系而解密,后一种方法进行逻辑分析的工作量虽然大大提高,但密钥本身容易被直接获取。因此我们用 m 序列产生器动态产生密钥,将解密的组合逻辑电路与时序逻辑相结合,而较复杂时序逻辑的分析是很困难的。

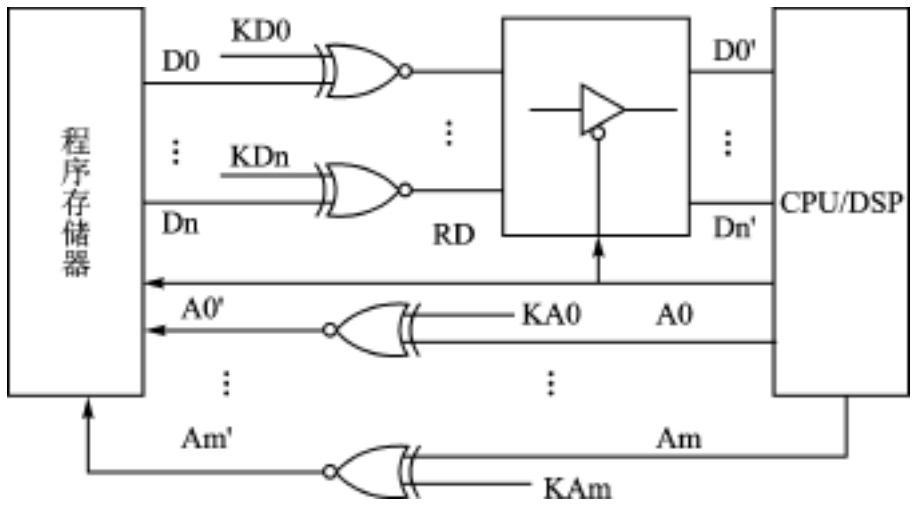


图 7.4-4 单片机/ DSP 异或加密原理电路

对 8031、MC6805 兼容系列的单片机系统,编程使开始的一段初始化程序顺序执行,系统复位时自动对移位寄存器设初值,复位后程序存储器的读信号同时作为移位时钟,使每条指令的密钥不断变化;在第一次执行循环、跳转指令前,程序发控制字阻断移位时钟,使以后的程序密钥相同。在高速 DSP 系统中,一般上电后将低速 EPROM 中的程序加载到高速 SRAM 中运行,可使 EPROM 的读控制信号一直作为移位时钟,使密钥不断变化;如果用串行口方式加载,程序解密操作与数据通信中的数据解密相同,数据输出时钟直接作为移位时钟。单片机/ DSP 作为微机系统的协处理器时,单片机/ DSP 的代码一般经总线装载,可将加密代码与移位寄存器输出的密钥异或解密,输出到单片机/ DSP 的程序 RAM。

系统运行时,用于 MCU/ DSP 程序保护的防跟踪计数器时钟一直有效,这样可以防止仿真器的跟踪。防跟踪计数器要用程序读或其相关信号清零,有的 DSP 从内部 RAM 运行程序时,程序读无相应输出信号,这时可用定时器中断或程序中及时插入的代码来清零。

上述介绍应用移位寄存器产生伪随机数据对程序进行加密的一些方法,曾在我们设计的系统中得到验证,整个电路的设计不复杂,占用 PLD 器件的资源不多,完全可结合在系统的硬件逻辑设计中。使用 8/ 16 位的移位寄存器时,密钥量有限,制约了加密的复杂度,使用者应根据设计要求和自己的经验,引入各种非常规的操作方式,这样就可以用简单的硬件电路,很好地实现软件和系统的保护。

参 考 文 献

1 王茂 . 单片机系统的加密技术 . 计算机工程与应用,1997(11)

2 董渭清,王换招 . 高档微机总线接口技术 . 西安:西安交通大学出版社,1995

选自《电子技术应用》月刊,2000 年第 6 期

7.5 增强单片机系统可靠性的若干措施

吉林医学院附属医院(132011) 肖丽君

一、监视定时器与复位电路

监视定时器(W.D.T 或 Watch Dog)提供了一种使单片机系统从偶然干扰造成的瞬时故障中自动恢复的能力。未应用监视定时器的系统显然不具备很高的抗干扰能力。它是单片机系统抗干扰措施的最后一关。

使用中常被忽视的有以下四点。

(1) 与复位电路的阻抗匹配要求 RESET 信号源有较大内阻,以保证监视定时器能在规定的时间内完成下拉(或上拉)复位端,否则监视定时器不起作用。典型电路如图 7.5-1、图 7.5-2 所示。

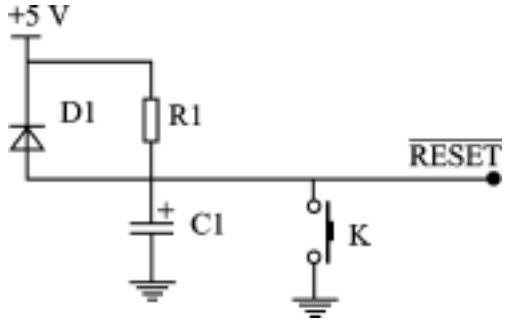


图 7.5-1 典型电路

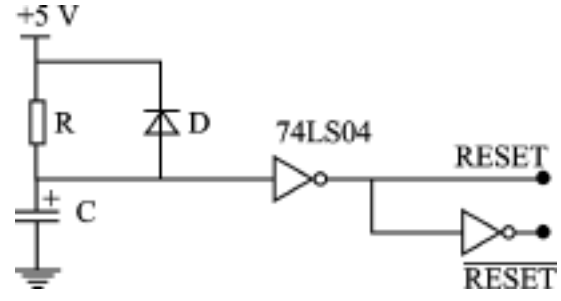


图 7.5-2 典型电路

图 7.5-1 可以满足监视定时器对复位电路阻抗的要求,而在图 7.5-2 则不能。改进方法有两种:第一种方法是简单地将非门 74LS04 换成集电极开路门,如 74LS06,但要在电路输出端接一个上拉($\overline{\text{RESET}}$)/下拉(RESET)电阻(1 kΩ)及对地去耦电容(0.01 μF),以提高抗干扰性能;第二种方法是在图 7.5-2 输出端串拉一个电阻(阻值 510 Ω)。

(2) 外围电路复位要受控于监视定时器。

这是必须的,其实质是复位同步问题,如图 7.5-2所示。该电路如果用在 8096 系列与 8255 芯片或 HD61830 液晶显示控制器等构成的单片机系统则十分不利。监视定时器使($\overline{\text{RDSET}}$)8096 单片机复位但不能复位($\overline{\text{RESET}}$)8255,则构成可靠性隐患。可如图 7.5-3 修改,当然如果 PCB 板上有剩余的非门,再在图 7.5-2 上增加一级非门也可以。

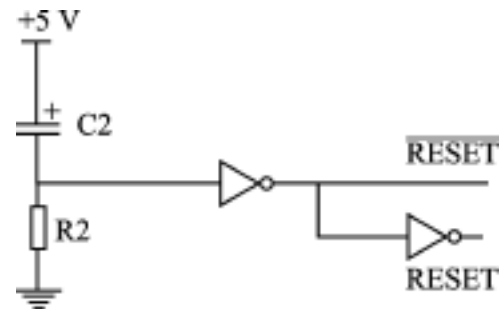


图 7.5-3 修改后的电路图

(3) 重要参数检查

监视定时器也不是万能的,虽然按照使用原则保证了复位监视定时器的指令只在全指令

空间出现一次,但仍有可能在某种情况下程序进入特殊的死循环而包括了这部分指令,这就要求对程序中影响程序跳转判断等重要参数进行检查。

也可能由于瞬间干扰改变了寄存器中的某些参数,而使程序虽能循环执行,但其某些功能出错,如图 7.5-4 所示。重要参数检查要放在复位监视定时器指令之后,足以保证不论程序怎样错误循环只要执行了复位监视定时器的指令,就一定能执行重要参数检查部分以期从错误中恢复过来。例如 OMRON 可编程序控制器程序流程就是这种设计思想,它甚至也包括硬件检查。虽然 OMRON 的流程中因程序运行时间关系三处复位监视定时器,但这并未影响其卓越的可靠性(见参考文献[2])。

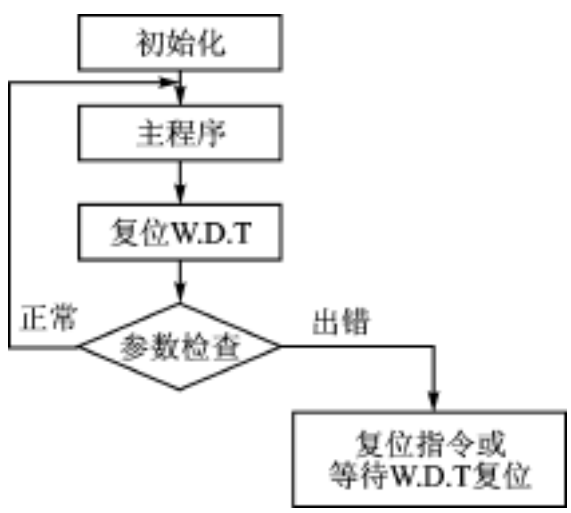


图 7.5-4 重要参数检查程序

(4) 复位状态下的输出如果发生干扰,监视定时器复位,单片机及外围电路均复位(I/O 一般置为输入状态),然后程序又正常运行。一般来看应该没问题了,但是对在工业上应用于继电控制,尤其是具有联锁(INTERLOCK)应用的系统则不一定。一瞬间的复位使个别输出端产生扰动从而改变了重要状态或错开/关了继电器从而引起事故的例子是有的。一般来说这个问题可通过增加锁存器来解决,如图 7.5-5 所示。其思想是只要锁存信号(或称时钟信号)无效,则输出不会转变。该方法给输出的稳定提供了保障,是有意义的。当然,具体电路应视单片机、输出芯片、锁存器型号及被控制电路等多种因素具体考虑。

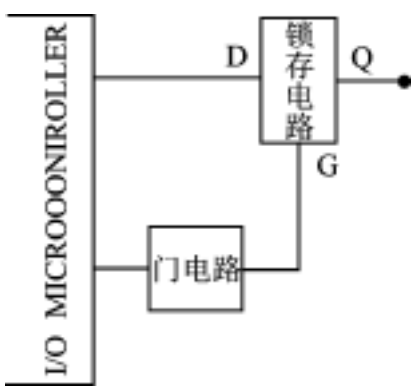


图 7.5-5 复位状态下的输出

二、电源问题

单片机系统对电源的要求表现在可靠性方面主要有长期安全运行及抗干扰两点要求。目前,设计得好的正品电源已完全可以可靠运行,且对串、共模干扰有很好的抑制。如果供电质量太差,则可在外部串接干扰抑制器及隔离变压器,重要场合甚至使用 UPS,但问题有时出在用电一方的内部干扰。

合理的应用应将数字电源、扫描显示电源、模拟电源及大功率大干扰电源在变压器处相互隔离。如因体积、成本等因素未能做到,则一定要分别独立取电、独立滤波、去耦,以防止相互干扰。实际中有些莫明其妙的干扰往往来自内部。

三、合理设计 LED 驱动

单片机类产品中,显示器件选用 LED 数码管的较多,设计中以下三个问题要给予足够重视。

1.驱动能力

许多公开发表的文章或批生产仪表包括单片机开发机常使用 74LS04、74LS164、8255 及其他一些 TTL 芯片以低电平灌电流作单个发光管或数码管的段驱动。其理论依据是 TTL 电路的低电平输出灌电流特性足以提供 LED 及共阳极数码管段显示的驱动电流(10 mA 左右),尤其是市场上有许多高亮度 LED 器件更是如此。

然而这显然是一种不可靠的设计思想。

首先,为满足可靠性要求,元件参数要求降额(30～50)%使用是公认的原则。而上法不仅不降额使用,反而超额使用,这会使器件寿命大大降低。例如:常用的 8 位移位寄存器 74LS164 芯片最大允许使用电流 27 mA(参考文献[3]),满额使用每个段位也只能分配不足 4 mA,这种应用既达不到一定的显示亮度要求,又降低了可靠性。8255 等芯片亦存在同样问题。驱动能力略强一些的芯片有 7406、7407、74244、74245、7446-49、74246-249 等,对环境亮度不高的场合它们基本上能满足小于 0.5 寸数码管与小于 Φ5LED 的驱动,但在中强光直射下几乎看不清。其实最好的驱动结构是使用储存成功率驱动电路或三极管驱动。

2.扫描显示电源

扫描显示电源要单独取电、单独滤波,防止对其他电路造成干扰。

3.显示亮度的环境适应性

同一个显示器,因其所处环境的照度不同,在使用者看起来,其显示亮度也就不同。结合面板的反射光考虑,其反差也不同。在一种环境下使用者感觉很好的显示效果,换种环境则让人无法接受(看不清或太刺眼)。这种矛盾使多数设计人员采取了以下保守的设计方法:宁亮、忽暗或折中。这在多数工业场合是可以接受的,因为工业现场在仪表盘的设计、安装、环境照明等方面已考虑过这个问题,对民用产品则更不适用(请读者注意一下身边的电扇、电视、VCD 等电器的指示灯)。要解决这个矛盾对应用单片机的电器早已谈不上技术难度了,这无非是个测量环境亮度,然后调节显示驱动电流或扫描开通时间比的过程,最简单地可以用两段调节法(高亮/低亮)。人机接口性能不容忽视,这是当今设计的原则。在可靠性探讨中论及此事是因为它涉及到三.1 节中关于提高驱动能力必要时的观点。

四、几个小问题

(1) 不要试图用低精度的 A/D 结合 D/A 转换器来增加 A/D 转换分辨力。虽然一些文章这样介绍,但那是满足不了精度要求的,尤其是漂移。

(2) 单片机系统外壳　作为一件产品,外壳的问题其实不是小问题,国际上现在有一种趋势,外壳多以塑料为主,包括很多可编程序控制器(PLC 其实也是一个单片机系统)也采用这种方法。但从可靠性角度出发这种设计是不可取的。

第一,从电磁兼容性角度来看,无法较好的屏蔽静电、高频及电磁干扰。在多种仪器混用仪表盘,尤其是当今无线通讯设备大量使用的情况下更是如此。

第二,增大了散热与防尘的难度,而散热与防尘两项设计都对电路长期运行可靠性有很大

影响,不仅对单片机小系统,就是国际大型自控设备公司如:西屋公司(WestHouse)设计的计算机集散控制系统(DCS)WDPF 系统的前几个版本就是因散热与防尘设计不合理而影响了长期运行的可靠性与维护性。而采用金属外壳工艺则可给解决以上问题带来方便。

(3) CMOS 与 TTL 目前来看 CMOS 集成电路的许多优点正在征服着电子工程师门,而且实践也证实了这一切,然而实践也同样发现它的可靠性较低。

(4) 在必要的位置设计去耦电容是一个好习惯,尤其是复位电路及重要 IC。

(5) 复位按钮 由于不可能绝对保证软件可靠性及对干扰的担心,大部分产品操作面板上提供复位按钮(从电子词典到智能仪表到个人电脑到……),这样做是正确的,但要注意不能将干扰由这个通路引入系统。

可靠性是产品质量中最重要的一项,如果出了问题,其牵扯的精力、浪费的时间、损失的金钱、失去的机会太多了。不能长期可靠使用,再好的高科技也是零。尤其是初入此行的工程师,对可靠性的充分认识及研究是保证走好第一步的关键。

参 考 文 献

1 魏庆福 . STD 总线工业控制机的设计与应用 . 北京:科学出版社,1992

2 O MRON(中国)有限公司 . O MRON 可编程控制器安装手册 . 北京:O MRON(中国)有限公司出版,Cat . No . zc196201A:5～6

3 陈清山 . 最新世界 TTL 数字集成电路及互换手册 . 长沙:中南工业大学出版社,1991

选自《电脑开发与应用》月刊,2000 年第 4 期

7.6 FPGA 中的空间辐射效应及加固技术

清华大学电子工程系(100084) 李冬梅 王志华 高文焕 张尊侨

电子系统在航天领域的应用越来越广泛,而空间辐射环境对电子系统的影响是不可忽视的。辐射会使器件的性能参数发生退化,以至失效,影响卫星的可靠运行,缩短卫星的寿命。据卫星资料统计,其异常记录中有 70%是由空间辐射环境引起的。

随着航天电子技术的发展,ASIC 开始受到设计者关注,尤其是可编程逻辑器件。

可编程 ASIC 中的现场可编程门阵列(FPGA)将半定制的门阵列电路的优点和可编程逻辑器件的用户可编程特性两者结合在一起,不仅包含大量的门电路,使设计的电子产品达到了小型化、集成化、可靠性高、速度快,而且为设计者提供系统内可再编程(或可再配置)的能力,使新一代电子系统具有极强的灵活性和适应性,可为许多复杂的信号处理和信息加工的实现提供新的思路和方法。同时大大缩短了设计周期,减少了设计费用,降低了设计风险。因此 FPGA 已经成为可编程 ASIC 中颇受宇航电子设计者们欢迎的一类器件。

近年来出现了不少抗辐射加固类型的产品,但由于成本较高,所以一些非加固的普通商用/军用产品仍然具有很强的吸引力。

一、空间辐射环境

近地空间是一个强辐射环境,主要包括太阳的电磁辐射及粒子辐射。

太阳的电磁辐射包括射线、X 射线、紫外线、可见光、红外线、微波及无线电波等。

粒子辐射是对空间飞行器影响最严重的环境,其来源主要有三种。

1. 地球辐射带

在地球周围,存在着被地磁捕获的大量带电粒子,这些粒子所占据的区域称为“辐射带”。

2. 太阳宇宙线

太阳风:太阳日冕层喷射出的高速粒子流。主要成分为质子,强度随太阳活动周期变化。

耀斑:伴随着大量高能带电粒子的发射,称太阳粒子事件。主要成分是质子,其次是 α 粒子,重离子。

3. 银河宇宙线

来自银河系的高能带电粒子。主要成分为质子,其次是 α 粒子,各种元素的原子核、重离子和微量电子。

空间电离辐射环境极其复杂,包括所有自然界中的元素的原子核,从质子(原子数 z = 1)到铀(z = 92)。这些粒子谱数值上达 15 阶,能量大于 10^{21} eV。其中大多数是全裸的,但另外一些离子在到达地球时仍保持着一定的原始电荷分布。其密度、组成、频谱随时间、地点及到达时的方向而变化。

二、FPGA 的类型

FPGA 的结构主要分为三部分:可编程逻辑块、可编程 I/O 模块、可编程内部连线。

可编程逻辑块和可编程互连资源的构造主要有两种类型:即查找表类型和多路开关型。

查找表型 FPGA 的可编程逻辑单元是由功能为查找表的静态存贮器(SRAM)构成函数发生器,由它来控制执行 FPGA 应用函数的逻辑。M 个输入的逻辑函数真值表存储在一个 2M×1 的 SRAM 中,SRAM 的地址线起输入的作用,SRAM 的输出为逻辑函数的值,由此输出状态控制传输门或多路开关信号的通断,实现与其他功能块的可编程连接。

多路开关型可编程逻辑块的基本构成是一个多路开关的配置。利用多路开关的特性,在多路开关的每个输入接到固定电平或输入信号时,可实现不同的逻辑功能。大量的多路开关和逻辑门连接起来,可以构成实现大量函数的逻辑块。

FPGA 由其配置机制的不同分为两类:可再配置型和一次性编程型。

一次性编程器件多采用基于反熔丝结构。反熔丝是在两层导体之间的一层很薄的绝缘介质。每个反熔丝占有等效于一个接触孔或通孔的面积,在电压加到此元件上时介质击穿,从而把两层导电材料连在一起。

可再配置器件主要为基于 SRAM 结构。利用 SRAM 单元来控制晶体管开关。每个晶体管开关的状态都由相应的 SRAM 中的值为确定。片上 SRAM 是配置存贮器,用来存储逻辑单元阵列(LCA)的配置数据。配置存储器控制功能、布线、特性、时序、I/O 驱动等。基于 SRAM 的 FPGA 在商业领域已经得到了广泛的应用,它是通过将状态信息载入 SRAM 单元来实现配置。为用户提供了最大的灵活性,使得系统内或在轨编程成为可能。另外,这类器件提供了再配置计算平台以最大功率;提供了改变需求的灵活性,以及纠正逻辑错误和恢复飞行中的故障的潜力,已成为空间飞行器电子器件的发展主题之一。

三、FPGA 中的辐射效应

FPGA 生产者所采用的不同的工艺和结构直接影响其辐射特性。信息存储结构的类型选择将决定辐射性能,并将决定关键功能的特性。

1.基于反熔丝型 FPGA 中的辐射效应

基于反熔丝型 FPGA 常用于非易失性空间飞行应用场合。反熔丝介质是一薄层 ONO 夹层,约为 80～90 埃的等氧层厚度。FPGA 设计中会有一定百分比的互连通道由介质击穿形成。当两个互相交叉的导体的逻辑层不同时,非编程的反熔丝就会存在偏压,这显然有赖于任务周期和两个信号的相位。

当工作在 5.5 VDC 限制范围内时,偏压反熔丝的隔离 ONO 中的电场强度大约为 6 MV/cm。当一个重离子撞击偏压反熔丝并随之发生穿通时,就形成了单粒子介质击穿,造成不期望出现的局部连接。这种连接可能表现为:

(1) 只是小电流增加(可能会增加故障率)。

(2) 由于减少的时间和电压裕量而引起的间断。

(3) 硬错误。

显然,一个特定单粒子介质击穿的严重程度取决于对反熔丝击穿环境下电路的考虑。

2.基于 SRAM 型的 FPGA 中的辐射效应

发生于基于 SRAM 型的 FPGA 中的较严重的辐射效应为配置翻转。

配置位容易受单粒子效应影响而发生翻转,并可能导致对集成电路控制能力的丧失。当受到重离子辐射时,器件会明显地失去所有的功能,直到电源重新启动,并伴随着器件电流的

变化。结构特性是影响FPGA辐射敏感度的因素。如:上电复位电路可能会翻转并改变再配置器件的状态。这对于用在关键部位的基于SRAM的门阵列具有重要意义。如果在基于SRAM的FPGA装载了错误的配置将会毁坏器件。系统资源可能会由于失去了对三态总线的控制或触发了系统的关键事件而导致崩溃。

基于SRAM的FPGA中的单粒子翻转很大程度上决定于工艺、结构特点和系统设计。例如:一个单粒子效应可以使芯片的两个输出驱动端口相连接,结果是出现意想不到的高电流状态,其电流密度超出可靠工作的要求。另外,总线与内部三态总线的冲突可能会引起过载;上拉电阻和三态总线的绝缘导致输入端和振荡器的悬浮;改变输出摆率会使时序混乱或进入暂稳态;将输入模块改为输出配置,引起FPGA和(或)板上其他元件的损坏;逻辑错误会使系统板发生故障,或是被检测到而进入保护状态并重新配置,或造成久损坏,若在关键电路中则发生状态改变。

在电路级,单粒子效应对基于SRAM的FPGA中的配置存储器有很强的影响。例如:常用的基于SRAM的FPGAXC4000的I/O单元,其器件的关键功能属性由SRAM控制。包括上拉和下拉电阻、输入阈值、输入和输出时钟极性、输入延时、输出极性、I/O是直通或寄存、该单元为输入或输出。对系统级的影响有:由于上拉电阻的使能使得功率微弱增加;由于改变输入延迟导致操作间歇;由于改变输出极性产生错误结果;由于被用来作高阻输入的单元的三态缓冲级被使能而导致系统崩溃。在器件内部,对于有些结构,配置SRAM翻转能导致布线网络中的驱动器冲突,三态总线中的总线冲突,若上拉电阻没有连接则总线悬浮,等等。这些单粒子效应可能使器件由于过载而牺牲硬件可靠性。

四、抗辐射加固措施

将FPGA应用于辐射环境是相当具有吸引力的工作。

1.基于反熔丝型FPGA的抗辐射加固措施

(1) 工艺加固措施

主要方法是增加反熔丝的厚度,并对输入缓冲级进行改进。

(2) 电路设计加固措施

目前在抗辐射加固电路设计中较多采用冗余技术来实现对故障的检测和隔离。主要包括三模冗余;复制及比较;编码及自查等方法。这些方法利用的是硬件冗余、信息冗余及时间冗余。

在基于反熔丝型FPGA中,由于其硬件资源较充裕,可以采用三模冗余及编码技术。

2.基于SRAM型FPGA的抗辐射加固措施

(1) 工艺加固措施

作为基础工艺,SRAM配置存储器比其他工艺具有明显优势,但是,也有很明显的结构上的弱点。即使考虑了各种门计数方法以后,对于商业器件,这些基于SRAM的器件现在仍然处于临界密度。因此,它不可能象其他FPGA用户存储器那样利用三模冗余或海明码来克服单粒子效应。而需要对非常大量的单元做单粒子加固。另外,还可以在FPGA内部采取辐射监控措施,随时检测和纠正错误配置。

(2) 电路设计加固措施

系统板上可以包括检查FPGA配置的逻辑。这可以通过如下方法实现:读取其中的内容

或让 FPGA 计算一个其内容的校验和与存在一个可靠寄存器中的计算值相比较。

当电路级器件的状态发生改变时,就需要重复再加载的过程。这需要错误监控电路来保证满足系统的可靠性,通过对配置恒定地监控来实现。这些可靠的电路将明显地消耗有效的版面空间,除非在 FPGA 内部采取了辐射加固监控措施。

单粒子容错的应用将使用监测电路来确保配置存储器内容的正确,以及在出现错误的情况下进行纠正。必须要保证不会发生永久性的电路故障。如果有必要进行重载或部分重载,系统设计必须能够允许电路运行过程中的暂停,同时要禁止任何错误的信号传播到系统的关键部分。

随着器件的几何尺寸进一步缩小,硅的成本在降低,也许可以使为每一配置位提供一个三模冗余加法表决器成为可能,通过非插入式的片上后台进程刷新配置存储器来实现。

参 考 文 献

1 atz R, Wang J J, Koga R, et al. Radiation effects on current field programmable technologies. IEEE transactions on nuclear science. 1997 Vol. 44(6)1945～1956

2 Katz R, Barto R, McKerracher P, et al. SEU hardening of FPGAs for space applications and Device characterization. IEEE transactions on nuclear science. 1994 Vol. 41(6):2179～2186

3 Gary Swift, Katz R. An experimental survery of heavyion induced dielectric rupture in Actel Field programmable gate arrays(FPGAs). IEEE transactions on nuclear science. 1996 Vol. 43(3): 967～972

4 Katz R, Wang J J, Koga R, et al. Current radiation issues for programmable elements and devices. IEEE transactions on nuclear science. 1998 Vol.45(6): 2600～2609

5 Xilinx Product Overview. 1998

选自《电子技术应用》月刊,2000 年第 8 期

7 .7　一种双机备份系统的软实现

上海交通大学自动化系(200030)　徐立云　邵惠鹤

当今社会对计算机系统可靠性的要求越来越高,计算机系统的可靠性是指计算机不发生故障,或即使发生故障也不影响系统正常工作的可能程度。根据可靠性的基本理论,容错就是对系统中可能发生的突发事件的控制。容错技术是近年来人们对计算机系统可靠性理论研究和实际工作提出来的一种有效手段,产生了如多处理机、磁盘阵列等容错技术,均取得较理想的效果,但大都成本较高或较复杂。本文要介绍的是利用串行口进行纯软件双机热备份的一种方法。

一、基本原理

双机热备份系统的主要功能是确保系统的不间断运行,主机和从机可以通过网络、串口、SCSI 等通道相互监视各自的运行情况,一旦某台机器发生故障,另一台机器将迅速自动接管它的全部资源,从而保证了系统的不间断性,也保证了系统数据的完整性。

众所周知,Windows 95 操作系统的特点之一就是由系统统一管理整个系统的软硬件资源,使多个应用程序可以共享。在 Windows 95 环境下,操作系统没有向用户提供控制串行口的硬中断,用户一般只能采用查询的方式来访问串口,这种方法过多占用 CPU 时间,浪费系统资源,仅适用于一些实时性要求很低的地方。为了保证系统对串口数据反应的及时性,我们引入多线程的思想。

Windows 95 操作系统支持基于多线程的抢先式多任务。线程是计算机执行任务的基本单元,一个进程可以有多个线程,系统在竞争的线程中间分配 CPU 的时间片。当前执行的线程在其时间片耗尽时挂起,让进程中的其他线程运行。当系统从一个线程切换到另外一个线程时,它保留所挂起线程上的上下文,并恢复队列中下一个线程的上下文。线程的优先级别不同,执行的顺序也不一样,系统首先响应优先级高的线程,再响应优先级低的。由于 CPU 的时间片很细小(大约 20 ms),从宏观上看起来好像是多个线程在同时运行。另外在同一个进程中,所有线程共享统一地址空间,可以访问进程中的全局变量,这样可以大大地简化线程之间的通信。

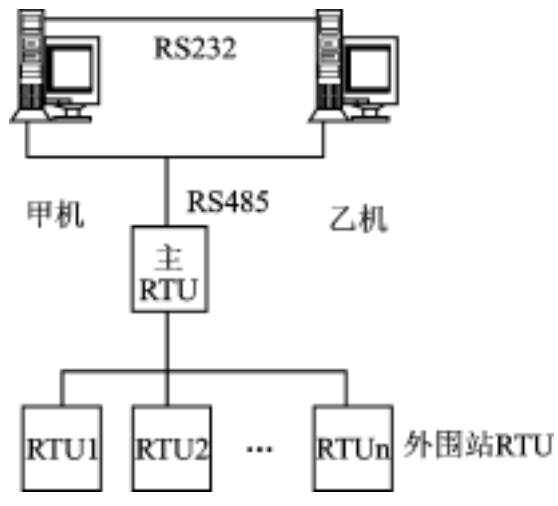


图 7 .7 - 1　双机备份系统简图

纯软件双机热备份系统就是采用多线程的编程思想。在应用程序中,专门生成一个辅助线程负责串口的读写,而主线程完成辅助线程的建立及其数据的计算、显示,主线程、辅助线程之间数据的交换是通过全局变量来进行的。如图 7 .7 - 1 所示,采用 CIM-PAC SCADA 系统,主 RTU 控制外围各站 RTU 间的数据传输和接收,并通过 RS485 和上位机

进行数据的通信和计算。当系统运行开始时,甲、乙两机均正常运行,同时接收从 SCADA 传来的数据。此时,甲机是主机,完成数据的计算和显示,并在其辅助线程中发送实时数据给乙机,乙机是备援主机,负责监控主机的任务,在其辅助线程中完成串口数据读取。当乙机接收到的数据为有效数据时,则说明甲机运行正常;如果乙机接收到的数据为无效数据时,或连续 5 次接收不到数据,则判断甲机处于非正常状态,提示用户对甲机进行检查,同时置本机器为主机,完成相应计算、显示的功能。当甲机故障排除后,它作为备援主机和乙机(此时乙机是主机)进行通信,重复前面相同的功能。为了确保备援主机处于正常工作状态,在备份主机的辅助线程中设立一个软件定时器,如果定时器溢出,则说明备援主机发生故障,提示进行修复。

二、相关函数简介

在 Windows 95 环境下,通信资源是提供单个双向、异步数据流的物理或逻辑设备,串口就是一个通信资源。对于每一个通信资源来说,有一个服务提供,由库或驱动器组成,它使应用程序能访问这些通信资源。WIN32 API 提供了一组对串口进行访问的通信函数。

(1) 打开、关闭串口。

```
ANDLE CreateFile(
 LPCTSTR lpFileName,                          / 要打开的串口号
 DWORD dwDesiredAccess,                       / / 对串口的操作模式
 DWORD dwShareMode,                           / / 共享方式
 LPSECURITY_ATTRIBUTES lpSecurityAttributes,  / / 安全属性
  DWORD dwCreationDistribution,               / / 打开方式
  DWORD dwFlagsAndAttributes,                 / / 属性
  HANDLE hTemplateFile
  );
BOOL CloseHandle(HANDLE hCom);                / / hCom 是要关闭的串口句柄
```

(2) 串口资源的配置。在 CreateFile()函数打开串口时,系统根据上次打开串口设置的数值配置串口,如果串口从来没有打开过,就采用系统的缺省配置。用 GetCommState()函数得到当前的串口 DCB 结构,然后用 SetCommState()函数设置指定的 DCB 结构。

(3) 串口的读、写操作。串口读写用 ReadFile()、WriteFile()两函数来完成。

```
OOL ReadFile(
  HANDLE hFile,                               / / 串口句柄
  LPVOID lpBuffer,                            / / 接收数据缓存区
DWORD nNumberOfBytesToRead,                   / / 读取数据的字节数
LPDWORD lpNumberOfBytesRead,                  / / 存放读取数据字节数的地址
LPOVERLAPPED lpOverlapped,                    / / 异步操作符方式数据结构地址
  );
```

WriteFile()函数的参数和 ReadFile()函数参数相差无几。

(4) 线程的创建。MFC 函数 AfxBeginThread 初始化 MFC 库创建新的线程,该函数运行时实际上是调用 API 函数 CreateThread 来启动线程的。函数原型如下:

```
winThread * AfxBeginThread(
```

```
    AFX_THREADPROC pfnThreadProc,                    //自定义新线程函数
    LPVOID pParam,                                   //传递参数
    Int nPriority,                                   //优先级
    UINT nStackSize,                                 //堆栈的大小
    DWORD dwCreateFlags,                             //线程运行标志
    LPSECURITY_ATTRIBUTES lpSecurityAttrs            //安全属性
            );
```

三、程序实例

由于篇幅原因,下面仅给出主线程、接收辅助线程的部分程序。

(1) 主线程:

```
  …
char InBuffer[5];
HANDLE hCom;
DCB dcb;
Int Count = 0;
UINT Com2Data;
……
hCom = CreateFile(
    "COM2",
    GENERIC_READ|GENERIC_WRITE,
    0,
    0,
    OPEN_EXISTING,
    0,
    NULL
            );
DCB Dcb;
GetCommState(hCom, &Dcb);                            //取得当前串口配置
Dcb .BaudRate = 9 600;                               //波特率设为 9600
Dcb .ByteSize = 8;                                   //8 为有效数据位
Dcb .Parity = NOPARITY;                              //无奇偶校验
Dcb .StopBits = ONESTOPBIT;                          //一个停止位
SetCommState(hCom, &Ccb);                            //重新设置串口配置
CWinThread * PWinThread;
PWinThread = AfxBeginThread(
    MyThread,                                        //创建辅助线程
    GetSafeHwnd( ),                                  //传递窗口句柄
    THREAD_PRIORITY_HIGHEST,                         //设置优先级
    0,
    0,
    NULL);
```

```
……
    CloseHandle(hCom);
……
```

(2) **辅助线程**:

```
  INT MyThread(LPVOID pParam)
{
   DWORD Num;
   SetThreadPriority(
     GetCurrentThread(),
     THREAD_PRIORITY_HIGHEST);
ReadFile(hCom, InBuffer,1, &Num, NULL);
   if(Num = 0)
     {Count + + ;
       if(Count > 5)
         {
           AfxMessageBox("主机发生故障,请检修 !");
::PostMessage(pParam, My_Message,0,0);                    //发消息,切换为主机
         return 0;
         }
     Com2Data = InBuffer[0];
       }
     Count = 0;
     return 0;
  }
```

四、结 论

在双机容错系统中引入多线程思想,大大降低了应用成本,并且串口操作简单方便,易于维护。实践表明:在数据采集频率要求不高的情况下,完全可以保证系统持续、稳定地运行,克服了以往串口通信反应不及时的现象,极大地提高了计算机系统的可靠性与安全性。

参 考 文 献

1 icrosoft Corporation . WIN 32 程序员参考大全(二) . 北京:清华大学出版社,1992

2 Stdfano Maruzzi . The Microsoft Windows 95 开发人员指南 . 北京:机械工业出版社,1995

3 Michacl J, Yong . Visual C + + 6 从入门到精通 . 北京:电子工业出版社,1998

选自《测控技术》月刊,2000 年第 8 期

7 .8　计算机系统容错技术的应用

中南工业大学信息工程学院(410012)　李　静

一、概　论

计算机的飞速发展使其在信息社会中占据了非常重要的地位。信息是人们前进的道路,计算机就好像电灯,有了它人们可以在信息高速公路上飞驰,否则,人们只能在黑暗中摸索前进。据统计,在商业市场中如果公司丢失了 10d 的数据,其中 65 %的公司会因此退出竞争市场。在目前信息社会中,信息的重要已经引起了商家的足够重视,不少的企业、商家、公司、机构等已经利用计算机来收集信息、数据,乃至分析、决策,在这些应用中最重要的是数据,数据是信息、是分析决策的根据,一旦数据丢失将造成企业停产、公司丢失市场、机构瘫痪的严重后果。笔者在多年的数据库应用的系统设计中已经越来越感觉到对数据进行安全保护的重要性。

对数据安全的要求,是否就是不允许有错误呢?任何事物不可能永远不坏,因此我们对数据安全性的研究就在于,一旦出现了错误如何及时处理、挽救。计算机系统的安全性是数据的安全性的首先条件,计算机系统的安全性分硬件安全性、软件安全性。软硬件的发展使商家认识到软硬件的可靠性是其占据未来市场的重要条件。IBM 公司就认为,计算机不具备容错性就无法占领市场。IBM 之所以能够占有全世界商用市场的最大份额,很大程度就是因为其机器性能可靠,用户信得过。微软公司最近也在中国投资兴建微软研究院,研究计算机容错和软件可靠性的问题。康柏公司去年收购容错机的最大生产厂——天腾公司,就可以看成是容错系统从专用计算机走向通用计算机的重要标志。

目前计算机系统容错技术在软硬件方面都有了很大发展,已经形成了一个独立的分支专业系统——容错系统,很多厂家也已有相应的产品,例如获得 96 美国 Comdex 最佳应用程序奖的 OCTOPUS 技术公司研制开发的 Octopus for Windows NT 软件,是双机热备份领域的领导者。

二、各种容错技术

1 . 数据备份

在计算机一开始应用于数据处理领域时,就有了数据备份的概念。数据备份指的是将计算机系统中硬磁盘上的一部分数据通过适当的形式转录到可脱机保存的介质(如磁带、软磁盘和光盘)上,以便需要时再输入计算机系统使用。脱机保存数据之所以重要,是由于以下两个原因:一是许多数据是不经常使用的,这些数据长期存储在硬磁盘上,既占用了宝贵的存储空间,增加了存储成本,又降低了存储设备的使用效率,降低了存取速度。为了能更有效地利用信息,通常把常用的信息放在联机的硬磁盘或磁盘阵列等设备上,组成联机的资料库,把不常用的、但有时又要检索的信息,放在联机的后备设备如磁带库、光盘库上。而大量的长时间不

使用的信息,则保存在脱机介质上——脱机保存。二是防止由于天灾人祸、计算机系统被破坏、相关信息的丢失、误操作、病毒或故意破坏行为等造成联机的数据丢失。为防范于未然,先将数据作备份保存,一旦发生事故,可及时调出备份,尽快恢复计算机系统的工作。

2.双机容错系统

当一个 CPU 板出现故障时,其他 CPU 板保持继续运行。这个过程对用户是透明的,系统没有受到丝毫影响,更不会引起交易的丢失,充分保证数据的一致性和完整性。系统的容错结构能够提供系统连续运行的能力,任何单点故障不会引起系统停机,系统提供在线的维护诊断工具可在应用继续运转的情况下修复单点故障。系统通过冗余的服务处理器统一监控系统所有设备的状态,监控服务器自身出现故障的概率可以认为是 0。系统运行及处理过程中,冗余的部件都在使用,处于"热"状态中,可以加快交易的处理,增加带宽,提高系统处理的速度和效率。

3.双机热备份

当 CPU 出现故障时由闲置状态的备份系统接替,但正在处理的交易有可能丢失,从而导致数据的不一致。系统关键部件如 CPU 的故障将导致主系统停止运行,对应用产生很大的影响当发生系统故障引起停机后,恢复运行时应用需要从磁盘或磁带上重新启动,需要耗费更多的宝贵时间。采用"心跳线"保持主系统与备用系统的联系,一旦"心跳线"部分发生故障,系统很难分清是"心跳线"还是系统其他部分的故障,往往需要人工干预才能解决问题,应用也将受到影响。备用系统的硬件和软件资源处于闲置的冷状态,浪费系统资源。

传统的高可靠性系统大多采用双机热备份方案。在双机热备份方案中,两台服务器都处于热机状态,如果一台服务器坏掉了,另一台服务器可以将所有的业务接管过来。双机热备份通常有两种方式,一种是 Online 方式,两台服务器都在工作,分别担负不同的任务,均衡负载。另一种是 Standby 方式,备份机不工作,只是监测作业机的工作状况。这两种方式各有利弊。Online 方式的投入成本大,管理上也存在一定难度;而 Standby 的缺点则在于服务器之间切换的时间较长,至少在 5 min 以上,而且前台的工作站还需要重新登录一次。

两个系统之间通过以太网连接,关键数据在两个系统之间呈镜像存在。在正常运行时,控制权在主用系统上,数据实时地镜像到备用系统上。当主用系统发生故障或主用系统检查到某种故障后,系统控制权切换到备用主机。由于采用以太网作系统的数据链路,主用系统可不干扰备用系统工作,自动脱离并在一个孤立的环境中进行故障的诊断和维修。主用系统修复后,控制权需再切回到主用系统。数据需要从备用系统恢复到主用系统,这个工作在后台自动完成,应用读取数据仍从备用系统上进行而不会中断。数据恢复完成后,双机系统进入正常工作模式。

4.三机表决系统

在三机表决系统中,三台主机同时运行,由表决器(Voter)根据三台机器的运行结果进行表决,有两个以上的机器运行结果相同,则认定该结果为正确。一般而言,三机系统的可靠性比双机系统要高,但三机系统也有其缺陷。首先三机系统的成本高是显而易见的,其次是当一台机器出现故障后表决器反复在剩下的两台机器中进行表决已经没有任何意义,其可靠性甚至比不上一个双机系统。因此现在三机系统中则较多采用的是将双机备份和三机表决两者结合起来的方式,当三机中坏掉一台后就当作双机备份系统来用,不再进行表决了。

5. 磁盘阵列技术

廉价的磁盘冗余阵列(RAID)技术是最常用的方法。RAID 技术将数据用各种校验算法处理后冗余存储在多块硬盘中,以此对硬盘进行物理容错。除提高了数据存储可靠性外,由于多块硬盘并行处理,还提高了硬盘的 I/O 速度,为解决 CPU 与 I/O 速度的瓶颈问题提供了一个有效途径。

实现磁盘阵列的主要方式有软件方式和阵列卡方式。前者如 SCO 虚拟磁盘管理(VDM),阵列管理软件运行在主机系统上。其优点是成本低,缺点是要过多地占用主机 CPU 资源,并且带宽指标上不去;而阵列卡方式把 RAID 管理软件固化在 I/O 控制卡上,从而可不占用主机 CPU 资源。

双机共享磁盘阵列系统是以磁盘阵列柜为中心的双机容错方案,磁盘柜通过 SCCI 线连接到两个系统上,并能被两个系统所访问。关键数据放在共享磁盘柜中,在正常运行时,控制权在主用系统上,当主用系统发生故障或主用系统检查到某种故障后,系统控制权就切换到备用机。主系统修复后,主备角色互换,双机系统进入正常工作模式。

6. 集群系统

均衡负载的双机或多机系统就是集群系统(Clusting)。DEC 公司最早在其 VAX 系统上实现了集群技术,多服务器集群系统的主要目的是使用户的应用获得更高的速度、更好的平衡和通信能力,而不仅仅是数据可靠性很好的备份系统。集群系统对于军队、金融、证券等大型关键业务系统,无疑是最好的选择。

7. 时间冗余和信息冗余

以上几种方法都是利用冗余的机器来确保系统的安全性,这种方式也称之为空间冗余。此外还有利用时间的冗余和信息的冗余确保系统安全性的方法。例如检查点(CheckPoint)就属于时间冗余的一种,将机器运行的某一时刻称作检查点,此时检查系统运行的状态是否正确,不论正确与否,都将这一状态存储起来,一旦发现运行故障,就返回到最近一次正确的检查点重新运行。

三、容错的误区

1. 拷贝 = 备份

目前很多用户将数据的安全性简单的理解为数据拷贝,认为只要将数据拷贝下来就可以保证数据的安全了,这是一种错误的观点。拷贝不等于备份,备份应是拷贝加上管理,管理是指可计划性/自动化操作历史记录和保存日志记录。

2. 双机容错等技术可以代替数据备份

“现在有双机容错、双机热备份、磁盘阵列等技术,它们可以代替数据备份。”这种说法正确吗?双机容错等技术解决的是计算机系统中数据可靠性问题,即只有在满足设计条件的情况下才有可靠性。例如磁盘阵列中一台磁盘坏了,可以恢复。如果两台坏了,就恢复不了。上面提到的因天灾人祸造成的数据损坏,双机热备份或磁盘阵列都克服不了,因此它们不能替代数据备份。通常,我们将在线备份称为热备份,而将脱机数据备份称为冷备份,以区别两种不同的备份概念。

四、结束语

早期的容错计算机大多是应用于国家重要部门,是造价昂贵的专用系统,随着计算机的普

及应用,人类的生产、生活活动已经越来越多地与计算机、网络、数据库联系在一起。人们对计算机的依赖程度越高计算机的可靠性就越重要,由此加大了对容错计算机的需求,同时推动了容错技术的发展。

参 考 文 献

1 纯软件方式容错系统的实现 . 计算机世界,1998

2 刘秀文 . 双机容错计算机系统的设计与实现 . 微电脑世界,1998

选自《电脑与信息技术》双月刊,2000 年第 1 期

7.9 容错系统中的自校验技术及实现方法

北京航空航天大学自动控制系(100083) 陈胜功 宋子善

容错是容忍错误的简称,容错系统是指在发生硬件故障或存在软件错误的情况下,仍能继续正确完成指定任务的系统。设计与分析容错系统的各种技术称为容错技术。有关计算机容错技术的各种理论及研究已经发展成为计算机学科的一个重要分支,称为容错计算(Fault-tolerant Computing)。设计容错系统的指导思想是:系统发生故障时能自动检出并使系统自动恢复正常运行。

为保证系统具有高可靠、长寿命和响应迅速,必须采用容错技术。从功能上讲,一个容错系统可用图 7.9-1 所示框图来描述,系统模块可由若干个功能相同的子模块并联组成,故障检测模块对系统模块的功能进行检测,检测到故障后由故障处理模块对系统模块进行重新配置,使系统在部分模块失效的情况下,仍能输出正确结果。在这个系统中,故障检测模块和故障处理模块起着至关重要的作用,它们往往是系统的薄弱点。因此,容错系统要有高的可靠性必须保证故障检测和处理模块的可靠性,故障检测系统不仅要检测系统模块的故障,而且还要能够检测自身的故障,使故障处理模块得以正确处理所发生的故障。

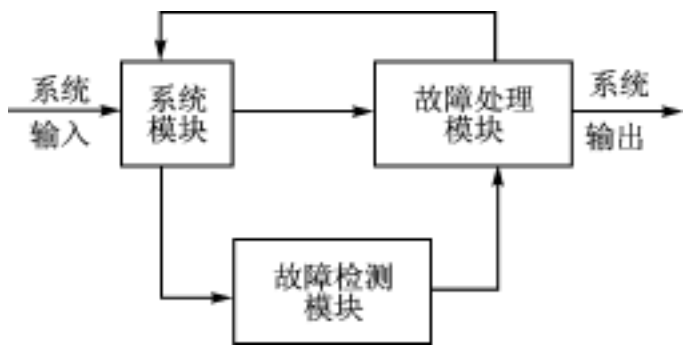


图 7.9-1 容错系统示意图

自校验技术是用于故障快速检测的一种有效手段,特别是具有完全自校验性质的自校验装置,它不仅能及时检出系统模块的差错,还能检测出自身的差错。在容错系统设计中,将自校验网络置于系统中,可大大提高系统对差错的反应能力,使差错潜伏期缩短,有效地防止错误传播。

一、自校验网络定义

一个容错系统,其所有可能的输出值组成的集合 U(称输出空间)由 S 和 U-S 两部分组成。当系统中无故障时,系统输出 S 中的元素,一旦系统发生故障则输出 U-S 中的元素。

如果一个电路的正常输出集 S 是一个检错码集,则该电路称为自校验电路。在无故障发生的情况下,自校验电路输出码向量;当预定故障发生时,则输出非码向量。连接在自校验电路输出端的校验器监督电路的输出,当非码向量出现时,校验器给出差错指示。自校验电路与校验器一起构成了自校验网络,其结构如图 7.9-2 所示。

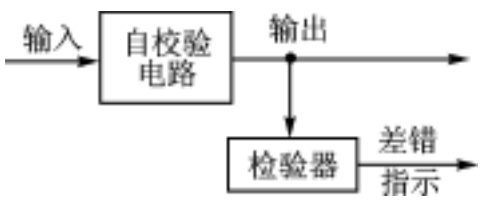


图 7.9-2 自校验网络结构

下面我们给出自校验网络的几个概念,设 S(X)是输

入向量空间，$S_n(X)$是合法输入向量空间，$S_a(X)$是非法输入向量空间，$S(F)$是输出向量空间，$S_n(F)$是合法输出向量空间，$S_a(F)$是非法输出向量空间，F_s^c 是组合逻辑网络所考虑的单故障集，网络的所有输入取自其合法输入集。

定义 1：组合逻辑网络对 F_s^c 是自测试的（简称 ST），若：对任意故障 $f \in F_s^c$ 存在 $X \in S_n(X)$，使网络输出 $G_f(X) \in S_a(F)$。

该定义是说对给定单故障集中的任何一个故障，总存在一个合法输入，它将导致组合逻辑网络产生非法输出，即用该输入可检测此故障。

定义 2：组合逻辑网络对 F_s^c 是故障保险的（简称 FSE），若：对任意故障 $f \in F_s^c$，存在 $X \in S_n(X)$，使网络输出 $G_f(X) = G(X)$，或者 $G_f(X) \in S_a(F)$，其中 $G(X)$ 是网络的正确输出。

该定义是说在合法输入条件下，对给定单故障集中的任何一个故障，组合逻辑网络将给出一个合法输出或一个非法输出，但不会给出同该合法输入不相对应的另一个合法输出。

定义 3：组合逻辑网络对 F_s^c 是完全自校验的，若它对 F_s^c 是自测试的和故障保险的。

对给定单故障集中的任何一个故障，完全自校验的组合逻辑网络在合法输入的情况下，或者产生一个合法输出，或者产生一个非法输出，且后者在故障存在期间是必然要出现的。这就使得网络在不产生不正确合法输出的同时，能并发检测 F_s^c 中的所有故障，这也正是在线测试对差错检测及时性的要求。

若采用检错编码技术实现自校验逻辑网络时，校验器的任务是用来检查功能模块输出码字的有效性，它能区分功能模块输出是否合法，这种特性被称为码分离特性，其定义如下。

定义 4：一个逻辑网络具有码分离特性，若满足：

(1) 任给 $X \in S_n(X)$，有 $G(X) \in S_n(F)$（合法输入映射为合法输出）

(2) 任给 $X \in S_a(X)$，有 $G(X) \in S_a(F)$（非法输入映射为非法输出）

定义 5：一个逻辑网络是完全自校验器，若它具有完全自校验及码分离特性。

设组合逻辑网络正确输入矢量为 $S_c(X)$，则矢量空间 $S_n(X) - S_c(X)$ 称作错误输入空间，记作 $S_e(X)$；空间 $S(X) - S_c(X)$ 被称作非法及错误输入空间，记作 $S_{ae}(X)$。由正确输入空间 $S_c(X)$ 经电路 G 可在 $S(F)$ 中产生一个子空间，这个子空间称为正确输出空间，记为 $S_c(F)$。同样，对于合法输入 $S_n(X)$，由组合逻辑网络可映射为合法输出空间 $S_n(F)$，它也是输出矢量空间 $S(F)$ 的子集。同样，空间 $S_n(F) - S_c(F)$ 被称作错误输出空间，记作 $S_e(F)$；空间 $S(F) - S_c(F)$ 被称为非法及错误输出空间，表示为 $S_{ae}(F)$。上述输出之间有如下关系：

$$S(F) = S_c(F) \cup S_{ae}(F) = S_n(F) \cup S_a(F)$$

$$S_{ae}(F) = S_a(F) \cup S_e(F)$$

由上面集合之间的关系我们可以看出，对于无故障组合网络的正确输入，其输出应落入正确输出空间 $S_c(F)$ 中。通过对网络的输出可部分判定系统工作是否正常（无法判断某些故障）。当网络发生故障时，可分成以下几种情况：① 非法及错误输入被映射成 $S_c(F)$；② 输入 $S_c(X)$ 被映射成为 $S_{ae}(F)$；③ $S_c(X)$ 映入 $S_c(F)$ 但已不是正确的映射关系，也就是说输入输出关系发生了变化。对于一个高可靠容错系统来说，必须能够以比较高的故障覆盖率来检测出以上三类差错（最好在一拍内检出），使系统及时采取措施，隔离故障，将其影响减小到最低限度。在三类错误中，第①类和第②类与第③类相比要好检测一些，高效检测第③类错误是提高系统故障覆盖率的关键，只有设计出对以上三类错误检出率均较高的检错系统，才能保证系统有较高的可靠性。

二、自校验网络的结构

自校验网络具有在无任何外加激励的情况下能自动检测其内部是否存在故障,这些故障或是永久性的或是暂时性的。设计自校验网络的主要技术有检错编码技术,基于自对偶函数的交替逻辑技术(交织逻辑技术),基于对偶函数的互补逻辑技术,还有基于多值逻辑的实现方法,下面我们主要讨论一些实用的实现方法。

1.双轨码校验器

双轨码校验器的原理图如图 7.9-3 所示。

图 7.9-3 双轨码校验器原理图

输入矢量为 (Y_1, Y_2),其中 $Y_i = (y_{i1}, y_{i2}, \cdots, y_{in}), (i = 1,2)$,输出矢量为 Z_e 且满足:

$Z_e \in S_n(Z_e) = \{(0,0),(1,1)\}$,若 $Y_1 \neq$ 矢 Y_2,且校验器无故障。

利用双轨码校验器的上述特点,设计一对偶组合逻辑网络,使其输出向量 Y_1 和 Y_2 恰好反相,将 Y_1 和 Y_2 加到双轨码校验器输入端,根据 Z_e 就可以判定系统是否发生故障。

2.可分码校验器

可分码校验器的结构如图 7.9-4 所示。校验器的输入矢量为 $Y = (Y_1, Y_2)$,矢量 Y_1 和 Y_2 分别对应可分码的信息分量和校验分量。其中,信息分量宽度为 $\| Y_1 \| = I$, $\| Y_2 \| = K$, $K = [\log_2 (I+1)]$是校验分量的宽度,且 $I + K = n$, $n = \| Y \|$。校验位生成电路根据信息位 Y_1 重新生成校验位 W,由双轨码校验器比较 W 与 Y_2 的一致性,在无故障的情况下,校验器的输出 Z_e 指示输入矢量的有效性。下面的定理给出了图 7.9-4 完全自校验可分码校验器的构造条件。

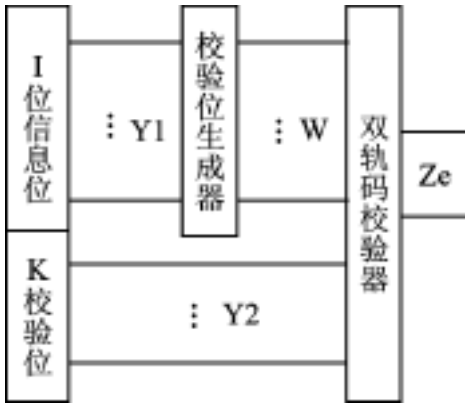


图 7.9-4 完全自校验可分码校验器

定理:图 7.9-4 所示的可分码校验器是完全自校验的,若校验位生成器是一个无冗余的组合逻辑网络,且比较器是完全自校验的。

3.互补逻辑网络

利用互补逻辑也可以构成自校验电路,如果某一逻辑网络其输入输出关系为 $f = g(x_1, x_2, \cdots, x_n, \cdot, +)$,则可构造一互补逻辑网络,使其输入输出关系为瑏= g〔瑏,瑏,⋯,瑏, +, ·〕其中 f 和瑏是互补的,在无故障的情况下,其输出是互补的;若发生输出相同,则两个逻辑电路中必定有存在故障的情况。互补逻辑网络实现原理比较直观,但对较复杂的系统,有许多故障它是检测不出来的。

4.交织逻辑网络

交织逻辑网络是基于自对偶函数的自校验逻辑网络。一个二进制变量是交替的,记作(x,瑏),若 x 在两个连续的时间间隔内所取的值互补。

对任意一个开关函数 $g(X)$, $X = (x_1, x_2, \cdots, x_n)$, $X \in B_2^n$,若假设 $x_1, x_2, \cdots, x_n$ 是交替二进制变量,且它们是同步交替的,则 g 的输入矢量可表示为(x,瑏) = $(x_1, x_2, \cdots, x_n; 瑏, 瑏, \cdots,$

瑏瑠),其输出可表示为(g(X);g(瑏瑡)),要使输出变量也是交替的,必须满足 g(瑏瑡) =瑏瑠(X),显然,g 必须是自对偶函数。利用交织逻辑网络的这个特点,可以检测出系统的一部分故障。

三、自校验网络实现方法

对于一些比较简单的应用场合,利用数字逻辑方法进行设计,使用 SSI 及 MSI 集成电路即可方便地构成自校验网络。但实际容错系统非常复杂,涉及大量逻辑设计,若仍采用传统的数字逻辑设计方法,不仅工作量大、容易出差错,而且修改和功能仿真都不方便。使用电子设计硬件描述语言 VHDL(或 Verilog HDL)对电路功能进行描述,用 FPGA 或 CPLD 器件实现自校验网络是比较现实的,对于大批量生产,可将 VHDL 描述的电路送半导体器件厂进行批量生产,VHDL 硬件描述语言实现自校验网络的步骤如下:

① 建立自校验网络的功能模型。对系统的输入/ 输出、状态转换、信号传递等进行详细的说明。

② 用 VHDL 语言或 Verilog HDL 语言对电路功能进行描述。对复杂系统可采用“自上而下”的设计方法,将系统分解成不同层次的、功能较简单的模块,利用 VHDL 语言对系统功能进行分层描述,减少系统描述造成的错误。

③ 对不同层次的模块进行功能仿真,以检验各模块设计的正确性,最后对整个系统进行功能仿真,及早排除系统设计中的错误。

④ 用 VHDL 或 Verilog HDL 综合编译器对设计好的系统进行编译,经过逻辑化简及综合布线,生成可对 FPGA 或 CPLD 编程的数据文件。

⑤ 将数据文件通过编程器写入 FPGA 或 CPLD,进行实际测试,若测试数据满足设计要求,则开发工作完成;否则,转①重新进行检查和设计。

采用自校验技术后,可有效地提高容错系统的可靠性,随着集成电路技术的飞速发展,可将一些自校验功能模块进行封装,作为标准单元使用,在模块级上提高容错系统的可靠性。采用高级语言和 FPGA 或 CPLD 开发容错系统具有重要的现实意义,可有效缩短开发周期,降低开发成本,提高系统可靠性,应在工程设计中加以推广应用。

参 考 文 献

1 袁由光编著 .实时系统中的可靠性技术 . 北京:清华大学出版社,1995

2 杨士元编著 .数字系统的故障诊断与可靠性设计 . 北京:清华大学出版社,1989

3 胡谋主编 .计算机容错技术 . 北京:中国铁道出版社,1995

4 Johnson B W . Design and Analysis of Fault-Tolerant Digital System . A . W . Publishing Company, Inc ., 1989

5 Habinc S, et al . Using VHDL for Board Level Simulation . IEEE Design & Test of Computers, Fall,1996

选自《电子技术应用》月刊,2000 年第 7 期

7 .10　基于 MAX110 的容错数据采集系统的设计

天津大学电气自动化及能源工程学院(300072)　刘　泽
内蒙古工业大学信息工程学院(010062)　常　佶

在工业过程控制系统的设计中,与工业现场环境相联系的接口电路的设计在保证性能的同时,要求有较高的可靠性和性能价格比。对于模拟输入通道的设计,串行接口的模拟数字(A/ D)转换器愈来愈受到设计者们的关注。串行接口 A/ D 转换器接口电路设计简单、芯片体积小、信号线大大减少、易于采取隔离措施,而且许多串行接口芯片的工作模式可编程、设计灵活。其中,MAXIM 公司的二通道±14bit 串行 ADC MAX110 在从串行接口读取数据的同时还完成下一转换过程控制方式数据的写入(包括转换启动控制),这样便可实现数据的自动采集;在程序设计时,数据采集进程独立设计,数据按预先设定的方式自动存储到循环队列中,完成和主进程的数据交换,给系统程序的设计提供了很大的方便。但由于系统受环境的干扰,A/ D 转换器控制数据可能会出错,导致 ADC 意外停止转换,使数据采集进程停止,本文提出了针对这种故障的容错[1] 设计方法。

一、基于串行 A/ D 转换器 MAX110 的数据采集系统的结构

工业过程控制的许多慢过程,对 A/ D 转换器转换速度要求较低,所以选用慢速 A/ D 转换器便可以满足设计要求。±14bit 串行 ADC MAX110BCPE 转换时间可达到 10 ms,且可编程控制,精度和速度也可以满足如温度控制对象的数据采集要求。作者在设计适用于温度控制的模糊智能调节器的过程中,数据采集部分使用了基于串行 ADC MAX110 的具有容错功能的数据采集设计方法。系统结构如图 7 .10 - 1 所示。

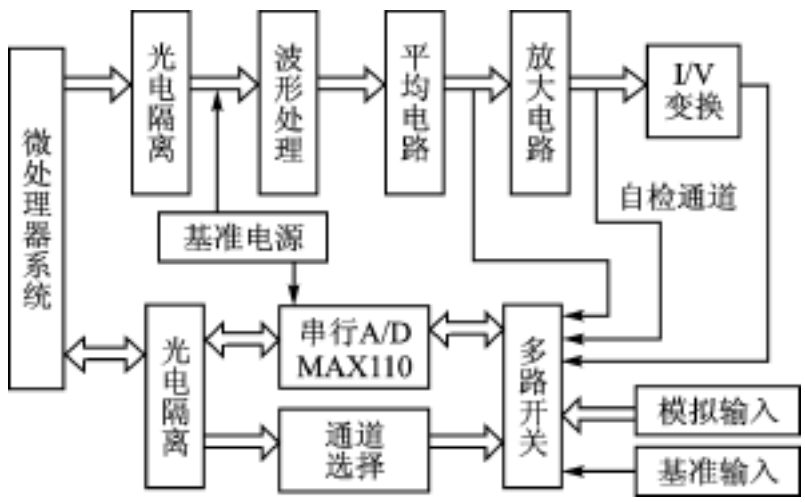


图 7 .10 - 1　MAX110 转换接口电路原理图

模拟信号隔离可选两种方法:选用模拟隔离放大器或在 A/ D 的数字接口采用光电隔离。模拟隔离放大器价格较高,所以选择数字隔离方法,而采用串行 A/ D 可大大减少数字信号隔离路数。如图 7 .10 - 1 所示,串行 A/ D 转换器 MAX110 的数字侧与微处理器全部采用了光

电隔离器件,将 MAX110 的数据输入输出信号、时钟信号、转换结束中断请求信号由 PC817 实现电气的隔离。在后续的模拟通道中采用了多路开关 CD4051 选通输入的各路模拟信号,其中通道选择信号也采取光电隔离的方法,这样使 A/ D 转换器之后的所有电路都与微处理器部分隔离。

二、A/ D 异常停止转换故障的自动恢复原理

1 .串行 ADC MAX110 的控制与数据传递

串行 ADC MAX110 的转换方式和标定由芯片控制字确定,控制字确定了 MAX110 A/ D 转换的通道、速度等各种工作方式。MAX110 的控制字包含:转换时间控制,SCLK 与过采样频率的比率控制、输入通道选择、增益标定控制、内部振荡器掉电控制、模拟部分掉电控制。每次转换按新送入的控制字工作。控制字格式及控制字作用见表 7 .10 - 1。

表 7 .10 - 1 控制字格式及控制字作用表

bit	NAME	描 述
15	NO-OP	NO-OP = 1 其余位送控制寄存器, $\overline{CS}$变高时新的转换开始; NO - OP = 0 控制寄存器保持不变, $\overline{CS}$变高时有没有新的转换开始
5,6,13,14	NU	Not Used,用于测试
9～12	CONV1～CONV2	转换时间控制位
7,8	DV2, DV4	XCLK 与过采样频率比率控制
4	CHS	通道选择,CHS = 0 通道 0 CHS = 1 通道 1
3	CAL	增益标定位,高电平表示增益标定模式
2	NUL	内部零偏移位,高电平选择零偏移模式
1	PDX	高电平选择振荡器电源降模式
0	PD	高电平选择模拟电源降模式

MAX110 采用与 Microwire™ SPI™ QSPI™ 串行接口通讯协议(SPI: Serial Peripheral Interface, QSPI: Quick SPI)兼容的串行接口标准[3]。其时序实现如图 7 .10 - 2 所示。当微处理器检测到 MAX110 转换完标志$\overline{BUSY}$变高时,AD 中断产生,中断服务程序首先将串行时钟信号 SCLK 初始化为 0,再将 MAX110 片选信号$\overline{CS}$置低,开始串行数据的全双工传输:送 MAX110 转换命令字,同时接收 MAX110 转换结果。数据的发送和接收过程是:首先,微处理器将要送命令字的最高位送到 MAX110 接收命令字的引脚 DIN,然后将 SCLK 置高,MAX110 利用 SCLK 时钟信号的上升沿将命令字最高位读入;这时微处理器将 MAX110 的转换结果最高位读入。微处理器再将 SCLK 置低,使 SCLK 出现下降沿;MAX110 在下降沿将转换结果的第二位送到 DOUT 引脚,微处理器将命令字的第二位送到 DIN 引脚,再将 SCLK 置高,MAX110 利用 SCLK 的上升沿读入命令字的第二位;微处理器读入 MAX110 转换结果的第二位。如此循环直到将十六位数据接收完毕,命令字写完毕,完成数据交换,最后将$\overline{CS}$引脚置高。当 MAX110 的$\overline{CS}$引脚被置高时,MAX110 开始新的转换,转换的工作方式由刚接收到的命令字确定。微处理器 A/ D 中断结束,直到 MAX110 下一次转换结束时间的

到来。

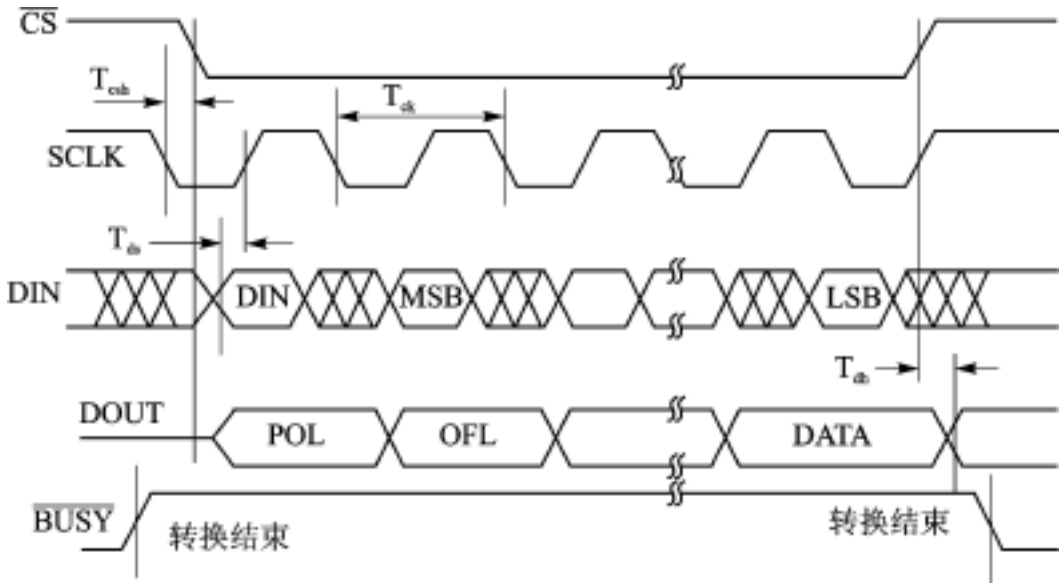


图 7.10-2　MAX110 接口时序图

2. A/D 异常停止转换故障的检测及自动恢复

A/D 转换器 MAX110 的启动包括标定和转换控制字的设置,耗时较长,设计时将 MAX110 设置为连续转换方式,本次转换启动下次转换,下次转换的转换控制字由本次提供,这样 MAX110 就可以连续产生采集数据中断,实现数据的自动采集。但这种转换方式有可能出现 A/D 转换停止的故障。因为系统受到外界干扰时,可能会影响到某些数据线信号波形的偶然畸变,而串行 A/D MAX110 每次转换都建立在收到正确转换命令的基础上,如受到干扰,转换命令字错误,有可能导致 MAX110 停止转换,或者进入休眠方式或模拟电路电压切离的低功耗方式,这样系统将停止数据采集。

解决这种停止转换故障的方法如图 7.10-3 所示。在数据的自动采集过程中,下次转换

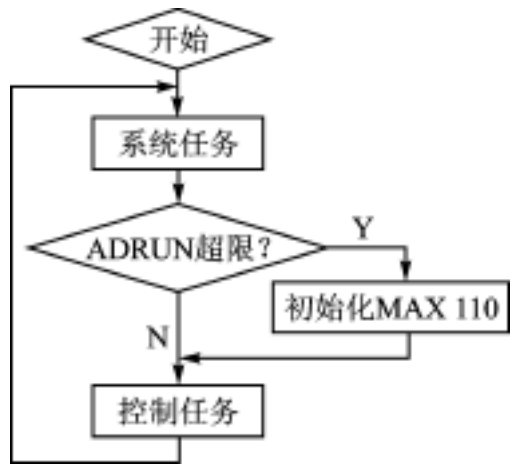


图 7.10-3　MAX110 停止转换故障恢复流程图

的正常工作建立在本次正确送入控制字的基础上,所以在微处理器的软件定时器中设置一个按一定时间间隔不停增长的计数器 ADRUN,在 A/D 转换中断服务程序中将 ADRUN 计数器值清零。若 A/D 正常转换,ADRUN 计数器的值将保持在某一范围内;若 A/D 停止正常转换,ADRUN 计数器的值将超出预先设定的范围。在系统程序的主循环中每次判断 ADRUN 计数器的值,若 ADRUN 计数器超限,说明 MAX110 停止转换,随即将 MAX110 重新初始化,使数据采集模块恢复正常工作,同时在系统的故障数据库中做出记录以备分析。这种设计方法保证了数据采集模块的稳定运行,使系统具备了一定的容错功能。

本文论述的容错数据采集系统已在自行设计的以 Intel 80C196 为微处理器的智能调节器中使用和测试。测试时,给数据采集系统注入故障,使 AD 采集进程停止工作,实验证明,在预先设定的时间内,故障得以排除并记录了故障情况,数据采集正常运行。但这种容错设计方法的可靠性依赖于探测故障时所使用的定时器的可靠性,若定时器异常停止工作,系统将不能检测到 A/ D 转换的故障。

参 考文 献

1 incenzo piuri . Design of Fault-Tolerant Distributed Control System . IEEE transactions on instrumentation and measurement, April 1994;43(2)

2 Isermann, Rolf . Model based fault detection and diagnosis methods . Proceedings of the American Control Conference, V3, 1995

3 Maxim Integrated Data Sheets . Maxim Integrated Products Inc . 1997

4 Application Note-Using the 80C196KB . Intel Corporation, November 1990

选自《电子技术应用》月刊,2000 年第 3 期

7 .11　冗余式时钟源电路

北京控制工程研究所　杨开雄

一、问题提出

根据‘资源一号”卫星的建造规范,作为整星惯性基准单元的惯性姿态敏感器应满足地面测试一年半、储存二年后仍能在轨工作二年的寿命要求。由于国产陀螺马达难以满足这种长寿命要求,因此整个惯性姿态敏感器采用了冗余技术,即 3 个陀螺为主份,3 个陀螺为备份,当主份陀螺失效时,立即切换至相应通道备份陀螺工作。似此,由 6 个陀螺通道组成的惯性姿态敏感器,如若每个通道设立一时钟源,共需 6 个,而据电路常识,多晶振电路极易产生拍频现象。为克服这一弊病,6 个陀螺通道需采用一公共时钟源(统一时钟源)。对于需满足 5 .5 年寿命要求、6 个陀螺通道公用的这一时钟源,其可靠性要求极高,若其失效,则整个惯性敏感器将无法工作,进而危及整星,因此必须采用冗余式时钟源。其由主、备份时钟源及切换电路组成。当主份时钟源失效后,能立即切换至备份时钟源工作,据此满足整星的长寿命要求。

二、时钟源电路简介

“资源一号”卫星惯性姿态敏感器中所使用的时钟源电路如图 7 .11 - 1 所示。由晶振、晶振切换电路、或门及分频电路组成。共有 8 路输出,其中 A、B、C、D、E、F 为 6 路相位依次互差 60°、频率为 400 Hz 的方波信号,它除为 6 路陀螺温控提供激励信号外(一相对应一路陀螺温控),还同时通过适当的组合,形成三相互差 120°的马达三相电源激励信号(其中 A、B、C 相为 3 个陀螺马达的激励信号,D、E、F 为另 3 个陀螺马达的激励信号)。另外两路信号中的一路为 V/ F 转换(模数转换)电路的时钟信号,另一路为陀螺传感器激磁信号。为提高晶振电路的可靠性,电路中设计了由晶振、晶振切换电路等组成的温备份电路。为减少整个设备的体积、重量,时钟源电路与其他 8 种陀螺伺服电路一起制作成为混合集成电路。单个时钟源电路失效率如表 7 .11 - 1 所列。

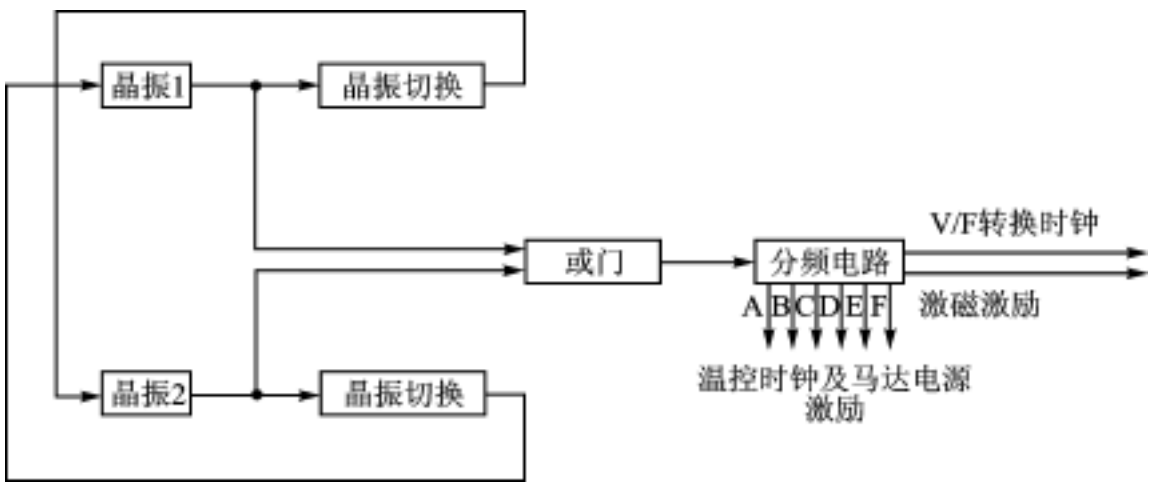


图 7 .11 - 1　时钟源电路原理框图

表 7 .11－1 单个时钟源电路失效率

名 称	数 量	单件失效率	失效率
CMOS	10	34 ×10⁻⁹	$\lambda_1 = 340 \times 10^{-9}$
电 阻	10	1 ×10⁻⁹	$\lambda_2 = 10 \times 10^{-9}$
涤纶电容	7	7 ×10⁻⁹	$\lambda_3 = 49 \times 10^{-9}$
云母电容	5	1 .3 ×10⁻⁹	$\lambda_4 = 6.5 \times 10^{-9}$
二极管	6	0 .7 ×10⁻⁹	$\lambda_5 = 4.2 \times 10^{-9}$
晶 体	2	60 ×10⁻⁹	λ_6

单个时钟源电路的可靠度为:

$$R_1 = e^{-(\lambda_1+\lambda_2+\lambda_3+\lambda_4+\lambda_5)t} \times [1 - (1 - e^{\lambda_6 t})^2] \quad (1)$$

若按 2 年卫星寿命周期计,则

$$R_1 = e^{-409.7\times10^{-9}\times17520} \times [1 - (1 - e^{-60\times10^{-9}\times17520})^2]$$

$$= e^{-0.718\times10^{-2}} \times [1 - (1 - e^{-0.105\times10^{-2}})^2]$$

$$\approx 0.99285 \times 0.99999$$

$$R_1 \approx 0.99284 \quad (2)$$

式中 $_1$ ——单个时钟源电路可靠度;

t——电路工作时间,2 年。

三、冗余式时钟源电路

1 . 电路原理简述

如图 7 .11－2 所示,整个电路由主、备份时钟源、失效信号检测、逻辑电路、切换电路及信

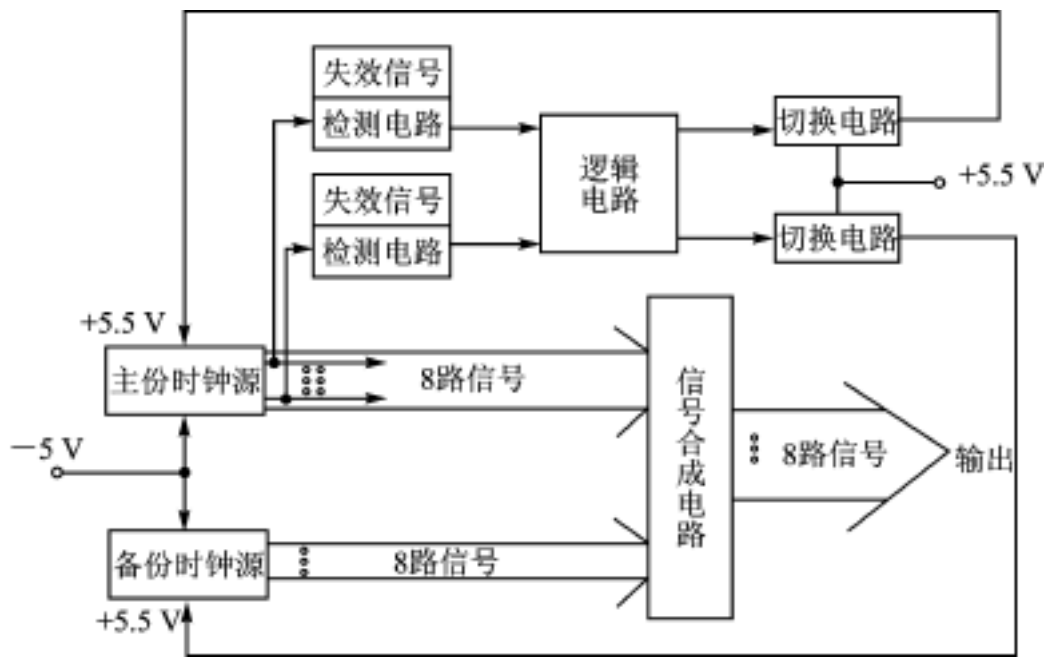


图 7 .11－2 冗余式时钟源电路原理框图

号合成电路组成。其原理可简述为:电路—上电时,设计为能准确无误地控制切换电路将 +5 .5 V电源加入主份时钟源,使其工作,其 8 路输出信号(如前所述)经信号合成电路输出至其他电路。当主份时钟源有 1 路或数路失效(即无输出)时,择其中 F 相 400 Hz 及 V/ F 转换时钟信号(电路原理决定选择此两路信号则能代表其他 6 路的失效情况,选择理由下节叙述)

为失效检测信号,此 2 路信号各经检测电路后,馈入逻辑电路、进而通过切换电路,切断加入主份时钟源的 + 5 .5 V 电源,使主份时钟源无输出,同时使 + 5 .5 V 电源加入备份时钟源,使其工作,其 8 路输出信号经信号合成电路输至其他电路。本电路的显著特点是,当且仅当主份时钟源工作失效时才切至备份时钟源工作,信号合成电路中瞬时仅有 1 个时钟源的 8 路信号通过。

2 . 失效检测信号的选取

如图 7 .11 - 3 所示,频率为 F_0 的时钟源信号,经 7 分频再经脉冲加速电路后得 V/ F 转换时钟(F_0/ 7),F_0/ 7 时钟经 2 分频后得激磁激励信号(F_0/ 14),F_0/ 14 的时钟经 180 分频后得 A、B、C、D、E、F 六路马达电源激励及温控时钟信号(F_0/ 2520)。由于处于分频源头的 F_0/ 7 时钟,还需经脉冲加速电路才成为 V/ F 转换时钟,故选择其作 1 路失效检测信号,其除表征晶振源起振状态外,也表征脉冲加速电路正常工作与否。据电路原理,由于 F 相 400 Hz 信号处于环形计数器的末级,故选择其为另 1 路失效检测信号。由此选择的 2 路失效检测信号,即可代表整个时钟源 8 路输出信号的失效状态。

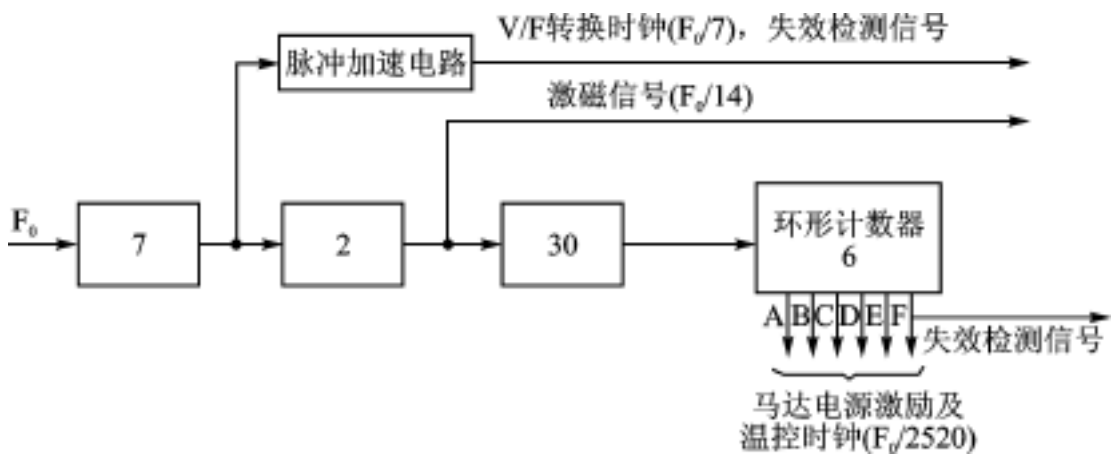


图 7 .11 - 3　时钟源分频电路原理框图

3 . 检测、逻辑、切换电路

图 7 .11 - 4 所示的检测、逻辑、切换电路,其原理为:当 V/ F 转换时钟及 400 Hz F 相信号

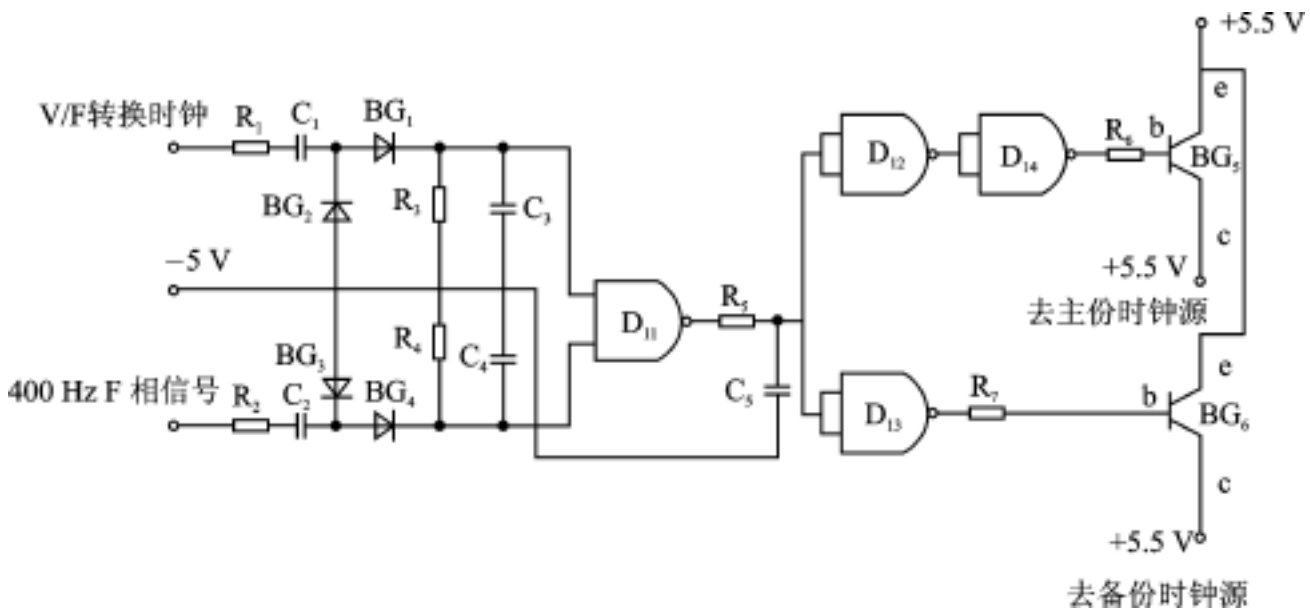


图 7 .11 - 4　检测、逻辑、切换电路

正常时,经各自的隔离、检波、滤波电路,而后送入与非门 D_{11} ,在 D_{11} 输出端为低电平信号,此信号经 D_{12}、D_{14} 与非门后,在 BG_5 的 b 端产生为低电平信号,致使 BG_5 导通, + 5 .5 V 电源经 BG5 C 极加入主份时钟源,使其工作。D_{11} 输出端的低电平信号经与非门 D_{13} 后,在 BG_6 的 b

端产生高电平信号,致使 BG_6 截止,+5 .5 V 电源未能加入备份时钟源,抑制其工作。当 V/ F 转换时钟、400 Hz F 相信号同时失效或其中 1 路失效时,与上述过程相反,在 BG_5 的 b 端产生高电平信号,致使 BG_5 截止,切断 +5 .5 V 电源加入主份时钟源的通路。同时 BG_6 的 b 端为低电平信号,BG_6 导通,+5 .5 V 电源经 BG_6 的 C 极加入到备份时钟源,使其工作,由此实现主、备份时钟源切换工作,达到冗余的目的,从而大大地提高电路的可靠性。

4 . 信号合成电路

图 7 .11 - 5 所示的信号合成电路由二片或门电路组成,主、备份时钟源各自的 8 路输出信号分别相应地输至或门的输入端,在或门的输出端形成 8 路输出信号,设计保证或门瞬时仅有主份时钟源的 8 路信号仰或备份时钟源的 8 路信号输出。

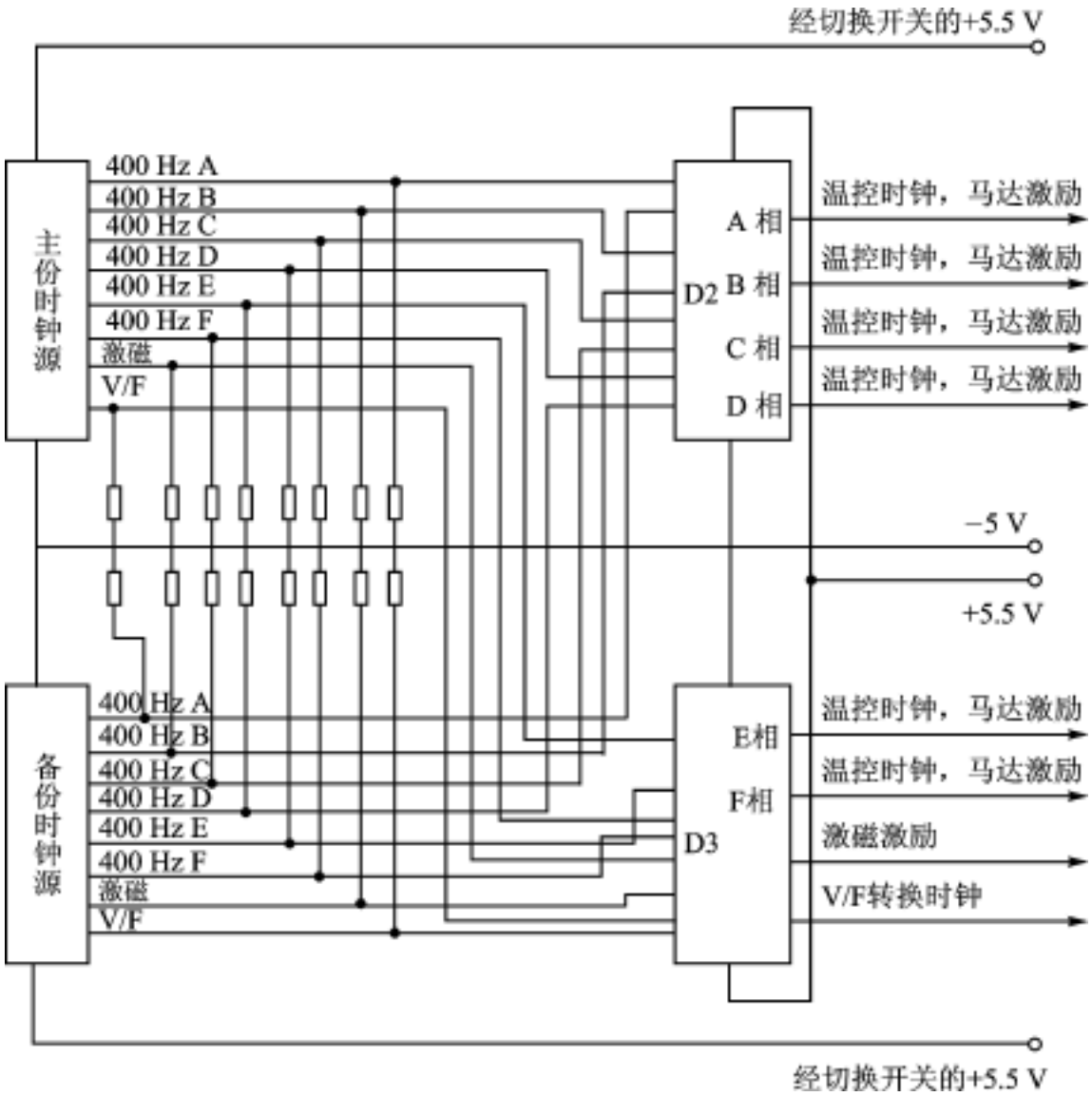


图 7 .11 - 5 信号合成电路

5 . 可靠度计算

据图 7 .11 - 2 可得冗余式时钟源的可靠性框图,如图 7 .11 - 6 所示。

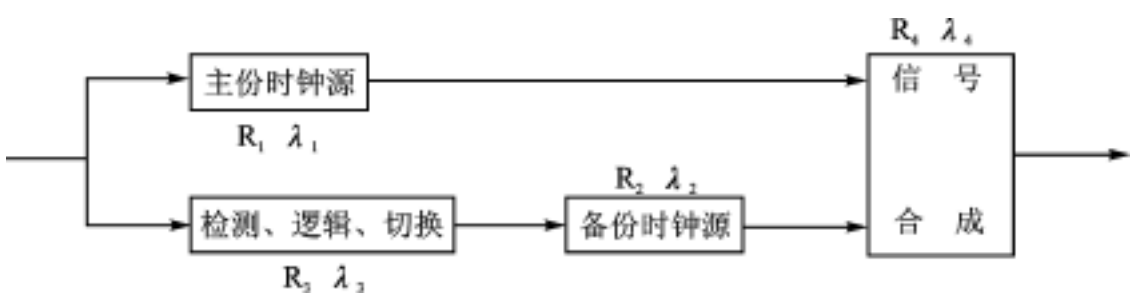


图 7 .11 - 6 冗余式时钟源电路可靠性框图

图中: $_1$、λ_1 主份时钟源可靠度、失效率;

R_2、λ_2 备份时钟源可靠度、失效率;

R_3、λ_3 检测、逻辑、切换电路可靠度、失效率;

R_4、λ_4 信号合成电路可靠度、失效率。

由于主、备份时钟源电路结构完全一样,据式(2)得

$$R_1 = R_2 = 0.99284 \tag{3}$$
$$R_3 = e^{-\lambda_3 t}$$

据图 7 .11－4 可计算得(t = 17520)

$$\lambda_3 = 62.9 \times 10^{-9}$$
$$R_3 = e^{-62.9\times10^{-9}\times17520} \tag{4}$$
$$R_3 \approx 0.99890$$
$$R_4 = e^{-\lambda_4 t}$$

据图 7 .11－5 可计算得

$$\lambda_4 = 84 \times 10^{-9}$$
$$R_4 = e^{-84\times10^{-9}\times17520} \tag{5}$$
$$R_4 \approx 0.99853$$

将式(3)、式(4)、式(5)代入图 7 .11－6 得图 7 .11－7。

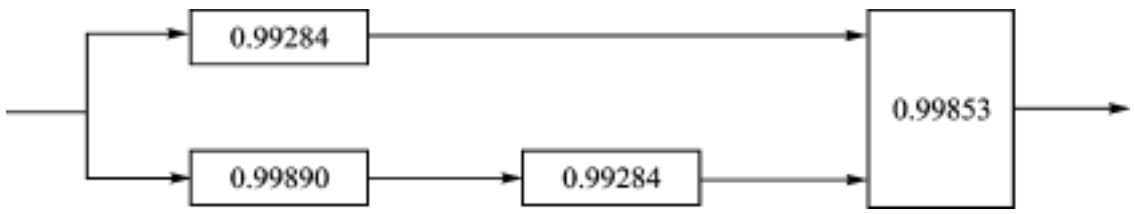


图 7 .11－7　冗余式时钟源电路可靠性简图

据图 7 .11－7 及可靠性原理可得冗余式时钟源电路可靠度为

$$R_\Sigma = (0.99284 + 0.99175 - 0.99284 \times 0.99175) \times 0.99853$$
$$= 0.99994 \times 0.99853 \tag{6}$$
$$R_\Sigma \approx 0.99847$$

四、可靠度对比

据式(2)、式(6),单个时钟源、冗余时钟源电路的可靠度为:

$$R_1 = 0.99284$$
$$R_\Sigma = 0.99847$$

二者相比,冗余式时钟源电路的可靠度比单个时钟源的可靠度有较大幅度的提高。电路达到了预期设计的目的。

五、实验室试验结果

冗余式时钟源电路与陀螺其他 8 种伺服电路一起制作成为了混合集成电路,且专门研制了测试设备对其进行验收与试验,测试设备中设计有人为地设置主份时钟源输出信号失效模式。实验证明,当主份时钟源有 1 路或数路输出信号失效时,能准确地切换至备份时钟源工作。本电路与陀螺其他电路一起,历经单板调试、系统联试及各种环境试验,工作正常,从未发生问题。

六、飞行试验结果

“资源一号”卫星自 1999 年 10 月 16 日发射升空至今,已在轨正常运行 8 个月有余。据遥测参数表明本电路及整个惯性敏感器迄今未发生任何故障,一直在轨正常运行。

七、结 语

为克服多晶振而极易产生拍频的现象,6 个陀螺通道采用一公共时钟源。为提高可靠性,设计了冗余式时钟源电路,经计算及试验证明,其可靠度与单个时钟源电路相比有较大幅度的提高,达到了预期的设计目的。冗余式时钟源电路对可靠度要求较高的设备极具应用价值。

参 考 文 献

廖炯生编 . 可靠性工程基础简编 . 航天工业部第五研究院,1986

选自《控制工程》双月刊,2000 年第 4 期

7 .12　微机控制系统的抗干扰技术应用

西安航空职业技术学院计算机与管理工程系　皇祯平

可靠性是微机控制系统的重要性能指标,它由多种因素决定。微机控制系统所在现场的各种干扰是影响可靠性的主要因素。干扰源有多种,干扰会影响传送信息的正确性;扰乱程序的正常运行,使程序“飞走”或进入死循环,还可能损坏微机的元器件。干扰是微机控制系统必须认真对待的问题。

一、硬件抗干扰技术

1 .电　源

微机所用的电源一般都由电网的工频交流电源经降压、整流等环节后提供。由于电网的影响以及生产现场大容量电气设备开停,会使交流电压含有高频成分、浪涌电压、尖脉冲或发生较大幅度的波动,这种干扰通过电源途径影响微机系统的正常工作。抑制交流电源的干扰,除了使微机电源尽量与大容量用电设备分别供电外,还经常采用滤波、屏蔽、隔离、稳压等措施。

(1) 采用滤波和屏蔽　图 7 .12 -1 所示是采用滤波和屏蔽的交流电源,图中画出了滤波器和电源变压器。

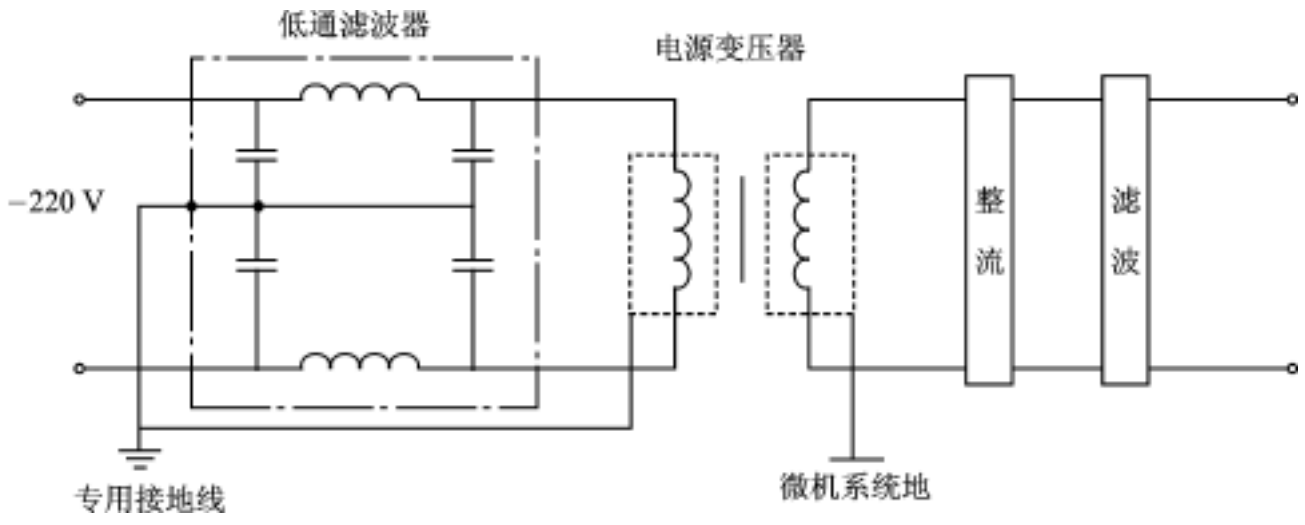


图 7 .12 -1　交流电源的滤波和屏蔽

低通滤波器由电感和电容组成,它对工频交流电的阻抗很小,而对于高频干扰信号具有很强的抑制作用。滤波器加屏蔽外壳,并使之良好接地。滤波器的进线端和出线端要离开一定距离,或采用屏蔽线以防止感应和辐射耦合。电源变压器采用双屏蔽形式,一次和二次绕组分别加屏蔽层,并分别接地。一次屏蔽层接专用地线,二次屏蔽层接即微机系统地。这样可阻断高频干扰信号经变压器的一、二次侧之间的耦合电容传播到微机系统,另外可以消除静电感应。

(2) 采用隔离和交流稳压措施　对于要求较高的系统,可在上述交流供电线路的基础上,

再增加隔离和交流稳压措施,如图 7 .12 - 2 所示。隔离变压器的电压比等于 1,它做成双屏蔽形式。隔离变压器的作用是阻止浪涌电压和尖脉冲通过,其屏蔽层能抑制高频干扰和静电感应作用。交流稳压器用于补偿电网电压的波动,其稳压精度并不需要太高,但要求它工作可靠且要有较快的响应速度。

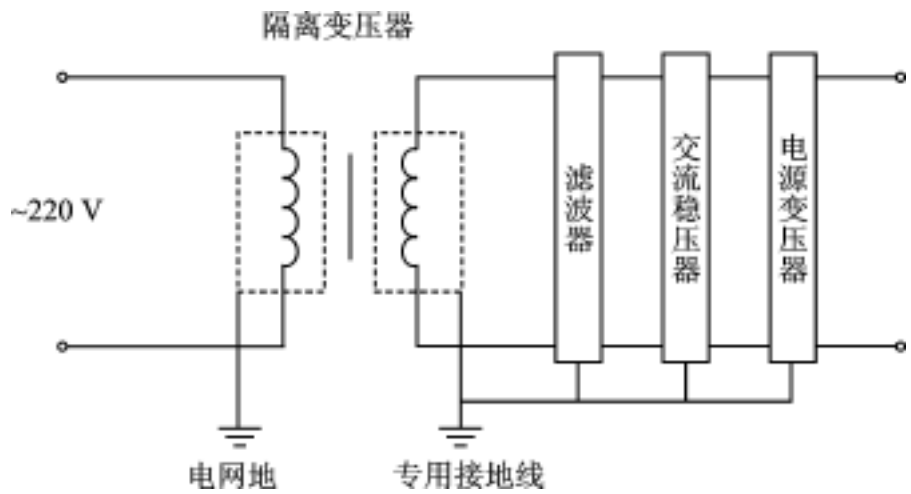


图 7 .12 - 2　对交流电源的隔离、滤波、稳压和屏蔽

(3) 采用分散的直流供电方式　为了提高微机直流电源系统的供电可靠性,可以各模板分别设置直流稳压电源,如图 7 .12 - 3 所示。现有多种规格的三端集成稳压块,型号为 78XX 或 79XX,能方便地用来组成直流稳压电源。采用各自独立供电还能消除相互之间通过电源产生的干扰。

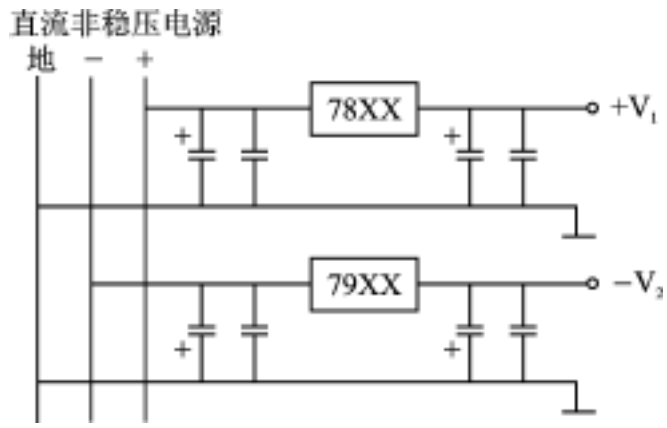


图 7 .12 - 3　分散的直流供电方式

(4) 采用高抗干扰稳压电源和干扰抑制器。

(5) 信号线不能与交流电源并行敷设,尽量远离交流电源线和大功率电气设备。

2 . 模拟量输入通道

(1) 对常态干扰的抑制　常态干扰是叠加在测量信号上的干扰信号。这种干扰信号一般是频率较高的杂乱的交变信号,其来源可能是传感器电路或传输线。

抑制常态干扰的方法有:

① 在输入电路中接入滤波器。常用双 T 滤波器,如图 7 .12 - 4 所示。这是种带通滤波器,对于高频和低频干扰均有抑制作用。

② 采用双积分式 A/ D 转换器,由于其积分工作的特点,具有一定的消除高频干扰的作用。

③ 接入光耦合器能有效地阻断噪声的传送。

④ 将电压信号传送改为用电流信号传送方式能提高抗干扰能力。如图 7 .12 - 5 所示,采

用 4～20 mA 电流传送信号,在进入 A/D 转换器之前在于 250 Ω 电阻上产生 1～5 V 的电压信号。

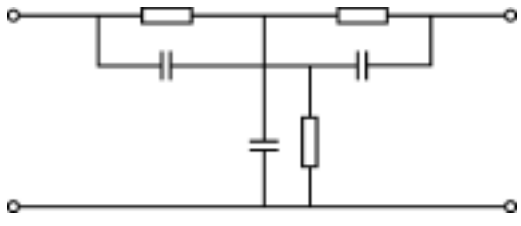


图 7.12-4　双 T 滤波器

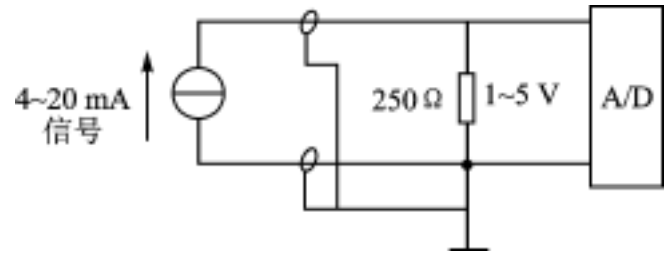


图 7.12-5　采用电流传输方式

(2) 对共模干扰的抑制　共模干扰是指信号的两根线上共有的干扰信号,是由于被测信号的接地端与微机系统的接地端之间存在一定的电位差而引起的。

抑制共模干扰的方法有:

① 采用双端输入的差动放大器,这种放大器具有很高的共模抑制比。

② 采用隔离方法消除不适当的共地带来的共模干扰,如使用带有光电隔离的测量放大器。如果信号传送距离长可以采用频率信号形式传送,便于实行光电隔离。

3.传输线

在微机控制系统中,从被测信号处和执行机构微机都有可能有相当长的距离。由于受空间电磁场的影响,会从这些传输线给微机系统带来干扰。消除这种干扰的措施有:

(1) 敷设线路时要使被测信号线、控制信号线与交流电源线、电气设备驱动线、大功率电气设备离开一定距离。

(2) 信号线使用双绞线,使空间电磁场一个个小环路中产生的感应电动势相互抵消;另外还可以采用屏蔽线或将信号线穿入金属管,屏蔽层或金属管的一端良好接地。

(3) 通过光耦合器将长线"浮置"起来,如图 7.12-6 所示。这样能有效地消除从传输线带入微机的干扰。

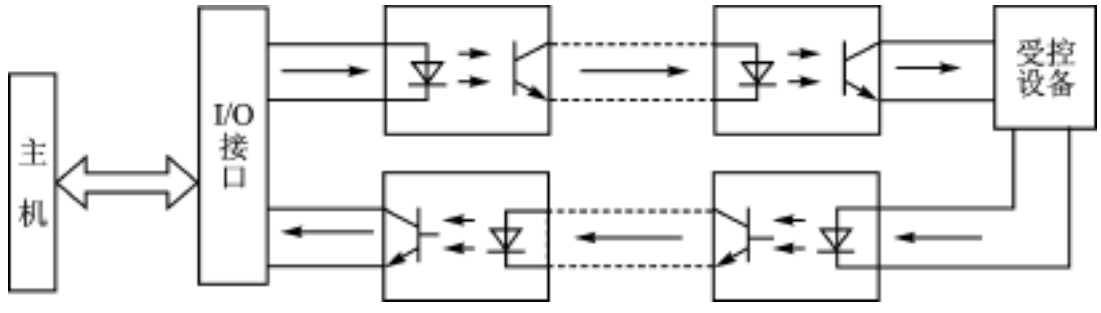


图 7.12-6　传输长线的光耦浮置处理

(4) 为了避免信号失真,对于长线传输要注意阻抗匹配。阻抗匹配的方法有:

① 终端并联阻抗;

② 始端串联阻抗;

③ 终端并联隔直阻抗;

④ 终端接钳位二级管,分别见图 7.12-7(a～d)。

4.接地系统

微机系统中的接地是一个重要问题,不适当的接地会形成产生干扰的回路,而正确的接地

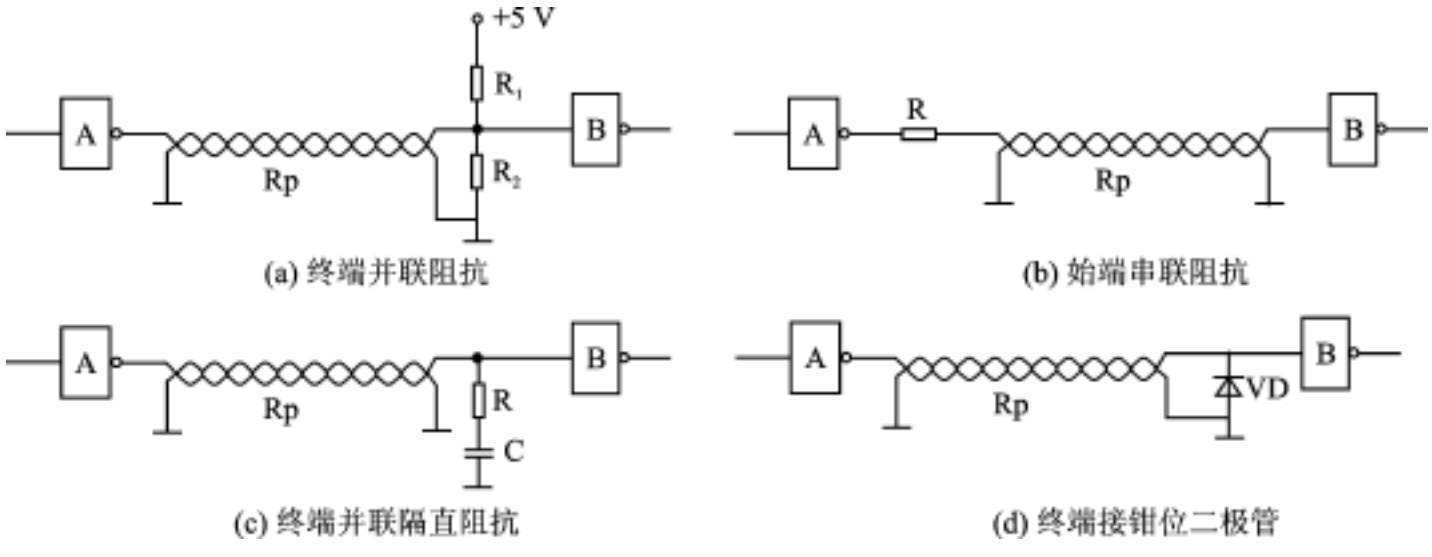


图 7.12-7 传输长线的阻抗匹配

能消除干扰,关键是对于不同的地要有不同的处理方法。

(1) 微机系统的地与信号地、功率地必须分开,也避免干扰信号通过地线传入微机。信号地与功率地也要分开,以免影响被测信号。

(2) 数字地与模拟地必须区分开,且只在一点相连,否则两种回路会相互影响。

(3) 长传输线的屏蔽层应一端接屏蔽地。屏蔽地都接到机柜,然后单独接大地。

(4) 当有多个元器件要接在同一地线上时,这些元器件应多点就近接地还是统一接地?在低频(<1 MHz)电路中,因元件和布线的电感不大,为减小地线环路造成的干扰,常采用一点接地。在高频(>10 MHz)电路中,元件和布线的电感和分布电容将造成各接地线之间的耦合,为缩短接地线,采用多点就近接地。当频率为 1～10 MHz 之间时,如采用一点接地,其地线长度不应超过波长的 1/20,否则应采用多点接地。

二、软件抗干扰技术

1.消除数据采集的干扰误差

(1) 算术平均法:对每一点的数值连续采样多次(一般取 3～5 次),计算其平均值,以此平均值作为该点的采样结果。这种方法可以减小系统的随机干扰对采集结果的影响。

(2) 比较取舍法:这种方法用来剔除采样数据中较大的偏差。具体做法是对每个采样点连续采样几次,根据所采数据的变化规律,确定舍取办法来剔除偏差数据。例如,对每个采样点连续采样 3 次,取其中 2 次相同数据为采样结果。

(3) 中值法:对一点数据连续采样 N 次(一般 N 为奇数),把 N 次的采样值依大小次序排列,取其中间值作为该点的采样结果。中值滤波对于偶然因素造成的波动或采样器件不稳定所引起的脉动干扰比较有效。

(4) 一阶递推数字滤波法:这种方法是利用软件完成 RC 低通滤波器的算法,代替硬件 RC 滤波,计算公式为:

$$Y_K = (1 - \alpha) Y_{K-1} + \alpha X_K$$

式中,X_K 为第 K 次采样值;Y_{K-1} 为第(K-1)次滤波结果输出值;Y_K 为第 K 次滤波结果输出值;α 为滤波平滑系数:

$$\alpha \approx T/\tau$$

式中,τ 为 RC 滤波器时间常数;T 为采样周期。

2.确保正常控制状态

(1) 对于开关量的输入,为了确保信息无误,在软件中可采取重复读入的方法(至少两次)。认为无误后再输入。开关量输出时,应将输出量回读(要有硬件配合),以便确认输出无误。

(2) 采用软件冗余方法。在条件控制系统中,对于控制条件的一次采样、处理、控制输出改为循环地采样、处理、控制输出。这种方法对于惯性较大的控制系统具有良好的抗偶然因素干扰的作用。

(3) 设置当前输出状态寄存单元。当干扰侵入输出通道破坏输出状态时,系统以及时查询该寄存单元的输出状态信息,以便及时纠正输出状态。

3.程序运行失常后的恢复

(1) 设置软件陷阱:这种方法是在非程序区设置拦截措施。当 PC 失控、程序"飞走"进入非程序区时,使程序进入陷阱,从而迫使程序返回初始状态。

(2) 设置时间监视器:设置软件陷阱能解决一部分程序失控问题,但当程序失控后进入某种死循环时,软件陷阱可能不起作用。使程序从死循环中恢复到正常状态的有效方法是设置时间监视器。定时时间稍长于正常执行一次程序循环所需的时间。当程序未受到干扰时,CPU 定时输出脉冲信号,使时间监视器复位,重新计时,即重新监视。一旦程序受到干扰、飞走而陷入死循环就没有复位脉冲信号加到时间监视器上。时间监视器便连续计时,直到超出定时范围,输出一个信号。利用这一信号,可以使程序恢复到初始状态,重新启动。

三、结束语

工业控制计算机工作在生产现场,各种电气设备的频繁启停造成的高低频电磁干扰和大幅度电网电压波动都直接影响计算机的正常运行。另外,生产现场还可能存在振动、粉尘以及湿度、温度等问题。因此,微机控制系统必须具有很强的抗干扰能力和适应环境的能力,以保证在恶劣的环境下正常工作。

参 考 文 献

1 卢存伟,等.微机原理及其应用系统设计.南京:河海大学出版社,1992

2 朱世鸿.IBM PC 微机接口和编程应用技术实验.合肥:中国科技大学出版社,1992

3 中国计算机服务公司.工业控制机原理及其应用.北京:电子工业出版社,1987

选自《电脑与信息技术》双月刊,2000 年第 5 期

7.13　单片开关电源瞬态干扰及音频噪声抑制技术

石家庄市河北科技大学(050054)　沙占友　薛树琦　唱春来

本文介绍抑制 TOPSwitch 和 TinySwitch 系列单片开关电源瞬态干扰及音频噪声的方法,这对提高其电磁兼容性(EMI)至关重要。

一、抑制瞬态干扰

瞬态干扰是指交流电网上出现的浪涌电压、振铃电压、火花放电等瞬间干扰信号,其特点是作用时间极短,但电压幅度高、瞬态能量大。瞬态干扰会造成单片开关电源输出电压的波动;当瞬态电压叠加在整流滤波后的直流输入电压 V_1 上,使 V_1 超过内部功率开关管的漏-源击穿电压 $V_{(BR)DS}$ 时,还会损坏 TOPSwitch 芯片,因此必须采用抑制措施。

1.瞬态电压的特点

瞬态电压的两种典型波形分别如图 7.13-1(a)、(b)所示。(a)图是由国际电工委员会制定的 IEC1000-4-5 标准中给出的典型浪涌电压波形,V_p 为浪涌电压的峰值,通常选 V_p = 3 000 V的测试电压。T 是浪涌电压从 0.3V_p 上升到 0.9 V_p 的时间间隔。T_1 为上升时间,T_1 = 1.67T = 1.2 μs±30%。浪涌电压降到 0.5V_p 所持续的时间为 T_2,T_2 = 50 μs。(b)图示出由 IEEE-587 标准中给出的典型振铃电压波形,其峰值也是 3 000 V(典型值)。第一个周期内的正向脉冲上升时间 T_1 = 0.5 μs,持续时间 T = 10 μs;负向脉冲的峰值已衰减为 0.6V_p。

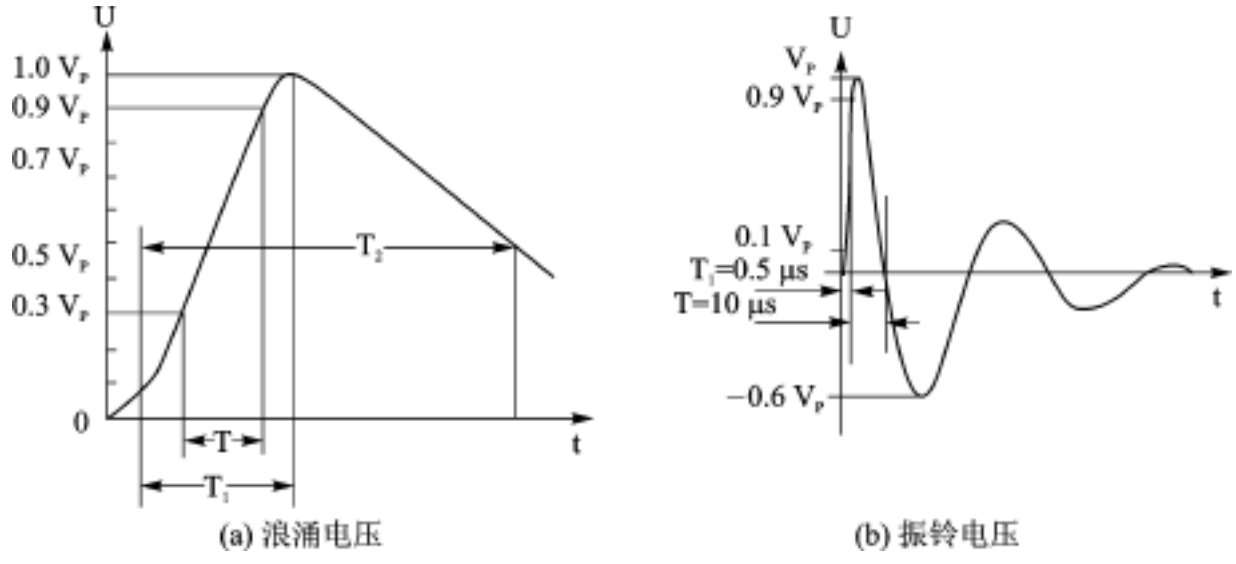


图 7.13-1　瞬态电压的两种典型波形

2.抑制瞬态干扰的方法

(1) 改进电路

现以 TOP202Y 构成 7.5 V、15 W 开关电源模块的电路为例,阐述抑制瞬态干扰的方法。其改进电路如图 7.13-2 所示,主要做了以下改进:① 将交流两线输入方式改成三线输入方式,G 端接通大地;② 采用两级电磁干扰(EMI)滤波器,为避免两个 EMI 滤波器在产生谐振

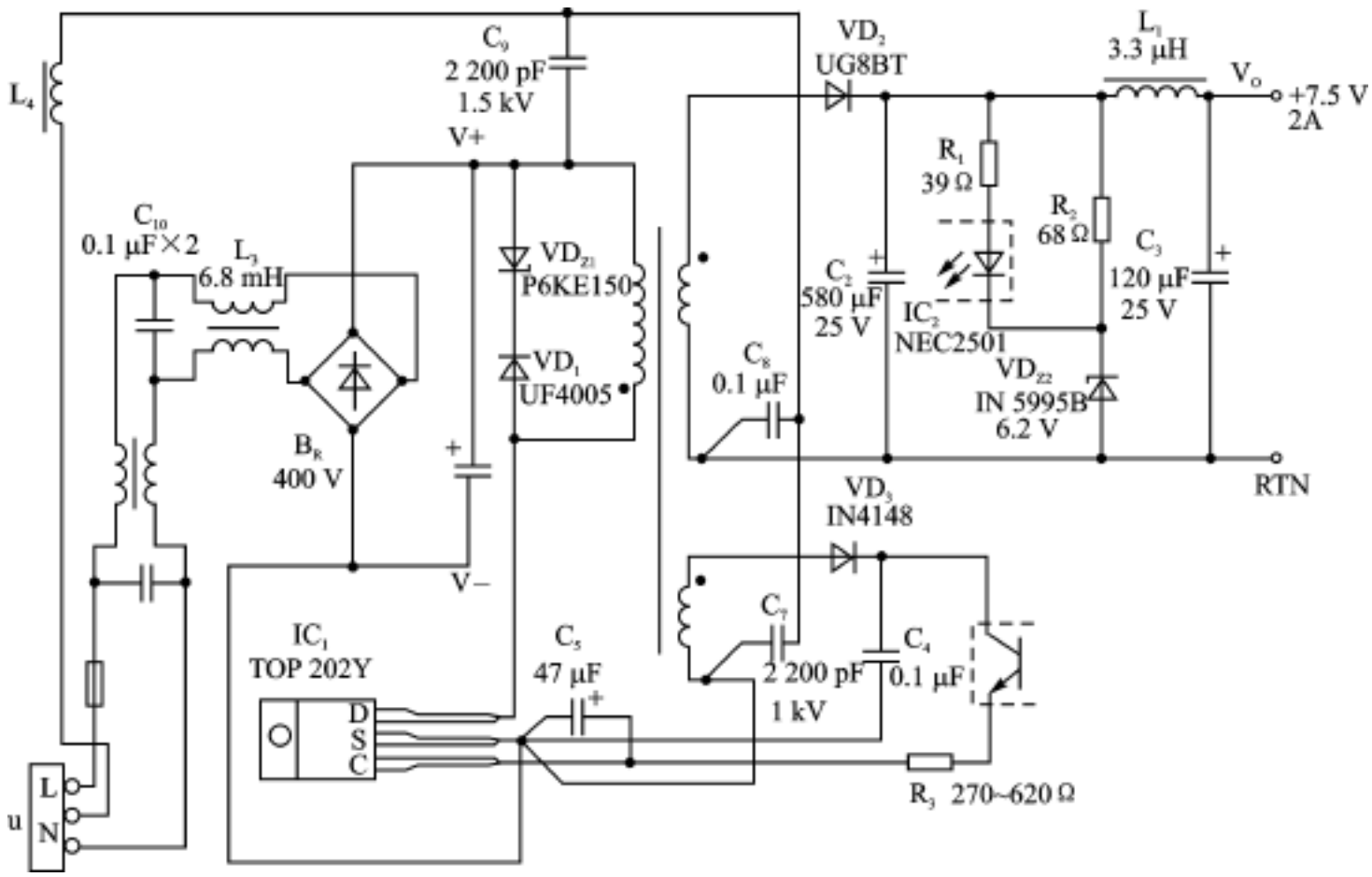


图 7 .13－2　能抑制瞬态干扰的改进电路

时的干扰信号互相叠加,应使 L_2 (L_3)≤10 mH, L_3 ≥2L_2;③ 增加 C_9 和 L_4,并将 C_8 换成 0 .1 μF 普通电容器。C_7～C_9 为安全电容,分别与高频变压器的引出端相连。其中,C_7 接初级直流电压的返回端,C_8 接次级返回端,C_9 接初级直流高压端。它们的公共端则经过滤波电感 L_4 接通大地。L_4 用铁氧体磁环绕制而成。设计印制板时,连接 C_7～C_9 的各条印制导线应短而宽。采用上述连接方式可保证瞬态电流被 C_7～C_9 旁路掉,而不进入 TOP202 Y 中。此外,反馈绕组接地端和 C_4 的引出端,各经一条单独导线接 TOP202 Y 的对应管脚。旁路电容 C_5 直接跨在控制端与源极上,以减小控制端上的噪声电压。④ 增加电阻 R_6,其阻值范围在 270～620 Ω,它与光耦合器发射极相串联。当控制环路失控时,R_6 能限制峰值电流,使之小于芯片中关断触发器的关断电流。

(2) 减小瞬态干扰的其他措施

· 为减小瞬态峰值电流,应在初、次级绕组之间绕 3～5 层 0 .05 mm 厚的聚酯绝缘胶布,使高频变压器的分布电容量降低。

· TOPSwitch 的外接散热器应与芯片上的小散热板连通。若两者之间加绝缘垫片,且散热器与电路连通位置又不合适,则散热器与小散热板的分布电容就会和电路中的电感发生谐振,产生高频振铃电压,使 TOPSwitch 中的关断触发器误操作。

· 提高整流桥耐压值并适当增加输入滤波电容 C_1 的容量。

· 利用共模扼流圈对过大的共模干扰电流进行抑制。

· 在 110～115 VAC 低压输入时,选择 TOPSwitch - II 系列产品,以提高芯片的漏 - 源击穿电压值。

·在交流进线端并联一只压敏电阻器(VSR),对瞬态电压进行钳位,电路如图 7 .13－3 所示。

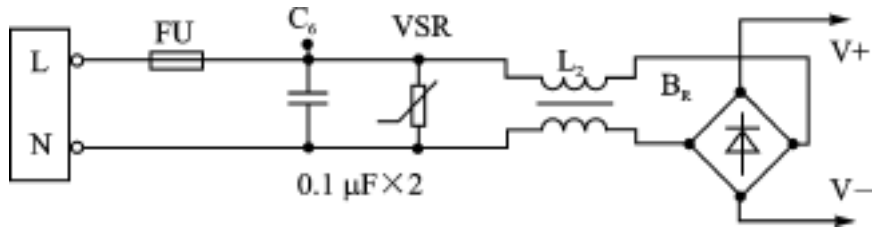


图 7 .13 - 3 利用压敏电阻器钳位瞬态电压

二、抑制音频噪声

单片开关电源的音频噪声用人耳即可听到,它主要是由电容器和高频变压器产生的。

1 . 电容噪声

电介质材料具有压电效应,其形变大小与电场力的平方有关,二者呈线性或非线性关系。某些非线性电介质在常温下就具有压电效应,例如在 TOPSwitch 漏极 RC 吸收回路中使用的耐高压陶瓷电容器,是由非线性电介质钛酸钡等材料烧结而成的,在周期性尖峰电压的作用下,使电介质不断发生形变,能产生较大的音频噪声。采用耐高压的聚脂薄膜电容能降低电容噪声。

2 . 高频变压器噪声

高频变压器 EE 或 EI 型磁芯之间的吸引力,能使两个磁芯发生位移,绕组电流相互间的引力或斥力,也能使线圈产生偏移。此外,受机械振动时能导致周期性的形变。上述因素均会使高频变压器在工作时发出音频噪声。10 W 以下单片开关电源的音频噪声频率,约为 10～20 kHz。

为防止磁芯之间产生相对位移,通常以环氧树脂作胶合剂,将两个磁芯的 3 个接触面(含中心柱)进行粘接。但这种刚性连接方式的效果并不理想。因为这无法将音频噪声减至最低,况且胶合剂过多,磁芯在受机械应力时还容易折断。国外最近采用一种特殊的“玻璃珠”(glass beads)胶合剂,来粘合 EE、EI 等类型的铁氧体磁芯,效果甚佳。这种胶合剂是把玻璃珠和胶着物按照 1∶9 的比例配制而成的混合物,它在 100℃以上的温度环境中放置 1 小时即可固体。但其作用与滚珠轴承有某种相似之处,固体后每个磁芯仍能独立地在小范围内变形或移位,而总体位置不变,这就对形变起到了抑制作用。用玻璃珠胶合剂粘接的高频变压器内部结构如图 7 .13 - 4 所示。采用这种工艺可将音频噪声降低 5 dB。

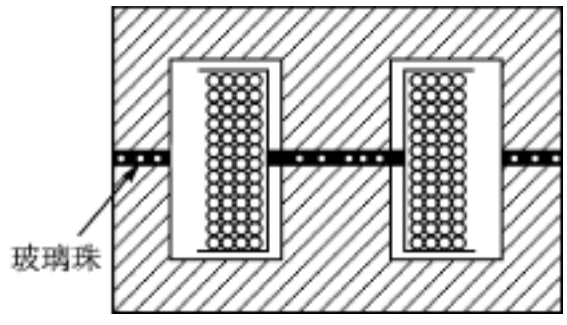


图 7 .13 - 4 高频变压器内部结构

给线圈浸入清漆、烘干之后,不仅能隔绝潮气,加强坚固性,还有助于减小音频噪声。不过灌入清漆后也会增大初级绕组的分布电容,降低高频变压器的固有谐振频率。

3 . 抑制 TinySwitch 开关电源中的音频噪声

由 TinySwitch 构成开关电源时,音频噪声主要是由钳位保护电路和 RC 吸收回路产生的。为此可采取以下几种措施:

(1) 将 RCD 型钳位保护电路中的二级管 VD,换成高压稳压管 VDz;把漏极 RC 吸收回路中的高压陶瓷电容器换成压电效应很小的聚脂薄膜电容器。改进电路如图 7 .13 - 5 所示。

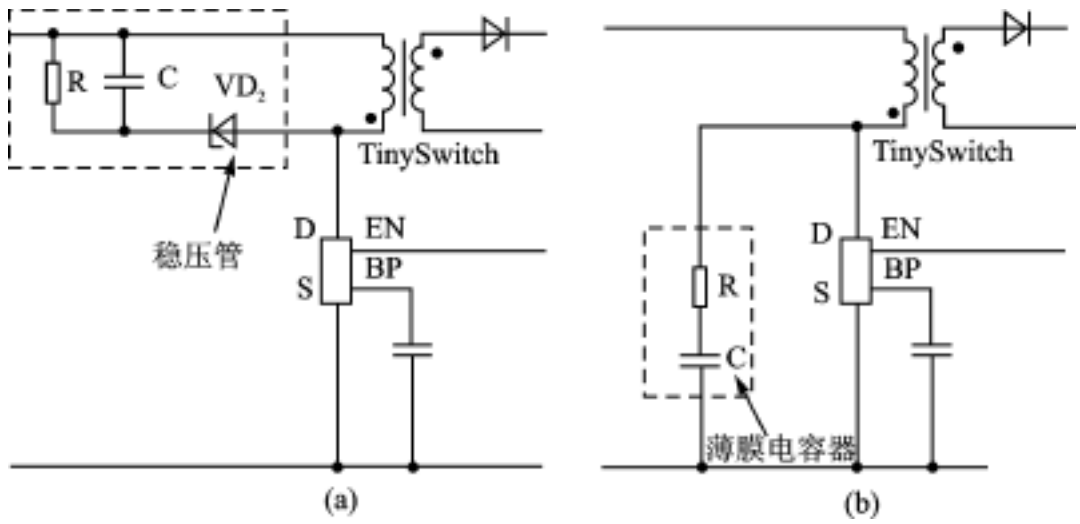


图 7 .13－5　TinySwitch 开关电源的改进电路

(2) 为进一步减小音频噪声,还可选择磁通密度较低的磁芯,例如当最大磁通密度从0 .3T 减小到 0 .2T 时,可使音频噪声降低 10～15 dB。

参 考 文 献

1　沙占友 .电磁兼容性的设计与测量 .电子测量技术,1997;(4)
2　沙占友 .单片开关电源的发展及其应用 .电子技术应用,2000;26(1)

选自《电子技术应用》月刊,2000 年第 12 期

7.14 单片机应用系统程序运行出轨问题研究

郑州电力高等专科学校(450004) 丁书文 杨雪萍 田玉珍
武汉水利电力大学电力系(430072) 张承学

随着单片机应用系统的发展,其应用范围越来越广泛,所处运行环境越来越复杂。如电力系统单片机应用系统的工作环境就具有较强电磁干扰。这些干扰虽然不能造成应用系统的硬件损坏,但常使单片机应用系统软件不能正常工作。其中一种现象就是干扰对单片机应用系统程序运行造成不良影响。虽然单片机应用系统在硬件设计时采取了多种常见的措施,如各种接地处理、屏蔽、隔离、滤波、退耦、旁路[1]等,抑制或消除了一定干扰,但这些措施并非万能,因单片机应用系统本身的整体配置、元器件(特别是新型器件)的使用及系统硬件、软件设计中的不完善因素,也会对可靠性产生重要影响。所以,从软件方面进行防护和抑制干扰的研究,软、硬件设计中对抗干扰进行综合考虑的研究,正逐渐受到人们重视。本文根据运用 8031 单片机的实际设计经验为例,提出若干种有效、实用的程序运行出轨对策,能使单片机应用系统在抗干扰方面具有更高的运行可靠性和程序出轨的自恢复能力。

一、程序运行出轨主要特征

程序运行出轨是指由随机干扰因素破坏了程序执行正常顺序而造成程序执行卡死现象。其主要特征是:数据码或指令码的个别字受干扰而发生跳变,使程序运行出轨。其中,最常见的错误情况是随机干扰因素破坏了程序计数器 PC 值,使 CPU 从 EPROM 中读取指令时出错,改变了程序正常运行的顺度,其最终结果是,如果这个误码 CPU 不认识,程序运行将发生中断;如果这个误码是可执行码,那么在执行了一系列非预期指令后往往磁到一条 CPU 不认识的指令操作码而停止工作,或进入一种死循环,使整个单片机应用系统控制失灵。这很可能会造成设备和生产的事故,严重时危及整个电力系统安全稳定运行。

二、程序运行出轨的软件举措

1.利用监视系统监视程序运行

防止程序运行出轨的常用方法,一是采用自身带有 Watchdog 的 CPU(及有些单片微处理器);二是设置监视跟踪定时器,使装置具有超时自动复归能力。这里介绍一种在微机保护装置软件设计中已经使用过,通过产品实际运行验证,抗干扰效果较理想的防范措施[2]。

使用定时中断来监视主程序运行状态时,定时器的定时时间要稍大于主程序正常运行一个循环的时间,而且要在主程序执行过程中执行一次对定时器时间常数的刷新操作,这就保证定时器在程序运行正常时不会出现中断,而当程序失去控制时,会产生中断,再利用中断服务程序使系统自动复位。在微机保护应用 8031 构成的硬件系统中,作为拦截程序运行出轨的一个实例,具体做法为:(1) 利用接口芯片 8155 中的一个定时器来作为定时中断使用,其产生的中断信号作为 8031 的外部中断源$\overline{\text{INT1}}$的输入信号,用 555 定时器作为 8155 定时器的外部时

钟输入。(2) 通过软件设计使 8155 定时器的定时值稍大于主程序正常循环时间,并且在主程序循环期间重置 8155 定时器的定时时间常数。(3) 若中断源只供抗干扰使用。只编制相应的中断服务程序即可;若与硬件中断合用,则对是硬件复位还是定时中断产生的自动复位进行判断。利用 8155 的好处是 8155 还含有三个扩充口和 256 个存储单元。当然,单片机 8031 中有定时器,如果系统不用,其内部定时器可供其他场合使用。

2.利用简单可靠的软件陷阱

当正常运行的单片机应用系统由于干扰破坏使 PC 值失控造成程序出轨后,CPU 离开原程序轨道而不断进入非程序区。在这种情况下,可在非程序区设置拦截措施。这就是使程序进入陷阱,然后强迫程序进初始入口状态。对 MCS-51 系列单片机的微机应用系统,理论上可用指令 LJMP#0000H,即在非程序区,程序存储器中(EPROM)写入指令码 020000H,实际上的情况要比这种情形复杂一些。因为 LJMP 的指令码是 02,而 JB 的指令码为 20,NOP 的指令码为 00,故可以连续使用这些指令的组合。LJMP#0000H,NOP,JB0,#000H,NOP,LJMP#0000H…的指令码之组合为 02000000200000000020000…,用这种码填满非程序区,不论 PC 失控后指向这串码中的哪个字节,最后都能导致程序执行 020000 指令码,返回到地址 0000H 处自动复位。通过复位后,单片机软件系统就可以重新开始运行原程序。

3.巧妙利用软件“看门狗”

当干扰使程序运行已出轨,正好进入一个重构的错误循环程序。若装置中设计有这方面的硬件电路,而该程序正好包含了“看门狗”电路的访问,且访问时间间隙小于“看门狗”的溢出时间,则上述一般“看门狗”电路和软件陷阱作用将失效。

对于上述问题,可在程序的另一处设计一段软件程序,使之具有“看门狗”的功能,即在 RAM 中找一个内存单元对程序循环次数进行记录,当记录数为 N 时,8031 单片机必须访问定时器以重新启动它,同时对该内存单元清零定时器的定时时间略大于 N 次循环时间 T,如果 T 时间内 CPU 未访问定时器,定时器将溢出引 CPU 复位。

4.程序模块设计中的综合考虑

对于电力系统中单片机应用实时监控系统,系统正常时,其程序一直在循环运行,而当其监视对象出现故障后,微机应用装置程序将通过一系列原定的程序模块运行而最终作出相应反映,若在这个过程中程序运行出轨,总希望引导系统尽快恢复执行刚才失控发生时的那个程序模块,而不希望这时的失控程序复位,这样更有利于装置的快速恢复正常工作和系统运行的稳定性。因此如何使出轨程序尽快、更好地重新进入程序正常运行状态是一个非常棘手的问题。这里介绍一种在程序模块设计中解决这个问题的一种方法。

对于微机实时监控系统的软件,是由完成各种功能模块的程序组成的。为了能使出轨程序尽快恢复到程序的正常运行状态,在进行软件编制时,就应考虑该问题,即将装置软件编制成模块化结构,这不仅有利于程序的调试而且可以使程序出轨恢复时,发现程序出轨更早、程序恢复运行速度更快。具体做法是,将细分的各个功能模块进行编号,每个功能模块在运行中需具有写入和记录功能,即设置 RAM 区的有效标志;记录功能模块编号和首地址;具有给运行监视系统发脉冲的功能等。在每个功能模块的结尾处将指定单元中保存的标志与本功能模块特有的标志进行对比来判断运行的程序是否出轨。通过标志字的对比,若相同则标明没有发生功能模块间乱窜现象,程序继续按原来程序顺序进行;若不同则表示程序运行出轨,这时就让程序转到指定单元中保有的标志所对应的功能模块去重新执行。对于功能模块内的出轨

问题,可根据本功能模块的具体情况,来对结果的合理性进行分析和判断(其判断条件会因模块不同而不同)。若判断结果认为合理,则进入下一功能模块继续程序运行;若判断结果不合理,则通过调转指令返回本功能模块重新执行。上述功能的程序流程图如图 7 .14 - 1 所示。

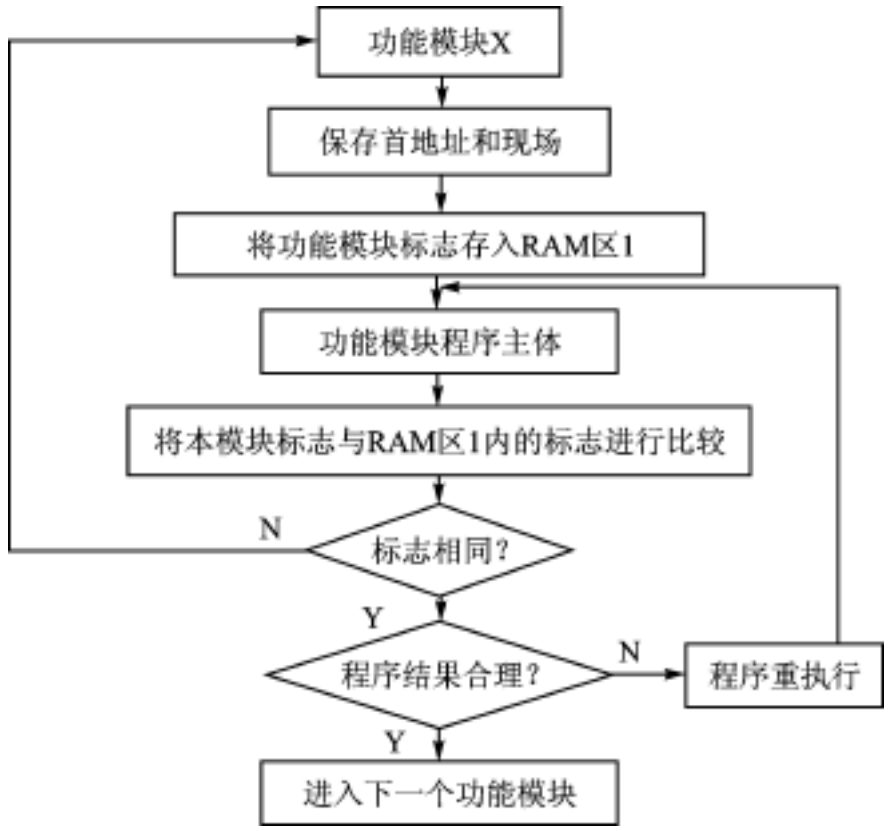


图 7 .14 - 1 程序运行重恢复功能模块流程图

三、利用硬件实现程序出轨自恢复

对于程序运行出轨问题,也可以利用专用的硬件电路来监测程序出轨,并实现自动恢复正常。这里介绍一种硬件自恢复电路方案如图 7 .14 - 2 所示,其中 A 点接至单片机应用硬件电路的某一点,例如并行接口的某一输出端口位,当程序没有运行出轨时,由软件安排使该点电位按一定的周期 T 在“ 1 ”和“ 0 ”之间周期性的变化。图中元件 t_1 为两个瞬时返还而延时 t_1 动作的元件,延时 t_1 应比 A 点电位变化的周期 T 长,因此在正常运行时两个延时元件都不会动作,“ 或 ”门输出为“ 0 ”。一旦程序运行出轨,A 点电位停止变化,不论它停在“ L ”态或是“ 0 ”态,两个延时元件中总有一个动作,因而使单片机应用系统重新初始化,恢复正常运行。这个电路的优势在于,不仅可以用来对付程序运行出轨,还可以用于在单片机应用系统主要元件(例如 CPU)损坏而停止工作时发出报警信号。因为在这种情况下,单稳触发器发出复位脉冲后,不能使 A 点电位恢复至周期性变化状态,这时将通过 t_2 的延时发出告警信号。

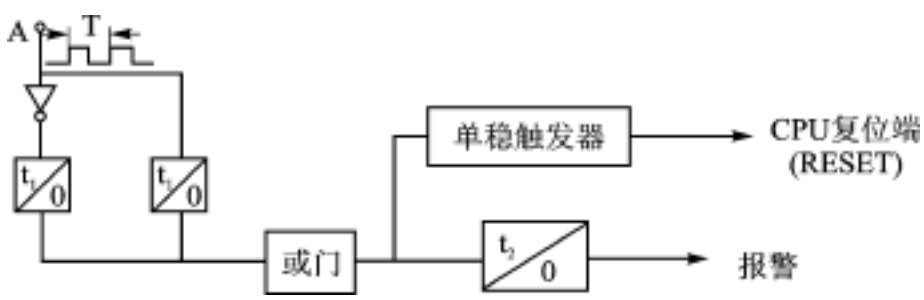


图 7 .14 - 2 程序运行出轨自恢复回路

四、利用软、硬件进行综合设计

一般的单片机应用系统中,若安排硬件自复位电路,就是所说的“看门狗”电路。在程序运行出轨后大多能实现硬件自复位功能[3]。但是,对于电力系统中的实时监控系统装置,由于装置的启动或判断通常要使用当时采集数据与过去记录数据进行比较判别。在程序运行出轨造成死机较长后复位,有可能会使保护装置启动误动作或判据失真,这可能造成严重后果。为尽快判别程序出轨和恢复程序正常运行,在硬件自复位前设置软件条件复位就具有十分重要的意义。

硬件自复位电路即“看门狗”电路本身就可看作是一个可被清除的定时脉冲发生器,它本身可以由单稳态触发器或计数器等构成。如果没有清除脉冲的话,它将产生一个固定频率的输出脉冲,这个脉冲可以引到系统的复位端。当主程序正常工作时,不断地发出清除脉冲,使脉冲发生器无输出。一旦程序运行出轨,清除脉冲消失,“看门狗”工作,将发出一个复位脉冲,使系统回到正确的工作状态。对上述一般的“看门狗”工作性质,初看起来很完美,但实际运行时仍有死机现象发生,其原因有:程序运行出轨后,“看门狗”发出复位脉冲,但程序刚刚正常尚不及发出一个脉冲时,程序再次被干扰而出轨,此时“看门狗”正处于稳态,将不再发出复位脉冲。因此,对于一个较理想的硬件复位电路应具有在检测出主程序运行出轨时发出的复位脉冲不应为单次脉冲,而应在主程序复位前期间内每过一定的时间间隙都要发出复位脉冲,直到主程序恢复正常工作为止[4]。

针对电力系统单片机应用系统对程序运行出轨进行复位的要求快速性和可靠性,一种软、硬件综合设计为:在系统正常运行时,由 CPU 定时输出一个脉冲信号,这可用单触发器与计数器组成的电路来进行检测与判断,利用计数器的输出 Q_i 提供软件条件复位信号,Q_j 作为硬件自复位信号($j > i$)。从程序运行出轨而死机到硬件复位开始的这段时间内,在硬件复位之前,硬件复位电路给出一个预复位信号,该预复位信号产生一个外部中断,在该中断服务主程序中分别判断系统当时的状态,根据不同状态条件,引导程序进入合适的入口地址。所以,这个中断服务程序在硬件自复位之前实现软件复位功能,其条件可能有多个,复位入口地址也各不相同。若进入该中断服务程序后,马上给出计数器的一个消除脉冲。

1 .硬件自复位电路的设计

图 7 .14 -3 为硬件自复位电路原理接线图,利用 8031 单片机的口线(如用 P1 .0)作为硬

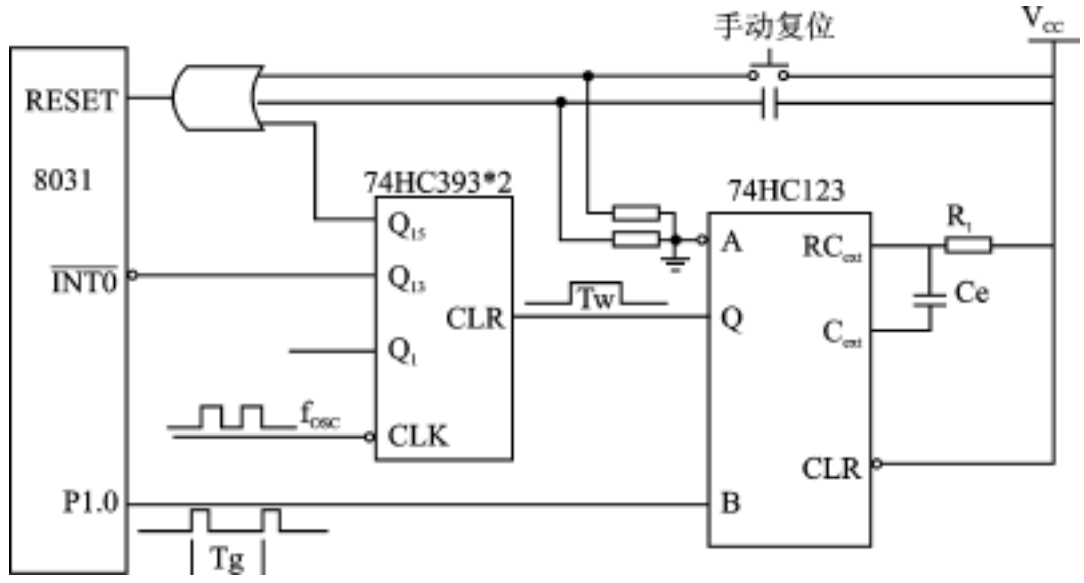


图 7 .14 -3　硬件自复位电路原理图

件复位电路的输入清除信号,用 74HC123 可再触发单稳触发器形成脉冲输出,用两片 74HC393 组成 16 位进制计数器。当程序运行出轨后,8031 口线 P1 .0 脉冲 Tg 清除消失,单稳态触发器形成一个脉冲信号 Tw 作用于计数器的复位端,使计数器的所有输出端 Q = 0。Tw 由电阻 R_t 和电容 C_e 大小决定,计数器的计数时钟直接利用 CPU 的时钟发生器。CPU 的复位输入用计数器 Q_{15} 的硬件自复位,Q_{13} 作为在硬件复位前的一个预复位中断信号接到 CPU 的$\overline{\text{INT0}}$上,提供软件复位的起始信号。若 CPU 不能响应$\overline{\text{INT0}}$中断,则等到硬件复位时间将由硬件完成复位。若选取 CPU 时钟振荡器频率为 12 MHz,通过计算,在软件条件复位不能实现的最不利情况下,从死机到完成硬件自复位的时间为 2 .73 ms,对于微机保护系统将不会产生太大的影响。

2 .软件条件复位的实现

软件条件复位是通过 CPU 接受外部中断信号$\overline{\text{INT0}}$后,来执行的一段中断子程序。这时 CPU 进入中断(1)关闭所有中断,以便进行故障的判别和处理;(2) 堆栈初始化以保证执行完成$\overline{\text{INT0}}$中断服务子程序后栈底为初始状态;(3) 对系统的有关状态与控制量进行比较、判断来决定程序的重新入口地址,这些地址是主程序中基本功能模块的程序首先地址。通过这种软件条件复位,可以使系统在无扰动或小扰动下,尽快进入正常运行状态,尽可能地减少对微机保护系统的冲击,尽量减少干扰对微机保护系统的影响。

五、总 结

电力系统中的各种电气设备存在着强大的电磁干扰,而电力系统的安全、稳定运行又要求单片机应用系统具有特别高的可靠性,干扰对微机应用系统的程序运行出轨问题一直是单片机应用系统设计中最棘手而又必须妥善解决的问题。本文在分析单片应用系统程序运行出轨具有的特征基础上,通过反复探索,就如何防范程序运行出轨,从软件、硬件和软硬件综合考虑等方面介绍了一些方案和实例,这里介绍的防范程序运行出轨的综合措施简单、可靠、易于实现,特别是为了避免单独硬件或软件自复位可能产生的大扰动而提出的硬件自复位与软件条件复位综合设计方法,已成功地应用在“自动励磁调节器”、“电力系统故障精确定位装置”等系统中。

参 考 文 献

1 陈德树 .计算机继电保护原理与技术 .北京:水利电力出版社,1992
2 丁书文等 .微机保护系统的软件抗干扰设计 .华北电力技术,1999(2):46～49
3 杨奇逊 .微型机继电保护基础 .北京:中国电力出版社,1988
4 何立民 .单片机应用系统设计 .北京:北京航空航天大学出版社,1998

选自《电气自动化》双月刊,2000 年第 4 期

7 .15　分布式系统故障卷回恢复技术研究与实践

国防科技大学计算机学院
文　梅　李宏亮　张春元　范金鹏　吴　涛　王志英

现代航天、航空、邮电、石油等重要应用领域,对计算机系统的实时性能和可靠性提出了越来越高的要求。而系统硬件和软件的故障,可能使操作失败,所以对这种时间严格受限的实时系统的容错要求是:在任何情况下,系统必须在一给定的时间内返回正确结果。卷回恢复是提高分布式系统容错性能的重要后向恢复技术。

一、卷回技术

卷回技术是指系统从故障中恢复时,从出错时刻以前的某一确定正确的时刻开始恢复。系统从故障中恢复到正常状态需要有两个过程:检查点设置和卷回恢复。关于检查点设置,有两种方法:

(1) 进程独立地设置检查点,并存储所有检查点。

故障发生时,系统通过进程间的相关跟踪和可能的消息记录,来确定一个一致的确定正确的检查点集合,系统从该集合中找到有效检查点卷回,能维持系统中各节点上内容的一致性。

这种方法的主要缺点是“多米诺效应”。对分布式进程系统来说,进程间通过消息传递进行通信。如果每个进程独立设置自己的检查点,当一个进程检测到错误时,就卷回到最近一个检查点。由于进程间的通讯,建立多个进程间的相关性关系,从而引起相关进程的卷回。如图 7 .15 - 1 所示,进程 1 和进程 2 独立地设置检查点序列,消息和检查点的交错使进程 1 和进程 2 的检查点集合不一致。如进程 1 发生故障,卷回到检查点 a,取消对 ml 的发送,由于进程 2 已接收 ml,为避免这种孤儿消息(没有发送源的消息),进程 2 必须卷回到检查点 b,由于进程 1 已收到 n1,同理,进程 1 必须再次卷回……。这种卷回操作在相关进程中无节制的传播现象,称为“多米诺效应”。对于实时系统来说,这是不可接受的。该方法的另一个缺点是,保存所有检查点要占用大量的存储空间。

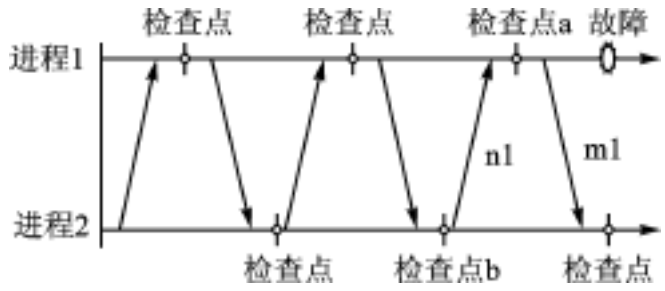


图 7 .15 - 1　多米诺效应

这种方法的优点是,在正常操作时,系统额外开销较小,能满足一般分布系统应用的高可靠性、高性能的需求。

(2) 协调设置检查点,进程只存储最近一个检查点。

系统中的检查点集合保证是一致的,故障发生时,系统恢复到这些检查点所维持的系统规范状态。

该方法克服了第一种方法的缺点,但进程异步卷回可能产生活锁问题。如图 7 .15 - 2 所示,进程 1 在收到消息 n1 前失败,卷回到检查点 x,并通知进程 2。进程 1 从 x 处开始运行,发出消息 m2,接收 n1。进程 1 恢复后,取消了对 ml 的发送,因此为避免 ml 成为孤儿消息,进程 2 应当卷回到检查点 y,以相应取消对 ml 的接收。进程 2 恢复执行后,发出 n2,接收 m2。进程 2 卷回后,取消了对 n1 的发送,而进程 1 却记录了 n1 的接收,因此进程 1 被迫再次卷回到 x,而进程 1 的卷回又将因取消了对 m2 的发送而迫使进程 2 的再次卷回,如此循环往复。虽然只发生了一次故障,但进程 1、2 互相迫使对方无限次卷回,产生活锁。为避免活锁,就要同步。正常操作的同步延迟较大,这是该方法的缺点。

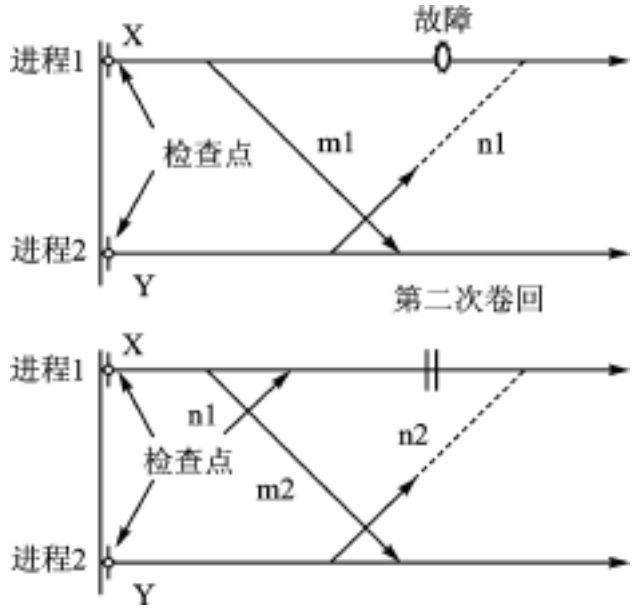


图 7 .15 - 2　异步卷回中的活锁

二、应用系统要求和计算模型

1 . 系统要求

应用系统是一个实时、高可用双工系统,具体技术要求是:200 tps 的持续流量;每个事务的独立计算量在 25～100 ms 之间;每个事务必须在 500 ms 内给出响应;高可用度要求每天 24 小时,每周 7 天提供处理服务。

2 . 计算模型

根据实际系统原型和性能要求,将每个事务请求的处理过程划分成 n 段,每一个任务段由一个处理节点来负责完成,完成时间不超过 5 ms。段 i 的输出即为段 i + 1 的输入 $(1 \leq i < n)$,没有反馈。这样,形成了宏流水双工分布式系统,该模型如图 7 .15 - 3 所示。

在以上模型中:

(1) 我们把一个完整处理过程的所有处理节点称为一个处理机组,整个系统同源输入,送给两个机组处理;

(2) 输出结果根据一定的策略从两机组的计算结果中选取;

(3) 对于两个处理机组之间的数据交换,采用专用高速互联网络。

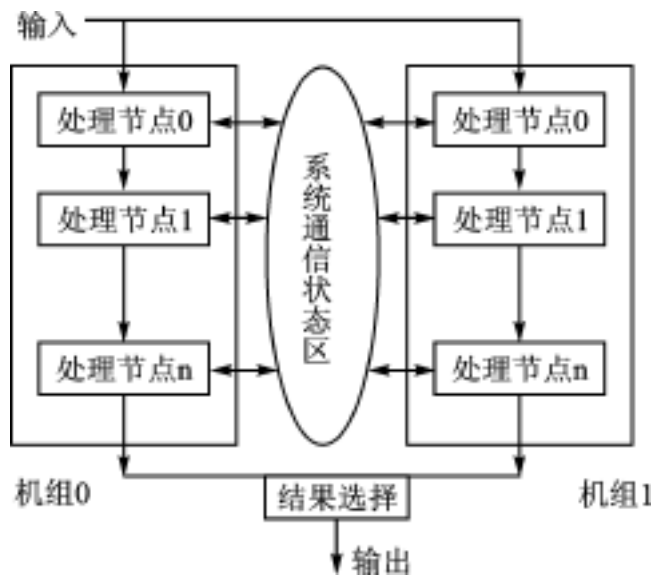


图 7.15-3 双工宏流计算模型

3.应用系统的故障恢复分析

根据实际情况,我们把应用系统的故障分成三类:系统软件故障、系统硬件故障和应用软件故障(不包括上述故障引起的应用软件故障)。对于前两种故障,有的操作系统有相应的寄存器反映其状态,或采用计数器、定时器等发现故障,并通过脱机修复等方法来恢复。本文主要讨论第三种故障的恢复。

三、应用软件故障的卷回技术

1.检查点的设置

(1) 检查点设置

应用系统采用独立的检查点设置模式,检查点为每个事务的结束点。前面已经论述过,独立的检查点设置模式不会产生“活锁”问题,但可能产生“多米诺效应”。为避免该效应,本系统采用的宏流水计算模型中,各节点段任务进程不进行断言操作,即消除了图 7.15-1 中的中间检查点,也就消除了产生“多米诺效应”的基础。经论证,这样处理可以保证系统性能要求[1]。

关于检查点的间隔问题是必须仔细权衡的。间隔时间过短,对系统打扰频繁,影响实时性,且保存检查点时内存开销大,冗余多;间隔时间过长,达不到实时性的要求,不能起到检查点应有的作用。如果每个任务段结束时设置检查点,无疑会提高检查点保存的开销,影响系统性能。而整个事务是响应时间在 500 ms 内的短事务,经分析,以事务结束点作为检查点,其设置间隔与应用的实时性要求是匹配的[2]。

(2) 断 言

每个事务完成后,两个节点的相应检查点进行比较。如果结果相同,则无故障,确认正确状态为断言;否则,有理由认为系统中至少一个机组已出现故障,需要进行故障恢复。

考虑到实际实时系统计算结果有时不可能完全一致,为了满足实时性的要求,我们采用质量评估的办法,给出容忍范围。如有一个结果满足一定精度则不认为发生故障,无需进行故障处理。

(3) 检查点状态数据

检查点状态为该时刻系统状态表中的有关数据。

为了及时检测各机组的状态,需要在系统通信状态区中设立“系统状态表”,其数据结构参

见图 7 .15 - 4。图中只显示了与检查点状态有关的数据。

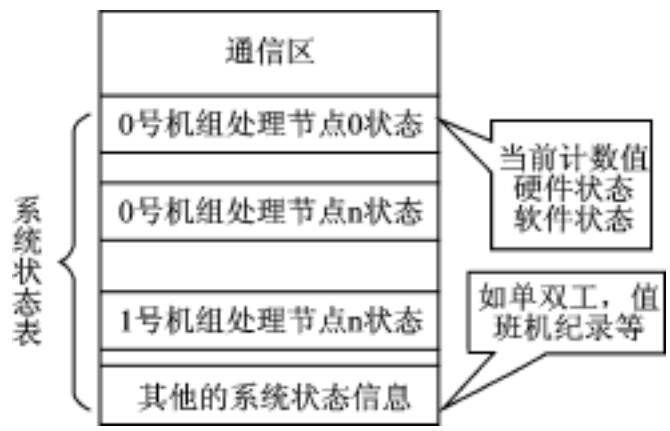


图 7 .15 - 4 应用系统的系统状态表

① 通信区的数据由所有处理机节点共享;

② 当前计数用于记录节点最近一次活跃的时间(假设各处理节点的时间均同步),由各个处理机自己填入;

③ 硬件状态、系统软件状态和应用软件状态分别由各自的处理机自行填入。

2 . 卷回技术

在断言确认故障的情况下,应用系统采用重复执行(简称复执)的卷回策略来进行故障恢复,即将故障事务提交,重新执行。为了保证系统实时性,对复执需要有所控制。

首先,能够不卷回执行,则不卷回,前面已讨论过。其次,对卷回执行结果的处理,正常执行结果应能使系统通过投票判断出故障机组,得出正确结果,并提交。但如果不能判断,也不能无休止地卷回,这样不能满足系统实时性要求。系统对事务的响应要求是 500 ms,正常执行时间为 100 ms,则卷回执行次数只能限制在 3 次以内。如果卷回执行次数已超限,则需要采用别的处理办法。应用系统运行标准测试即通过一些标准程序的运行,来判断故障机组。当卷回执行和标准测试都不能做出故障判断时,则需要进行进一步的故障处理,这不在本文讨论内容之列。具体流程如图 7 .15 - 5 所示。

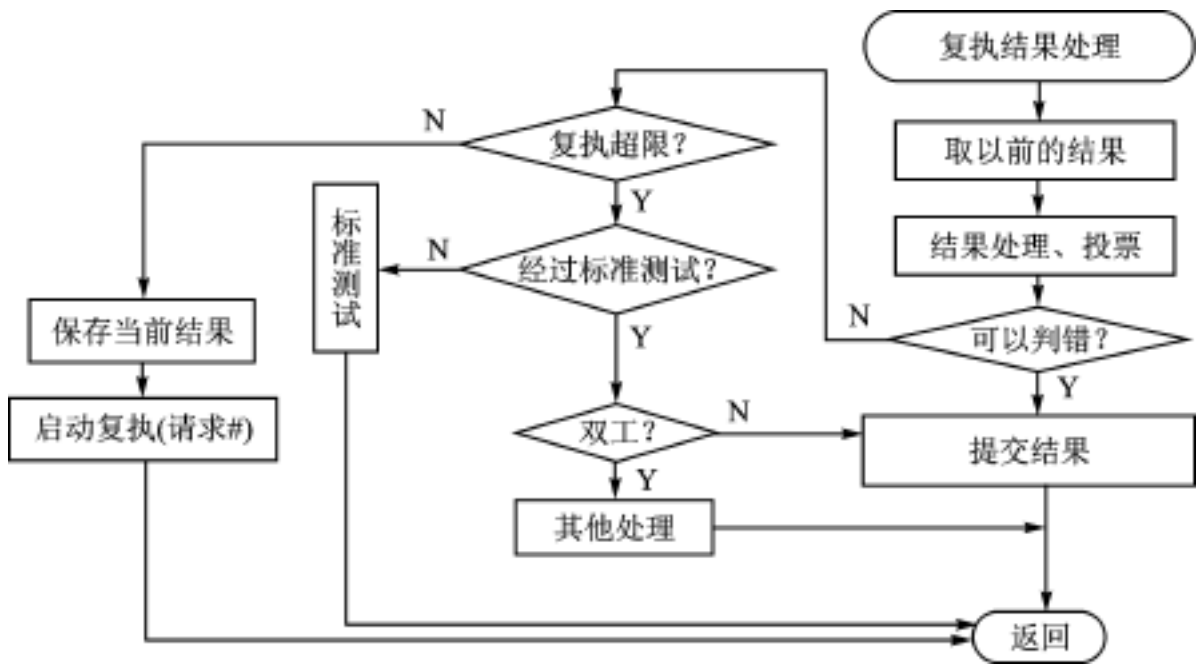


图 7 .15 - 5 卷回技术流程图

四、结束语

本文探讨了分布式实时系统中的故障恢复技术,着重讨论了实时和高可用双工系统中的应用软件故障卷回技术。对一个通用的双工系统来说,其故障检测与恢复都应该与应用程序无关,但这种系统的检错与故障恢复时间一般都较长,不能满足强实时任务的要求。因此,在设计中需要权衡用户应用程序编写的自然性和系统的效率。我们的侧重点是在保证系统实时性的前提下,尽可能减少对应用程序编写的限制和附加要求。以上所述的技术方法,已通过理论分析和原型系统的模拟证明,达到了系统的技术指标要求。

参 考 文 献

1 王志英,张春元,朱海滨 . 高可靠性群机系统通信和切换[J] . 计算机工程与科学,1997,19(3):1～8

2 王东升 . 并行机群系统后向恢复技术研究[R]:[博士后研究报告] 清华大学,1996

3 金士尧,王志英,胡华平 . 强实时高可靠的群机系统设计与论证[J] . 计算机工程与科学,1997,19(A1):1～5

4 Koo R, Toueg S . Check-Pointing and Rollback-Recovery for Distributed Systems[EB/ OL] . http:// www . elib . stanford . edu, 1985 - 10

选自《计算机工程与科学》双月刊,2000 年第 5 期

第八章

典型应用实例

8.1 基于单片机系统采用 DMA 块传输方式实现高速数据采集

南京大学物理系(210093)
刘先昆 潘红兵 纪圣谋 王 勇 徐健健

PC 机中外设与内存储器之间数据直接传输的 DMA 功能以其高效、高速、CPU 资源占用少等特点已被广泛应用,这一功能通过安装在主板上的专用 DMA 控制器芯片或集成在外围控制芯片来实现。单片机的应用领域也常常需要有高速数据传输或数据采集,虽然近些年单片机速度有所提高,仍然无法应付类似单脉冲信号捕获、周期信号频谱分析等需要采用高速 A/D 的场合。对于速率在 100 kbps 以上的数据采集或传输一般的中断查询法就不易实现,因此考虑通过直接存储存取操作才能完成。然而单片机内部设计通常不具有 DMA 功能,也没有现成的控制芯片可以利用。而目前通用 DSP 芯片对于开发小型仪器仪表而言价格过高。

数字式磁通表主要应用在对恒定磁场的磁感应强度或脉冲场磁感应强度峰值的测量。是采用闭合线圈作为探测线圈,穿过线圈的磁通 Φ 变化时,探测线圈中感应电动势: $\varepsilon = -\frac{d\Phi}{dt}$,瞬间将线圈由 0 磁场移到磁场最大点,记录下整个过程中感应电动势 ε 的变化。变化过程结束后用软件对 ε 进行积分,计算出磁感应强度。本设计所需解决的是在瞬间记录下 ε 的变化曲线。

本文讨论一种采用数字逻辑电路设计的 DMA 控制电路。结合在“数字式磁通表”中的实际应用,给出高速 A/D 芯片 MAX153 与 89C51 单片机系统 DMA 接口电路的主要原理图和主程序流程。

一、系统构成

系统结构如图 8.1-1 所示。

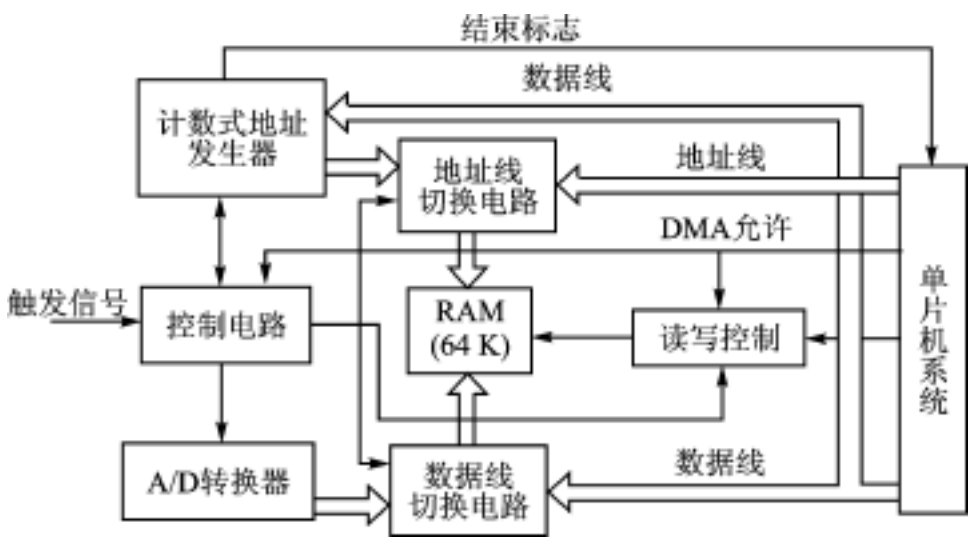


图 8.1-1 系统框图

设计采用 89C51 单片机芯片组成的单片机系统,外部 64KB 的 RAM 采用 62512 芯片,存

储器的全部 64K 地址范围允许作为外设的 A/ D 转换电路在 DMA 周期写入,以及单片机在非 DMA 周期读出写入。RAM 的数据线和地址线由总线切换电路来控制,选择连接单片机系统总线或连接 A/ D 转换电路的数据输出和地址发生器输出地址。总线切换电路不仅切换地址线和数据线,同时切换 RAM 的写控制线 WRI,控制实现 DMA 周期外设写入和非 DMA 周期的单片机写入。RAM 读控制线接单片机$\overline{\text{READ}}$。

DMA 周期时,RAM 的写入地址由地址发生器产生。它是一个可预置数的 16 位二进制计数器。其溢出信号作为 DMA 周期的结束控制和结束标志。

二、硬件组成

1 . 单片机芯片

采用 ATMEL 公司的 89C51,片内有 4 KB FLASH ROM。设计使用其最大时钟频率 20 MHz,可以有 1 μs 以下的指令周期。

2 . A/ D 转换器

采用 MAXIM 公司的 MAX153 芯片,转换精度 8 位并行输出,转换时间 660 ns、采集速率 1 Mbps。设计采用循环采集方式。

3 . 静态 RAM

静态 RAM62512 芯片为 64 KB,典型存取时间 200 ns。

4 . 总线(数据线、地址线和 WR)切换电路

总线切换采用 74HC245 三态总线收发器并联,分别选通。如图 8 .1 - 2 所示。

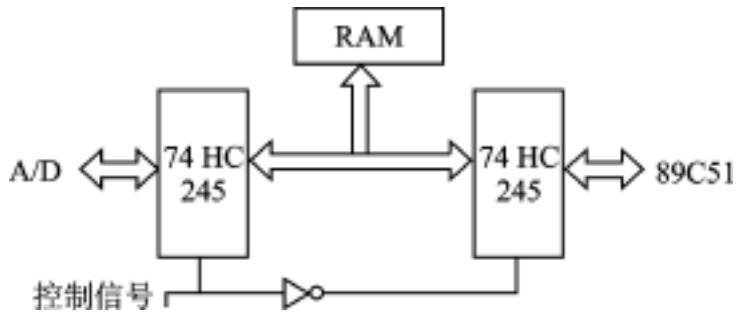


图 8 .1 - 2　总线切换电路

5 . 写地址产生电路

DMA 写地址产生电路是一可预置数的 16 位 2 进制计数器,电路是采用四片 4 位可预置计数器 74HC569 级连方式。因此可以预置 16 位地址的初值,就是 DMA 传输数据块的起始地址。计数脉冲反向输入 74HC569 的 CP 端以保证写操作的时序。

三、工作原理

1 . DMA 允许与响应

单片机系统开机运行时或进行内部数据处理、显示时应置 P1 .0 高,使其处于 DMA 禁止状态。单片机完成初始化或进入 DMA 准备就绪状态时,先向地址发生器写入数据块的起始地址,将地址溢出标志位置零,再发出 DMA 允许信号释放 RAM 的控制权,等待 DMA 结束标志和 DMA 控制信号。当 DMA 允许为 1 且地址计数器溢出标志为 0 时,进入 DMA 预备状态,等待外部触发信号输入。DMA 触发信号可以是周期信号的过零脉冲,也可以是单脉冲信

号放大整形输出,或者是 A/ D 转换完成信号。DMA 控制部分如图 8 .1 - 3 所示。本设计采用将感应电动势信号放大整形,其脉冲前沿触发。DMA 允许后的第一个触发脉冲到来,D 触发器输出 Q 为 1。开通四输入端与非门输出与振荡器同频同相的周期信号,开始数据采集和传送。由 NE555 和阻容元件组成的方波振荡器的振荡频率决定采样频率的大小,其频率应小于等于 A/ D 转换器的最大转换速率 1 Mbps。与非门输出信号同时作为 RAM 写信号、A/ D 芯片读$\overline{RD}$,经反相后作为地址计数器的计数输入。

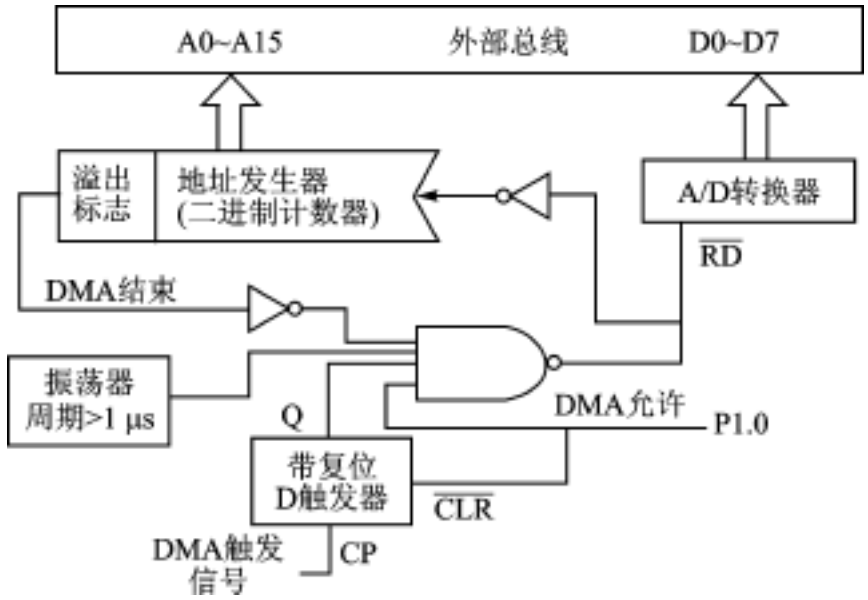


图 8 .1 - 3 DMA 控制部分

2 . 数据线与地址线的控制

总线的选择控制由 DMA 允许信号控制两组 74HC245 三态总线收发器,使其分别处于开通和高阻状态。此两组总线收发器一端并接至 RAM,另一端分别接单片机系统总线和 A/ D 转换外部总线。当 DMA 禁止周期时 DMA 允许信号为低,选通系统总线允许单片机对 RAM 进行读写操作。反之当 DMA 周期时 DMA 允许信号为高,选通外部总线允许 DMA 控制器对 RAM 写操作。数据线有 8 根(D0～D7),对数据线的切换需要两片 74HC245,而地址线有 16 根(A0～A15),切换地址线需要 4 片 74HC245 才可以。另外还需两片用以对 RAM 的读写线的切换,对读写线的控制采用相同的方法,也是由 DMA 允许进行两周期的控制权切换。

3 .DMA 块数据传输

与非门输出 1 MHz 的振荡信号,连接 A/ D 芯片$\overline{RD}$脚,在低电平开始时已采集的数据被输出到数据线。地址计数器被设计为下降沿触发,因此下降沿开始后地址计数器将在预设的起始地址的基础上加 1,形成新的地址输出至地址线。数据和地址的形成均在下降沿后的 160 ns(由 74HC 系列计数器性能参数可知其最大传输延迟时间为 40 ns,有四片级连总计 160 ns。由 MAX153CPP 的手册可知其读写模式下 Data-AccessTime 为 160 ns)内完成,其小于振荡信号低电平停留时间 500 ns。在下一个振荡的下降沿到来之前数据地址保持不变,在此后的上升沿时数据被写入 RAM 的指定地址,第二个下降到来后重复这样的过程,地址计数器加 1 形成新的地址和读出 A/ D 转换器中新的数据,再写入储存器。工作时序参见图 8 .1 - 4。

4 . 响应过程的结束

DMA 过程的结束设计在地址溢出时。设计利用地址计数器的溢出位,当地址溢出即大于(FFFF)H 时溢出位为 1,经反向器反向后至四输入端与非门的输入端,使其输出常为高而达到封锁的目的。直到单片机系统重新初始化地址计数器,清溢出标志,并重新 DMA 允许后

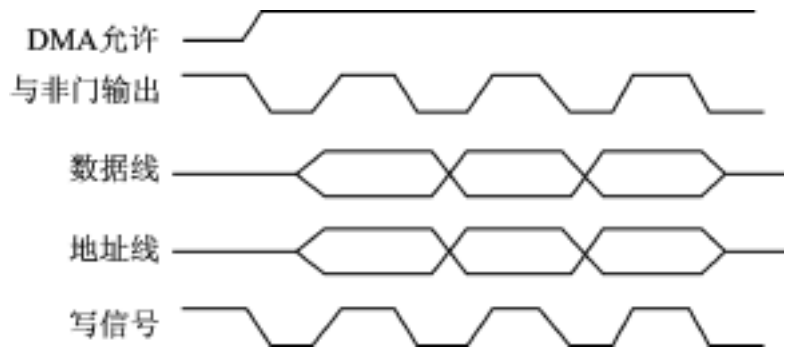


图 8.1-4　DMA 写工作时序

才能再次进入 DMA 准备就绪状态。

四、软件设计(主流程)

主程序流程图见图 8.1-5。

单片机系统以其方便、简洁、灵活、廉价为主要特点,所以在 DMA 电路设计中一定要结合实际应用简化设计,软件硬件综合设计避免系统过于复杂,才能达到优质廉价的最终目的。利用本设计研制的数字式磁通表综合性能达到了预期指标,并获得满意的性能价格比。

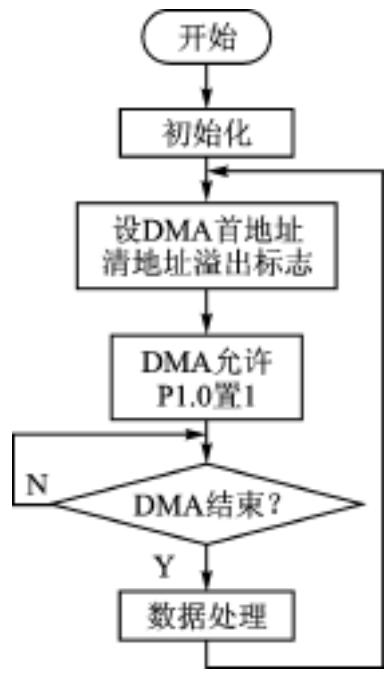


图 8.1-5　主程序流程图

参 考 文 献

1　AXIM 公司新产品数据手册(四). 1995

2　何立民. 单片机应用系统设计. 北京:北京航空航天大学出版社,1990

选自《电子技术应用》月刊,2000 年第 7 期

8.2 GPS数据采集卡的设计

成都中国科学院光电技术研究所(610209) 谢小平 姜凌涛

GPS具有全球、全天候、连续实时的精密三维导航与定位能力,有着广泛的应用价值和发展潜力。GPS数据接收和存储目前主要采用计算机的串口和数据采集卡,用中断方式进行。本文介绍由单片机和双口RAM等外围电路组成采集卡,用通用的串行通信口与GPS接收机通信,将GPS的数据采集存储于双口RAM中,当采集完一帧数据后,通过中断方式使计算机读取双口RAM的数据。

一、GPS数据输出格式

本系统所用GPS接收机为NOVTAEL的GPSCARD™,它是标准的MODEM异步串行传送方式,并采用D型RS-232C电缆接口。在使用异步串行传送时,不要求有严格的时间控制和同步协议,但要求在电文中作一些规定,在数据传送前,需送一个"开始位",传送完一帧数据后,再送一个"停止位"。传送的一般格式为:

波特率:9 600 bps

数据位:8位

奇偶校验:无

开始位:1

停止位:1

NOVTAEL的GPSCARD™可接收30多种命令格式,并可按用户给定的格式输出数据。其用户给定记录格式为:

LOG PORT,DATATYPE,[TRIGGER,{PERIOD},{OFFSET}]

其中:

PORT:控制面板、COM1,COM2。

DATATYPE:ASCII或二进制数据记录有效值。如POSA以ASCII码输出。

TRIGGER:选择LOG测量频率,频率按PERIOD和OFFSET设定。

PERIOD:仅ONTIME时有用,时间间隔。

OFFSET:仅ONTIME时有用,以秒为单位,以一秒一个记录输出。

数据类型中,"A"表示以ASCII输出数据,"B"表示以二进制形式输出数据。当用户给出上述格式的命令后,GPS接收机输出用户给定类型的数据格式。NOVTAEL的GPSCARD™输出数据以"$"为开始位,以"[0D][0A]"即回车和换行符为结束符。其输出格式为:

$xxxx,data field…,data field…,data field…*XX[0D][0A]

其中以"$"为开始标识符,"xxxx"为记录类型,接着为记录的信息内容,每个信息长度依赖于数据类型格式。不同数据信息以",”分隔;"*"结束数据信息;"XX"为总校验和;"[0D][0A]"为每个记录结束标记。

本系统仅要求 GPS 输出 POSA/B,SPHA/B,VLHA/B,RGEA/B,MKPA/B 这五种命令格式。由于 ASCII 易于识别和应用。在本系统中采用 ASCII 码。例如用户要求 GPS 输出 POSA/B 的记录,以 ASCII 码形式输出同时以一秒间隔时间,则用户输出记录格式为:

LOG COM1,POSA,ONTIME,1

GPS 接收用户要求的格式后,按下述格式输出数据。其输出记录格式为:

$ POSA,data field…,data field…,data field… * XX[0D][0A]

二、GPS 数据采集卡的硬件组成框图

根据以上的 GPS 数据输出格式,GPS 数据采集卡由 CPU 监控系统、可编程异步通信接口、数据存储器和地址译码器等组成,其组成框图示于图 8.2-1。下面分别对各部分加以说明。

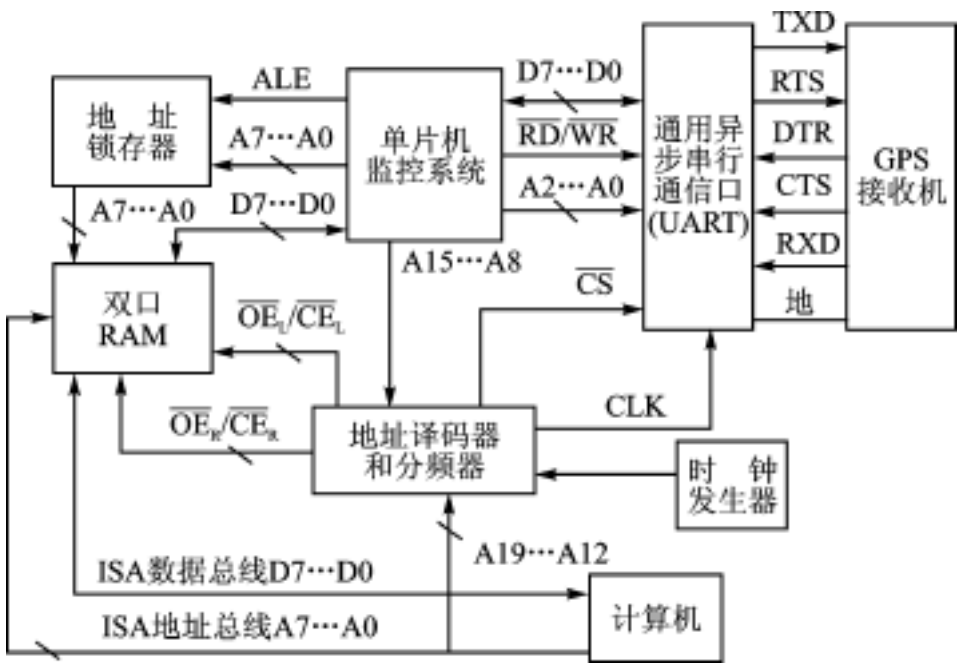


图 8.2-1 系统框图

1.监控系统

监控系统 CPU 由单片机 AT89C51 组成,AT89C51 本身内部带有 2 KB 的 ROM 程序存储器,256 B 的数据存储器,工作频率最高可达 24 MHz,当程序在 2 KB 以内时,不需要扩展外部程序存储器,可以减少外围电路体积,它是目前广泛采用的单片机。AT89C51 负责对可编程异步通信口写入通信规约,同时判断通信口转换数据是否为有效的开始符和结束符,当收到结束符后负责向主机发中断信号。

2.可编程异步通信接口

GPS 按标准串行异步通信格式输出数据。在此,采集卡采用可编程串行异步通信口作收发器。单片机有专门的异步收发器,可用定时器 T1 作波特率发生器,用 T_{XD} 和 R_{XD} 作发和收。但是单片机 AT89C51 在扩展外部数据存储器时,ALE 作为锁存地址低位字节,并以不变的频率周期输出。正是这个原因,对外部数据存储器某单元进行操作时,低位地址相同的单元内容将同时被改变,影响了数据操作的可靠性。为了克服这种现象,可在内部寄存器 SFR 8EH 地址上置"0"来禁止 ALE 的输出。此时 ALE 只有在执行 MOVX 指令时才起作用,保证了外部数据存储器数据操作的惟一性。由于 SFR 8EH 地址上置"0",即禁止了单片机波特率发生器。因此在此选用 TL16C550 可编程异步通信口。TC16C550 波特率的设置由内部参数完

成,具有标准可编程异步通信接口,接收和发射的各状态控制字完全独立,且具有 MODEM 控制功能,完全适应 GPS 的数据接收。TL16C550 与 TL16C450 兼容,在 FIFO 运行方式时,接收和发射都具有 16 B FIFO 存储器,这样可以大大减少中断数;TL16C550 可编程异步通信口初始化后即可与 GPS 进行通信。TL16C550 向 GPS 发送 LOG 格式,本系统向 GPS 输出如下的 LOG 格式:

```
LOG COM1,POSA,ONTIME,1;
LOG COM1,VLHA,ONTIME,1;
LOG COM1,RGEA,ONTIME,1;
LOG COM1,SPHA,ONMARK;
LOG COM1,MKPA,ONMARK;
```

TL16C550 发送上述 LOG 后,开始接收 GPS 数据。GPS 接收上述的 LOG 命令后,一秒内同时输出 POSA、VLHA 和 RGEA,并以一秒的间隔输出下一个记录。TL16C550 每接到一个有效数据后,中断请求单片机 AT89C51 读取 FIFO 的数据。AT89C51 判断是否为开始符,当开始符有效时,将串口转换数据存入双口 RAM 中,直到[0D][0A]结束当前记录的存储。

3 . 数据存储器

在 GPS 所输出的记录中,POSA、VLHA、SPHA 和 MKPA 中的每个记录最多为 84 个字节,每个卫星的 RGEA 字节长为 74 个字节,目前最多可收到 12 颗卫星,则 RGEA 最多为 1 000个字节。所以在选择数据存储器时,选用双口 RAMCY7C136 2K * 8 位存储器就能满足要求。双口 RAM CY7C136 具有两个独立的端口,各自均有一套相应的数据总线、地址总线和控制总线,允许两个端口独立、异步地对存储器中的任何存储单元进行存取操作。当两个端口同时对存储器中的同一单元进行存取操作时,可由其内部的仲裁逻辑决定优先权。双口 RAM 的左边数据线、地址线与单片机相接,双口 RAM 右边数据线、地址线与计算机 ISA 总线相连。因此,单片机和计算机可以同时对 RAM 进行读写操作。

4 . 地址发生器

地址发生器由可编程器 Lattice isp 1032E 组成,它产生整个采集卡所有的控制信号和片选信号,同时产生异步通信口的时钟信号。可编程器在数字电路中广泛应用,当它用于产生地址信号和时序控制信号时,对于调节地址范围和控制时序关系时,不需修改硬件电路,仅作软件处理即能完成。特别是在计算机接口电路中,数据存储器的控制信号必须是惟一时,地址控制相当严格,这时,用可编程器就显得比较优越。

三、GPS 数据采集软件设计

在上述硬件支持下,软件设计是比较容易的。按 GPS 数据规约初始化通信口,确定数据收发频率和数据传输位数,并使通信口以中断方式输出数据。异步通信口发出采集卡所要接收的 GPS 数据格式。异步通信口在接收时,每当转换完一个字节就产生中断信号,采集卡以中断处理方式读取 GPS 数据。软件设计流程图如图 8 .2 - 2 所示。

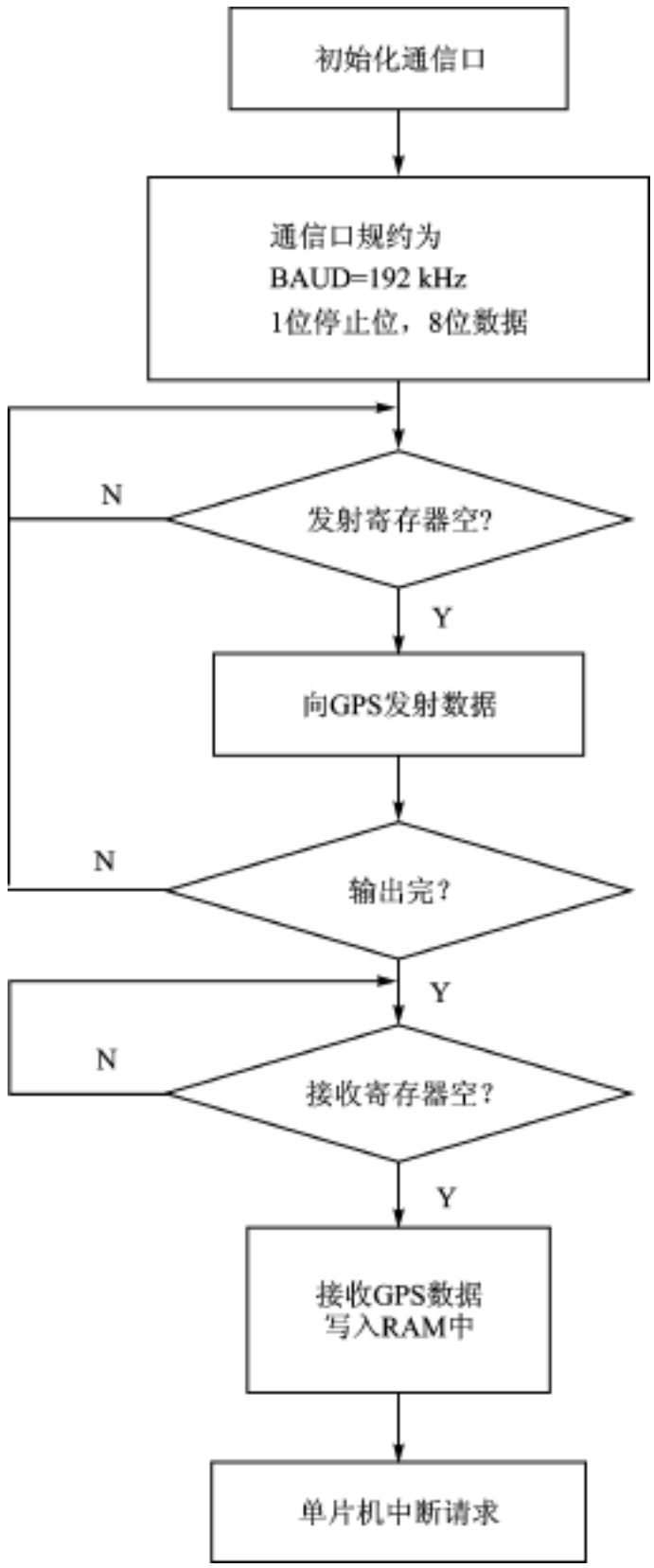


图 8.2-2 软件流程

四、结束语

用单片机 AT89C51 作监控系统,当程序在 2 KB 以内时,不需要扩展外部 ROM,可以减小电路体积;可编程逻辑器件产生各种时序信号和控制信号,易于适应不同时序控制要求。异步通信口采用 TL 16C550 可编程通信口,适应于不同的通信协议。因此,本接口电路不仅用于 GPS 数据接收,而且可用于所有标准的异步接口电路。

附:

GPS 卫星星座的最新状态

轨道面	第一轨道面					第二轨道面				第三轨道面			
	A1	A2	A3	A4	A5 *	B1	B2	B3	B4	C1	C2	C3	C4
SV 编号	14	25	38	27	19	22	30	13	35	36	33	31	37
卫星型号	2A	2A	2A	2A	2	2A	2A	2	2A	2A	2A	2A	2A
发射时间	1993 0626	1992 0223	1997 1106	1992 0909	1989 1019	1993 0203	1996 0912	1989 0610	1993 0830	1994 0310	1996 0328	1993 0330	1993 0513

轨道面	第四轨道面					第五轨道面					第六轨道面				
	D1	D2	D3	D4	D5 *	E1	E2	E3	E4	E5 *	F1	F2	F3	F4	F5 *
SV 编号	24	46	17	34	15	51	21	40	23	16	29	26	18	32	43
卫星型号	2A	2R	2	2A	2	2R	2	2A	2A	2	2A	2A	2	2A	2R
发射时间	1991 0704	1999 1007	1989 1211	1993 1026	1990 1001	2000 0511	1990 0802	1996 0716	1990 1126	1989 0818	1992 1218	1992 0707	1990 0124	1992 1122	1997 0723

SV = Space Vehicle

* 表示同一轨道面内除四个主要的卫星星位外附加星位。

参 考 文 献

1 张守信编著 . GPS 卫星测量定位理论与应用 . 长沙:国防科技大学出版社,1996

2 何立民主编 . 单片机应用技术选编 . 北京:北京航空航天大学出版社,1999

3 NOVATEL 用户手册 .1999

选自《遥测遥控》双月刊,2000 年第 4 期

8 .3　一种新型非接触式 IC 卡识别系统研究

北方交通大学　陈满才　周希德

一、射频 IC 卡识别系统概述

IC 卡分为有接触点 IC 卡(接触式 IC 卡)和无接触点 IC 卡(非接触式 IC 卡)两个大类。前者通过与卡的读写设备的触点连接,形成电的直接联系,进行数据的读写,达到信息传递与存储的目的;后者无须与卡的读写设备接触。二者之间不产生电的直接联系,而是通过一些非接触式如微波、红外、光及磁场等读写技术建立联系,从而进行数据读写,实现信息的传递与存储。由于非接触式 IC 卡与读卡器之间没有接触点,因而具有无磨损、使用寿命长、安全、可靠、美观等优点,而且非接触式卡可以同时操作多张卡,可以大大提高操作速度。本文讨论的 IC 卡是基于射频识别技术的非接触式卡。

一个典型的射频识别系统由应答器和阅读器两部分组成。阅读器包含一个无线收发机,应答器包含一个感应线圈及存储器。工作时,阅读器不断地发射无线激励信号及相应的询问信号,当应答器进入阅读器的工作区域(刷卡)时,应答器接收阅读器的激励与询问信号被激励发射应答信息,阅读器接收该应答信息进行相应的处理。

射频识别系统需要的外围部件少,功耗低,能在较低电压下工作,射频识别的媒介无线电波在介质边沿及表面具有反射、衍射和穿透介质的能力,使得射频识别系统具有不受灰尘、冰、雪的影响、衰减小等优点。

射频卡识别系统是 IC 卡技术与射频识别技术结合的产物。在这种识别系统中,采用射频识别技术,把应答器作为 IC 卡形式,形成非接触式的射频 IC 卡识别系统,简称射频卡识别系统。射频卡识别系统综合具有非接触式 IC 卡与射频技术的低功耗、高的抗干扰特性、可靠方便、使用方便快捷,廉价美观等优点。射频卡识别系统在门禁系统(汽车门禁、庭院门禁等系统)、遥控系统(报警系统、电视、录象机、遥控遥测等系统)、数据传输系统(公路铁路调度、窗库管理等系统)以及自动售票系统(地铁、公共汽车售票系统)等方面有着广泛的应用前景。

二、射频识别系统功能及组成

1 .射频识别系统基本模块组成

射频卡识别系统主要由应答和阅读器(基站)两个模块构成,如图 8 .3 -1 所示。该识别系统在 CPU 的控制下实现数据传输与存储功能。对于一个实际的射频卡识别系统,应答器模块在被动、非接触式条件下获取能量,并完成数据传递、密码等简单的逻辑校验和数据存储功能。阅读器模块以非接触的方式向应答器提供能量,并完成传输数据、接受 CPU 指令、读取并处理应答器数据以及向 CPU 输出。

2 .射频识别系统各模块功能

射频卡系统的工作过程主要包括电源获取、数据读和数据写三个部分。系统工作时,在阅

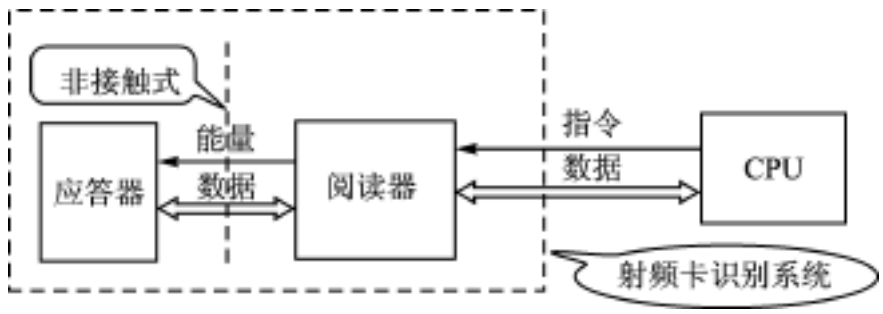


图 8.3-1 一个典型的射频卡识别系统

读器线圈上产生一个交变的磁场,应答器线圈在此磁场中感应出一个电压,该感应电压经整流稳压后为应答集成电路供电,不需要电池以及连接电缆。

“数据读”是实现数据从射频卡到阅读器的传递,这是一个信号解调、整形、增益及传送给CPU进行处理的过程。当要读取应答器的数据信息时,先由CPU向基站输出读信号命令,基站对命令进行调制通过基站与应答器线圈的相互感应把‘读命令’发送到应答器,应答器对命令进行判断后向基站发射数据信息。基站从应答器接收到的信号较弱,只能在阅读器线圈上产生微弱的电压调制,所以还要对该信号进行整形放大后,输出供CPU处理。

应答器以Manchester编码方式向基站发射信号,图8.3-2为Manchester的编码方式。

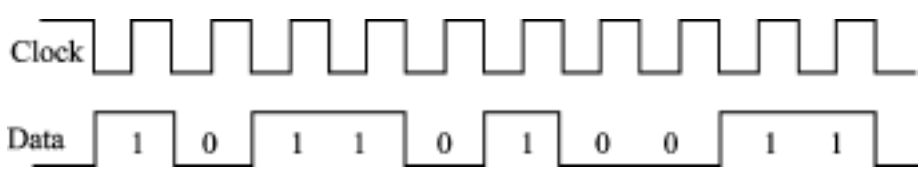


图 8.3-2 Manchester编码

经解调整形放大后的信号以Manchester码的方式输出供CPU处理,CPU必须先进行解码,才可以作为数字进行进一步的利用(如发往上位计算机或者存入指定的EEPROM存储单元)。图8.3-3是Manchester码的解码流程。“数据写”与“数据读”是一个相逆的信号调制的过程。CPU的数据先发送给阅读器,在阅读天线上进行信号的调制。当应答器工作时(指扫过阅读器天线的读区域),应答器线圈中的电流产生一个磁场叠加到阅读器的调制的磁场上。TEMIC公司的E5550卡采用改变发射场负载(即应答器按基站的数据信号来改变谐振电路的工作状态)的方法,在应答器上形成数据信号,实现‘数据写’的功能。

采用这种方法的数据编码如图8.3-4所示。

三、射频识别系统的实现策略

本系统采用TEMIC公司生产的E5550作为应答器IC卡。E5550是在125 Hz范围的用于非接触式读/写识别的IC卡,在E5550 IC的外部加上一个简单的线圈构成调谐回路,封装成卡片形式即可组成一个实际可用的应答器,如图8.3-5所示。

图中的调谐回路线圈用于向IC提供工作电源,并作为双向通信的收发接口。E5550卡内具有264位用于数据信息的EEPROM,这些存储区共分为8块,每块33位。由于射频卡能存储的数据量较小(最多264位),所以射频卡一般用于存放钥匙信息(诸如门禁密码等各类个人密码)以及预付金额(如电话卡、买电卡、信用卡等)用途。

阅读器(基站)负责向应答器供能,并由构成应答器(射频存储卡)与微机之间信息交换的通道。本系统采用U2270B作为阅读器的基本芯片,简单设计其相应外围电路,即可组成实用

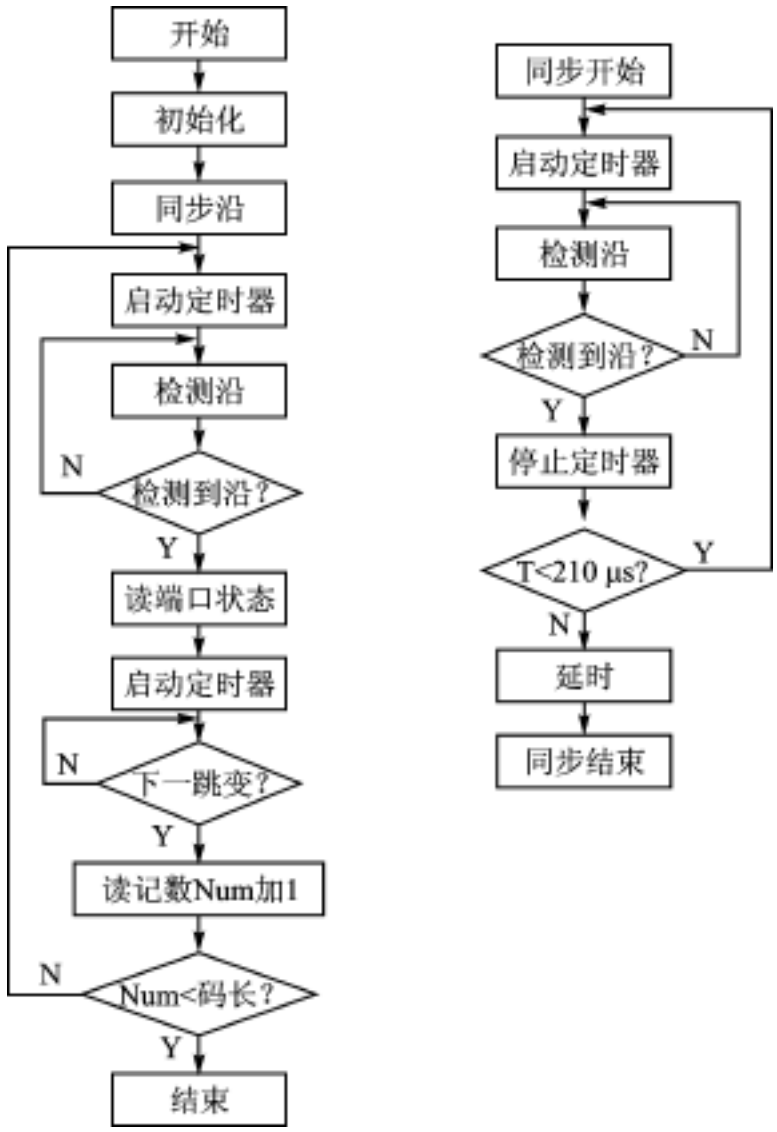


图 8 .3 - 3　Manchester 解码流程图

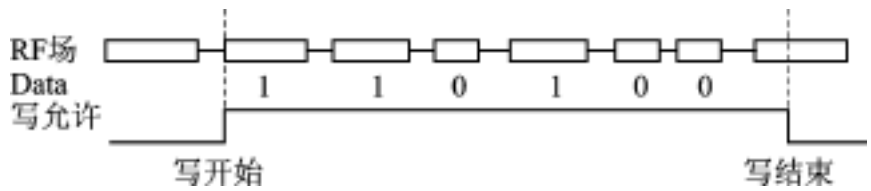


图 8 .3 - 4　数据写的编码方式

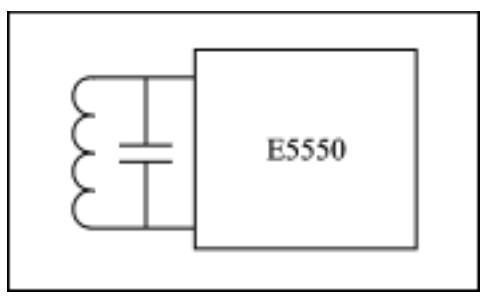


图 8 .3 - 5　应答器示意图

的阅读器,其电原理图如图 8 .3 - 6 所示。图中,有一个需要手工绕制的电感量约在 1 .71 mH 左右的天线,该天线用于向射频卡传送能量并发送、接收调制数据信息。

系统已通过调试与试验,并配置在国家九五攻关项目电动汽车地面应急充电机上,目前运行情况良好,工作可靠,性能稳定。

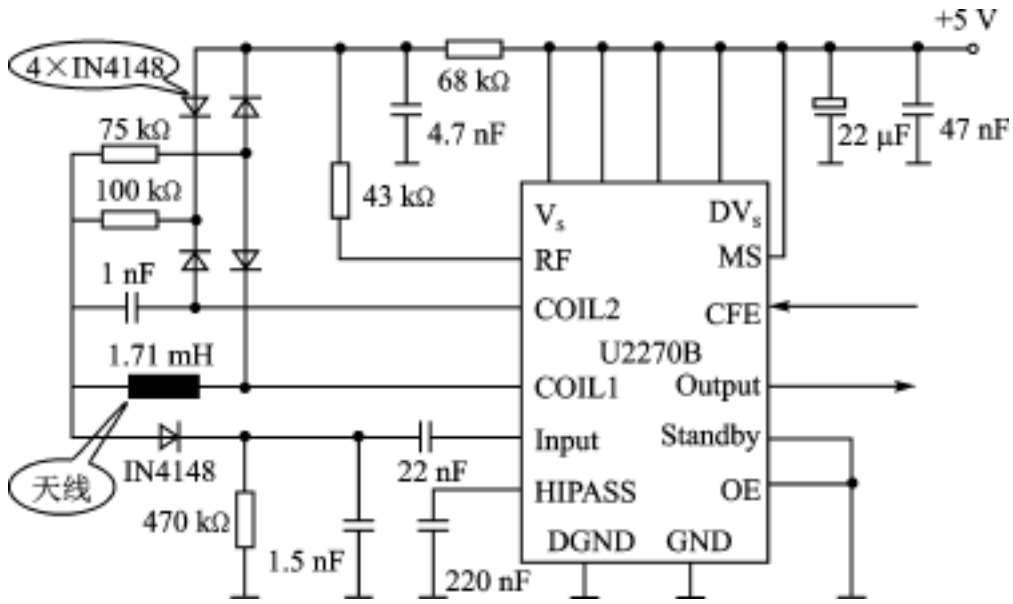


图 8.3-6 阅读器原理图

四、小 结

本文介绍了一种非常有前景的 IC 识别技术——射频卡识别。文中回顾了 IC 卡以及射频技术的发展历史和现状,并简单地介绍了射频 IC 卡识别的基本原理。在此基础上,制作了一个具体的射频卡识别系统,目前试运行情况良好。

参 考 文 献

1 白晓梅 . 非接触式控制系统集成电路原理及设计方案 . 北京:兵器工业出版社,1999
2 唐庆博 . 非接触式 IC 卡电度表的研制:[学士学位论文] .北方交通大学,2000
3 ATMEL 公司 . IC 智能卡技术手册 .1999
4 李华等 . MCS-51 系列单片机实用接口技术 . 北京:北京航空航天大学出版社,1993

选自《电子测量技术》月刊,2001 年第 2 期

8.4　自适应调整增益的单片机数据采集系统

哈尔滨工程大学自动化学院　田景文

一、引　言

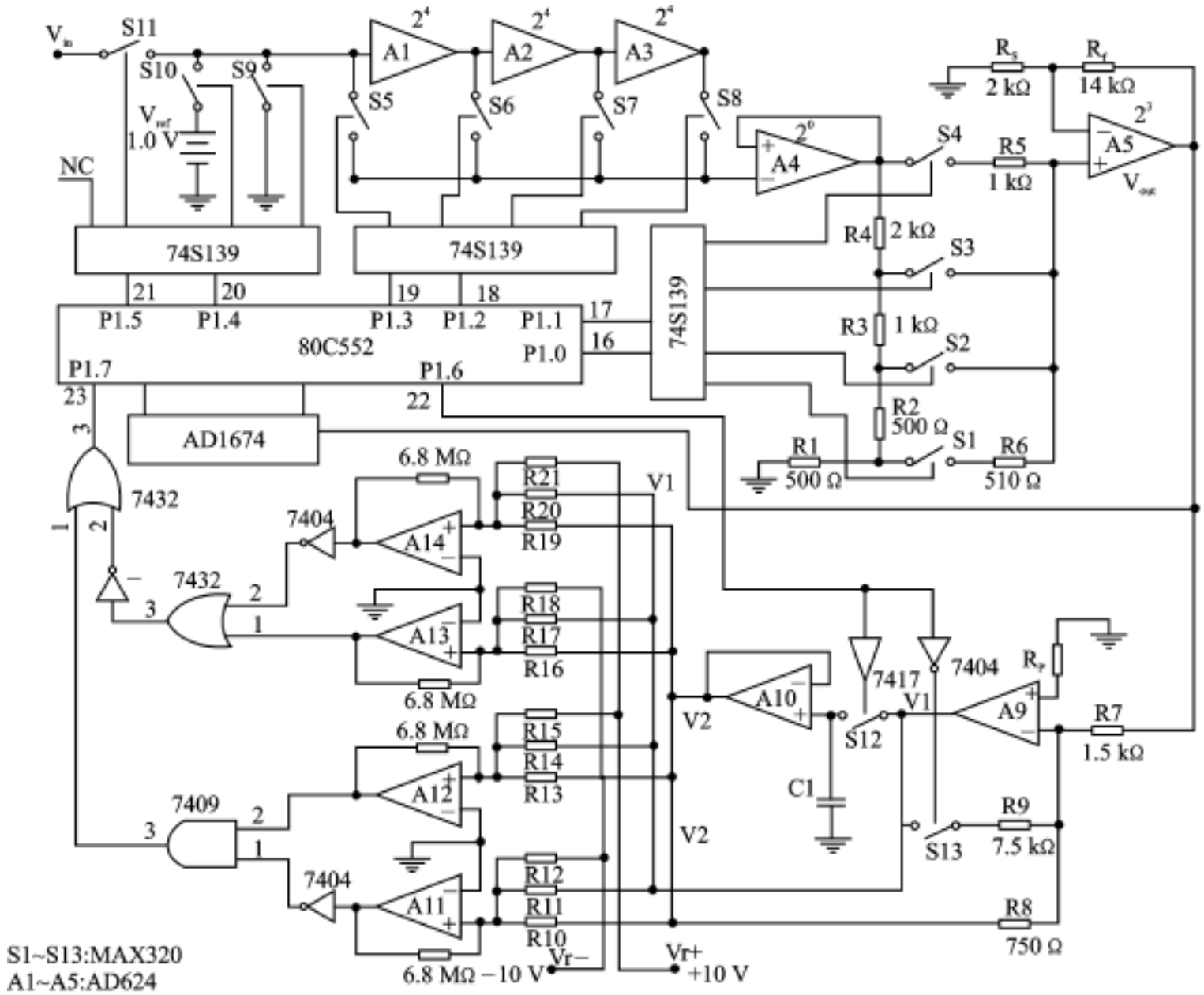
随着单片机技术的进步和发展,智能仪器仪表不断的发展和普及,智能仪器中的 A/D 转换器的输入窗口电压一般为 -5～5 V,但对于一些输入信号动态范围较大的情况,通常采用手动换档实现改变增益从而改变采集系统的输入范围。本文提出一种单片机控制的自适应调整增益的数据采集系统它可以根据每一个采集信号的大小,经过四次对信号的预测比较和调整增益,用最佳的增益放大信号,使放大后的信号落入设定的窗口范围之内。提高采集信号动态范围和 A/D 转换精度。同时此系统能自动消除零漂的影响,系统通过自动定标电路消除由于放大电路自身增益的不精确或增益的变化对信号的影响。

二、数据采集系统简介

图 8.4-1 是简化的放数据采集系统原理图。它由以下 4 个部分组成。

1.单片机控制的 16 台阶最佳增益放大电路

基本放大电路由两部分组成即主放大电路和细调放大电路如图 8.4-1 上部,主放大电路由同相放大器 A1,A2,A3(每一级的增益为 2^4)和集成模拟开关 S5,S6,S7,S8 组成。S5,S6,S7,S8 单独依次接通控制主放大电路的增益分别是 2^0,2^4,2^8,2^{12};A4 增益为 2^0,它作为隔离级,它在主放大电路和细调放大电路之间起隔离作用。细调放大部分由电阻衰减网络 R3～R8 和运算放大器 A5 及集成模拟开关 S1,S2,S3,S4 组成,A5 接成同相放大电路增益为 2^3,电阻衰减网络依次接成 2^{-3},2^{-2},2^{-1},2^0 如图 8.4-1 中所示,这样 S1,S2,S3,S4 单独依次接通,控制部分的放大电路增益依次为 2^0,2^1,2^2,2^3。主放大电路控制大范围的增益变化,而细调放大部分则控制小幅度的增益变化,放大电路工作时主放大电路部分的开关 S8～S5 只有一个接通,同时细调放大部分的开关 S4～S1 也只有一个接通,两组开关的不同组合,可以构成 2^0,2^1,2^2,…,2^{15} 共 16 个可选增益值。表 8.4-1 列出细调增益控制开关与 16 个数字增益值及单片机控制码和增益码的对应关系。



R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21
8.66 kΩ	4.75 kΩ	1.96 kΩ	8.66 kΩ	4.75 kΩ	1.96 kΩ	9.09 kΩ	4.99 kΩ	40.2 kΩ	9.09 kΩ	4.99 kΩ	40.2 kΩ

图 8.4-1 简化的单片机数据采集系统工作原理电路图

表 8.4-1 增益值、控制开关、控制码、增益码对应关系表

放大电路增益	开关状态(1 表示接通;空表示断开)								控制码 P1.3～P1.0	增益码 2 进制	增益码 16 进制
	S8	S7	S6	S5	S4	S3	S2	S1			
2^0				1				1	0000	0000	00H
2^1				1			1		0001	0001	01H
2^2				1		1			0010	0010	02H
2^3				1	1				0011	0011	03H
2^4			1					1	0100	0100	04H
2^5			1				1		0101	0101	05H
2^6			1			1			0110	0110	06H
2^7			1		1				0111	0111	07H
2^8		1						1	1000	1000	08H
2^9		1					1		1001	1001	09H

续表 8.4-1

放大电路增益	开关状态(1 表示接通;空表示断开)								控制码 P1.3～P1.0	增益码 2 进制	增益码 16 进制
	S8	S7	S6	S5	S4	S3	S2	S1			
2^{10}		1				1			1010	1010	0AH
2^{11}		1			1				1011	1011	0BH
2^{12}	1							1	1100	1100	0CH
2^{13}	1						1		1101	1101	0DH
2^{14}	1					1			1110	1110	0EH
2^{15}	1				1				1111	1111	0FH

2.单片机控制预测增益比较电路

比较电路由运算放大器 A9、A10 和集成比较器 A11～A10 和电阻 R10～R21、电子开关 S12、S13、电容 C 及数字逻辑门 7432、7409、7404 等组成,如图 8.4-1 下部所示。一般来说输入的信号是随时间变化的量,在极小的时间范围内可以用 $V_i = V + Kt$ 来表示,当比较电路的输入在 t_1 时刻的信号 $V_i(t_1) = V + K \times t_1$ 与设定的窗口范围比较,确定一个增益 G_1,逻辑控制部分要经过一系列的逻辑判断和逻辑控制过程(经历 ΔT 时间)才使放大电路的增益变为 G_1 进行放大,此时放大电路中的信号为 $V_i(t_2) = V + K(t_1 + \Delta T)$,$t_2 = t_1 + \Delta T$ 也就是用来确定增益的样值信号与当前被放大的信号有一个滞后的时间差 ΔT,这个时间差有可能造成放大器的输出超出设定的窗口范围,即:虽然 $V_{omin} \leq V + K \times t_1 \leq V_{omax}$ 但是 $V + K(t_1 + \Delta T)$ 却超出窗口范围。为防止上述情况的出现,在比较电路中加入预测电路,此电路可以在 t_1 时刻预测到 ΔT 以后的信号即在 t_1 时刻预测到 t_2 时刻的信号($t_2 = t_1 + \Delta T$),即 $V_i(t_2) = (V + K \times t_1) + K \times \Delta T$,利用这个预测的样值信号经过比较逻辑电路确定的增益值,再对真正 t_2 时刻($t_1 + \Delta T$)的信号进行放大时,就不会出现超出窗口范围的现象,保证放大器的输出一定落入设定的窗口范围内,即,$V_{omin} \leq |V_o| \leq V_{omax}$。预测电路原理简介如下。

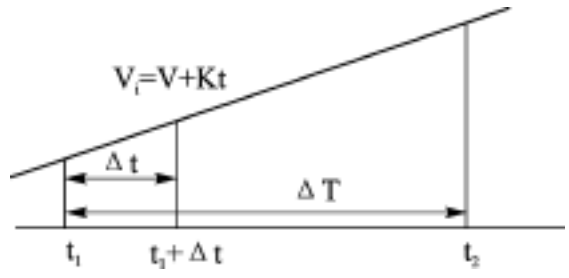


图 8.4-2　输入信号示意图

图 8.4-2 输入信号示意图,预测电路的理论推导简述如下:

预测增益比较电路分两步工作。

第一步:开关 S12 通 S13 断,则 A9、A10、R7、R8、C 构成如图 8.4-3(a)等效电路。

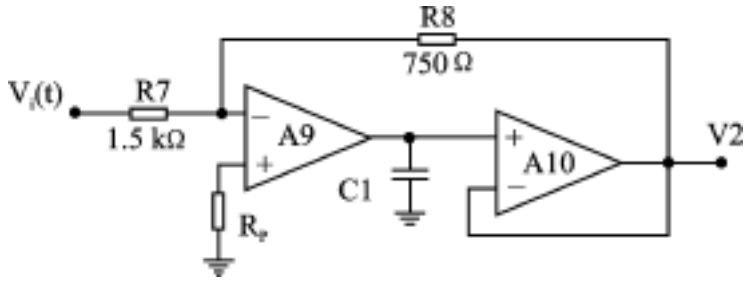


图 8.4-3(a)　第一步等效电路图

由图 8.4-2 可知,$V_i(t) = V + Kt$(以下把 $V_i(t)$、$V_o(t)$ 简写为 V_i、V_o)

$$V_2(t_1) = -\frac{R8}{R7}V_i(t_1) = -\frac{1}{2}V_i(t_1) \quad \text{即 } V_{c1} = -\frac{1}{2}V_i(t_1)$$

第二步:S12 断 S13 通,则 A9、A10、R7、R8、C 构成等效电路如图 8.4-3(b)所示。

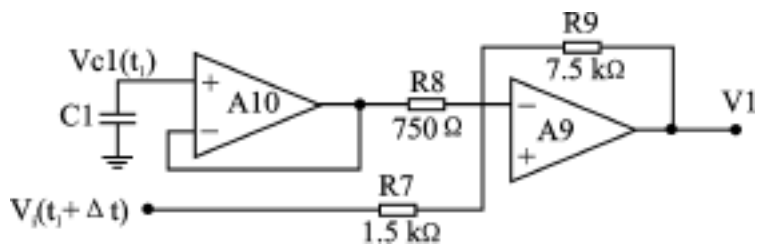


图 8.4-3(b) 第二步等效电路图

$$V_1 = -\frac{R8}{R7}V_i(t_1 + \Delta t) - \frac{R9}{R8}V_{c1}(t_1) = -5k\Delta t$$

取 $\Delta t = 2\ \mu s$,则

$$V_1 = -10, V_2 = -(1/2) \times V_i(t_1)$$

因为 V_1 代表信号的斜率特征,V_2 代表 t_1 时刻的信号幅值,我们可以利用在比较器 A11、A12、A13、A14 的输入电阻的权值关系把 V_1、V_2 在每个比较器 A11～A14 的入口处构造出相当于如下信号:

$$V(t_1 + \Delta T) = V(t_1) + \Delta T * K$$

取 $\Delta T = 22\ \mu s$ 这说明我们利用 t_1 时刻和 $t_1 + \Delta t(\Delta t = 2\ \mu s)$时刻的信号值可以预测 20 μs 以后的信号值。整个放大器根据预测信号的大小调整增益使放大器自动预测跟踪信号的大小完成对最佳增益放大。

设 V_o 的窗口范围为 $1\ V < |V_o| < 4.5\ V$,根据以上条件确定比较电路的电路参数如图中符表所示。

设 V_o 的窗口范围为 $1\ V < |V_o| < 4.5\ V$,根据以上条件,比较器的逻辑设计如下:

$$|(V + Kt_1) + \Delta t \times K]| > 4.5\ V, P = 0 \text{ 增益减小};$$

$$|(V + Kt_1) + \Delta t \times K] < 1.0\ V, P = 1 \text{ 增益增加};$$

为使放大器的输出 V_o 落入窗口的上部,比较逻辑设计为:

$$1.0V < |(V + Kt_1) + \Delta t \times K| < 4.5V, P = 1 \text{ 增益增加};$$

单片机程序中设定当第四次(最后一次)调整增益时如 P = 1,则增益保持不变,这样使输出的信号向窗口的上端靠拢又不至于超出上限。为实现以最短的时间搜索到某一个特定信号的最佳增益,控制逻辑采用对折搜索法。图 8.4-4 是比较电路的四次比较四次调整增益的流程示意图。需要注意的是最后一次增益调整时当 p = 1 时增益保持不变。

3. 输入信号选择部分

输入信号的选择部分由集成模拟开关 S9、S10、S11 和 2 线—4 线译码器 74S139 组成,它们在单片机的控制下完成对三种信号的选择,即输入信号(V_i),基准源信号(Vref),接地调零(GND)。在任何时刻 S9、S10、S11 只有一个导通。

4. 单片机及 AD 转换部分

单片机选用 PHILHS 公司的 80C552 型微控制器和 12 位 AD 转换器 AD1674,因为对于这样的高性能智能放大器配备 12 位以上的 AD 才能充分体现它的优越性,当然配 8 位或 10

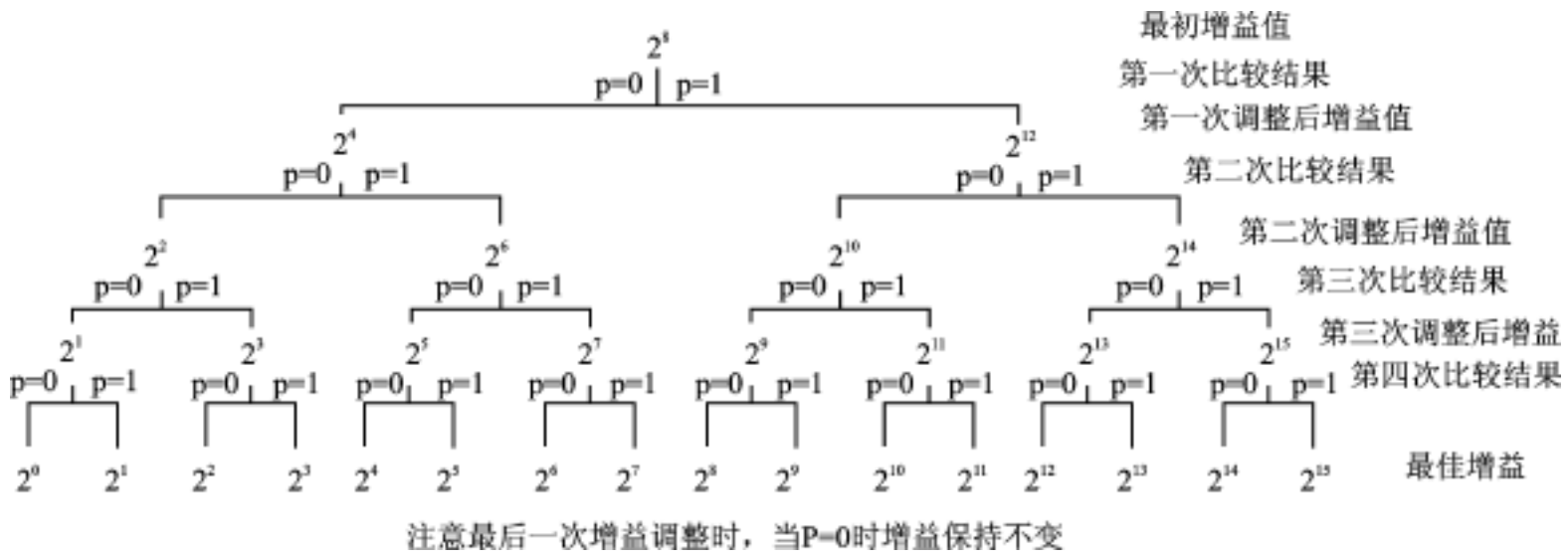


图 8.4-4 放大器四次增益调整过程示意图

位 AD 也可以。本文的 AD1674 的输入设定为 - 5～ + 5V。

三、单片机控制自动零漂校正和自动定标原理

对于精密的仪器来说设计放大电路时不但要考虑放大电路的零漂问题(即自动零漂校正问题),同时还要考虑放大电路的增益的变化和不精确带来的影响(即自动定标问题)。如果不用单片机而纯用逻辑电路来解决上述两个问题将是比较困难,即使能够实现也会使放大电路的电路过于复杂化,现在利用单片机的控制功能和运算功能,使解决零漂和增益影响的问题变得简单,实现起来也比较容易。对于本文介绍的放大电路其自动零漂校正和自动定标过程及原理简述如下。

设零漂为 V_{os},定标基准电压为 V_{ref}。对于某一特定信号的工作过程如下:(1)S10, S9 断开,放大电路,输入信号 V_i,放大电路经过四次增益比较调整,从 16 档增益中确定某一增益 G_b 作为对该信号的放大倍数,放大电路完成对信号的放大后,得到 $V_1 = (V_i + V_{os})G_1$;(2)单片机控制增益 G_1 不变,同时改变输入选择开关,使 S11、S9 断;S10 通,此时输入信号为定标基准电压 V_{ref},放大电路得到 $V_2 = (V_{ref} + V_{os})G_b$;(3)单片机控制增益 G_1 不变改变输入选择开关使 S11、S10 断,S9 通此时放大电路输入接地,放大电路得到输出 $V_3 = G_b V_{os}$。V_1、V_2, V_3 转换为数字量后,单片机对上述三次测量的结果作如下数学运算,可得

$$\text{结果值} = \frac{V_1 - V_3}{V_2 - V_3} = \frac{(V_i + V_{os})G_b - V_{os}G_b}{(V_{ref} + V_{os})G_b - V_{os}G_b} = \frac{V_i}{V_{ref}} \cdots$$

本文中 $V_{ref} = 1.0$ V,则计算出的结果值 = V_i,可见最后的结果完全消除了零漂 V_{os} 和增益 G 的影响,解决了零漂校正和增益变化对放大电路的影响问题。只要选择高精度的 AD 转换器和高精度的基准源(V_{ref}),就可以使数据采集系统达到很高的精度。

四、单片机控制放大电路工作过程

整个放大电路的工作过程是以单片机为核心控制完成的,完成一次对信号放大过程可用下面图 8.4-5 的流程框图来表示。

图 8.4-5 放大电路完成一次信号放大及 AD 转换工作流程图

五、结 论

本文介绍的数据采集系统,具有自动选择最佳零漂校正和自动定标的功能,使测量精度提高,增大采集信号的动态范围,不需要设置量程变换开关。充分利用 80C552 单片机的输入输出能力,使数据采集系统智能化。

参 考 文 献

1 王玉,等编著.集成检测电路原理与设计.北京:兵器出版社,1996
2 [美]R F 格拉夫.电子电路百科全书.北京:科学出版社,1992
3 黄晨武.最新集成电路应用大全.北京:北京希望电脑出版社,1991,10
4 李朝青.单片机原理及接口技术.北京:北京航空航天大学出版社,1994
5 张友德编.飞利浦 80C51 系列单片机原理与应用技术手册.北京:北京航空航天大学出版社,1992

选自《电气自动化》双月刊,2000 年第 6 期

8.5 利用光纤发射/接收器对实现远距离高速数据采集

长沙电力学院电力系(410077) 邓本再 谢玉梅 周力行

电力变压器局部放电在线定位监测系统的关键环节之一是准确、可靠的数据采集与传输。根据电力变压器局部放电产生的超声波和电脉冲信号频率高、频谱宽、幅度变化大,环境电磁干扰严重,以及电力变压器与中心控制室距离一般在 500 m 范围以内,而多路检测信号集中在变压器周围等特点,电力变压器局部放电在线定位监测系统的信号检测采用现场集中数据采集方案,数据传输直接利用一种低价、高速、中远距离多模光纤 HFBR-1414/2416 发射/接收器对实现远距离光纤高速同步传送。这样可大大简化控制器的电路结构,省去繁琐的数据传输编码和解码,数据传输速率也得以进一步提高。

一、HFBR 光器件简介

由惠普公司生产的光纤 HFBR-1414/2416 发射/接收器对具有如下主要技术特性:

- 发射光波长 820 nm;
- 最高数据传输速率 155 MBd;
- 最长传输距离 4 km;
- 适用光纤:50/125 μm,62.5/125 μm,100/140 μm,200 μm HCS;
- 工作温度范围: -40～+85 ℃;
- ST、SC、SMA 和 FC 四种连接头选择;
- 8 管脚 DIP 封装。

HFBR-1414/2416 光纤发射/接收器对的内部结构与管脚排列如图 8.5-1 所示。HFBR-1414 光纤发射器内含一个高效光功率激励的铝砷化镓光发射器,该光发射器在 60 mA 直流电流激励下能向光纤中馈入光波长为 820 nm 的光功率 -12 dBm。HFBR-2416 光接收器由一个高效 PIN 光电二极管和一低噪声跨阻前置放大器电路组成。该放大器由于跨阻抗作用使得放大器的带宽和非线性得到了较大的改善,同时也获得了较大的动态范围。光信号经光电二极管转换成模拟电信号,放大后由射极输出器缓冲输出,最大动态范围达 23 dB,频率响应从直流到 125 MHz。

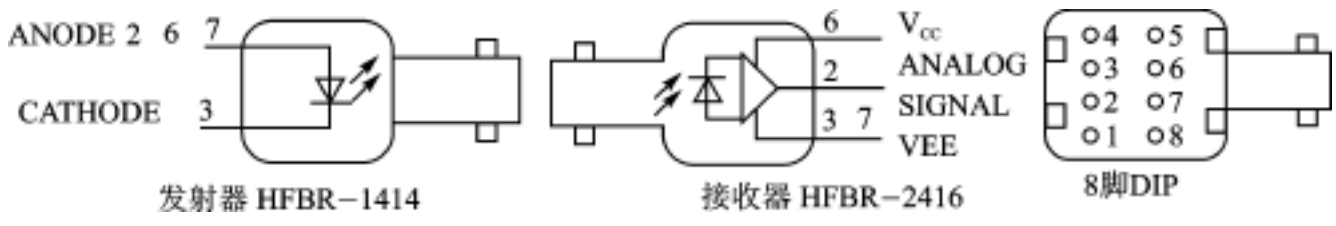


图 8.5-1 HFBR-1414/2416 发射/接收器对内部结构与管脚排列图

二、发射/ 接收器对的 TTL 接口电路

对于 HFBR - 1414 光发射器必须提供足够的正向驱动电流 I_F 才能发出所需光功率,由高性能的 CMOS 数字电路四二输入正与非门 74ACT00 设计的驱动电路如图 8 .5 - 2 所示。具有 TTL 电平的串行数据(或同步时钟)从与非门 U1D 的输入端 12 输入,反相后由 74ACT00 的三个与非门 U1A～U1C 并联输出驱动 HFBR - 1414 光发射器,产生足够的光功率。为避免驱动器的电流特性对光发射器光学开关特性的影响,运用频率补偿技术改善光信号的上升/下降沿,通过给 HFBR - 1414 光发射器提供一定的预偏置电压,来减小传播延迟误差所引起的脉宽失真。电路参数可由下列公式计算:

$$R1 = \frac{(V_{CC} - V_F) + 3.97(V_{CC} - V_F - 1.6)}{I_{FON}(A)}(\Omega)$$

$$R2 = \frac{1}{2} \cdot \frac{R1}{3.97}(\Omega)$$

$$Re = R2 - 1(\Omega)$$

$$R3 = R4 = R5 = 3Re(\Omega)$$

$$C1 = \frac{2\,000(ps)}{R2}(pF)$$

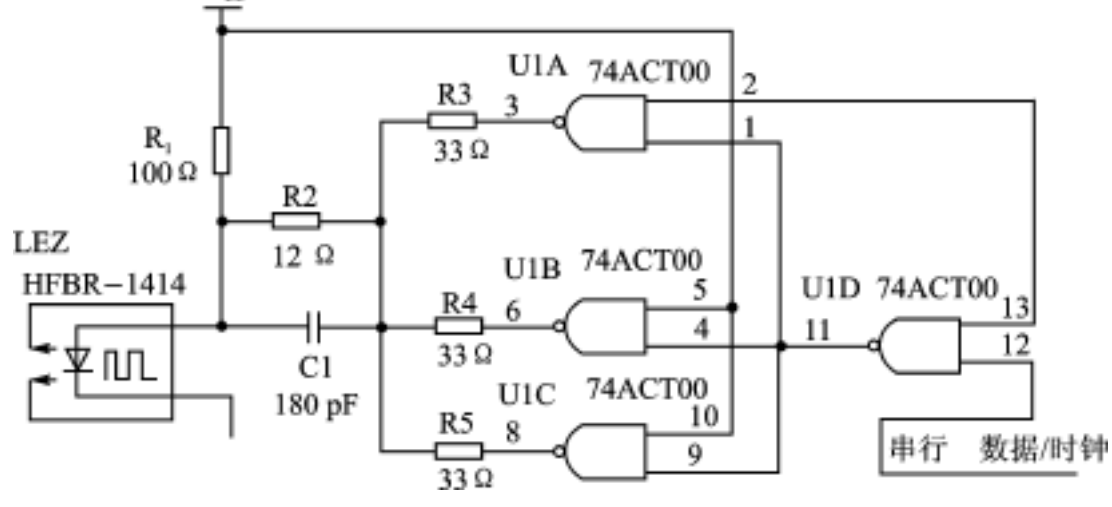


图 8 .5 - 2 HFBR - 1414 光发射器驱动电路原理图

HFBR - 2416 光接收器的输出为模拟电信号,为了与数字系统兼容,须通过图 8 .5 - 3 所示的高速比较电路进行变换。比较电路由超高速、低功耗、高精度 TTL 比较器 LT1016 集成芯片和一些外围元器件组成。HFBR - 2416 光接收器输出的模拟信号经 C1、C2 耦合到比较器的两模拟输入端,经 U1 比较后恢复出与发射端相一致的 TTL 信号,以相反极性从两输出端输出。电路中的 R8、R9 和 C4 组成低通滤波器,用以提高光接收器对电源和外部噪声的抑制能力。比较器的输入耦合电容由下式确定:

$$C1 = C2 = \frac{2}{3(R1 + R2)[\text{Data Rate(Bd)}]}$$

由光发射/ 接收器对与上述 TTL 接口电路所组成的通信通道,最大数据传输速率可达 32 MBd,最大传输距离 800 m。

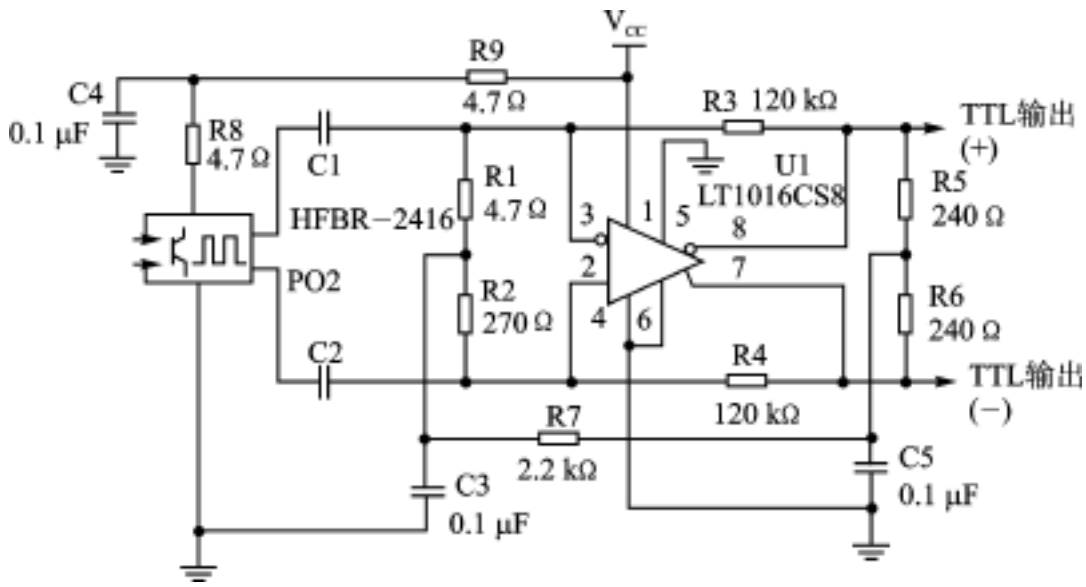


图 8.5-3 HFBR-2416 光接收 TTL 比较器电路原理图

三、远距离高速数据采集与传输系统设计

电力变压器局部放电在线定位监测系统的数据采集与传输系统结构如图 8.5-4 所示。整个系统由 7 路数据传输通道和 1 路同步时钟传输通道组成,分为现场数据采集与传送和主控数据接收与存储两大部分,彼此之间用 8 芯光缆连接;数据的采集、传输和存储统一由数据采集与传送控制器的同步时钟协调控制。

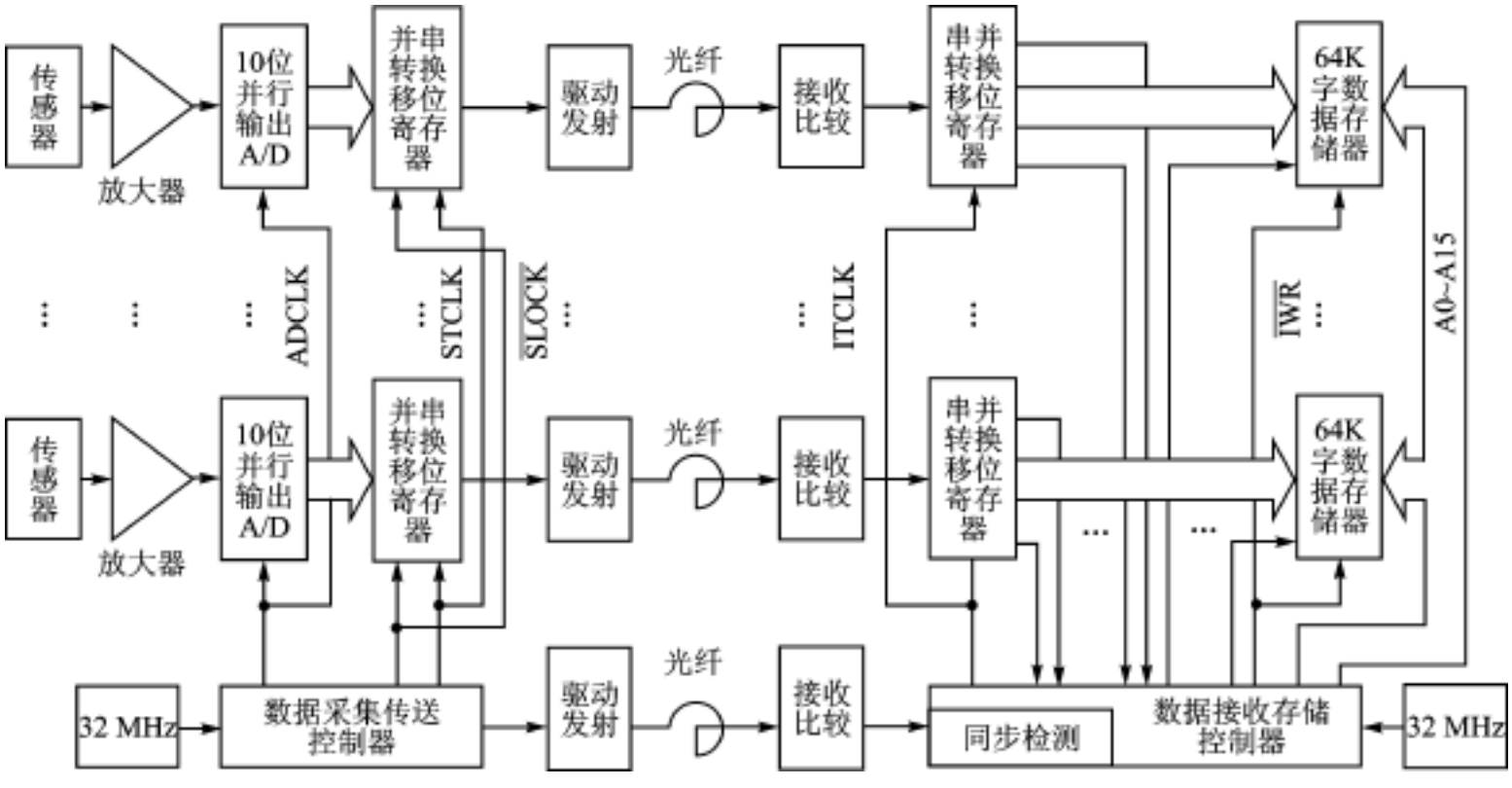


图 8.5-4 远距离高速数据采集与传输系统原理框图

1.数据采集与传送

数据采集由 TLC876 模数转换器完成,该芯片为 CMOS 低功率 10 位 20 Mbps 模数转换器,以多级流水线结构原理设计而成,内部串联五级 ADC 子模块。芯片在输入时钟控制下,转换过程分级进行,每级转换获得 2 位结果,逐级精化,直至最后一级达到 10 位分辨率输出。输出结果就一个采样值而言,仅需一个时钟周期,但从模拟值采样到 10 位数字量输出有 5 个

时钟周期的延迟。传感器检测到的变压器放电信号,经放大器放大到 - 5～+ 5 V 电压范围后,被送到 A/ D 转换器的模拟输入端。A/ D 转换器在 2 MHz 时钟 ADCLK 控制下连续不断地对其采样和输出。

TLC876 输出的 10 位并行数据由数据采集与发送控制器的锁存脉冲/ SLOCK 将其锁存于 2 片并串转换移位寄存器 74F165 中的 D1～D10 中。而寄存器的 D0 位则锁存固定电平“1”作为发送数据的起始位,D11～D15 位锁存固定电平“0”作为发送数据的停止位。被锁存的 16 位并行数据在传送控制器的 12 个移位脉冲 STCLK 作用下,被转换成数据队列 D0～D11 串行输出。移位时钟速率 32 MBd。

数据采集与传送控制器作为采集与传送控制核心,除负责产生 A/ D 转换时钟 ADCLK、数据锁存脉冲/ SLOCK 和同步移位脉冲 STCLK 外,还必须将同步移位时钟脉冲 SCLK 传送到接收端去,用于串行移位数据的串并转换同步移位时钟和同步信息的检测与恢复。时序关系如图 8 .5 - 5 波形的采集与传送时序部分所示。由图 8 .5 - 5 可见,一个采样数据传送周期需 16 个 OSC1 时钟周期,其中 12 个 OSC1 时钟周期用于传送 12 位数据,其他 4 个 OSC1 时钟周期作为同步信息发送时间,这段时间不发送时钟脉冲,接收端控制器将根据间隔时间的检测恢复同步信息。数据采集发送控制器采用一片可编程逻辑阵列芯片 GAL22V10 设计,通过编程实现内部 4 位二进制计数器对输入时钟计数,其 4 位输出结果与逻辑组合电路一起实现各项时序。控制器输入时钟频率 32 MHz,由有源晶振提供。

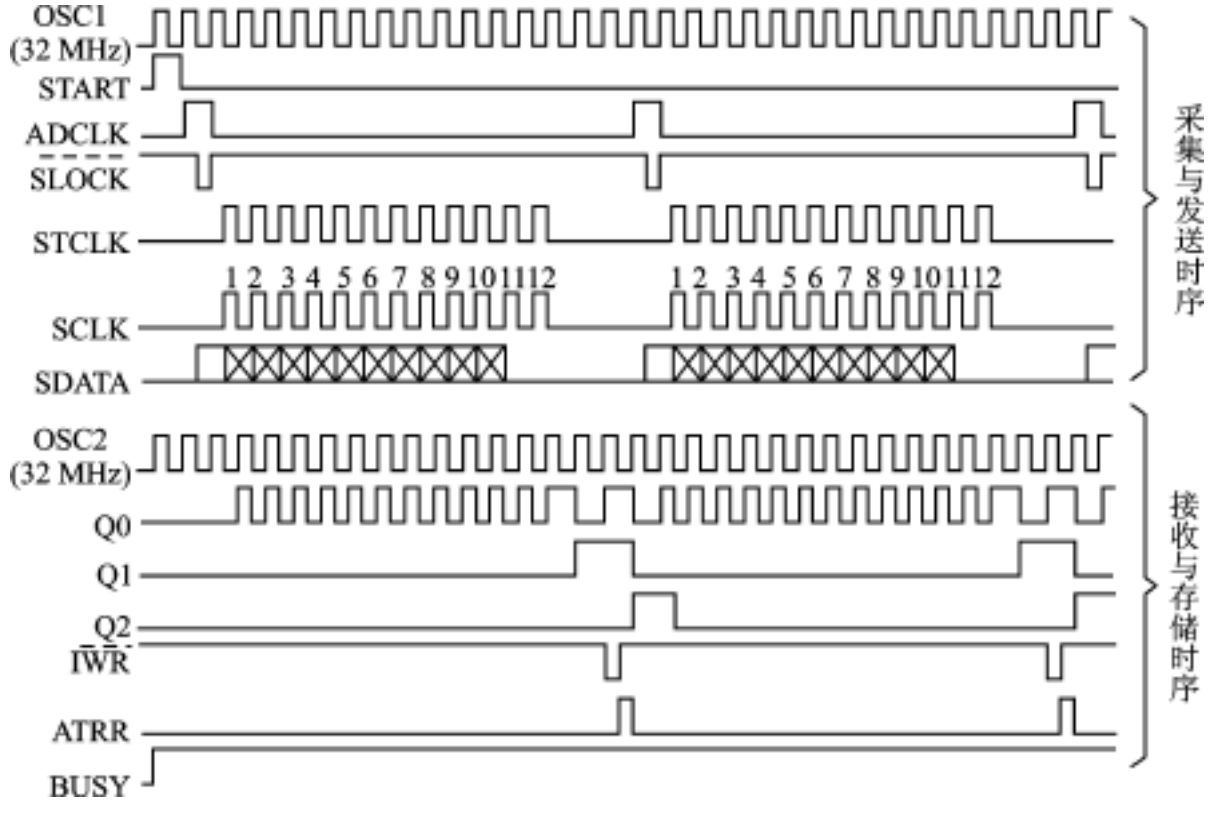


图 8 .5 - 5 数据采集发送与接收存储时序波形图

由并串转换移位寄存器输出的采样数据流,经光发射驱动电路功率放大后激励 HFBR - 1414 铝砷化镓发光器发光并馈入光纤。

2 . 数据接收与存储

接收端的光接收器 HFBR - 2416 将光纤传来的微弱光信号转变成模拟电信号,再经 TTL 比较电路恢复成与发送端相一致的 TTL 电平数据流,送入串并转换移位寄存器的串行数据输入口。

串并转换用2片串并转换移位寄存器74F164串联组成16位并行输出,串行输入的转换电路。从数据接收通道送来的数据流在同步移位时钟作用下,被转变成与发送端数据一致的并行输出数据。由于同步时钟是通过同步传输通道,由发送端与数据一起传输到接收端的,它们的传输条件一致、时序与时延一致,所以数据的移位同步性很好,经转换的并行数据可靠性也得到了充分保证。

串并转换输出的12位并行数据中的D1～D10共10位数据为A/D采样输出数据,被直接送到64 kW数据存储器RAM数据总线的D0～D9位。而起始位D0和终止位D11则被送到接收与存储控制器的内部数据校验器中进行同步校验,所得校验结果RAMD15 =/(D0./D11)作为数据正误标志("0"正确,"1"错误)送到存储器RAM的数据总线D15位。被送到存储器总线上的数据在接收与存储控制器产生的写脉冲/IWR作用下,存储在相应的存储单元中。

数据接收与存储控制器由CPLD复杂可编程逻辑器件EPM7128S的一部分(另一部分作控制接口设计)设计而成,通过编程实现如下电路功能:

- 同步移位脉冲计数器;
- 7路数据起始与终止位同步校验;
- 写脉冲/IWR形成;
- 存储器地址A0～A15发生器;
- 存储地址计数脉冲ATRR形成;
- 同步信息检测与同步控制。

接收器与发送器的数据同步由74F161计数器与CPLD的内部逻辑电路组成同步信息检测器,对同步传输光路传输来的移位时钟进行检测并实现同步控制。74F161计数器计数时钟由OSC2提供,频率与发送时钟频率相一致,由32 MHz有源晶振产生。计数器的清零输入来自经CPLD内部反相器反相的SCLK,同步时序见图8.5-5的接收与存储时序部分。在数据移位期间,每来一个移位时钟脉冲,计数器就清一次零。而在每个移位脉冲的周期内,计数器的计数时钟输入端有同频率的时钟脉冲OSC2输入,计数器可能对其计数。从理论分析,它有两种情况:第一种情况是计数脉冲的上升沿在复位脉冲的低电平时出现,计数器因处于复位状态而不会计数;第二种情况是计数脉冲的上升沿在复位脉冲的高电平时出现,这时计数器会加1。但不管怎样,在整个移位脉冲期间,计数器的计数值不会超过1。当12位数据发送结束,接下来是4个不发脉冲的同步信息时间,在这段时间内,计数器的计数值会累加到4,即$Q_D Q_C Q_B Q_A$ = 0100B。通过CPLD的内部逻辑组合,产生出写脉冲/IWR =/(/Q_D /Q_C .Q_B .Q_A .OSC2)和地址修改脉冲ATRR = (/Q_D /Q_C .Q_B .Q_A /OSC2),用于数据的存储和地址的修改。

上述远距离高速数据采集与传输系统在上电后,发送端A/D转换器就连续不断地对信号进行采集并传送。接收端是否存储传送来的数据,由启动控制信号START确定。当启动信号产生一个正脉冲后,忙信号BUSY由低电平跳变为高电平,开始一段存储时序周期。首先控制器检测同步信息,一旦检测到第一个同步信息,存储器地址发生器A0～A15立即清0,地址指针指向存储器的0单元,开始数据接收与存储。存储数据容量由预置计数器的初值决定。存储过程中,每存一个字数据,计数器就减一,直到计数器的值减为0,使忙信号BUSY由高电平跳变为低电平,结束这段存储周期。每通道存储数据容量最大64 KB。

由 HFBR - 1414/ 2416 光发射/ 接收器对设计的多路远距离高速数据采集系统,在电力变压器局部放电在线定位监测系统中应用,数据采集传输性能稳定可靠。该方法也非常适用于传输距离不太远、需大容量数据高速传输的一些实时测控系统。

参 考 文 献

1 光电子器件技术手册 . 惠普公司
2 TI 模数/ 数模转换器数据手册 . 武汉力源电子股份有限公司
3 秦文明 . 变压器局部放电的超声定位方法 . 变压器, 1995(3)
4 赵曙光 . 可编程逻辑器件原理、开发与应用 . 西安:西安电子科技大学出版社

选自《电子技术应用》月刊,2001 年第 12 期

8.6　一种频率编码键盘的设计与实现

北京理工大学电子工程系信息系统研究室(100081)
徐元军　陶　然　王卫江

单片机在消费电子、自动化仪表、工业控制等领域已得到广泛的应用,它以灵活的设计、低廉的成本、微小的功耗在电子器件市场中占有十分重要的地位。今天越来越多的芯片厂商在不遗余力地竞争这个应用空间,如 INTEL、NEC、MICROCHIP 等公司都已形成了自己强大的产品线,给产品的设计带来了越来越多的选择。

几乎在每一个单片机应用系统中,键盘都是必备的人机交互的主要输入设备。传统的按键识别方法是采用编码式键盘芯片,如 8279;或采用软件控制多条 I/O 线扫描的方法。这种方法用到的 I/O 引脚数常常在 4 条以上。然而单片机 I/O 引脚资源有限,特别是在引脚数少、功耗低、系统成本敏感的场合,成本和功耗决定了设计人员不可能另外扩充 I/O 空间,如采用 ATMEL 公司的 AT89C1051/ AT89C2051 及 AT90SXX 系列、MICROCHIP 公司的 PIC16CXX 系列的单片机时就是这样。如何利用有限的 I/O 资源实现多个按键的识别是经常遇到的问题。作者根据实际的开发经验,结合单片机自身的特点,提出一种利用单片机的定时器/ 计数器和测频原理、用一个 I/O 引脚实现多个按键识别的方法,并给出了 MCS-51 单片机的汇编源程序。由于在各种型号的单片机中,定时器/ 计数器几乎是一种必备的配置资源,因此其原理很容易移植到其他型号的单片机应用系统中。

一、硬件电路的分析

频率编码式键盘的硬件电路如图 8.6-1 所示。由 NE555 定时器组成的多谐振荡器产生

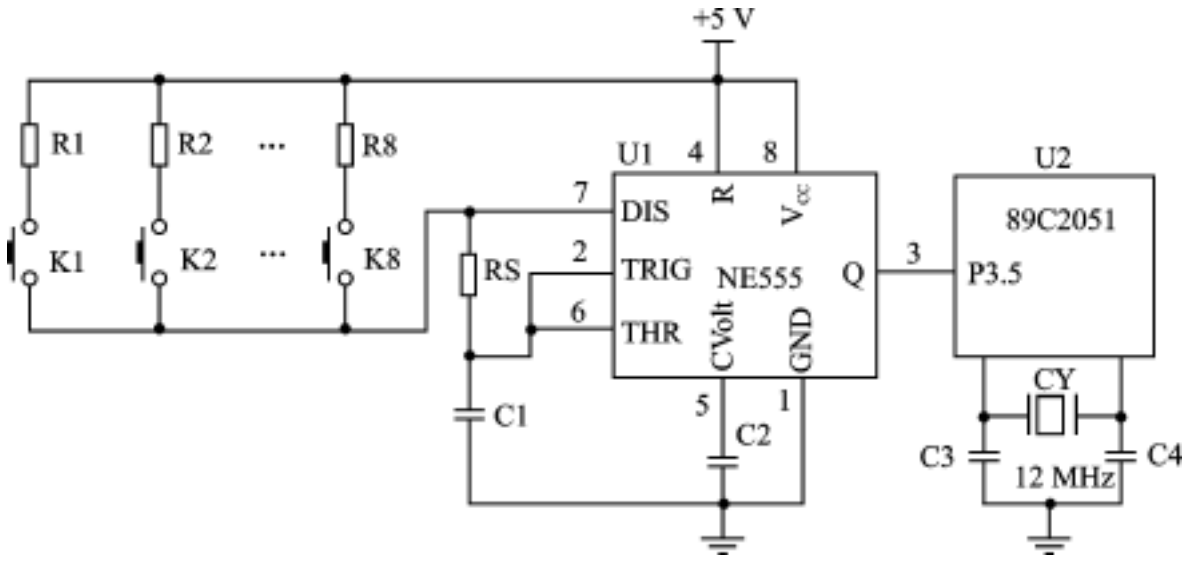


图 8.6-1　频率编码键盘原理图

一定频率的方波[1],单片机利用其定时器/计数器对这个方波的频率进行测量。NE555 定时器组成的多谐振荡器的放电时间常数为:$\tau_{放} = RS \cdot C1$,而充电时间常数为:$\tau_{充} = (R_i + RS)C1$,(i = 1,2,…,n),n 为按键的数目。

当不同的按键按下时,NE555 定时器组成的多谐振荡器的充电时间常数不同,放电时间常数不变,因而输出方波的频率也不同,使得不同的键按下时对应不同的输出频率。只要准确地测量出 NE555 定时器的输出频率,就可以精确地识别出被按下的按键号,这就是频率编码式键盘设计的理论依据。

在实际应用中,考虑到电阻阻值和电容容量的分散性以及电路的时间稳定性和温度稳定性,在满足单片机测量频率的分辨率和量程的基础上,应尽量将各个键之间的频率间隔拉大。这样即使每一按键输出的频率有一定的误差,但只要保证输出的各个频率互不相同,就可以通过软件去判断被测的频率究竟落在了哪一个范围,而不是判断落在了哪一个频点上。这样使设计的软件对电路产生的误差具有一定的适应性,从而摆脱了本电路对元器件参数的高度敏感性,大大加快了电路调试和批量生产的速度。

二、汇编程序的设计

在作者设计的系统中,采用 AT89C2051 单片机,外接晶振频率为 12 MHz。单片机仅具有 15 个 I/O 线,由于系统采用电池供电,需要进行低功耗设计。而采用本电路后,简化了系统的硬件,满足了低功耗的要求,另外的 14 个 I/O 线能满足一般的便携式应用。本系统用到 8 个按键,键盘电路中的 C1 = C2 = 0.01 μF, RS = 150 Ω,其余的阻容值和设计的中心频率如表 8.6-1 所列。其中,中心频率是指在电路参数误差为零时对应的频率。由于实际电路中误差总是存在的,所以频率就落在此中心频率附近。

表 8.6-1 键盘设计参数表

键号	R_i/Ω	中心频率/kHz	50 ms 内的按键频率计数值及判断区间		
			中心测量值	下限值	上限值
K1	2.1 k	60.112	301	250	350
K2	1.5 k	80.150	400	350	450
K3	1.1 k	103.05	505	450	550
K4	900	120.22	600	550	650
K5	730	140.07	700	650	750
K6	600	160.30	801	750	850
K7	500	180.34	901	850	950
K8	420	200.37	1002	950	1050

在软件设计上要实现以下三个功能:(1) 判断有无键按下;(2) 有键按下时,进行按键消抖;(3) 正确识别被按下的按键编号。首先设置单片机定时器/计数器的工作方式,让定时器/计数器 T0 设为外部计数方式,允许 T0 中断;并给 TL0、TH0 赋初值 FFH,一旦有键按下时,T0 便产生中断,由此可以判断是否有键按下;然后延时 8 ms 实现按键抖动的消除;接着将 T1 设为内部定时方式,定时时间为 50 ms,T0 在这 50 ms 的时间里对 NE555 输出频率信号进行

计数,通过对计数值的大小范围的判断就可以识别按键的编号。有关延时和测频的程序很常见,读者可以参考有关资料。按键识别这部分的汇编程序如下文[2]。程序入口参数 FRQH、FRQL 分别是 50 ms 定时时间内对外部频率计数的高位和低位,判断结果放在 KEYCODE 中。

```
        EYCODE    QU   30H        存放键值(1～8)有效;
                                   ;0FFH 无键按下,0EEH 出错
        FRQH      EQU  32H        ;频率测量值高位
        FRQL      EQU  31H        ;频率测量值低位
        ORG       0000H
        MOV       DPTR, # TABLE
        MOV       R0, #0
        MOV       KEYCODE, #0
NEXT:   MOV       A,R0
        MOVC      A,@A + DPTR
        INC       R0
        CJNE      A,FRQH,J1
        MOV       A,R0
        INC       R0
        MOVC      A,@A + DPTR
        CJNE      A,FRQL,J2
        MOV       KEYCODE, # 0EEH ;频率在边界上出错
        LJMP      WAIT
J1:     JNC       WAIT
        INC       R0
NNT:    INC       KEYCODE
        LJMP      NEXT
J2:     JNC       WAIT
        JMP       NNT
WAIT:   MOV       A,KEYCODE
        ; …
        ; …       添加用户应用程序
        ;判断频率区间上下限的数据表:
TABLE:  DB  00H,0FAH,01H,5EH,01H,0C2H
        DB  02H,26H,02H,8AH,02H,0EEH
        DB  03H,52H,03H,0B6H,04H,1AH
        DB  04H,7EH,0FFH,0FFH
```

三、特点及注意事项

采用频率编码具有抗干扰力强、接口简单的优点,并且易于实现光电隔离。在键盘与主机分离的场合,还可以大大简化二者互连的电缆。另外,若将此信号去调制红外发射组件,也可以实现遥控键盘而无需额外的编码逻辑。但是它对多键的同时按下和单键的连击检测能力较差,在与实际应用结合的时候,应尽量避免这两种情况的出现并恰当地进行处理。同时NE555的上限工作频率是500 kHz,采用传统的MCS-51单片机测量外部频率时,最高可测的频率为晶振频率的24分频,设计中应考虑可用的频率范围。采用其他型号的单片机时,也应注意这一点。

参 考 文 献

1 陈永甫.555集成电路应用800例.北京:电子工业出版社,1992

2 涂时亮.单片微机MCS-51用户手册.上海:复旦大学出版社,1990

选自《电子技术应用》月刊,2001年第4期

8 .7　高准确度时钟程序算法

广州增城市电力局(511300)　黄沛芳

电子计时器通常以石英晶振为时钟源。时钟源的频率通常为几十 kHz 乃至几十 MHz,而常用时钟的最小计时单位一般在 0 .01～1 s。高频的时钟源脉冲通过分频器后产生基本定时脉冲。电子计时器的计时部分就是对基本定时脉冲进行累加,产生秒、分、时等时间信息乃至日、月、年等日期信息。

一、引起计时误差的因数

一个常规电子计时器的计时准确度,取决于晶振标称频率(f_s)与实际频率(f_o)的频率偏差和晶振频率的时漂、温漂等离散参数。普通晶振的实际频率与标称频率有较大的偏差,可达万分之五(59),折算到一天计时误差就是 43 .2 s。一般室内气温变化在每天 10 ℃左右,对应晶振频率温漂 $< 10^{-5}$,若以一段较长的时间取温漂的平均值则更小。因此电子计时器的误差主要取决于晶振实际频率与标称频率的偏差。

二、减少计时误差的方法

1 .纯硬件方法

对于纯硬件计时电路,因分频系统 N 固定不变,要提高计时准确度只能调整 f_o,使得它尽可能接近于 f_s。常规减少计时误差的方法是:微调元件 L、C、R 的参数,调节硬件频率,使得时钟源的频率误差减小。但此方法操作复杂,没有一定的电子技术知识和专用仪器很难校准,而且会降低晶振频率稳定度。

2 .纯软件方法

由微控制器控制的实时时钟,可以采用软件的方法消除晶振实际频率与标称频率间误差引起的计时误差。

令晶振标称频率为 f_s,而实际频率为 f_o,则 $f_s = k \times f_o$。若 $f_s = f_o$ 则 $k = 1$,否则 $k \neq 1$。

$$T_{NS} = N_s \times \frac{1}{f_s} = N_s \times \frac{1}{k} \times \frac{1}{f_o}$$

N_s 是在标称频率 f_s 下定时 T_{NS} 的分频系数。

由微控制器控制的实时时钟,可以用软件模拟,将$\frac{1}{k}$归入总的计时程序中,从而消除 f_s 和 f_o 间偏差引来的计时误差。对于专用硬件时钟电路如 DS1320、PCF8583 等,可以采用每小时或每 10 min 读出时间,然后乘上$\frac{1}{k}$再写回芯片的方法校正。对于采用可编程分频定时器,由软件模拟时钟功能的软件实时时钟,则有更好的提高计时准确度的方法。因为定时器的分频系数是可以动态改变的,如 89C52 内置的 16 位计数器,分频系数可以在 $1 \sim 2^{16}$ 内任意选取。

令 $N = \frac{1}{k}N_S$ 作为分频系数写入计数器,这样每个基本计时周期 $T_N = T_{NS}$,从而实现软件校正定时周期。

在 $\frac{1}{k} \times N_S$ 刚好为整数时,可以使得计时误差为 0。大部分的情况 $\frac{1}{k} \times N_S$ 并不是整数,若将四舍五入后的值作为 N_S,就会带来量化误差,最大可达 $\frac{1}{2N}$,这是一个不容忽视的问题。以 12 MHz 的 89C52 T_2 定时器定时 10 ms 为例,每天最大量化误差累加是:

$$24 \times 3\ 600 \div (2 \times 10\ 000) = 4.32\ s$$

若在片内 RAM 中定义 1 个字节尾数,令它的满码值为 $\frac{1}{N}$,则最大量化误差就从原来的 $\frac{1}{2N}$ 下降到 $\frac{1}{2 \times N \times 256}$。对应于上述的 10 ms 定时程序,其最大量化误差的累加值由原来的 4.32 秒/天减少到 0.016 875 秒/天,这是很大的改进。根据精度要求,可以在片内 RAM 中定义 2 个字节,令它的满码值为 $\frac{1}{N}$,这样最大量化误差就可降为 $\frac{1}{2 \times N \times 65\ 536}$。

减少量化误差的具体算法是:对于使用 89C52 的 T2 定时器,若标称为 12 MHz 的晶振实际长期平均振荡频率 $f_o = 12.000\ 6$ MHz,量化精度取 1 字节,取 $T_{NS} = 10$ ms,则分频系数为:

$$N_0 = f_o / 12 \times T_{NS} = 12.000\ 6 / 12 \times 10^6 \times 10^{-2} = 10\ 000.5$$

令 $N = INT(N_0) = 10\ 000$

$$\begin{cases} NT = INT[(N_0 - N) \times 256 + 0.5] = 128 \\ NT_i = NT_{i-1} + NT \\ N_i = N + (NT_i \text{ 进位}) \end{cases} \quad (1)$$

式中,N_i 为第 i 次定时值,可能是 10 000 或 10 001,这取决于 NT_i 的进位;NT_i 为第 i 次尾数暂存值。

每次定时中断服务程序均执行式(1),取得第 i 次的定时计数值,然后实时时钟增加 10 ms,完成时钟功能。值得注意的是,N_i 是实际的计数值,至于实际写入特定定时器的数值,则须根据具体定时器的递减、递加计时性质分别写入 N_i 或 N_i 的补码,同时定时器在溢出到新的定时值装入并开始新定时周期这段时间,将 TLOAD 考虑在内。例如 89C52 T2 工作于自动重装定时初值、递加定时方式时,实际写入定时器 T2 的捕获/自动重装载寄存器(Rcap2H, Rcap2L)的值是 N_i 的补码,即 65536 - N_i;而对于 89C52 T0 和 T1 定时器则实际写入的定时初值是:N_i 的补码 + TLOAD 对应的机器周期数。

三、测量晶振实际长期振荡频率

没有专用仪器,怎样测得晶振实际长期振荡频率?有一个很简单的方法。以标称频率下的定时计数值 N_S 作为实际计数值,在电台报时时将时间设置正确,然后让它运行一段较长的时间,再与电台的报时比较求出误差的秒数,即可算出实际频率。例如晶振标称频率是 12 MHz,时钟运行了 10 天,快了 432 s,则

$$f_o = \frac{10 \times 24 \times 3\ 600 + 432}{10 \times 24 \times 3\ 600} \times 12 = 1.000\ 5 \times 12 = 12.000\ 6\ \text{MHz}$$

若将上述算法编成程序,让用户直接输入 N 和 N_T 的值;或输入运行了多少天、时、分、秒,

快或慢了多少秒,让系统自动算出 N 和 N_T,将会为从根本上校准时钟带来极大方便。任何人都可以轻松地提高时钟准确度而无需专业知识和专用仪器。

现在微控制器已广泛应用于人们日常生活的各个方面,电子时钟也随着它融入到各种电器和设备中,如专门时钟功能的石英表和各种附带电子计时器的电器如手机、普通液晶显示电话、VCD 机、DVD 机、电视及高档音响、空调遥控器、电力系统微机自动化设备等。高档专用计时器如高档石英表,因为计时是它的主功能,须保证计时准确度而对计时时钟源准确度要求严格,每天误差在 1 s 以下。以上提到的其他电器,其时钟只是它的一个附带功能,出厂时一般不严格校正,甚至根本不做任何校正。所以误差通常在 1 s/天以上,有些达 10 s/天以上,每天都需校正,否则运行几天就会因误差太大而变得不可信,令用户不胜烦恼。

在电力系统中,无人值班变电站须安装无功自动控制设备。它根据一天中的不同时间段和电网无功情况自动投退电容器组,使得电网的功率因数尽可能接近于 1,以利于经济运行。但有些设备内部时钟每天误差 > 5 min。若将本文算法编入计时程序中,让用户自己校正定时参数,将大大提高各种附带时钟的计时准确度。

将基于软件提高实时时钟准确度的算法应用于普通石英晶振,利用 89C52 T2 定时器的软实时时钟,未作校正前每天快 11 s;进行软件计时校正后,每 10 天的计时误差 < 1 s。本文提出的基于软件提高时钟准确度的算法,具有极高的实用价值。

参 考 文 献

1 何立民 .单片机应用技术选编(1) .北京:北京航空航天大学出版社,1994

2 罗 娟 .计算机时间校准方法 .微计算机信息,1999(4)

选自《电子技术应用》月刊,2001 年第 8 期

8.8　旋转编码器的抗抖动计数电路

北京中国科学院空间中心与应用研究中心(100080)　陈敏捷　田国璋

旋转编码器应用于角度定位或测量时,通常有 A、B、Z 三相输出。旋转编码器的输出波形见图 8.8-1。A 相和 B 相输出占空比为 50%的方波。编码器每转一周,A 相和 B 相输出固定数目的脉冲(如 100 个脉冲)。当编码器正向旋转时,A 相比 B 相超前四分之一个周期;当编码器反向旋转时,B 相比 A 相超前四分之一个周期。A 相和 B 相输出方波的相位差为 90±45°。编码器每转一周,Z 相输出一个脉冲。由于编码器每转一周,A 相和 B 相输出固定数目的脉冲,则 A 相或 B 相每输出一个脉冲,表示编码器旋转了一个固定的角度。当 Z 相输出一个脉冲时,表示编码器旋转了一周。因此旋转编码器可以测量角位移及位移方向。

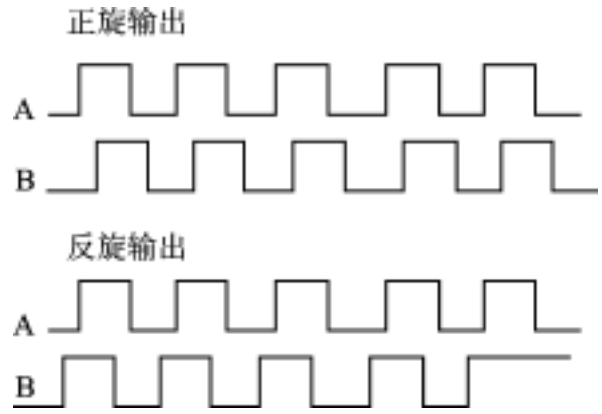


图 8.8-1　旋转编码器输出波形

问题出在伺服系统停止工作时,若无锁定,则旋转轴受外力(如风力影响)可能自由晃动,因而引起编码器输出波形抖动,如图 8.8-2 所示,从而引起误计数。在这种情况下,就不能对波形进行正确计数。虽然可以通过软件设置标志状态,用记录历史状态的变化来滤除误计数,但是程序耗费颇大。因此,本人设计了一个抗抖动计数电路,它能够自动消除抖动造成的误计数。

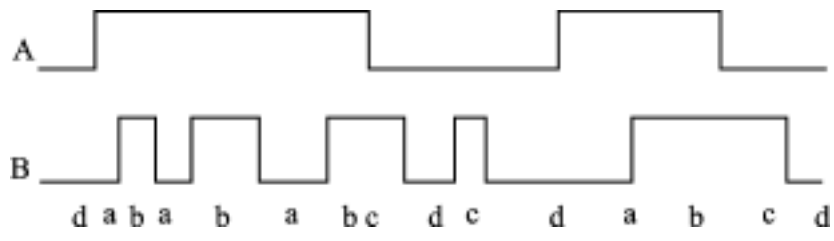


图 8.8-2　编码器抖动时输出波形

一、抗抖动计数电路原理图

图 8.8-3 是抗抖动计数电路原理图。此电路滤除了旋转编码器输出波形的抖动现象。该电路分为四个部分:译码电路 U4A;互锁电路 U5A、U5B;正旋计数链 J1、J3、J5 和反旋计数链 J2、J4、J6。U4A 为二四译码器,U5A、U5B 为与门,J1～J6 为 D 触发器。正旋计数链负责对编码器正向旋转的计数,反旋计数链负责对编码器反向旋转的计数。

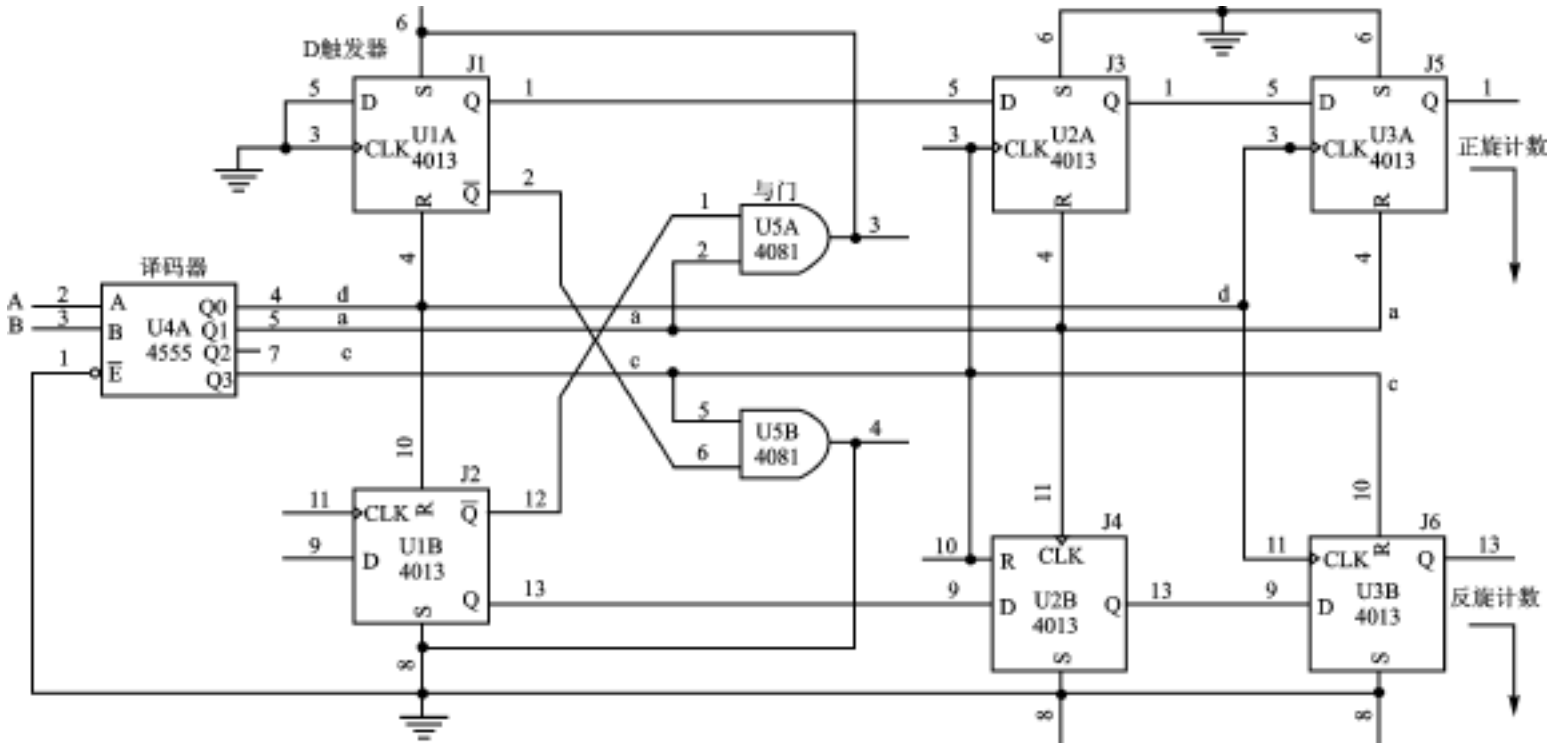


图 8.8-3　抗抖动计数电路原理图

二、抗抖动计数电路工作分析

图 8.8-4 为二四译码器输出的波形。译码器产生 d、a、b、c 四种不同的状态。在图 8.8-3 中当 B = 0、A = 0 时,译码器 Q0 输出为 d 状态,d 状态为高电平。当 B = 0、A = 1 时,译码器

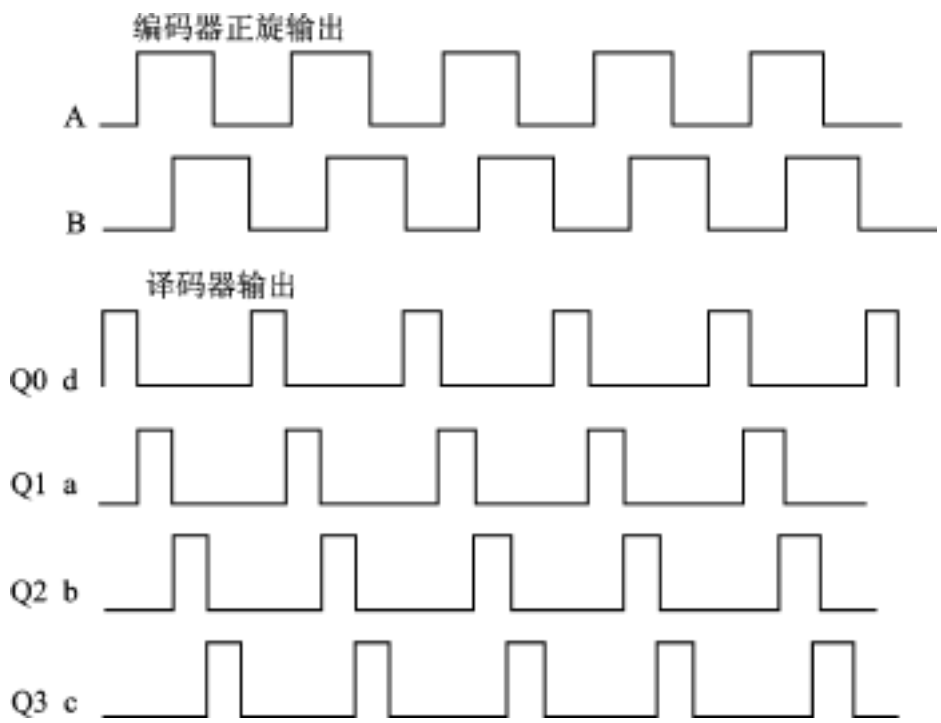


图 8.8-4　译码器输出波形

Q1 输出为 a 状态,a 状态为高电平。当 B = 1、A = 1 时,译码器 Q2 输出为 b 状态,b 状态为高电平。b 状态不影响计数和方向确定,在图 8.8-3 电路中没有使用。当 B = 1、A = 0 时,译码器 Q3 输出为 c 状态,c 状态为高电平。

当旋转编码器正向旋转时,译码器输出的状态顺序为 d、a、b、c、d、a、b、c……。如图 8.8-4 所示。当 B = 0、A = 0 时,进入 d 状态,与门 U5A 的 Pin2 = a = 0(Pin 是管脚的意思),于是 U5A 的输出 Pin3 = 0。D 触发器 J1 的 R = d = 1、S = 0,因此 J1 被清 0。与门 U5B 的 Pin5 = c = 0,于是 U5B 的输出 Pin4 = 0。D 触发器 J2 的 R = d = 1、S = 0,因此 J2 也被清 0。这时 J1、J2 的 $\overline{Q}$ 端都为 1,与门 U5 的 Pin1 = Pin6 = 1,U5A 和 U5B 都处于等待开门状态。当进入状态 a 时,Q1 = a = 1,U5A 的 Pin2 = a = 1。由于 c = 0,所以 J2 的 $\overline{Q}$ 端仍为 1,U5A 的 Pin1 = 1,U5A 的输出 Pin3 = 1。J1 的 R = d = 0、S = 1,因此 J1 被置 1。J1 的 Q = 1,$\overline{Q}$ = 0。J1 的 Q = 1,正旋标志送到了 J3 的 D 端。同时 J1 的 $\overline{Q}$ 端关闭了 U5B。在下一个 d 出现之前,所有的 c 脉冲都不会改变 J2 的状态。这就是说,J1、J3、J5 组成的正旋计数链被打开,J2、J4、J6 组成的反旋计数链被阻断。U5A、U5B、J1、J2 完成互锁的功能。在进入状态 a 时,J3 的 R = a = 1、S = 0,J3 被清 0,J5 的 R = a = 1、S = 0,J5 被清 0。在进入状态 c 前,J3 的 R = a = 0、S = 0、D = 1,J3 处于待触发状态。J3 的 CLK = c,当 c 脉冲上升沿过后,D = 1 被打入 J3 的 Q 端,正旋标志送到了 J5 的 D 端。在进入状态 d 前,J5 的 R = a = 0、S = 0,J5 处于待触发状态。J5 的 CLK = d,当 d 脉冲上升沿过后,D = 1 被打入 J5 的 Q 端,正旋标志送到了正旋计数输出端。正旋计数输出端由低电平变为高电平。到此为止,完成了一次正旋计数。当由状态 d 进入状态 a 时,J5 的 R = a = 1、S = 0,J5 被清 0,正旋计数输出端由高电平变为低电平。由此可知,当旋转编码器正向旋转时,对应 A 相和 B 相的每一个完整周期,正旋计数输出端都会产生相应的一个脉冲。

a 的出现抢到了正旋计数权。只有在 d 重新出现后,脉冲 c 才可能有机会抢到反旋计数权。从而保证了一旦进入正旋计数状态,不全完成这一过程,就进入不了反旋计数状态。运行时有可能从状态 a 返回状态 d,结果这只不过释放正旋计数权。因这时正旋标志还只在 J3 输入端上,a 出现时已把 J3 清 0。d 状态只会把 0 送给 J5 的 Q 端,因此不会产生误计数。

当旋转编码器反向旋转时,译码器输出的状态顺序为 d、c、b、a、d、c、b、a……。这种情况的分析方法与正向旋转的分析方法相同,这里不再叙述。

这就好比接力赛。在一个只允许上一个人的封闭的环形跑道上依次均匀设 d、a、b、c、四个站,立四个裁判员。d 点为起止点、出入口,持棒运动员沿环形跑道一站站往下跑。d 裁判长的职责是:运动员往 a 去(顺行),告示:“有人,正向”;往 c 去(逆行),告示:“有人,反相”。其他裁判员的职责是给到达本站的运动员发签证(计数标志),往回跑,撤销签证。d 裁判长的计分规则是:凭其他裁判的签证齐全加牌示,正,加一分,反,减一分,然后撤销签证。签证不齐到 d,则不计分,撤销签证。运动员在跑道内来回跑或坐时,d 裁判视而不见。

例如在图 8.8-2 中,当从状态 d 进入状态 a 时,正旋标志送到 J3 的 D 端。此后的 a、b、a、b、a、b 都不起作用,只是把 J3 和 J5 反复清 0。当从状态 b 进入状态 c 时,正旋标志送到 J5 的 D 端。当从状态 c 进入状态 d 时,正旋标志送到正旋计数输出端,同时 J1 和 J2 被清零。在从状态 d 进入状态 c 后,反旋标志送到 J4 的 D 端。同时,D = 0 被打入 J3 的 Q 端,这时 J5 的 D 端为零。在从状态 c 回到状态 d 后,反旋计数权被释放。但是,由于 J5 的 D 端为零,虽然这里再次出现状态 d,该 d 脉冲不会发生计数,这就是抗抖动。

图 8.8-5 为抗抖动计数电路的输出波形。此电路计数频率可达 10 MHz。A 相和 B 相

输入前应予整形,必要时还要进行光电隔离。

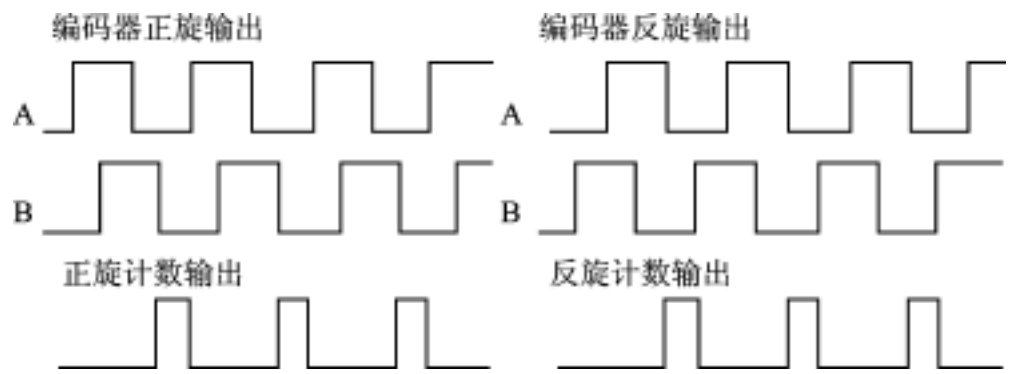


图 8.8-5 抗抖动计数电路输出波形

本人把此电路用于对天线云台角度的定位与测量。天线云台有两个旋转角度,俯仰角为 0～90°,方位角为 0～354°。旋转编码器用的是 OMRON 公司的 E6A2。此编码器每转一周,A 相或 B 相输出 100 个脉冲。由此可得,抗抖动计数电路每输出一个脉冲,编码器旋转的角度是 3.6°。如果加上变速齿轮将会提高角度测量精度。把抗抖动电路的正旋计数输出和反旋计数输出接到单片机的中断管脚上,其下降沿时给单片机发中断。单片机把处理的角度信息送到显示屏上显示出来,从而完成了对天线云台角度的定位与测量。这套天线角度显示系统在实际使用中运行一直很稳定,没有出现过误计数现象。

参 考 文 献

1 王毓银.脉冲与数字电路.北京:高等教育出版社,1992
2 Victor P Nelson.数字逻辑电路分析与设计.北京:清华大学出版社,1997
3 John M Yarbrough(美).数字逻辑应用与设计.李书浩译.北京:机械工业出版社,2000
4 陆 坤,奚大顺,李之权.电子设计技术[M].成都:电子科技大学出版社,1997

选自《电子技术应用》月刊,2001 年第 11 期

8.9 利用 X9241 实现高分辨率数控电位器

武汉理工大学(430079) 沈维聪
长江轮船总公司职工大学 刘义菊

美国 Xicor 公司生产的非易失性数控电位器是很具特色的一类产品。系列产品中有从 32 个抽头到 100 个抽头的、电阻值从线性分布到对数分布的、一个封装中有 1～4 个各种规格型号的电位器,可以满足各种分辨率和控制精度的要求,而且还不受机械振动的影响,可以取代几乎所有的模拟电路中的机械电位器。这就为用数控的方法精确地调整模拟电路中的电压、电流并实现遥控等创造了条件。用数控电位器代替机械电位器来调整模拟电路的参数,比用“A/D→数控→D/A”的方法要方便得多。

一、X9241 芯片简介

X9241 是 Xicor 公司 X9 系列芯片中的一种,它把四个非易失性数控电位器集成在一个单片的 CMOS 微电路中。它的功能方框图如图 8.9-1 所示。在 X9241 所包含的 4 个电阻阵列中,每个阵列包含有 63 个电阻单元。在每个单元之间和 2 个端点都有可以被滑动单元访问的抽头点。

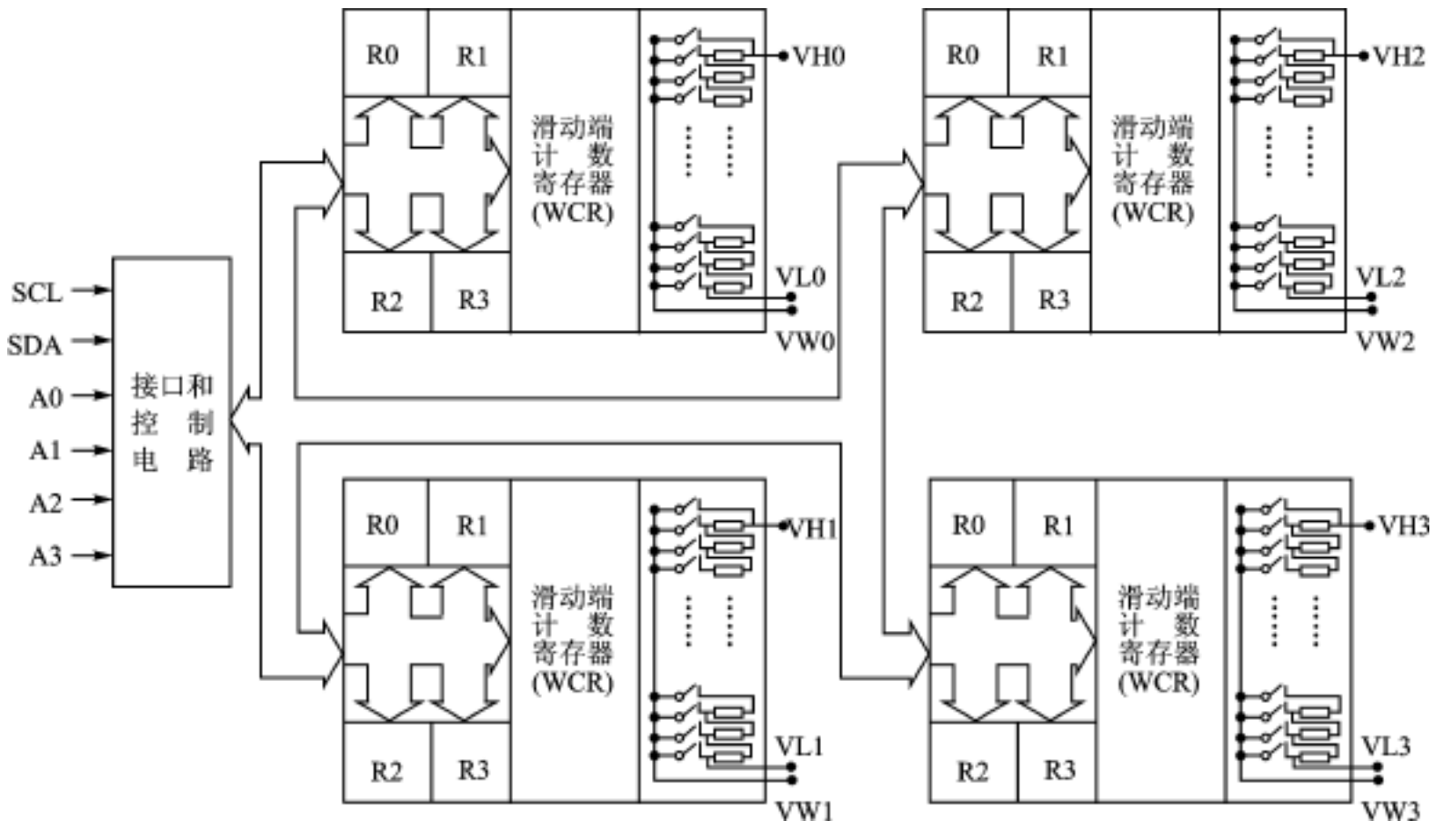


图 8.9-1 X9241 功能方框图

X9241 芯片引脚数为 20,有 DIP、SOIC 和 TSSOP 三种封装形式;二线串行接口;16 字节

的 E^2 PROM;电阻阵列值有 2 kΩ、10 kΩ、50 kΩ 三种;工作电压为 5 V;VH、VL 对地电压 ±5 V; 最高时钟频率 100 kHz。

二、器件工作原理

X9241 支持双向总线的 I^2C 数据规约。这个规约定义总线上的任何器件,当它把数据送至总线时为发送器,而当它从总线接收数据时为接收器。一个控制传输的器件是主机,而被控制的器件则为从机。由主机提供数据传送所需的时钟。X9241 在所有应用中都被考虑成一个从属器件,它通过 SCL(时钟线)和 SDA(数据线)与主机交换命令的数据。在 SDA 线上的数据只有在 SCL 为低的期间才能改变状态;当 SCL 为高时 SDA 状态的改变被保留。数据传送通常以 8 个二进制位为 1 个传送单位,包括地址、命令和电阻的抽头值。在进行数据传送前必须由 1 个开始条件作引导,这个开始条件就是当 SCL 为高时,SDA 出现由高至低的跳变。所有的通信也必须由 1 个终止条件来结束,这个条件就是当 SCL 为高时 SDA 由低至高的跳变。

在开始条件的后面,主器件必须输出它所要访问的从器件的地址。从器件的高 4 位地址是器件类型辨识符。对 X9241 来说,这个辨识符固定为 0101B。从器件的低 4 位地址是该器件的物理地址,物理地址由 A0～A3 决定。X9241 把输入的串行数据与地址输入端的状态进行比较,若相等则 X9241 就作出一个应答响应。应答是一个软件规约,这个规约用来在主、从器件的总线间提供一个正确的握手信号,以表示数据接收成功。发送器件(不管是主或从)在发送 8 位数码以后将释放 SDA 总线。主器件将产生第 9 个时钟周期,而在这个期间接收器把 SDA 线拉低,作为成功地接收了前 8 位数据的响应。在识别出开始条件和它的从地址以后,主器件将向 X9241 发送片内地址、操作命令以及抽头数据(见图 8.9-2)。其中,P1、P0 位用来选择电阻阵列,R1、R0 位用来选择电阻阵列中的 4 个寄存器,CM、DW 为抽头端子控制位,D5～D0 表示 64 个抽头位置,I3～I0 为指令编码。指令共有 9 条,以决定串口与 WCR、寄存器与 WCR、四通道同步寄存器与 WCR、串口与寄存器之间的数据传递以及 WCR 步进式增减等 9 个操作。

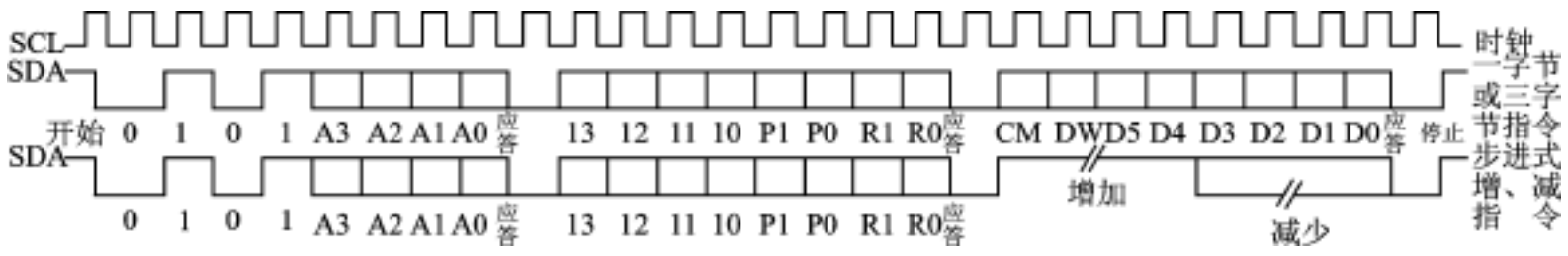


图 8.9-2　I^2C 总线数据交换时序图

三、使用方法

1. 滑动端计数寄存器(WCR)

X9241 包括 4 个滑动端计数寄存器(WCR),每个电位器包含有 1 个。WCB 可以被认为是一个 6 位并行和串行装载的、带有输出译码的计数器,用来选择电阻阵列的六十四选一的开关。WCR 的内容可以有四种方法来改变:它可以由主机通过串口直接写入(串行加载);可以将 4 个辅助数据寄存器之一的内容直接写入(并行装载);它也可以通过步进式增/减指令一步一步地修改;最后,它还可以在上电时装入它的数据寄存器 0(R0)的内容。WCR 是 1 个易失性存储器,即当 X9241 断电时它的内容将失去。

2 . 数据寄存器(R3～R0)

每个电位器有 4 个非易失性数据寄存器。这些寄存器可以被主机直接读出或写入,而且数据可以在 4 个数据寄存器的任 1 个和 WCR 之间传输。必须注意,在这些寄存器中的任何 1 个改变数据的操作都是非易失性的操作,将花去 10 ms 的时间(最大)。如果在应用中不需要对电位器有多个置数的储存时,这些寄存器可以被用作通用的存储器单元,它可以储存系统参数或用户的参考数据。

3 . 连续增加/ 减少工作方式

步进式增/ 减指令连续改变电阻值。一旦这个命令发出而 X9241 已经用一个应答来响应时,主机才能够以时钟来触发选定的滑动端升或降一个电阻段;这样,为主机提供了一个精细的调整能力。当 SDA 为高时每一个 SCL 时钟脉冲将使选定的滑动端向 VH 端移动一个电阻段;类似地,当 SDA 为低时每个 SCL 时钟脉冲将使选定的滑动端向 VL 端移动一个电阻段。

4 . 与 MCS－51 单片机的连接

X9241 与 MCS－51 单片机的连接非常简单,对不具有 I^2C 串行接口的单片机可将 SCL 和 SDA 接至单片机的任何一个 I/ O 脚(接 P0 口的 I/ O 脚时需加上拉电阻)。串行通信由程序按 X9241 的约定(见图 8 9－2)进行,并注意 SCL 的最高频率不得超过 100 kHz。

四、用 X9241 实现高分辨率的数控电位器

在有些应用中单独使用 X9 系列数控电位器有时会感到分辨率偏低。独立使用 X9241 各个电位器时,其分辨率仅为 1/ 63。若以该分辨率的电位器来控制 0～100 V 的幅值信号,则单步幅值为 100/ 63 = 1 .56 V。可采用以下两种方式来提高分辨率。

1 . 串联方式

X9241 提供一个把电阻阵列串联起来的机构。即,一个阵列的 63 个电阻元件可以与 1 个相邻阵列的电阻元件串联(链接)起来。其数据字节的高 2 位(CM、DW)就是为串联方式的使用而设置的。其中 CM 位是串联方式使能或禁止(正常工作)位。当 CM 位被置为“0”时,则电位器是正常工作方式;当被置为“1”时,则电位器与它相邻的高序号的电位器串联连接。例如,电位器 WCR2 的位 7 被置为“1”,则 POT2(电位器 2)将与 POT3(电位器 3)串联。DW 位是滑动端使能或禁止位。当 DW 位被置为“0”时,则滑动端被使能;当被置“1”时,滑动端被禁止。如果滑动端是禁止的,则该滑动端电气上是隔离的并且是浮空的。当工作于串联方式时,被串联的阵列的 VH、VL 及滑动端这 3 个输出端必须在电气上与外部连接,除了 1 个滑动端以外其余的滑动端必须被禁止。

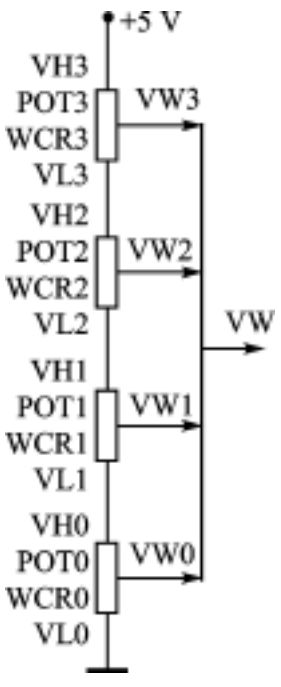


图 8 9－3 多电位器串联阵列

将电位器串联使用时,可获得 1/ 127(2 个串联)、1/ 190(3 个串联)和 1/ 253(4 个串联)的分辨率(见图 8 .9－3)。

2 . 组合方式

将 1 个或 2 个电位器(串联使用)作移动抽头电位器使用,另 2 个电位器则用来对该移动抽头电位器设置 VH 和 VL 的电平(见图 8 .9－4),并使 2 个电位器的滑动端始终保持为 1 位置相隔,那么将有 63 个不同的电压间隔施加到作

移动抽头的电位器上。此时的分辨率便可达到(1/63)/63 = 1/3 969(3 个组合)或(1/63)/127 = 1/8 001(4 个组合)。注意:图 8.9-4(b)中的 VW 按串联方式使用。

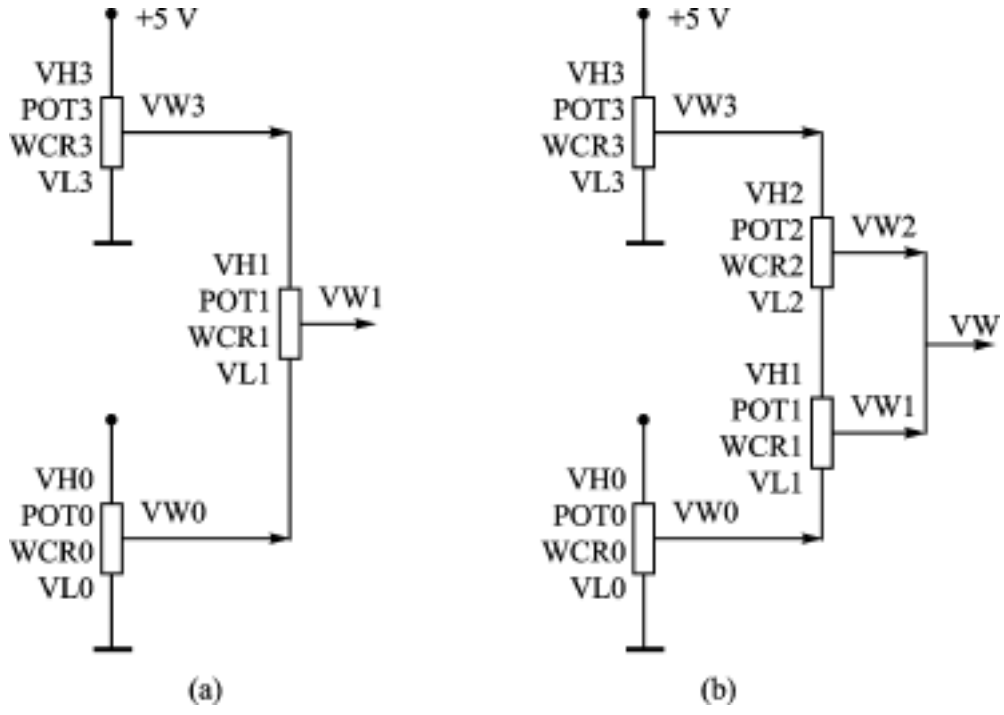


图 8.9-4　电位器组合连接

五、应用举例

1.用 51 系列单片机构成的组合型高分辨率电位器

主机采用 51 系列单片机,键盘等其他部分省略,仅给出 51 单片机与 X9241 的连线,SCL、SDA 分别接至 P1.0、P1.1,组合采用图 8.9-4(b)所示的形式,见图 8.9-5。

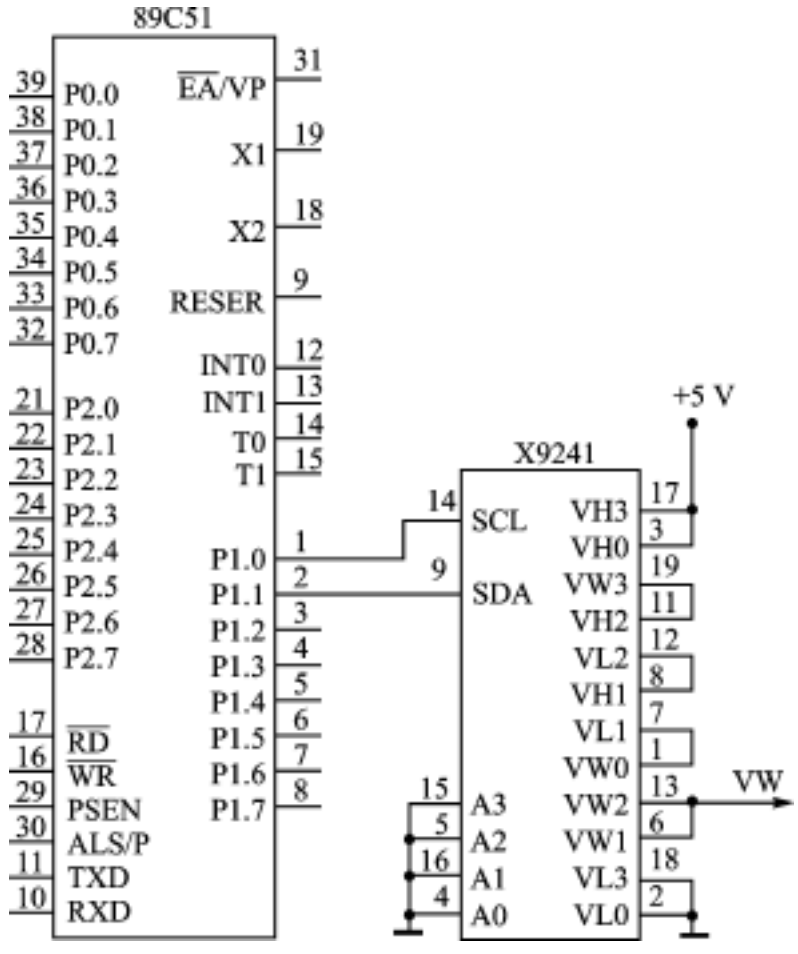


图 8.9-5　组合型电位器线路原理图

2 . 调节流程

按图 8 .9 - 5 的连线, 调节 POT0、POT3 就为粗调, 其单步分辨率为 1/ 63。调节 POT1、POT2 即为细调, 其单步分辨率为(1/ 63)/ 127 = 1/ 8 001。图 8 .9 - 6 给出的是步进式增/ 减指令连续改变电阻值调节流程。

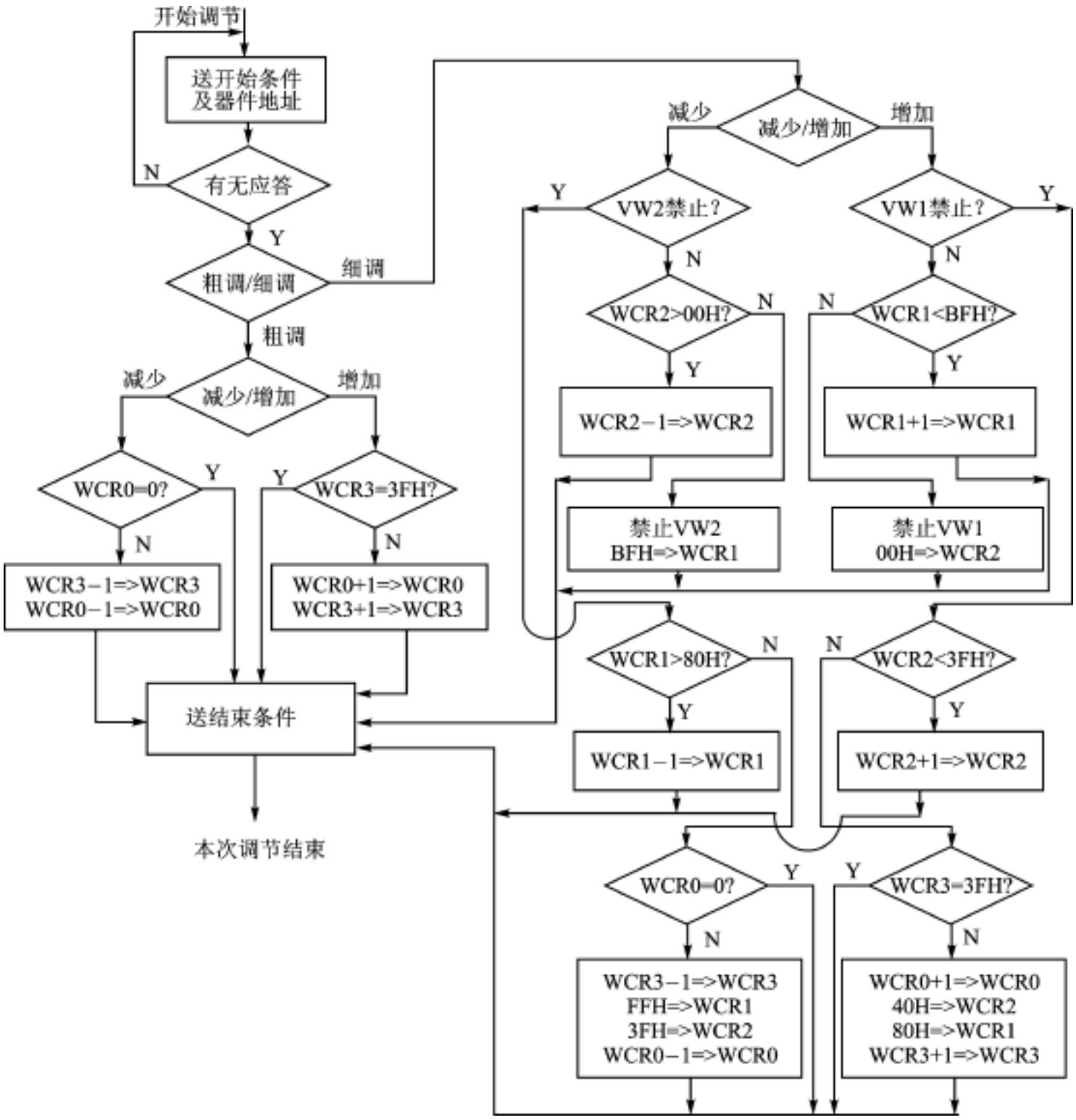


图 8 .9 - 6 调节流程

选自《电测与仪表》月刊, 2001 年第 10 期

8 .10　基于 AD2S80A 的高精度位置检测系统及其在机器人控制中的应用

山东矿业学院机器人研究中心(250031)
周凤余　李贻斌　苏学成　刘　明　李彩虹

一、前　言

基于电磁感应原理的旋转变压器(resolver),有时又称之为解算器,是一种精密微控制电机,原先主要应用于航天、航空、航船以及陆地军事装备中,如用于飞行器姿态控制和检测、导弹制导控制、火炮瞄准和控制、雷达天线跟踪等角位置伺服控制系统中,完成轴角位移信息的检测、传输、接受和变换功能。近年来,在移动机器人、数控机床、计算机辅助制造(CAM)、工业自动化(FA)等方面也得到越来越多的应用。由于旋转变压器是模拟机电元件,为适应测量系统、伺服系统数字化和普遍采用微型计算机控制的要求,需要一定的接口电路,即旋转变压器 - 数字转换器(RDC),以实现其模拟量信号到控制系统数字量的转换。随着电子技术的飞速发展,RDC 已由分立元件发展到小型固态厚膜或薄膜混合集成模块。目前美国 AD 公司又将它发展成为一系列单片集成电路,弥补了模块体积大、笨重的不足,给应用带来了极大的方便。本文着重介绍了基于旋转变压器和 AD2S80A RDC 的高精度位置检测系统的设计与实现方法,以及在大型喷浆机器人控制系统中的应用。

二、位置检测电路设计方法

1 . AD2S80A RDC 结构

AD2S80A RDC 是美国 AD 公司最新推出的 RDC,它将先进的 CMOS 逻辑电路与高精度双极线性电路相结合,以 BIMOS Ⅱ工艺制作的单片集成电路。该集成电路具有两种封装形式: 40 引脚的 DIP 封装(如图 8 .10 - 1 所示)和 44 引脚的 LCC 方形封装。表 8 .10 - 1 给出了 DIP 封装引脚号及功能描述。

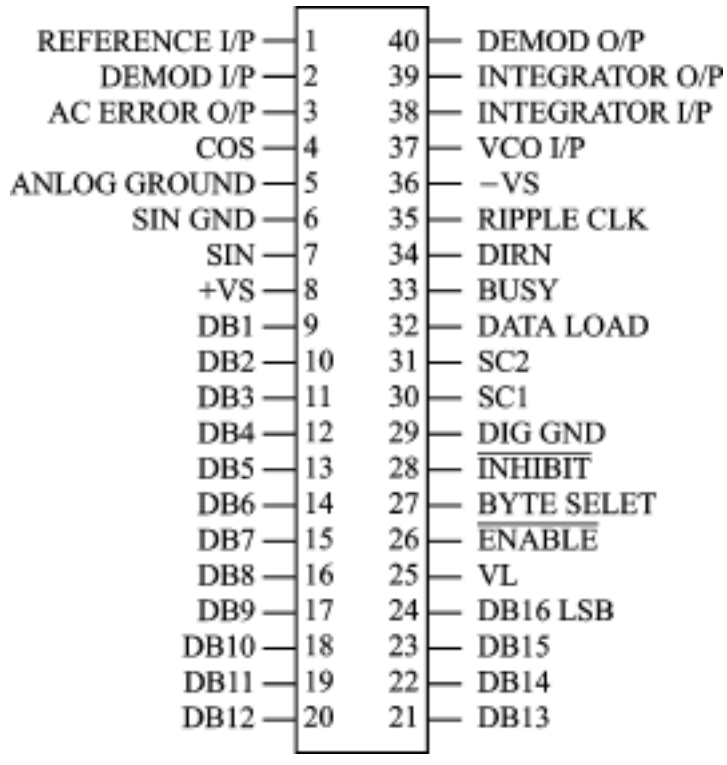


图 8 .10 - 1　DIP 封装引脚图

表 8 .10 - 1 AD2S80A 引脚功能描述

引脚号	符 号	功能描述
1	REFERENCE I/ P	基准信号输入
2	DEMOD I/ P	解调器输入
3	AC ERROR O/ P	比率乘法器输出
4	COS	余弦信号输入
5	ANLOG GROUND	电源地
6	SIN GND	旋转变压器信号地
7	SIN	正弦信号输入
8	+ VS	正电源
9～24	DB1～DB16	并行输出数据
25	VL	逻辑电源
26	$\overline{\text{ENABLE}}$	逻辑高电平—输出高阻态 逻辑低电平—数据存于输出锁存器
27	BYTE SELET	逻辑高电平—高字节送至 DB1～DB8 逻辑低电平—低字节送至 DB1～DB8
28	$\overline{\text{INHIBIT}}$	逻辑低—禁止数据送入输出锁存器
29	DIG GND	逻辑地
30 , 31	SC1 , SC2	选择转换分辨率
32	DATA LOAD	逻辑低—DB1～DB8 逻辑高—DB1～DB16 输出
33	BUSY	逻辑高时, 数据无效
34	DIRN	表示转向
35	RIPPLE CLK	转换器输出从全 1 到全 0 或相反时出现 正脉冲, 表示输入轴转一转或一个节距 (感应同步器)
36	- VS	负电源
37	VCO I/ P	VCO 输入
38	INTEGRATOR I/ P	积分器输入
39	INTEGRATOR O/ P	积分器输出, 即速度输出 VELOCITY
40	DEMOD O/ P	解调器输出

AD2S80A 的工作条件为:

- 电源电压(±VS) 直流:±12(1±0 .1)V;
- 逻辑电源电压(VL) 直流: + 5(1±0 .1)V;
- 模拟输入信号电压(sin 和 cos) 均方根值:2(1±0 .1)V;
- 参考电压 1～8V(peak);
- 输入信号与参考电压失真 10 %(max);

· 输入信号与参考电压间相移　±10 ′(max);

· 工作温度　℃～ + 70℃(商用级),

- 40℃～ + 85℃(工业级),

- 55℃～ + 125℃(军品级)。

2 . AD2S80A 运行特点

AD2S80A 的内部原理框图及外围电路如图 8 .10 - 2 所示。按图 8 .10 - 2 连接后, AD2S80A 就真正成为一个运行于Ⅱ型伺服环的跟踪式 RDC,其数码输出能以选取的最大跟踪速率自动跟踪轴角输入。由于它在把旋转变压器信号转换为自然二进制数时,采用的是比率式跟踪方法,输出数字角仅依赖于 SIN 和 COS 输入信号的比值,而与他们的绝对值大小无关。因此 AD2S80A 对输入信号的幅值和频率变化不是很敏感,也不必使用精确、稳定的振荡器来产生参考信号,转换环路中相敏检测器的存在确保了对参考信号中的正交分量有很高的抑制能力。

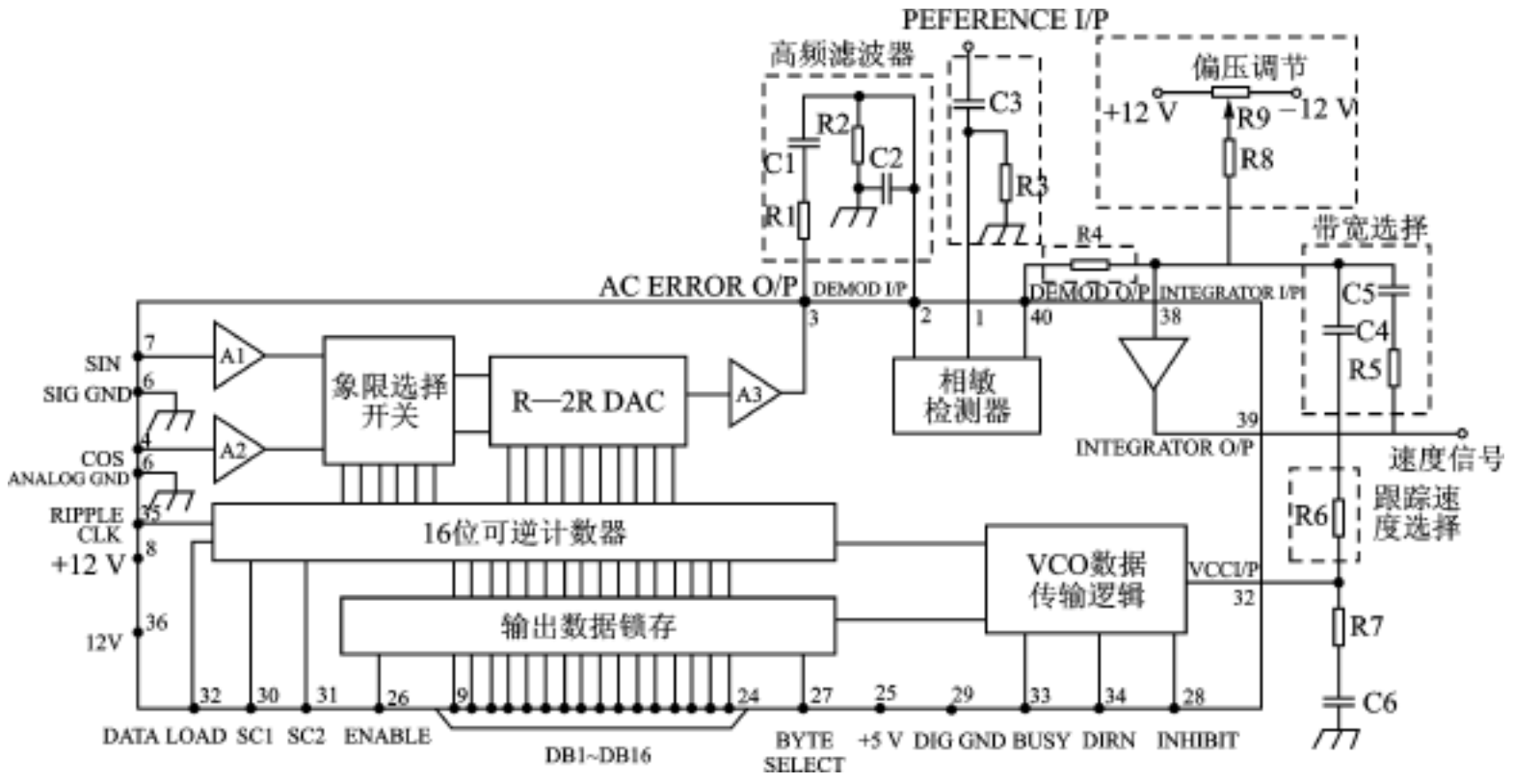


图 8 .10 - 2　AD2S80A 内部原理框图及外围电路

AD2S80A 突出优点还在于它可由用户选择相应参数以优化整个系统的性能。其数字输出分辨率可由 SC1 和 SC2 两个引脚的逻辑状态设置为 10、12、14 或 16 位,见表 8 .10 - 2。而动态特性:如带宽和跟踪速率由外部元件决定。

表 8 .10 - 2　输出分辨率的选择

SC1	SC2	输出分辨率/ bit
0	0	10
0	1	12
1	0	14
1	1	16

3.检测电路设计方法

通过选择外围元件的参数,可决定系统的带宽及最大跟踪速率等动态特性。在选择过程中应保证选用最接近理想数值的元件,并运行于允许的温度范围之内。选用误差等级为5%的元件不会降低转换器的性能。

(1) R1、R2、C1、C2 的选择

这四个元件组成一个 HF(高频)滤波器,其作用是除去所有直流偏置和减少输入到 AD2S80A 的信号中的噪声,因为这些噪声进入相敏检测器后会影响输出。元件的参数按下列式子进行选择计算:

$$15\ \text{k}\Omega \leq R1 = R2 \leq 56\ \text{k}\Omega \tag{1}$$

$$C1 = C2 = \frac{1}{2\pi R_1 f_{REF}} \tag{2}$$

其中,f_{REF} 为参考频率。

若取 R2 = R3、C1 = C2,可省去电阻 R1 和电容 C2。

需要注意的是,该滤波器会影响环路的增益,因为它对输入到相敏检测器的信号有3倍的衰减。

(2) R4 的选择

R4 为增益比例电阻,若 R1、C2 满足式(1)和式(2)的关系,那么

$$R4 = \frac{E_{DC}}{100 \times 10^{-9}} \times \frac{1}{3} \quad \Omega$$

否则

$$R4 = \frac{E_{DC}}{100 \times 10^{-9}} \quad \Omega$$

其中 00×10^{-4} = 电流/ LSB

EDC 160×10^{-3} 0 位分辨率

= 40×10^{-3} 12 位分辨率

= 10×10^{-3} 14 位分辨率

= 2.5×10^{-3} 16 位分辨率

= 电压中直流误差的比例

(3) R3、C3 的选择

R3 和 C3 起着耦合参考输入的作用,合适的 R3 与 C3 可使信号在参考频率点上没有明显的相位移,两元件参数应满足:R3 = 100 kΩ

$$C3 > \frac{1}{R3 f_{REF}} \quad \text{F}$$

(4) R6 的选择

VCO 的输入电阻 R6 用来设置转换器的最大跟踪速率。若在最大跟踪速率时,速度输出为±8 V,则可按下式计算 R6

$$R6 = \frac{6.32 \times 10^{10}}{T \times 2^n} \quad \Omega$$

其中 ——最大跟踪速率,不得超过参考信号频率的 1/16;

n——输出分辨率。

(5) C4、C5、R5 的选择

闭环带宽由这 3 个元件的参数决定。选择带宽 f_{BW} 时,应保证参考频率与 f_{BW} 的比率不要超过下列指标:

分辨率	参考频率与 f_{BW} 的比率
10	2.5∶1
12	4∶1
14	6∶1
16	7.5∶1

当参考频率为 400 Hz 时,带宽的典型值为 100 Hz;当参考频率为 5 kHz 时,带宽的典型值为 500～1000 Hz。f_{BW} 确定以后,C4、C5、R5 的数值分别按下列三式计算,

$$C4 = \frac{21}{R6 \times f_{BW}} \quad F$$

$$C5 = 5C4$$

$$R5 = \frac{4}{2\pi f_{BW} \times C5} \quad \Omega$$

(6) C6、R7 的选择

C6、R7 能对 VCO 进行相位补偿,按下面数值给定:C6 = 470 μF,R7 = 68 Ω

(7) R8、R9 的选择

积分器输入端的零漂与偏置电流能引起转换器输出端额外的位置漂移,如漂移能被忽略的话,可省略 R8、R9,否则取 R8 = 4.7 MΩ、R9 = 1 MΩ(电位器)。

为了校正零点漂移,首先要确保信号输入端没有与旋转变压器连接,外围电路元件均按上述公正已匹配合适。然后把 COS 和 REFERENCE INPUT 二个管脚相连,SIN 与 SIGNAL GROUND 二个管脚相连,加上电源和参考信号,调节电位器 R9 使数字输出为全“0”即可。

三、机器人位置检测系统设计与实现

大型喷浆机器人是我们课题组承担的国家 863 计划项目,它具有 6 个自由度,其中 4 个需要用高精度伺服控制。由于喷浆机器人是一个用于水电、水利、铁路、公路和地铁等隧道和涵洞中的工业产品,常年工作在潮湿、震动、温度变化范围大的环境中,因此对伺服控制系统的位置检测单元应慎重考虑。另外,由于机器人杆件较长,底端很小的转动就会造成末端较大的移动。因此,在确保可靠的情况下,还必须满足精度要求。经过多方面比较,我们最终选定了由无接触式旋转变压器和 AD2S80A RDC 单片集成电路构成位置检测单元的设计方案。

1. 位置检测单元硬件电路的设计与实现

根据实际需要,系统要求 16 位的分辨率,最大跟踪速率为 2.5 r/s,并选取参考频率为 400 Hz、带宽为 100 Hz。按照这些性能指标,利用上述计算公式对 AD2S80A 外围电路元件进行选择后,得到图 8.10-3 所示的位置检测单元硬件连接图。

在具体实现这个电路时,建议在 + VS、- VS 电源线与 ANALOG GROUND 之间, + VL 与 DIGITAL GROUND 之间在靠近转换器的地方分别并联 100 μF(陶瓷)和 10 μF(钽) 的去耦电容。如果一块印刷电路板上下只一块转换器时,每个转换器都应有自己单独的去耦电容。旋转变压器的两个信号地在转换器的 SIGNAL GROUND 管脚处相连,以减少正、余弦信号间

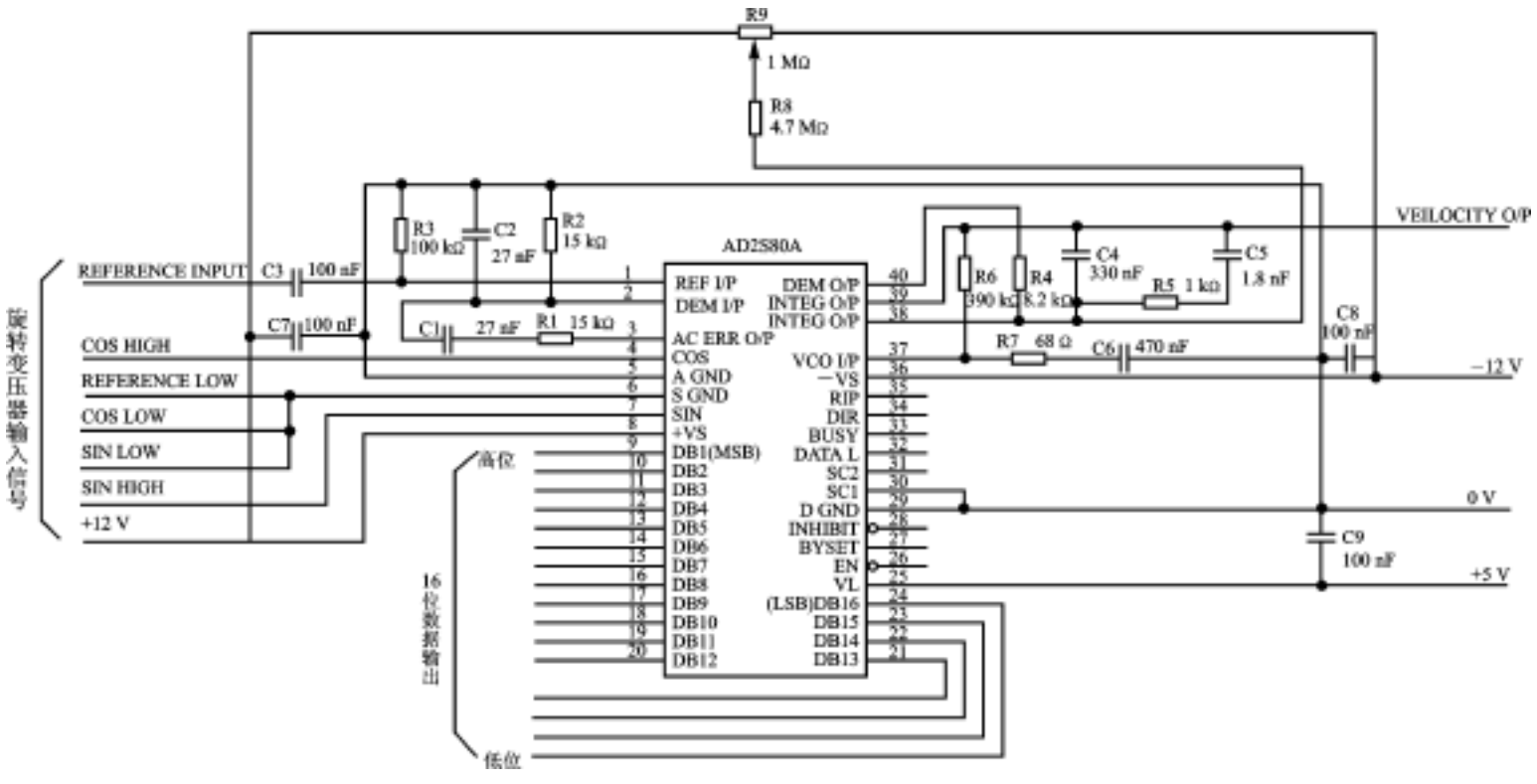


图 8 .10 - 3 位置检测单元硬件连接图

的耦合,为此建议旋转变压器的正、余弦信号,参考信号应分别使用单独屏蔽的多股低容抗电缆。外围电路元件选用 1/ 4 W,精度为 5% 的电阻,耐压 100 V,精度为 10% 的陶瓷电容即可满足要求,由于 AD2S80A 内部没有输入变压器,当信号和参考电压大于 2Vrms 时,可用一个简单的电阻分压电路来产生适合于变换器的正确信号,但应保证正弦线与地之间的电阻和余弦线与地之间的电阻的比值保持不变,稍有不同将导致产生额外的位置误差。

2 . AD2S80A 与 80C196KC 单片机的接口技术

为了实现数据的采集与处理,需将 RDC 与单片机进行接口。接口电路很简单,只要把 RDC 的$\overline{\text{INHIBIT}}$引脚与单片机的片选信号相连,$\overline{\text{ENABLE}}$引脚与$\overline{\text{RD}}$信号相连,BYTE SELECT 引脚与计算机的字节选择信号(比如 P1 .0)相连,BUSY 引脚与计算机的输入口线(比如 P2 .1)相连即可。另外,当用户使用 8 位宽度的总线时,只需 RDC 的高 8 位数据输出线与计算机的数据总线相连;当用户使用 16 位总线时,RDC 的 BYSEL SELECT 引脚应接高电平(或悬空),输出全量与总线一一相连。按上述方法连接后,单片机就可以对 AD2S80A 进行操作,读取数据。在介绍操作过程之前,有必要对转换器的控制信号加以简单的说明。

BUSY 信号:输出信号是否有效由 BUSY 输出状态决定。当输入到变换器的数据变化时,BUSY 管脚将输出一列 TTL 电平的脉冲,输入模拟量每变化 1LSB 就产生一个 BUSY 脉冲,内部计算器被加数或减数。当 BUSY 为高电平时,表示转换器内部正处于跟踪转换状态,输出数据不可靠,当 BUSY 为低电平时,表示转换结束,数据有效,可以读取,该引脚为单片机检测转换状态提供了方便。

$\overline{\text{INHIBIT}}$信号:$\overline{\text{INHIBIT}}$信号只禁止数据从可逆计数器传输到输出锁存器中,因此并不干扰跟踪环的运行,释放该信号,转换器会自动产生一个 BUSY 脉冲并刷新输出锁存器。

$\overline{\text{ENABLE}}$信号:$\overline{\text{ENABLE}}$信号决定输出数据的状态。高电平时,使输出数据管脚保持高阻状态;低电平时,把锁存器中的数据传输到输出管脚上,对该信号的操作对转换器的运行没有影响。

BYTE SELECT 信号:BYTE SELECT 信号选择位置数据的高位还是低位出现在数据输

出线 DB1～DB8 上。无论该信号的状态如何,只要$\overline{\text{ENABLE}}$为低电平,低位字节都将出现在数据输出线 DB9～DB16 上。需注意的是,当设置的分辨率小于 16 位时,未使用的数据线要下拉为低电平。当 BYTE SELECT 为高电平时,高 8 位数据输出到数据线 DB1～DB8 上;当 BYTE SELECT 为低电平时,低 8 位数据复制到高 8 位输出线上。

速度信号:速度信号是由积分器输出的与转速成比例的直流模拟信号,在许多情况下,可使用该信号来代替传统的测速发电机。

下面介绍单片机读取数据的过程:单片机首先检测 BUSY 信号,在检测到 BUSY 为低电平时,对 AD2S80A 施加$\overline{\text{INHIBIT}}$信号,阻止锁存器的刷新,在$\overline{\text{INHIBIT}}$被置为低电平并延迟 600 ns 后,单片机对 BYTE SELECT 操作,然后通过$\overline{\text{ENABLE}}$信号分别读取数据的高位字节和低位字节。读完后,立即释放$\overline{\text{INHIBIT}}$信号,把它恢复成高电平。该过程可用图 8 .10－4 的时序图来表示。

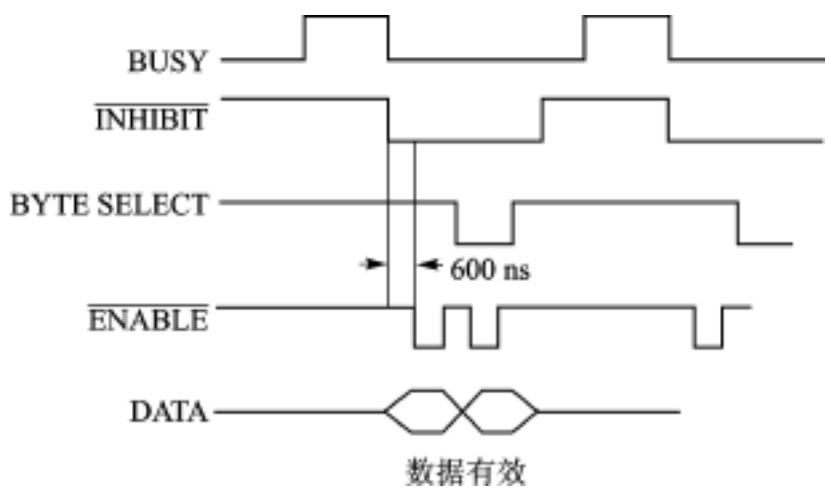


图 8 .10－4　单片机读取数据的时序图

关于系统的软件设计,由于比较简单,也限于篇幅,这里不再赘述。

四、结束语

本文对 AD2S80A RDC 集成电路的结构和外围器件进行了较详细介绍与分析,设计并实现了由该集成电路和旋转变压器构成的检测单元,并把这一位置检测单元与单片微型机 INTEL80C196KC 相结合组成位置伺服系统,运用于大型喷浆机器人控制系统中。实际运行结果表明:该单元最高精度可达 3,与由分立元件或 RDC 模块构成的位置检测单元相比较,有体积小、结构紧凑、抗干扰能力强、可靠性高、操作灵活方便等优点,特别适用于恶劣环境中的位置检测系统,具有很高的应用价值。

参 考 文 献

1　NALOG DEVICES 公司 . 美国 . Variable Resolution Monolithic Resolver-to-Digital Converter AD2S80A

2　谭建成 . 电机控制专用集成电路 . 北京:机械工业出版社,1997

3　孙涵芳 . INTEL 16 位单片机 . 北京:北京航空航天大学出版社,1995

选自《电气自动化》双月刊,2000 年第 2 期

第九章

文章摘要

一、专题论述

1.1 微控制器的发展趋势

摘自《电子世界》月刊,2000 年第 9 期

该文作者认为,微控制器(单片机)发展趋势主要有以下 7 点:

1.进一步改进 CPU 的性能,加快指令运算速度和系统控制的可靠性,包括采用双 CPU 结构以提高处理能力,采用流水线结构,增加数据总线宽度,通过提高时钟频率来大力提高运行速度。

2.存储器功能日臻完善,包括不断扩展存储器容量,集成在片内的存储器种类越来越多,片内存储器具有超强加密功能。

3.外围电路内装化。

4.I/O 功能增强,包括适合微控制器技术特点的接口种类增多,I/O 口驱动能力增强,进一步加快 I/O 的传输速度。

5.向极端参数发展,包括工作电压极低,电压允许波动范围极宽,微功耗,极小封装。

6.采取多种措施提高芯片及组成系统工作的可靠性。

7.制造工艺不断改进。

1.2 系统微集成技术的发展

摘自《探测与控制学报》季刊,2000 年第 3 期

20 世纪 90 年代以来系统微集成技术得到迅速发展。该文作者总结了近年来发展的如下一些情况:

1.硅晶片和集成芯片规模迅速变大。包括硅晶片与集成电路芯片尺寸迅速变大,可加工线条尺寸(λ)越来越精细,每个芯片包含的门电路数迅速增加,芯片上的 I/O 引线数迅速增加,芯片上的焊(盘)点间距逐步变小。

2.系统微集成技术出现几个新分支学科,包括机械电子学(Mechatronics),微机电系统(MEMS-Microelectro Mechanical system),微光机电子学(Microoptomechatronics)。值得注意的是,系统微集成工艺设计人员在进行技术性能设计的同时,还要特别注意生态学与经济学设计,即所谓 ECO(Economyand Ecology design)设计。

3.采用柔性芯片的微集成系统。

4.实现三维微集成的技术途径之一——积木模块。

5.新型印制电路板。

6.三维高密度集成方法之一——柔性 PCB 折叠。

文中对 THT、THT 和 SMT 混合、SMT、CSP、FC、SMT 以及 3D BGA,TB BGA 等各种电子装配工艺进行了比较。

1.3 多芯片组件技术及其应用

摘自《航空电子技术》季刊,2000 年第 2 期

多芯片组件技术(MCM-Multichip Module Technology)是一种用先进的微电子组装技术封装而成的高级混合组件。该文阐述了 MCM 的发展历史,MCM 技术中 MCM 的关键技术,包括:基板材料,介质材料,导体材料,制造技术和它在计算机、军事、通信及消费类电子产品中的应用。

1.4 MCS-51 和 80C51 系列单片机

摘自《电子世界》月刊,2000 年第 8 期

该文概述了 Intel 8 位单片机的发展。从最初的单片机发展到如今的新一代单片机,大致经历了三代。第一代:以 1976 年推出的 MCS-48 系列为代表。第二代:以 MCS-51 的 8051、8052 为代表。第三代:以 80C51 系列为代表。还包括了飞利浦、西门子、ADM、富士通、OKI、Atmel 等公司以 80C51 为核心推出的大量各具特色、与 MCS-51 兼容的单片机。

新一代 80C51 系列单片机最主要技术特点是向外部接口电路扩展,以实现微控制器完善的控制功能为己任。由于 80C51 系列所具有的一系列特点,其获得广泛使用将是指日可待的。

1.5 PSD813 器件在单片机系统中的应用

摘自《电子技术应用》月刊,2000 年第 10 期

该文介绍 PSD813 系列产品的特点,并结合实例介绍了系统硬件设计以及使用流程。文中专门介绍了 PSD813 专用开发软件 PSDsoft 可编程外围器件 PSD813 应用于单片机系统后,可大幅度简化 CPU 外围电路的设计,减小系统体积,降低功耗,增强系统可靠性。

1.6 主辅单片机系统的设计及应用

摘自《工业控制计算机》月刊,2000 第 5 期

采用廉价单片机作为辅助单片机,完成系统的某些特殊功能,可取代一些可编程外围接口芯片,使系统设计更加灵活,结构更加简单。该文介绍以廉价单片机作为辅助单片机的主辅单片机系统的设计及应用。

文中给出了系统设计的一些原则并介绍了应用实例。采用廉价的单片机作为辅助单片机,可在不提高成本的基础上优化单片系统的设计,使系统结构简单,功能强化,不失为单片机系统设计的一个重要方向。

1.7 一种双单片机结构的微机控制器

摘自《工业控制计算机》月刊,2000 年第 5 期

该文介绍在研制开发螺旋钢箍机控制系统时,采用的双规制的 CPU 结构模式。文中给出了硬件结构和软件结构。实践证明,这是一种非常有效的设计方案,整套系统并没有增加原器件,就可以生产出两种性能完全不同的系统来,并且可保持较长的时间,运行情况可靠。

1.8 用 PC 机直接开发单片机系统

摘自《微型机与应用》月刊,2000 年第 5 期

该文介绍一种不依赖于仿真器、通过 PC 机串行口直接调试目标系统的单片机自开发方法。文中介绍了设计思路,硬件和软件及开发过程。用本方法开发应用系统,允许目标系统只有外部 RAM 或只有外部 ROM,调试期间代码的执行环境与开发成功后的实际运行环境是一样的。同时,由于系统具有自我固化的能力,使其真正摆脱了对专用开发工具的依赖,不但能被专业开发者使用,也为业余开发者提供了一个方便途径。

1.9 单片机系统大容量存储器扩展技术

摘自《电测与仪表》月刊,2001 年第 10 期

高速采样、分析系统往往数据量很大,需要几百 KB 甚至数 MB 的数据存储器,故其寻址空间需要扩展。本文重点介绍在扩展中可能遇到的影响存储器系统甚至 CPU 正常工作可靠性的问题及其解决办法。着重论述了存储地址的无冲突设计和基本/ 扩展存储器间数据线的隔离问题。无论选择具有输出锁存功能的端口作页面控制,还是通过数据端口实现页面控制,都应该设置页面寄存器。在软件设计中,每次完成对扩展存储器的访问后,一定要立即将页面指针恢复指向基本空间,不要等到欲访问基本空间时再恢复指针。所有这些措施,都对具有扩展存储器的单片机系统的可靠运行极为有利,应在系统开发设计时正确运用。

1.10 高性能微处理器性能模型设计

摘自《航空电子技术》季刊,2000 年第 2 期

当前国际上集成电路芯片研制和设计中,大都采用自上而下的正向设计技术。在高性能微处理器设计中,性能仿真模型能在设计的前期获得对于系统体系结构的性能评估,解决流水线的性能、组织、相关等关键技术问题。本文介绍的性能模型,是针对 32 位 RISC 微处理器 i960KA 的指令集体系结构,将 i960KA 五级流水线更改为超标量流水线,对超标量处理器的体系结构进行性能分析,为设计者进行结构级改进提供依据,以获得更高的系统性能。高性能微处理器性能模型的设计研究,将为进一步研制具有独立版权的高性能微处理器打下坚实的基础,具有较大的实际意义。

1.11 闪速存储器的选择与接口

摘自《航空电子技术》季刊,2000 年第 2 期

该文在对 INTEL、AMD、SST 等公司产品不同特点进行比较的基础上,选择 28F016XS 作为研制海量存储器的存储芯片。文中介绍了 28F016XF 的突出特点;并详细说明 28F016XS 与 DTEM/ MM 分系统的微处理器 80960KB 的硬件接口设计方案。按照 28F016XS 的输入要求和 8090KB 的总线时序,采用状态机作为接口的核心,实现正常的寻址和读写操作。

1.12 串行存储器接口的比较及选择

摘自《电测与仪表》月刊,2001 年第 3 期

该文对常见的 I^2C、SPI 和 MPS 三种串行存储器接口进行了分析比较,并侧重分析了相应

的串行器件在进行写操作时的抗干扰性能。作者的结论是:I²C 接口具有占用端线少的突出优点,并且采用这种接口模式的串行器件非常广泛,设计人员可以非常方便和灵活地选用这类器件。然而,I²C 接口也存在速度偏低,对噪声干扰较敏感的缺点,在选择和应用时应加以考虑。SPI 接口传输速度较高,抗噪声干扰性能较好,并有大量器件(尤其是核心器件),采用这种接口,方便设计人员的选择和应用。但相比 I²C 接口,在 I/O 端线较紧张和线路板布线密度较高的情况下,SPI 接口占用端线较多的缺点就会显现出来。MPS 接口传输速度最高,且对噪声干扰的敏感程度低,但包括数据,地址及各种控制线在内的接口线显然偏多。

1.13 移位寄存器分析方法的研究

摘自《航空计标技术》季刊,2000 年第 2 期

该文对串行输入、并行输出移位寄存器,就移位方向的判断,研究移位的顺序和确定移位的规律进行了剖析。作者认为移位方向由位置判断法决定;移位顺序以数据输入端为准,先远后近移送;移位规律(画时序图的顺序)从数据输入端出发,先近后远画出。通过对移位功能中移位方向、顺序、规律全面、细致地分析,明晰了移位寄存器状态表和时序图的内在联系,揭示了分析移位寄存器功能的普遍性规律,对于正确分析、使用移位寄存器具有实际意义。

1.14 GPS 的时频系统

摘自《空间电子技术》季刊,2000 年第 4 期

GPS 时已经成为目前重要的标准时间,应用于各种各样的系统中,GPS 系统优越的定位精度也主要依赖于其完善的时频系统。建立三维卫星定位系统极为迫切。为了为我国的下一代卫星定位系统提供可借鉴的思路,该文详细叙述了作者对 GPS 时频系统的研究结果。包括:GPS 时频系统的基本方案,地面控制系统的时频技术,卫星及其时频实现以及用户接收机精确时间的获得。本文是作者在占有资料很少的情况下的研究结果,许多部分是作者的推理和猜测,希望能起到一定的启发作用。

1.15 一种基于 C 语言的虚拟仪器系统实现方法

摘自《宇航计测技术》双月刊,2000 年第 5 期

虚拟仪器是在通用计算机平台上,用户根据自己的需求定义和设计仪器的测试功能,其实质是将传统仪器硬件和最新计算机软件技术充分结合起来,以实现并扩展传统仪器的功能。该文介绍虚拟仪器及其开发环境 Lab VIEW 的特点,分析了 LabVIEW 调用外部程序代码途径,实现了将 Lab VIEW 与 C 语言接口的高级技术-CIN。实践证明,该方法高效、易行,是增加 Lab VIEW 整体性能的一条很好的途径。由于在 Lab VIEW 中引入了 C 语言的强大功能,从而提高了 Lab VIEW 的数据处理能力,并极大地增强了 Lab VIEW 与其他 Windows 应用程序之间的数据共享能力。

1.16 智能家庭网络研究综述

摘自《计算机应用研究》双月刊,2001 年第 9 期

智能家庭网络是指在家庭内部通过一定的传输介质(如电力线、双绞线、同轴电缆、无线电、红外等)将各种电气设备和电气子系统连接起来,采用统一的通信协议,对内实现资源共

享,对外能通过网关与外部网(如 Ethernet, ISDN, 3DDW 等)互连进行信息交换。该文逐一介绍了国际上比较重要的六种智能家庭网络系统包括美国的 X-10, CEBus, LonWorks;日本的 HBS,欧洲的 EIB, EHS,分别介绍各自的特点,并列表进行比较,指出它们的共同特点以及智能家庭网络的未来发展趋势;最后分析了智能家庭网络需要研究的主要问题。

在我国,智能家庭网络的研究刚刚起步,迫切需要建立一套适合我国国情的智能家庭网络的国家标准,以使不同厂家的家用设备实现互操作,同时带动一大批企业开发具有我国自主知识产权的家庭网络产品。

1.17 用 C51 实现电力部多功能电能表通信规约

摘自《电测与仪表》月刊,2001 年第 9 期

电力部的《多功能电能表通信规约》DL/ T 645-1997 规约,规定了多功能电能表的费率装置与数据终端设置进行数据交换时的协议。该文介绍了采用 MCS－51 单片机,以 C51 为编程语言,实现规约的方法。详细介绍了 MCS－51 的串口工作在中断方式下,用状态图描述规约的接收过程,以及带缓存的发送过程,并给出了相关的代码。目前,已将《多功能电能表通信规约》扩充应用到‘水电气三表自动抄表系统’的通信规约中。实际运行情况良好并为将来进一步修改现有的协议打下了很好的基础。

1.18 测控系统中采样数据的预处理

摘自《测控技术》月刊,2000 年第 8 期

由于测控系统(尤其是大的工业现场控制系统)中总会有各种干扰存在,需要对采样数据从软件上进行预处理。该文结合作者多年从事计算机测控系统的研制开发实践,总结出测控系统中常用的一些对采样数据进行预处理的方法,包括刻度和标度变换、数字性能和平滑处理这些方法分别在研制开发的‘SDCL-2000 分布式计算机数控测井系统’和‘SL-3000 计算机测井系统’中得到应用,经历数年现场使用说明效果很好。

1.19 数据采集系统动态特性的总体评价

摘自《航空计测技术》双月刊,2000 年第 4 期

数据采集系统动态特性的辨识和评价,是其总体性能评价中至关重要的一个目标。本文通过选取频带宽度,传递函数等 8 个相互比较独立的指标来总体描述数据采集系统的动态特性,然后通过几个特殊的模型化方法将其动态特性分别以动态有效位数、传递函数、阶跃响应曲线和幅频特性、相频特性的方式给出,从而解决了数据采集系统动态特性的总体评价问题。通过实验验证,本文所述方法可用于数据采集系统的动态特性评价。

1.20 一个高速准确的手写数字识别系统

摘自《计算机应用研究》月刊,2001 年第 11 期

手写文字(数字)的识别是国际上众多科技工作者研究的一个热点。该文对于手写体数字的识别提出了一种新思路,阐述了整个识别步骤中的关键算法。系统可以自动解析包含二值化手写体数字 BMP 文件,在经过裁剪、细化、映射归一、获取数字结构特征、匹配等一系列处理之后,可以最大限度地减少因为不同手写风格而带来的干扰,从而快速准确地识别出每个数

字。基于所描述的算法而实现的程序,具有很好的识别性能,应用前景广泛。

1.21 日本理光实时时钟集成电路发展历史及现状

摘自《电测与仪表》月刊,2001 年第 10 期

实时时钟集成电路是一种高密度集成的专用时钟集成电路,适合于一切需要微功耗计时的场合,如手机、电视机、复费率电表、高精度时钟、可编程时间控制以及数码相机等等。该文简要总结了日本理光实时时钟集成电路的发展历史及其现状,并提醒用户在实际使用中需注意的问题及使用技巧。

1.22 单片开关电源的发展及其应用

摘自《电子技术应用》月刊,2000 年第 1 期

小功率开关电源正朝着单片的方向发展。单片开关电源集成电路具有高集成度、高性价比、最简外围电路、最佳性能指标、能构成高效率无工频变压器的隔离式开关电源等优点。本文详细阐述近年来问世的 TOPSwitch-II 系列三端单片开关电源, TinySwitch 系列四端开关电源和 MC33370 系列五端单片开关电源。文中分别给出各自的性能特点与典型应用。

二、综合应用技术

2.1 MCS-51 系列单片机在 SDH 系统中的应用

摘自《电子技术应用》月刊,2000 年第 7 期

同步数字序列 SDH(Synchronous Digital Hierarchy)是一种全新的传输网体制。在我国,干线网络基本采用了 SDH 网络。SDH 大规模专用集成电路套片包括高阶复用芯片 MXH0155-2,低阶映射芯片 MXL021E1-3,基于这两个芯片,可以实现一个基于双向 SDH 环路的 ADM(ADD/ DROP Multiplexer)站点。该文选用了 Dallas 公司的 DS80C320 做为处理器来实现对此 ADM 系统的管理。文中介绍了微处理器系统的硬件实现,微处理器系统的软件构成。通过此微处理器系统的设计、仿真和实际验证,证明了一个复杂的 SDH 双向环路的 ADM 站点可以通过简单的微处理器系统实现完全的管理和监控,为 SDH 大规模专用集成电路的推广应用奠定了基础。

2.2 公共闪存接口在 Flash Memory 程序设计中的应用

摘自《电子技术应用》月刊,2000 年第 7 期

公共闪存接口(Common Flash Interface,简称 CFI)是一个开放的从 Flash Memory 器件中读取数据的标准接口。该文介绍了公共闪存接口结构以及系统软件如何利用 CFI 获取 Flash Memory 的各种参数,实现对各种 Flash Memory 的程序设计。文中给出了查询操作程序框图。

2.3 应用 IA MMX™ 技术的离散余弦变换

摘自《电子技术应用》月刊,2000 年第 7 期

IA MMX™ 技术最重要的特点是单指令多操作数据(SLMD),这一技术允许一条指令并行处理多个数据元素。它的设计目的就是用来加强多媒体和通信方面的应用。该文在简要介绍 IA MMX™ 技术的基础上,重点讨论了应用 IA MMX™ 技术的函数余弦变换(DCT)快速算法及其优越性能。IA MML™ 技术在数据处理的许多算法中都能发挥优越的性能,成倍地提高运行速度。在多媒体应用中,IA MMX™ 技术大大降低了程序对处理器的占用率,使处理器能够同时更有效地执行其他任务。

2.4 串行实时时钟芯片 DS1302 程序设计中的问题与对策

摘自《电子技术应用》月刊,2000 年第 7 期

Dallas 公司的串行接口实时时钟芯片 DS1302 可对时钟芯片备份电池进行涓流充电,具有体积小、功耗低、接口容易、占用 CPU I/O 口线少等主要特点,可作为实时时钟,广泛应用于智能化仪器仪表中。该文指出了串行实时时钟芯片 DS1302 程序设计中几个易被疏忽而导致错误的问题,包括读操作出现的错误,禁止涓流充电出现的错误及受干扰时钟/ 日历信息出现

的错误。为避免以上错误的产生,文中分析了问题的原因,并给出了解决问题的方法。

2.5 数字传感器及其应用

摘自《测控技术》月刊,2000 年第 5 期

该文分析了 DS1821 数字温度传感器的原理结构、工作方式及操作指令,提出了用 DS1821 进行热电偶自由端温度补偿的新方法,并将其应用于智能温度测控仪表的设计中,具有准确、线性、完全补偿的特点,从根本上解决了桥式冷端补偿器的非线性与型号匹配问题。DS1821 在冷端补偿器设计中的引入,优化了系统结构,解决了通用性和补偿精度问题,是热电偶测温中一种有益尝试。

2.6 电阻式温度传感器的系列化设计及其应用

摘自《测控技术》月刊,2000 年第 11 期

国内对于宽量程电阻式湿敏元件的研究,程度不同地存在着线性不好、响应慢、稳定性差、怕结露、不耐污染等问题。该文针对上述问题进行了多方位的探索和研究,通过对湿敏元件的系列化研究和组合,实现了传感器的系列化设计。文中介绍了包括选择主链具有环形结构的高分子材料做电解质材料的载体和改性加工和配伍材料等六项研究重点,并介绍了电阻式湿敏元件在变送器中的应用。目前,ZSM 系列湿敏传感器全部使用国产材料和元器件制成,价格低,已实现系列化,符合市场要求已应用于工业与民用部门。

2.7 温度传感器及其与微处理器接口

摘自《电子技术应用》月刊,2000 年第 3 期

根据温度传感器输出方式及接口方式的不同可以分为模拟温度传感器和数字温度传感器。模拟温度传感器输出的模拟信号,必须经过专门的接口电路转换成数字信号后才能由微处理器进行处理。随着计算机及半导体技术的飞速发展,数字接口的半导体温度传感器得到了广泛应用和快速发展。该文通过具体芯片的应用,介绍了各自与微处理器的具体接口。重点讨论了具有数字接口的半导体温度传感器包括单线输出的数字温度传感器和基于总线协议输出的数字温度传感器。数字半导体温度传感器由于其廉价、精确、线性、低功耗、小型化等特点必将得到更大的发展。

2.8 AD7416 数字温度传感器及其应用

摘自《电子世界》月刊,2000 年第 6 期

AD7416 是 AD 公司生产的内置 I^2C 串行总线接口的新型数字温度传感器,具有价格便宜、测量精度高、低功耗、可以直接与微处理器接口的特点,可广泛应用于个人计算机、电子通信、家用电器等各种设备的温度测量和控制。该文介绍了 AD7416 的内部结构和功能、工作时序和方式以及在多点温度测量系统中的应用。

2.9 隔离放大器及其应用

摘自《电子技术应用》月刊,2000 年第 3 期

该文介绍了具有代表性的隔离放大器包括: ISO 122, AD202/ 204, 3650、3652 以及

HCPL7800/ 7800A/ 7800B,讨论了隔离放大器应用时应注意的问题,包括降低辐射及线性光耦的死区。

2 .10 高速 A/ D 转换器动态参数

摘自《电测与仪表》月刊,2001 年第 3 期

该文重点介绍和分析 A/ D 转换器的动态参数包括信噪比(SNR),信号与噪声 + 失真之比(SINAD),有效比特位数(ENOB),谐波失真(THD),无杂散动态范围(SFDR),双音交调失真(双音 IMD),多音交调失真及电压驻波比(VSWR),说明了这些参数对 A/ D 变格器性能的影响。随着数字信号处理技术迅猛发展及滤波、变频、调制/ 解调等其他一些处理任务进入数字领域,对于作为模拟系统和数字系统之间桥梁的 A/ D 转换器的要求越来越高,应在理解 ADC 性能指标定义的基础上,根据具体需要合理选择 ADC。

2 .11 V/ F 变换在单片机系统中的应用

摘自《电子技术应用》月刊,2000 年第 2 期

电动汽车单片机测控系统的工作环境十分恶劣,要求在设计单片机系统时必须考虑到各种影响其正常工作的因素,并采取相应的有效措施。该文介绍了将 V/ F 变换用于电动汽车单片机系统的抗干扰措施。V/ F 方式的使用,有效地消除了空间和过程通道的干扰。

2 .12 微处理器内嵌式模数转换器在精密仪器中的应用研究

摘自《测控技术》月刊,2000 年第 4 期

微处理器内嵌式 A/ D 集成电路在精密测试仪器应用中,经常会在模拟输入信号输入幅度、信号极性以及分辨率等方面遇到问题。本文分析和研究了微处理器内嵌式 A/ D 在精密测试仪器应用中可能遇到的问题及解决方法,包括模拟输入信号的幅度限制,模拟输入信号的极性变换及 A/ D 分辨率的提高。以上方法已成功地应用在分辨率为 0 .01 '的高精度自准直系统中,获得了理想的应用效果。

2 .13 电子秤非线性自动修正方法

摘自《电子技术应用》月刊,2000 年第 2 期

本文介绍用线性插值方法进行的一种方便、准确、有效的非线性修正方法,并给出了应用实例。文中以高性价比的单片机 Z86E08 为核心,通过与 EEPROM 接口进行数据交换,用内码自动标定来修正误差,提高了测量精度,降低了成本,简化了调校过程。

2 .14 光耦传输的非线性校正

摘自《电子技术》月刊,2001 年第 1 期

由于光耦内部发光二极管状-安特性的非线性,使得光耦的输入与输出之间不呈线性关系。该文介绍一种能不失真地隔离传输模拟量的电路,该电路可用于不共地的线性信号在传输过程中保持不失真。

2.15 高斯滤波器在实时系统中的快速实现

摘自《电子技术应用》月刊,2000 年第 3 期

该文介绍在一个用 MCS－51 单片机组成的强噪声背景下的通信系统中,高斯滤波器的快速实现。文中介绍了算法原理,均值滤波器的 MCS－51 快速实现方法。这种方法,特别适用于实时数据采集、处理、控制系统中的滤波。实践表明,这种方法具有很高的实用价值,值得推广。

2.16 用在系统可编程模拟器件实现双二阶型滤波器

摘自《电子技术应用》月刊,2000 年第 8 期

Lattice 公司最近推出的在系统可编程模拟电路(in system programmability Programmable Analog Circuits,简称 isp-PAC)允许设计者使用 EDA 开发软件、利用计算机设计和修改模拟电路,进行电路特性模拟,最后通过编程电缆将设计方案下载至芯片中。该文阐述了在系统可编程模拟器件的特点以及用它设计双二阶型、连续时间低通和带通滤波器的方法。

2.17 最小二乘法在高精度温度测量中的应用

摘自《传感器技术》月刊,2000 年第 1 期

铂热电阻的非线性和不平衡电桥的非线性,给温度测量带来较大的误差。为了解决铂热电阻不平衡电桥测温法中的非线性误差,本文通过硬件电路对铂电阻 Pt100 温度传感器的非线性进行改善。在此基础上利用最小二乘法,给出了一个适用于 0～800℃之间的测温多项式,从而得到较高的测量精度。由于此方法速度快、精度高、方法简单,因此,适用于许多高精度的温度测量中。

2.18 提高实时频率测量范围和精度新方法

摘自《电子测量技术》月刊,2001 年第 4 期

如何兼顾很高和很低转速的测量,而又有足够的测量精度及测量实时性是转速测量和频率测量研究的一个重点。精密游标式实时频率测量法解决了低频的测量,但实际使用中还存在高频测量的问题,本文提出了一种扩展高频测量范围以及精度的新方法,并介绍了用单片机组成的测量电路。文中给出了采用单片机计数频率范围扩展电路的游标式测频电路。采用单片机计数频率范围扩展电路,扩展了高频段的测量范围,提高了测量准确度和实时性。

2.19 具有微控制器的智能仪表设计与应用

摘自《兵工自动化》月刊,2001 年第 1 期

该文介绍了一种以微控制器基本系统及测控软件、通信接口和输入/输出接口构成的全数字化智能仪表的构成与功能、与上位机 PC 的通信协议、和仪表软件的抗干扰技术。同时介绍了基于智能仪表的集散型计算机测控系统的结构及所实现的功能及其应用。

2.20 用 C 语言编程的数据采集系统

摘自《仪表技术》双月刊,2001 年第 4 期

利用 PC 机和 AD 板构成微机控制的数据采集系统是较好的选择。它可以在 PC 机显示器上给出此单片机数据采集系统更为生动的彩色显示,此外,还可以利用 C 语言编程的优越性取代汇编语言编程,从而不仅使程序大大简化,而且使其与机器无关,即在具有 I/O 扩展槽的不同型号的 PC 机上都可使用。该文介绍一种利用 PC 机和 ADC 构成的、并用 C 语言编程的数据采集系统。文中介绍了硬件连接和如何用 C 语言编写 A/D 转换程序。

2.21 大动态范围浮点 A/D 数据采集器的设计

摘自《电气自动化》双月刊,2000 年第 2 期

船用捷联导航系统要求其陀螺仪能测量小至 0.03°/h,大至 60°/s 的转动角速度,其动态量程达 10^7,由于陀螺仪的动态测量范围很大,即使目前使用的高精度 16 位 A/D 转换器也不能满足要求,本文提出了用浮点 ADC 提高动态范围的方法来采集陀螺仪的输出角速度信号。该文介绍了浮点 A/D 数据采集原理以及船用捷联惯导系统中基于高性能十六位单片机 80C196KC 的浮点 A/D 采集转换电路的设计。浮点 A/D 转换技术能很好地采集到陀螺的微弱信号,对陀螺的大动态范围信号进行压缩。浮点 ADC 数据采集系统的大动态范围技术已在捷联惯性导航系统中陀螺仪和加速度计的数据采集系统中应用,有效地解决了信号的动态范围要求,可以获得高分辨率的陀螺和加速度计数据。

2.22 基于 PCI 高速数据采集系统

摘自《电子测量技术》双月刊,2001 年第 4 期

该文介绍一种基于 PCI 的高速数据采集系统。该系统采用了采样率达 100MHz 的 AD9054 芯片和读写周期为 10ns 的 FIFO 芯片,并利用高速的 ispLSI PLD 芯片实现采样窗和其他逻辑控制电路。数据的传输使用了符合 TIA/EIA-644 标准的低电压差分信号技术。数据通过 PCI 接口存储到计算机硬盘中,该 PCI 接口是利用现场可编程门阵列 FPGA 芯片实现的。文中给出了系统总体结构及工作原理,A/D 转换和采样控制电路,数据传输及 PCI 接口和数据接收电路。

2.23 一种基于 PC 机的高速 16 位并行数据采集接口

摘自《数据采集与处理》季刊,2000 年第 4 期

该文介绍了一种在 PC 机上实现的高速 16 位并行数据采集接口。该接口由高速光电隔离电路,双端口 FIFO 存储缓冲器电路及由 FPGA 芯片构成的计算机接口逻辑与控制电路等组成。该接口电路将终端显示处理系统与前端数据处理系统通过光电耦合器隔离开来,避免了它们之间的相互干扰,较好地解决了 16 位并行数据高速传输中存在的电磁干扰问题和大数据量实时有效传输问题。采用现场可编程门阵列 FPGA 芯片,使硬件设计软件化,既实现了复杂逻辑功能设计,又减少了硬件电路规模,提高了系统的可靠性,在雷达、声纳等复杂系统中具有很好的应用价值。

2.24 数据采集系统中增强型并行接口(EPP)电路的设计

摘自《电测与仪表》月刊,2001 年第 3 期

EPP(Enhanced Parallel Port)是 IEEE1284 协议定义的增强型并口,它的突出特点是能双向传送数据,且有接近于 PC 机 ISA 总线的数据传输率。本文介绍了有关 EPP 的基本特性、信号定义、操作模式及其程序设计方法,给出了利用 EPP 进行便携式高速数据采集原理及相应程序。

2.25 用增强型并行接口 EPP 协议扩展计算机的 ISA 接口

摘自《电子技术应用》月刊,2000 年第 11 期

利用微机开发便携式的数据采集、控制系统一直是微机应用系统开发者十分关心的课题。该文提出了一种用 EPP 协议和 CPLD 扩展笔记本电脑的 ISA 接口的方法,给出了详细的技术解决方案。扩展的 ISA 接口可以达到 1310 KBps 的双向通信速度,与 ISA 本身的速度相当。用户可以通过扩展的 ISA 接口直接使用市场上的基于 ISA 的接口卡。采用 EPP 扩展计算机的 ISA 接口具有非常高的性能价格比,能够达到绝大多数基于 ISA 接口的数据采集和控制系统的通信速度要求。这种方案大大扩展了笔记本电脑对于 ISA 接口设备的适应能力,省去了用户对扩展箱的需求。

2.26 基于增强型并行接口 EPP 的便携式高速数据采集系统

摘自《电子技术应用》月刊,2000 年第 12 期

增强型并行接口 EPP 不但与 SPP 兼容,而且其最高传输速率可达 ISA 总线的能力(2MHz)。由于便携式计算机日益普及,基于 EPP 协议开发的便携式微机采集系统将会是一个发展趋势。该文针对基于 EPP 协议的并行端口设备开发的特点与趋势,开发了由 A/D 转换器 AD1671 和 FIFO 存储器 ID7202 构成的 1.25MHz、12 bit 的高速数据采集系统,并通过 IDT7202 与 EPP 的接口电路实现了采集数据的高速回传。文中介绍了该采集系统工作原理,包括 AD1671 控制及采集系统工作原理,FIFO 与 EPP 的接口电路。

2.27 增强型并行接口 EPP 协议及其在 CAN 监控节点中的应用

摘自《电子技术应用》月刊,2000 年第 4 期

该文针对微机并行接口在 CAN 监控节点中的实际应用,详细介绍了微机并行口 EPP 协议和 CAN 监控节点的设计,给出了 EPP 协议并行口与 SJA 1000 及 82C250 连接电路图,介绍了监控节点的连接和软件的使用。

2.28 利用增强型并行接口协议传输图像文件

摘自《电子测量技术》月刊,2001 年第 4 期

该文提出了一种利用 EPP 协议传输图像文件的方法,给出详细的技术解决方案,包括传输数据系统硬件结构及并口发送数据的软件实现。经实际应用,基于 EPP 模式的传输文件速率最高可达 1.8Mbps。

2.29 用并行接口进行数据采集

摘自《电脑开发与应用》月刊,2000 年第 9 期

该文利用笔记本电脑的并口配合自制的数据采集接口卡进行数据采集,从而使得整个设备的体积不到两个笔记本电脑的大小。数据采集采用了多路 A/D 转换器 ADC0809,转换精度 > 0.4%,文中给出了数据采集接口卡电路图。

2.30 高信噪比的 VFC/DPLL 数据采集装置

摘自《电子测量技术》月刊,2001 年第 1 期

该文讨论了由 VFC 芯片鉴频,鉴相器(PFD)组成的数字锁相环路(DPLL)抑制噪声的原理和有关参数的选择和计算,探讨了由 DPLL 和微处理机结合组成的数据采集系统,信噪比改善的可能性。该装置 V/F 转换频率调制、数字锁相快速频率捕捉、微机实时计数积累平均及数字平滑解调,使系统信噪比得到肯定的改善,系统总信噪比为

$$SNIR = SNIR_{(V/F\text{调制})} \cdot SNIR_{(DPLL)} \cdot SNIR_{(\text{计数平均,平滑解调})}$$

总信噪比的改善可达 20dB 以上。

2.31 高精度数字式转速测量系统的研究

摘自《测控技术》月刊,2000 年第 5 期

该文介绍了一种利用单片机技术实现高精度数字式转速测量系统的方法。给出了测量系统的基本组成和系统的主要技术性能指标。采用单片机技术来实现转速的测量,可以提高转速测量的准确度,并且加快了采样的速率,具有较好的实时性。适用于高、低转速的测量,测量准确度与转速无关,因而具有较宽的应用范围和广阔的应用前景。

2.32 用单片机测量相位差的新方法

摘自《电脑开发与应用》月刊,2000 年第 7 期

该文提出了一种利用 AT89C52 单片机测量两正弦信号相位差的测量原理、给出了电路组成和工作波形图,并分析了引起误差的原因。该方法测量准确、稳定,与被测信号频率、计数信号时钟频率无关。

2.33 交流采样在电力系统中应用

摘自《电子测量技术》月刊,2001 年第 2 期

交流采样是指交流信号经过低通滤波除去无用的高频分量后,遵循采样定律,通过时间离散化后得到离散序列完成采样过程。该文介绍了交流采样如何应用于电网监测中。详细阐述了其工作原理,推导了电流,电压,有功功率,无功功率,功率因数等计算公式,并进行了详细的误差分析。最后介绍了应用实例以及典型电路,并指出了在设计各环节中应注意的要点。该交流采样装置,已形成一定的生产批量,正在国内一些供电部门可靠运行。

2.34 同步图形存储器 IS42G32256 的电源与应用

摘自《电子技术应用》月刊,2000 年第 1 期

IS42G32256 是高速 16 Mbit CMOS 同步图形存储器(SGRAM),性能优良,速度快,适合于图形工作站、显示卡、电视机顶盒、游戏卡、二维/三维图形处理等高性能、高带宽的应用场合。该文介绍其功能、特点、工作原理及其应用。SGRAM 以其高带宽、高速度、低功耗,特别是在图形处理方面的特点,在图像处理等场合已得到广泛应用。

2.35 IBM-PC 处理 10MHz 高速模拟信号的研究

摘自《电脑开发与应用》月刊,2000 年第 1 期

在许多场合,特别是航空、航天科研中,需要对瞬时高速模拟信号进行数据采集。本系统用普通微机 IBM-PC 配接高速 A/D 转换器 TRW TDC1007、D/A 转换器 TDC1006、高速缓冲器等组成的视频信号接收卡对信号进行采集与处理,取得了良好的效果。本系统已用于 8MHz CCD 的输出信号接收,解决了 CCD 用户信号处理的困难。本系统同样适用于其他高速信号采集的场合,具有一定的推广价值。文中给出了系统的基本结构,描述了系统工作原理及实验结果。

2.36 MCS-51 系列单片机存储容量扩展方法

摘自《电子测量技术》月刊,2001 年第 1 期

该文中提出了一种扩展超容量外部程序存储器(EPROM)和数据存储器(RAM)的方法——体选扩展法,阐述了体选扩展原理,给出了体选法扩展硬件电路的设计和软件设计。文中还给出了 512K 随机存储器(RAM)扩展原理图和转体程序流程图。

2.37 用单片机实现数字相位变换器的设计方法

摘自《电子技术应用》月刊,2001 年第 11 期

该文提出了一种单片机实现数字相位变换器(又称脉冲调相器)的设计方法。文中描述了数字相位变换器的工作原理,基于 AT89C2051 单片机的硬件电路的工作原理以及软件设计的基本思想和程序设计方法。文中还给出了脉冲调相器电路原理图和脉冲调相程序设计框图。

2.38 一种新的可重配置的串口扩展方案

摘自《计算机应用研究》月刊,2001 年第 12 期

现代大型、复杂系统中各微控制器之间通过串口进行信息交换是一种成熟且普遍采用的通信方式。传统的多机串行通信实时性、可靠性差;该文提出了一种能够大大提高系统通信实时性和可靠性串口扩展的新方案。文中给出了串口扩展方案的原理框图和系统状态迁移图。该方案硬件开销极小,且能提供实时性、可靠性高的串口通信能力,并已在实际系统中获得了成功应用。

2.39 VB 环境下对双端口 RAM 物理读写的实现

摘自《电子技术应用》月刊,2001 年第 3 期

该文介绍一种应用双端口 RAM 芯片设计的智能型高速并行通信卡。针对 VB 语言环境下,用 DLL 函数链接方式,对采用内存直接映象技术的双端口 RAM 进行读写,实现主、分机之间数据高速并行传输。文中给出了系统描述,硬件设计和软件设计。本文具有通用性,在自动控制领域设计 A/D、D/A 转换、实现闭环控制以及设计虚拟仪器、视频信号数字化处理、语音处理、各种系统动态仿真等需要高速数据传输的场合,都具有借鉴意义。

2.40 双 CPU 实现远程多键盘鼠标交互

摘自《微处理机》双月刊,2000 年第 1 期

该文提出一种利用双 89C2051CPU 实现远程键盘和鼠标的交互方法,给出了电原理图、通信协议和软件流程图。用双 CPU 实现远程多键盘鼠标交互,其硬件结构简单、通信稳定可靠、价格低廉,具有大的应用前景。但软件的工作量和难度都很大,调试困难,特别是通信的稳定性和可靠性方面需要作相当精确和周密的考虑。

2.41 两种电阻-时间变换器设计与分析

摘自《电子测量技术》双月刊,2001 年第 3 期

该文分别给出基于张弛振荡器原理以及脉宽调制原理的两种电阻-时间变换器电路。对电阻时间变换器线性误差分析表明,两种变换器在宽的电阻变化范围内具有良好的线性及稳定性,特别适合用于电阻式传感元件的信号处理。本文给出的两种电阻-时间变换器,具有电路设计简单,变换线性好,分辨率高,无需添加另外的补偿电路等特点。特别是当采用集成化设计后,有可能进一步减小温度漂移、失调电压的影响,也将大大提高电路的可靠性。

2.42 液晶显示器的接口和编程技巧

摘自《电测与仪表》双月刊,2001 年第 5 期

液晶显示器(LCD)具有体积小、外形薄、重量轻、耗能小、工作电压低、无辐射,特别是视域宽、显示信息量大等优点。该文介绍了 MGLS240128T 图形点阵式液晶显示器的内部结构及其与 MCS-51 单片机的接口电路,针对液晶显示器及其控制器的特点,详细阐述了汉字显示、清屏等功能的程序设计,并讨论了程序优化问题。

2.43 一种简单的电机变频调速方案及其应用

摘自《自动驾驶仪与红外技术》2000 年第 4 期

该文提供了一种实用的电机变频调速方案,控制部分采用单片机控制专用 PWM 波形发生器,功率部分采用 IPM 智能模块,扩展了一片 ADC0809 负责信号采集,单片机负责信号处理和过程控制。文中给出了方案的基本硬件配置,软件流程图。本方案已应用在新型红外探边仪中,试验效果良好。

2.44 基于单片机的火控系统符号产生器电路原理设计

摘自《计算机工程》月刊,2000 年第 7 期

符号产生器用于产生瞄准符号,经光学系统使驾驶员观察到一组与外界背景重叠在一起的动态符号图像,是完成平面显示器 HUD 状态画面显示的关键部件之一。该文给出了符号产生器电路及其与单片机的接口设计,并利用示波器 CRT 作为显示屏,研制了简易的 HUD 地面模拟器,用于教学和实验,画面逼真,效果很好。

2.45 A/D 转换器性能的改善方法

摘自《电脑开发与应用》月刊,2000 年第 5 期

怎样使用经济可行的办法提高 8 位 A/D 转换器的性能,本文提出了两个改善方法:

1 采用程控放大器提高 A/D 转换器性能

2 单信号多通道输入法提高 A/D 转换器性能,并给出了单信号多通道分段输入电路和转换驱动子程序。

2.46 快速小波变换算法与信噪分离

摘自《电子测量技术》月刊,2001 年第 1 期

该文对小波理论和快速小波变换算法进行了分析,着重阐述了信号的小波分解与重构过程的实现。文中对快速小波变换算法进行信噪分离的原理和步骤进行了研究,并以一个具体的实例展示了该算法应用到机械振动测试信号的信噪分离中的效果。快速小波变换算法应用到信噪分离,不但能消除噪声,而且保持信号的阶跃突变点的位置不变,保证了信号原有的绝大多数重要信息与特征不丢失,该算法的卓越性能显示了广阔的应用前景。

2.47 80C196MC/MD 单片机多个中断程序的同步问题

摘自《微处理机》季刊,2000 年第 3 期

8XC196MC/MD 具有指令丰富、外设完善、功能强大等特点,其外设间的协同工作,在应用中十分必要。本文介绍了其外设中断服务程序间的同步控制和程序设计应注意的问题,并给出了中断过程的同步处理程序设计实例。文中所有程序都在实际电机调速系统中调试通过,经验证可以正常工作。

三、操作系统及软件技术

3.1 嵌入式软件技术的现状与发展动向

摘自《计算机应用》月刊,2000 年第 7 期

该文介绍了一类标志性嵌入式设备的技术特征,分析了嵌入式软件技术面临的挑战和发展动向,最后,提出了发展我国嵌入式软件技术的一些设想。有关嵌入式软件技术面临挑战,文中列出的有:

- 对应用开发提供强大支持;
- 为设备网络通信提供标准接口;
- 支持小型电子设备实现小尺寸、微功耗和低成本;
- 提供精巧的多媒体人机界面;

有关影响未来的若干软件新技术,文中提出的有:

- 日趋流行的行业性开放系统和备受青睐的自由软件技术;
- 无线网络操作系统初见端倪;
- IP 构件库技术正在造就一个新兴的软件行业;
- JAVA 编程和平台技术将对嵌入式软件的发展产生深远影响。

3.2 什么是嵌入式实时操作系统

摘自《单片机与嵌入式系统应用》月刊,2001 年第 7 期

实时操作系统(RTOS)是开发计算机嵌入式应用产品的有力工具,研究和掌握 RTOS 的思想方法有益于提高开发人员的水平,从而缩短产品开发周期、提高产品质量。该文是《嵌入式 RTOS 讲座》之一,文中包括:

1. 计算机操作系统的发展历史和概念;
2. 什么是实时操作系统;
3. 嵌入式实时操作系统;
4. 嵌入式应用中使用 RTOS 的必要性;
5. 使用实时内核的优缺点;
6. 我国计算机嵌入式应用开发的特点。

3.3 实时多任务系统中的一些基本概念

摘自《单片机与嵌入式系统应用》月刊,2001 年第 8 期

实时操作系统(RTOS)是开发计算机嵌入式应用产品的有力工具,研究和掌握 RTOS 的思想方法有益于提高开发人员的水平,从而缩短产品开发周期、提高产品质量。该文是《嵌入式 RTOS 讲座》之一,文中阐述了下列问题和概念:

1. RTOS 系统的必要性;

2 .任务与多任务;
3 .任务切换与调度;
4 .嵌入式实时多任务系统;
5 .实时内核(the real time kernel) ;
6 .任务优先级分配;
7 .优先级反转问题;
8 .任务间的竞争;
9 .死锁;
10 .可重入性;
11 .时钟节拍;
12 .信号与信号量(semaphore)。

3 .4 一个源码公开的实时内核

摘自《单片机与嵌入式系统应用》月刊,2001 年第 9 期

实时操作系统(RTOS)是开发计算机嵌入式应用产品的有力工具,研究和掌握 RTOS 的思想方法有益于提高开发人员的水平,从而缩短产品开发周期、提高产品质量。该文是《嵌入式 RTOS 讲座》之一。文中介绍了一个免费的、源码公开的、有详尽注解的实时内核 μC/ OS,μC/ OS 的应用已覆盖了诸多的领域,如照相机业、医疗器械、音响设施、发动机控制、网络设备、高速公路电话系统、自动提款机、工业机器人等。文中介绍了 μC/ OS-II 的特点和 μC/ OS 的任务调度机制。提出要掌握和拥有一个属于自己的实时内核,并介绍了嵌入式实时操作系统的未来。

3 .5 Windows CE 的实时性分析

摘自《测控技术》月刊,2000 年第 1 期

该文介绍了 Windows CE 2 .0 的特点,对其实时性问题进行了分析,并给出了衡量系统实时性强弱的操作系统时间延迟的测量结果。Windows CE 最突出的优点在于模块化的设计以及它的 API 基于 W32 API。另外,从操作系统的体系结构来看,Windows CE 通过其模块化特点很容易升级,而且很容易获得第三方的软件支持。Windows CE 的确适合于设计简单、功能比较单一的小型实时控制系统,但不适合应用在中型或大型的硬实时嵌入式系统中。对 Windows CE2 .0 需要在中断嵌套、优先级数目和 IST 延迟等方面作进一步的改进,才能在嵌入式实时操作系统中有所作为。

3 .6 串口通信多线程实现的分析

摘自《计算机应用研究》月刊,2001 年第 11 期

该文分析了多线程实现串口通信的机制。通过 UML 活动图,详细展示了串口通信中各个线程并发执行的过程,并讨论了多线程同步问题。通过两个通信线程并发执行,加以同步管理,应用程序能够在发送数据的同时接收数据,实时响应性强,高效可靠,有效地避免了数据丢失、程序锁定等问题。

3.7 基于中间件的开发研究

摘自《计算机应用研究》月刊,2001 年第 8 期

中间件是为解决分布异构问题而提出的,它是位于平台(硬件和操作系统)和应用之间的通用服务,这些服务具有标准的程序接口和协议。中间件的使用,将使在异构环境中进行开发和应用变得更方便和容易。该文论述了中间件技术,探讨了基于 C++ 的 CORBA 的一般开发方法。尽管在开发过程中还存在许多细节和困难需要探讨,但是这方面的研究无疑是有意义的。

3.8 Windows 95 下实时控制软件设计的研究

摘自《测控技术》月刊,2000 年第 12 期

该文分析了中断的原理与延迟以及系统响应的实时特性,介绍了 Windows 95 下对硬件 I/O 端口和内存物理地址的访问原理,分析了 DMA 编程的特点。了解和掌握这些特性,就能在系统对实时性要求不高时,利用 Windows 95 操作平台设计控制系统的软件。文中所述技术在实际的振动控制系统的设计中得到应用并取得很好的效果。

3.9 Windows NT 4.0 下设备驱动程序的开发与应用

摘自《电子技术应用》月刊,2000 年第 9 期

Windows NT 以其安全、稳定及界面友好等特性正逐渐成为工业控制领域的前台操作系统。该文介绍了 Windows NT4.0 内核模式设备驱动程序开发中的一般性过程。文中通过一个最小化驱动程序的核心代码,解释各组成部分的结构功能和使用方法。用户可利用 NT SDK 及 DDK 开发工具包,并根据自身需要,对以上核心代码进行扩充完成各自所需任务。

3.10 Windows 98 下硬件中断驱动程序的开发

摘自《测控技术》月刊,2000 年第 3 期

该文介绍了 Windows 98 的内核管理机制和应用程序权限级别,简述了在 Windows 98 下进行虚拟驱动程序开发的几种工具和编程方法,并给出了借助 VtoolsD 用 C++ 语言编写的处理硬件中断的程序实例。文中提到的虚拟设备驱动程序的开发工具和基本编程方法包括:微软提供的设备驱动程序工具箱(DDK),美国 Vireo Software 公司推出的 VtoolsD 为开发 VxD 提供的方便快捷方法。VxD 作为现在流行的编程技术已逐渐受到广泛关注。

3.11 Windows 下实时数据采集的实现

摘自《计算机全国研究》月刊,2001 年第 3 期

该文介绍在 Windows 的实时数据采集程序设计中,如何利用多媒体定时器来代替普通的定时器,获得时间上的高精确度。使用多线程技术,避免实时采集与其他数据处理可能会产生的时间上的冲突,进一步保证数据采集的实时性。用 Windows 的消息机制来协调各线程的运行,避免使用循环查询方式,使程序具有更高的运行效率和更好的协调性。从而解决 Windows 下的包括数据采集在内的多种实时性问题。

3.12 Win 95 下虚拟设备驱动程序设计开发

摘自《电子技术应用》月刊,2000 年第 4 期

该文介绍 VxDs 与 Win 32 应用程序通信的几种常用方法,并给出了用 VtoolsD 开发 VxDs 的具体实例。目前应用广泛的 VxD 开发工具主要有两大类,一类是 Microsoft 提供的对应于不同版本的 Windows 的 DDK(Device Driver Kit),其中包含了开发 VxD 所需的各种类库及汇编工具等。另一类是 Vireo Software 提供的 VtoolsD,比较而言,使用 VtoolsD 要比使用 DDK 简单、快捷。

3.13 Win 95 环境下测控软件中端口读写的快速实现

摘自《测控技术》月刊,2000 年第 3 期

编写测控软件时会涉及端口读写,但 Windows 中的设备驱动程序编写复杂。本文介绍了一种在 Windows95/ 98 下无需写设备驱动程序而快速实现端口读写的简便方法。文中介绍了实现的原则,并给出了端口读写的一个实例,本方法简单可行,无需编写复杂的设备驱动程序即可实现对端口的读写,对在资源丰富、操作友好的 Win 95 下编制新的控制程序和移植 DOS 程序都不失为一种方便的途径。

3.14 Linux 系统中 ARP 的编程实现技术

摘自《计算机应用研究》月刊,2001 年第 4 期

该文简要介绍了 TCP/ IP 互联网中的地址解析协议(ARP)以及 Linux 系统提供的基于低层的应用程序开发接口(API),重点分析了三种访问数据链路层方法,即(1)PF_INET 协议族中 SOCK_PACKET 类型的套接字;(2)PF_PACKET 协议族中 SOCK_RAW 类型的套接字;(3)PF_PACKET 协议族中 SOCK_DGRAM 类型的套接字。利用 API 接口,用 C 语言开发了 ARP 协议应用程序。描述了具体编程步骤,并提供了相应的系统调用及程序分析。文中提出,第一种访问数据链路层方法在新的 Linux 版本中已被后两种方法代替,但仍得到支持。可以看出,Linux 提供的基于数据链路层的应用程序开发具有高度的编程灵活性,用户可以自定义数据报文的格式,有利于信息保密,这在一些特殊的应用领域具有重大意义。

3.15 Linux 中 System V 进程通信机制及访问控制技术的改进

摘自《计算机应用研究》月刊,2001 年第 4 期

该文简述了消息队列、共享内存和信号量这三种通信机制的结构及其实现,并以此为基础,提出了通过实现进程间通信的强制访问控制来提高系统安全性的方法。本文提出了两种方案,并简要介绍了它们的优缺点。

3.16 VC++6.0 中动态创建 MSComm 控件的问题及对策

摘自《电测与仪表》月刊,2001 年第 12 期

该文讨论了在使用 VC++6.0 设计 Windows 监控软件时,静态和动态创建 MSComm 控件的原理和方法。分析了动态创建时的许可码问题,并给出了完备的解决方案。本文具有较强的实用性和针对性,动态创建 MSComm 控件的原理和方法已成功应用于几种测量仪器的

上位计算机软件的开发中。整套软件使用方便,性能稳定,非常符合便携式仪器配套软件的特点。

3.17 在 Visual Basic 下使用 I/O 接口程序

摘自《测控技术》月刊,2000 年第 7 期

该文介绍了如何利用 Visual C++ 指令编制直接访问 I/O 接口地址以及处理字节数组合的各位功能的函数,通过动态链接库方式在 Visual Basic 中应用。文中给出了实现方法和应用实例。在 Visual Basic 中能够很方便地完成对 I/O 端口地址的访问,合理利用 Visual Basic 语言强大功能,就可以很好地解决应用中的实际问题。

3.18 VB 应用程序速度的优化技术

摘自《计算机应用研究》月刊,2001 年第 1 期

在用户评测应用程序性能的诸多指标中,速度是决定性的因素。该文介绍了 Visual Basic 应用程序速度的优化技术,包括真实速度优化、显示速度优化、感觉速度优化,以及编译时有利于提高速度的优化选项等。文中提出,优化并不是在所有的情况下都是有益的,所以选择优化的具体类型有赖于应用程序的性质及其目的。要选择最受用户关注的地方进行优化,有关优化的一个普遍存在的误解是,认为只是在应用程序开发周期的最后阶段(即编译时)才进行优化,而实际上,为了创建真正优化的应用程序就必须在开发时就实行优化。

3.19 嵌入式实时操作系统在机车微机测控软件开发中的应用

摘自《测控技术》月刊,2000 年第 6 期

本文以嵌入式实时多任务操作系统 VRTX32 为例,介绍其在机车微机测控软件开发中的应用。重点论述了实时多任务软件开发中板支持程序(BSP)的结构、原理以及用户任务的划分与驱动方法。由于多任务机制下的程序结构简单、模块化程度高,软件的调试、维护与扩充都十分方便,因此可以相信,随着嵌入式实时多任务操作系统性能的不断提高(例如:可裁减 RTOS 的出现)以及与 RTOS 配套的集成开发、调试环境的出现,基于 RTOS 的嵌入式微机测控系统的软件开发将会更加简单、方便。

3.20 结构化程序方法在汇编语言中的应用

摘自《电脑开发与应用》月刊, 2000 年第 2 期

本文介绍设计选择结构程序、循环结构程序、过程子程序及其他用于结构化程序设计的一些语句及用于多条件判断的逻辑表达式。利用这些语句可设计具有高级语言结构风格的结构化汇编语言程序。

3.21 AVR 单片机编程特性的应用研究

摘自《微型机与应用》月刊,2000 年第 9 期

AVR(Advanced RISC)是美国 ATMEL 公司生产的增强 RISC 内载 Flash 的高性能 8 位单片机,它用 1 个时钟周期执行 1 条指令,具有良好的性能价格比。该文结合在制作 AVR 单片机串行编程器及设计可下载程序用户板过程中的实际经验,指出了经常遇到的一些问题,并

给出了相应的解决办法。AVR 单片机的 MTP 及 ISP 技术为用户提供了极大的便利,因而对 AVR 单片机编程特性的研究及串行编程器的制作具有实用价值。

3.22 一种有效的 51 系列单片机软件仿真器

摘自《电子世界》月刊,2001 年第 6 期

单片机仿真器分为硬件,软硬结合及纯软件仿真器。其中纯软件仿真器是指完全用软件实现对单片机的硬件模拟、指令模拟、运行状态模拟,从而完成应用软件开发。该文介绍 Kei 公司开发的 uVision for Windows,它是一个集成软件开发平台,集成了 C51 编译器、A51 汇编器、L51 连接器、实时操作系统、项目管理器、调试器等,可以完成编辑、编译、连接、调试、仿真等整个开发流程,可以用 IDE 编辑 C 或汇编源文件。文中还介绍了一些应用实例。

3.23 PIC 单片机软件模拟仿真时输入信号的激励方式

摘自《电子世界》月刊,2001 年第 2 期

该文介绍微芯公司软件开发平台 MPLAB IDE 内含的软仿真工具 MPLAB SIM,利用它可以在 PC 操作系统 Windows 平台上模拟 PIC 单片机内部的程序运行和引脚的输入/ 输出,以便完成软件的调试工作。文中介绍了 MPLAB SIM 提供异步信号激励(Asynchronous Stimulus)、引脚连续激励(Pin Stimulus)、时钟激励(Clock Stimulus)和寄存器内容激励(Rigister Stimulus)四种信号激励的方式。

3.24 基于 LabVIEW 的分布式 VXI 仪器教学实验系统设计

摘自《仪表技术》月刊,2001 年第 2 期

随着仪器与测试技术的发展,VXI 仪器正进入大学实验室。文中介绍利用先进的 LabVIEW5.1 开发环境设计完成的一个分布式 VXI 仪器网络测量系统。定义了 VXI 网络测量模型及传输报文格式,给出了系统软件的总体结构。分布式 VXI 仪器教学试验网络系统的优点主要是系统结构具有极强的开放性,用户可根据实际需要在客户机上迅速开发功能强大的测试应用。实际应用结果表明,采用分布共享式系统结构是开展 VXI 教学实验的最佳选择。

四、网络、通信及数据传输

4.1 单片机网络的组成与控制

摘自《计算机自动测量与控制》月刊,2000 年第 1 期

该文依据单片机网络的基本构想,构造了一个实用的微机控制下的单片机网络,内容主要包括:网络管理器,网络通信等,并给出了应用实例。

4.2 实现 ARINC 429 数字信息传输的方案设计

摘自《电子技术应用》月刊,2000 年第 6 期

ARINC 429 数字信息传输规范(DITS)33 为在航空电子设备之间传输数字信息制定了航空运输工业标准,该文以 ARINC 429 数字信息传输规范为基础,分别以 8031、80C196 及 TMS320F206 为核心设计了三种实现 ARIN 429 通信的系统方案,同时对三种系统的特点及适用的通信速率范围做了详细说明。从三种不同设计方案中可以看到,在不同的功能要求条件下,系统核心器件的选择存在着差别。

4.3 结合电力线载波和电话通信的报警网络系统

摘自《电子技术应用》月刊,2000 年第 11 期

该文介绍一种报警系统,其采用电力线载波和电话通信相结合的通信方式,兼有二者优点,与其他报警网络系统比较,不仅功能齐全,成本低,系统组成灵活,对所控区域大小选择性强,而且适于小区和报警指挥中心的保安管理,是一种新型的报警网络系统。文中给出了系统总结构简图,主流程图,巡检流程图和报警处理流程图,并对用户端报警装置和报警指挥中心分别进行了介绍。这种新型报警网络系统,充分结合电话和电力线载波两种通信方式的优点,弥补了二者在报警系统通信中的不足。

4.4 网络电子密码锁监控系统的设计与实现

摘自《测控技术》月刊,2000 年第 11 期

密码电子锁网络监控系统能够自动完成对联网的电子密码锁进行实时监控、动态管理和报警处理等功能,满足对安全性、可靠性和易于管理等性能的要求。该文详细介绍了密码电子锁监控系统的整体结构、硬件设计、系统通信网络方案以及系统软件的开发和具体实现。文中给出了分布式电子密码锁网络监控系统的多机通信结构,电子密码锁控制器电路结构及监控系统软件设计。

这种监控系统的硬件和软件采用标准化、模块化和系统化的设计,系统的配置具有通用性强、系统组态灵活、控制功能完善、数据处理方便、显示操作集中、人机界面友好以及系统安装、调试和维修简便等特点,适合于宾馆酒店的客房管理、单位智能化大厦、居民住宅小区等的安全保密、自动控制、防盗报警和现代管理应用。

4.5 IRIG-E 标准 FM-FM 解调器的有关技术

摘自《电子技术应用》月刊,2000 年第 1 期

该文介绍了遥测中的 FM 技术和 IRIG 标准,讨论了 IRIG-E 标准 FM-FM 调频解调器的原理。文中主要对其中起关键作用的滤波器设计作了比较详细的说明,主要是分路滤波器和输出低通滤波器,同时也介绍了有关的一些产品和设计思路。

4.6 基于 TCP/IP 的多媒体通信实现

摘自《计算机应用研究》月刊,2001 年第 4 期

该文提出了一种数据流方式下实现多媒体组播的方法。在 WindowsNT 平台上,用 Visual C++6.0 编程实现了基于 TCP/IP 的数据流传输方式下的端-端通信和多媒体组播。首先实现了端-端通信的客户机/服务器模式,在此基础上实现了端-端通信的端-端对等模式。而后针对 TCP/IP 数据流方式下不能进行组播的缺点,本文提出控制帧法实现多媒体组播功能,并有效地减小了通信占用的带宽。

4.7 基于 TCP/IP 的多线程通信及其在远程监控系统中的应用

摘自《电子技术应用》月刊,2000 年第 1 期

该文提出了一种在 Windows NT 下基于 TCP/IP 协议的多线程通信的设计与实现方法,给出了多线程通信在蓄电池远程监控系统中的应用实例。根据本文给出的在 Windows NT 下基于 TCP/IP 协议的多线程通信的基本方法,进行修改和扩充,便可设计出符合具体应用的高质量的多线程通信程序。

4.8 基于 Internet 的远程测控技术

摘自《国外电子测量技术》双月刊,2001 年第 6 期

该文介绍了基于 Internet 远程控制系统的组成,详细分析了远程测控技术的现状,最后介绍了基于 Internet 远程测控系统的应用和前景。内容主要包括基于 Internet 远程测控系统的构成,系统实现的关键技术包括网络数据库技术,现场总线技术,远程监控网络中的延时处理技术,网络化仪器仪表技术,故障诊断技术,网络安全技术以及基于 Internet 远程测控的应用与前景。

4.9 Windows 95 串行通信的几种方式及编程

摘自《微型机与应用》月刊,2000 年第 3 期

该文详细介绍 Windows 95 平台下串行口通信的 4 种方式和部分程序代码,并对应用场合提出建议。四种通信方式为:同步方式、查询方式、异步方式和事件驱动方式。

4.10 在 Windows 95 下 PC 机和单片机的串行通信

摘自《微型机与应用》月刊,2000 年第 1 期

该文实现了基于 32 位操作系统 Windows 95 的工作,控制 PC 机和单片机间的串行通信,给出了用 VC++5.0 和 PL/M-96 语言编写的串行通信程序。包括:主控机的 VC++5.0 串

行通信程序,下位机的 PL/ M-96 串行通信程序,VC + +5 .0 程序和 PL/ M-96 程序。

4 .11 基于 80C196KC 微处理器的高速串行通信

摘自《电子技术应用》月刊,2000 年第 1 期

该文讨论了基于 80C196KC 微处理器串行通信口的硬件设计、波特率设置和软件开发的技术途径。文中简要介绍了串行通信在某型雷达仿真平台中的应用,其主要性能和可靠性达到了系统的设计要求。内容包括:串行口的通信模式、串行口控制、串行通信波特率的计算、串行通信的软件设计及串行通信应注意的几个问题。

4 .12 使用 PC 机并行口与下位单片机通信的方法

摘自《计算机工程》月刊,2000 年第 6 期

在一些应用系统设计中,PC 机与下位单片机间互传数据常常采用串行异步通信方式,其硬件电气连接简单,编程方便,但是受传输速率的限制。为了解决这个问题,同时又不在 PC 上增加设备,系统利用了 PC 机并口作为通信接口。编程环境采用了广泛采用的 VB,但是 VB 自身的接口不能直接访问并口寄存器,所以利用 DLL 完成了这一功能。文中介绍了 PC 机并口与下位机通信的一种硬件连接方法以及 PC 机上 VB6 .0 环境中利用 DLL 解决对并口寄存器直接访问的方法。文中给出了 8255 的扩展电路,PC 机向下位机传数据和下位机向 PC 机传数据的时序及通信软件。

4 .13 双向并口通信的开发

摘自《电脑开发与应用》月刊,2000 年第 3 期

该文着重从逻辑界面、连接和时序图三方面讨论了双向并行口(PS/ 2)、增强型并口(EPP)和扩展并行口(ECP)。可以看出,微机的并行口已经由原来只具有单纯的打印功能发展成为微机与外设双向交换数据的比较可靠的接口。在实际应用中,选定并行口的哪种模式还要根据具体情况而定。

4 .14 DSP 和计算机并口的高速数据通信

摘自《国外电子测量技术》双月刊,2001 年第 4 期

计算机并口已经从标准并行口(SPP)模式发展到了 EPP 和 ECP 模式。该文提出了一种用 EPP 协议和 EPLD 实现 DSP 与计算机并口的双向高速数据传输方法,给出了详细的技术解决方案。利用 EPLD 可方便实现 EPP 模式,再加上 FIFO 的使用,可实现 DSP 和计算机并口之间的双向高速数据传输,并且较少占用 CPU 时间。该电路设计已成功地应用到某测试模块的信号处理系统中,在测试仪器、计算机外设等领域有较好的应用前景。

4 .15 一种高可靠性的 PC 机与单片机间的串行通信方法

摘自《电气自动化》双月刊,2000 年第 3 期

该文提出了一种高可靠性的 PC 机与 80C196KC 单片机之间异步串行通信的方法——握手通信约定法。详细讨论了实现这一通信方法的硬件接口电路和软件设计,并给出了实例。这种通信方法,系统的硬件电路简单,软件编程可靠,并在全数字交流伺服驱动系统中得到了

很好的应用。握手通信约定法也适合于 PC 机与 PC 机之间,PC 机与大多数单片机之间的串行通信,具有推广价值。

4.16 单片机与 PC 机串行通信的实现方法

摘自《电子世界》月刊,2001 年第 10 期

该文主要介绍 PC 机与 51 系列单片机实现通信的一般方法和步骤。文中给出了 RS-232 电平转换和 PC 机的接口电路,单片机串口工作方式,单片机串口的控制方式,单片机串口的速率设置以及单片机串口通信程序的实现方法。

4.17 89C51 单片机 I/O 口模拟串行通信的实现方法

摘自《电子世界》月刊,2001 年第 2 期

该文介绍一种用单片机普通 I/O 口实现串行通信的方法,可在单片机的最小应用系统中实现与两个以上串行接口设备的多机通信。文中给出了实现通信的硬件接口电路,接口程序设计。

4.18 TMS320C50 与 PC 机高速串行通信的实现

摘自《计算机自动测量与控制》月刊,2000 年第 5 期

文中提出一种 DSP TMS320C50 与 PC 机的高速通信电路。文中给出了 TMS320C50 与 PC 机通信的硬件电路,软件设计包括 PC 机,TMS320C60 及 TL16C750 的初始化和通信协议等。通过扩展串口完成 TMS320C50 与 PC 机通信,具有硬件接口简单,数据传送距离远及使用经济的特点。通信实践证明,在波特率为 38400 下可实现可靠通信。

4.19 DSP 和 PC 机的异步串行通信设计

摘自《电测与仪表》月刊,2001 年第 3 期

该文介绍 DSP 系统中数字信号处理最小系统的基本框图,DSP 与上位机的通信,并提供了解决 DSP 与上位机异步串行通信的软硬件解决方案。文中给出了 UART 硬件详图,DSP 对 UART 的初始化,PC 机通信程序设计。目前此 DSP 系统已应用于 500kV 光纤电压电流传感器系统中,运行稳定,通信性能良好。

4.20 基于 MCS 单片机与 PC 机串行通信电平转换

摘自《电子测量技术》月刊,2001 年第 4 期

该文介绍了 MCS 单片机与 PC 机串行通信中 TTL 与 RS-232C 电平转换的三种方式:采用分立元件实现电平转换;标准 RS-232C 电平转换以及利用 MAX3232 芯片实现电平转换。给出了各自的实现方法并进行了简单比较。

4.21 一种简单的光电隔离 RS-232 电平转换接口设计

摘自《电测与仪表》月刊,2001 年第 1 期

该文提出了一种简单实用的光电隔离 RS-232 电平转换接口设计方法。文中给出一个 MCS-51 单片机与 PC 机通信的硬件连接电路、完整的通信协议和软件设计流程。这种转换

电路,由光电耦合器直接完成电平转换。不需要通用的 RS－232 电平转换器,在 RS－232 标准的有效距离内可实现高达 4800 波特率的可靠通信。

4.22 ISA 总线工业控制机与单片机系统的数据交换

摘自《电气自动化》双月刊,2000 年第 5 期

ISA 总线工业控制机与单片机之间有效的数据交换是实现主从系统的关键。该文在比较各种主要通信方式的基础上,着重讨论了单片机系统与主机之间通过 ISA 总线交换数据的两种实现方法:(1)共用单片机系统外部数据存储器法。(2)静态数据传送方法。分别给出了共用数据存储器传递数据和使用 8255 实现静态传送的结构框图。

作者认为,共用单片机数据存储器可以实现使用少量端口进行大量数据的成组传送,具有较高的速度,但其电路复杂,使用时需注意信号的隔离。使用 8255 构成的静态传送电路结构非常简单,只占用少量资源,扩展容易,并且具有非常灵活的使用方式,是一种廉价有效的方法,适用于各种单片机构成的智能接口卡。

4.23 RS－232/422/485 综合接口

摘自《电子世界》月刊,2001 年第 6 期

如果在设计通信设备时,将 RS－232/422/485 接口综合起来考虑,做成一种多用接口,必将大大方便使用,并且几乎不增加成本。本文提出一种 RS－232/422/485 异步串行通信综合接口。文中给出了异步串行通信综合接口电原理图,分别介绍了 RS－232 接口,RS－422 接口,RS－485 全双工接口,RS－485 半双工接口,CONTROL 信号的形成以及接地问题。

4.24 基于 RS－485 接口的单片机串行通信

摘自《兵工自动化》双月刊,2000 年第 4 期

该文介绍了 MAX483E RS－485 收发器的特点,阐述了用 MAX483E RS－485 收发器实现单片机系统双机及多机串行通信的方法及技巧,给出了用 MAX483E 构成的双机和多机串行通信原理框图以及 MAX483E RS－45 收发器构成的半双工串行通信时序配合应遵守的原则。

这种 MAX483E RS－485 收发器构成的串行通信已成功地应用于某精密实时测控系统中,串行通信的数据稳定、可靠,满足系统要求。

4.25 在 VC++中利用 ActiveX 控件开发串行通信程序

摘自《电子技术应用》月刊,2000 年第 6 期

该文探讨使用 Visual C++编程,利用 Microsoft Communications Control 控件编写串行通信程序的方法,并给出了例程,具有一定的实用意义。这一探讨也显示了 ActiveX 技术的强大功能、充分的灵活性和易用性。

4.26 上位机和多台下位机的 485 通信

摘自《自动化技术与应用》双月刊,2001 年第 6 期

在自动控制领域中,经常要实现上位机和下位机的通信,有时甚至要求上位机和多台下位

机建立通信。该文给出了实现上位机与多台下位机 485 通信实际常用的 VB6 .0 和 C 语言的编程方法,文中针对实际系统给出了通信协议和上下位机程序实例。

4 .27 计算机与 CAN 通信的一种方法

摘自《测控技术》月刊,2000 年第 7 期

为实现计算机与 CAN 之间的通信,该文设计了一种具有 RS-232 和 CAN 两种接口的通信模块,该模块应用单片机技术实现了两者之间的"透明"双向通信,并可以同时连接于两个不同的 CAN 网络。文中详细阐述了模块的软硬件设计。包括:RS-232—CAN 通信转换模块硬件简图,计算机一侧的软件设计,通信转换模块一侧的软件设计。本文为计算机与 CAN 之间的直接通信提供了一种有效的方法。

4 .28 用 VB 语言实现对端口 I/O 的访问

摘自《微型机与应用》月刊,2000 年第 12 期

该文介绍了 Visual Basic 通过动态链接库(DLL)实现对端口 I/O 的二种访问的方法,即用 VC 语言和汇编语言实现和用现成的软件包实现的方法。

4 .29 异种单片机共享片外存储器及其与微机通信的方法

摘自《电子技术应用》月刊,2001 年第 1 期

该文介绍了 AT89C51 与(DSP)TMS320C32 通过共享片外存储器实现板间通信的方法,给出了总线隔离硬件电路与软件控制流程。文中还简要分析了 C51 与微机进行串行通信的软硬件设计,通过扩展 AT89C51 间接实现了 TMS320C32 与通用单片机或微机之间的通信。文中分别给出了信号处理板共享存储器硬件接口原理图和数据采集板共享存储器硬件接口原理图,信号处理板上的 AT89C51 与控制系统的串口通信电路以及通信软件编程。

4 .30 单片机与 MODEM 接口技术及其在智能仪器中的应用研究

摘自《工业控制计算机》月刊,2000 年第 2 期

该文以 89C51 为例介绍单片机与 MODEM 的硬件接口与编程技术,文中还给出该技术在无人值守变电站智能故障录波器中的应用实例,包括:89C51 与 MODEM 接口原理图和接口软件的设计。本文提及的方案可以用于其他领域,尤其对自动化仪器的有关信息远传具有一定的实用性。

4 .31 采用 MCS-51 单片机实现 CPFSK 调制

摘自《电子技术应用》月刊,2001 年第 9 期

用模拟电台传输数字信号时,调制方法多采用频移键控(FSK)。该文介绍一种用 MCS-51 单片机直接产生 FSK 信号的方法。此法首先采用 PWM 调制方法产生正弦波信号,再根据串行口输出的高、低电平产生 FSK 信号。文中分别介绍了正弦波的调制,信号输出,CPFSK 调制及波形的优化。该调制方式已在全国许多地区的水情自动测报系统中应用,运行结果理想。

4.32 一种新型编码芯片及其驱动程序的设计方案

摘自《电子技术应用》月刊,2000 年第 5 期

DVxpert-II 是 C-Cube 公司新开发的视频处理单芯片,该芯片能实现视频编码、解码和编/解码三种功能,可实现运动补偿、分块/离散余弦变换压缩算法。该文介绍了 Dvxper-II 的内部结构、功能及其特点。文中提出了将 DVxpert-II 集成于一块 PCI 插卡上的设计方案,并论述了编码器虚拟驱动程序的实现方法。DVxpert-II 提供的 PCI 接口和编程接口使其可方便地集成到 PC 插卡上,而且可由驱动程序对其进行较灵活的控制。目前该 VxD 已付诸使用,编码器设备工作良好。

4.33 DTMF 远程通信的软硬件实现技术

摘自《电子技术应用》月刊,2000 年第 5 期

远程通信除通常采用 MODEM 外,还可以考虑一种 DTMF 通信方式。该文详述了采用 DTMF 技术实现远程通信的软硬件关键技术。硬件实现技术描述了通信接口电路设计、发码电路设计、收码电路设计、450Hz 信号检测电路和振铃与防盗检测电路等。软件实现技术中给出了从机发起通信程序框图、从机应答主机程序框图、数据通信程序框图。

本套软硬件实现技术具有接口电路简单、可靠性高、成本低、灵活性强等优点。适用于数据通信量不大,速率要求不高的远程通信场合。可应用于远程分布式数据采集系统、家用自动防盗报警装置、远程室内监控系统以及公话集中管理系统等。

4.34 采用 DTMF 方式通信的电度表管理系统

摘自《仪表技术》双月刊,2001 年第 1 期

该文提出了一种采用 DTMF 信号方式通信的电度表管理系统,给出了控制分机及用户电表数字化接口的电路构成,重点分析了 DTMF 数据收发电路的工作原理及通信软件的设计方法。文中给出了系统框图、控制分机框图、数字化电度表电路、DTMF 发送电路、DTMF 接收电路及通信程序流程图。

本系统能准确抄录用户电表数据,并及时传送给电力部门计算机管理系统,对控制分机和用户电表功能进行扩展后,可对变压器低压侧电流、电压及功率因数、变压器及电缆温度等参数进行综合监测,对各路负荷进行合理均衡,使供电更科学、安全、可靠。

4.35 基于 TAPI 的电话语音系统设计方法

摘自《微计算机应用研究》月刊,2001 年第 9 期

电话语音系统是一种集成电话和计算机技术的应用系统。大型电话语音系统又称为计算机电话集成,它可以同时支持多路电话接入,为多个电话用户进行服务。本文介绍了一种使用调制解调器基于电话应用程序编程接口的小型电话语音系统设计方法。文中提出了一种描述电话流程的计算机语言,并介绍了分析执行这种语言的解释器的设计方法。

由于表示电话流程的语言命令都以中文方式提供,用户可以方便地构建、修改自己的电话流程,实现了灵活性和易用性的较好结合。本文介绍的思想也可以作为设计大型电话语音系统和相关产品的参考。

4.36 语音芯片 APR9600 及其在电话遥控系统中的应用

摘自《电子技术应用》月刊,2000 年第 1 期

APR9600 是采用模拟存储技术的新型语音处理芯片。该文介绍 APR9600 的功能及典型应用电路,重点阐述了其与单片机 AT89C51 构成的电话遥控系统的设计。文中给出了系统原理框图、系统流程图。

4.37 串行红外收发模块及其控制器在红外抄表系统中的应用

摘自《电子技术应用》月刊,2000 年第 9 期

该文详细介绍了 Vishay 公司先进的 RFDx4xxx 系列串行红外收发器及其控制器 TOIM3232,描述了 TOIM3232 的功能结构框图及各引脚符号、功能等。文中还给出了红外抄表系统的实现,包括红外抄表系统的原理图。实验证明,抄表系统的红外通信性能良好,抗干扰能力满足实际需求。

4.38 HSP50214B PDC 及其在软件无线电中的应用

摘自《电子技术应用》月刊,2000 年第 12 期

软件无线电核心思想是将模数/数模变换器尽量靠近天线,在对信号充分数字化的基础上依靠软件来定义无线电的各项功能。HSP50214B 是 Intersil 公司推出的单通道数字下变频器 DDC 中性能最好、功能最强的新款产品。该文介绍了 HSP50214B 可编程下变频器(PDC)的结构组成、工作机理、引脚功能、技术特性和应用范围,文章最后给出了一个实用的数字中频软件无线电实验系统。

4.39 变速率 CDMA 系统软件无线电多用户接收机

摘自《电子技术应用》月刊,2000 年第 7 期

该文介绍了变速率 CDMA 系统及其相关接收机数学模型,特别描述了满足每一个符号最大信号干扰比(SIR)的线性多用户接收机。针对这种线性多用户接收机模型,提出了一种基于软件无线电技术的多用户接收机实现方案。该方案可以针对不同的 QoS 标准在固定速率和变速率系统之间进行切换,也可以实现变速率系统不同指标的多用户接收机之间的动态转换。文中给出了软件无线电接收机结构及变速率多用户接收机与传统匹配滤波器系统实现两者的对比。

五、新器件及应用技术

5.1 全帧读出型面阵 CCD 光电传感器在图像采集中的应用

摘自《电子技术应用》月刊,2002 年第 7 期

面阵 CCD 传感器被广泛应用于电视、多媒体技术及医疗电子设备。全帧读出型面阵 CCD 传感器不存在转移区或暂存区,使其在读出时不能“曝光”,在“曝光”时不能读出。要求光源必须是闪烁的。其体积小,可以微型化,适合应用在医用及工业电子内窥镜中。该文介绍全帧读出型面阵 CCD 光电传感器的原理及其在图像采集中的驱动和输出电路的设计,文中给出了实际测得的输出信号波形。

5.2 光电码盘四倍频分析

摘自《电子技术应用》月刊,2002 年第 12 期

当前常用的转角位置传感器增量式光电码盘,常采用四倍频的方法提高其测量精度。该文针对一些精度和稳定性不高的四倍频电路在应用中造成的误差,详细分析了两种可应用于不同环境的四倍频电路,从原理上说明了电路的精度和稳定性,其结论在实际应用中也得到了验证。文章内容包括:四倍频电路设计原理,面向通用计数器的四倍频及判向电路,面向可逆计数器的四倍频及判向电路设计。该文对电机伺服电路设计可提供一定参考。

5.3 H8/300H 系列单片机及其应用

摘自《微计算机应用》双月刊,2002 年第 3 期

H8/300H 系列单片机是日立公司推出的一种高性能单片机,具有功能强大、资源丰富和使用方便等特点,非常适于一些高性能要求的应用场合。该文分析了该系列单片机的特点和性能,并结合用 H8/3003 开发的一个系统,介绍了一些使用 H8/300H 单片机的注意事项和经验。

作者认为,目前我国开发人员选择单片机有单一化趋势,主要集中在 Intel 单片机,应该扩大选择范围,选择最适合系统要求的单片机,通用仿真器的出现也为开发人员扩大单片机的选择范围提供了条件。

5.4 PIC 16F877 单片机的键盘和 LED 数码显示接口

摘自《电子世界》月刊,2001 年第 8 期

该文是如何利用 PIC 单片机进行开发的系列文章之一。内容包括一个采用 8 个按键组成的小键盘,4 只共阴极 LED 数码管的硬件接口电路及软件流程程序。

5.5 PIC16F877 单片机实现 D/A 转换的两种方法

摘自《电子世界》月刊,2001 年第 11 期

该文是如何利用 PIC 单片机进行开发的系列文章之一。文中介绍两种使用 PIC16F877 单片机实现 D/A 转换的方法。一种是利用 PWM 输出功能实现的低精度 D/A 转换;另一种是利用 PIC16F877 单片机的同步串行口 SSP 部件的 I^2C 总线与 MAX518D/A 芯片接口实现的 D/A 转换。文中给出了实现 D/A 转换的硬件设计和软件设计。

5.6 P89C51RX2 的 PCA 原理及设计

摘自《电测与仪表》月刊,2001 年第 11 期

PCA(可编程计数器阵列)是 P89C51RC2 和 P89C51RD2 的一个特色。该文介绍其主要特点、工作原理和实际应用。文中给出了 PCA 定时器和中断部分的功能与原理图,介绍了 PCA 捕捉模式、PCA16 位软定时器模式、PCA 高速输出模式、PCA 的 PWM 模块及 PCA 模块 4 看门狗模式。

5.7 ADμC812 中串口及其应用

摘自《电测与仪表》月刊,2001 年第 10 期

ADμC812 是一种新型的集成 12 位数据采集系统,它在单个芯片内包含了三种串行 I/O 外设接口,即 UART 串行外设接口、2 线 I^2C 串行外设接口、SPI 串行外设接口。

该文详细阐述了上述三种串行接口的原理、结构、功能和操作。ADμC812 串行接口可以实现程序调试时的程序下载以及在短距离内进行主机与从机的数据传递,并且有多种可调的传输方式,连接电路简单,使用方便等优点。为实现主机和从机及从外围设备的通信提供了一种简单易行的方案。

5.8 INTEL96 系列单片机中若干问题的讨论

摘自《微处理机》季刊,2000 年第 2 期

16 位 MCS-96 系列单片机是我国目前应用最广泛的单片机之一。该文作者指出了许多文献资料中关于 MCS-96 系列单片机中断响应时间的不正确说法,并对存储器空间及定时器 T1、T2 的计数值为闭合空间的问题进行了讨论,还对相关概念给出了更为清晰的阐述。

5.9 关于 INTEL96 系列单片机中 HSO 事件的设置

摘自《微处理机》季刊,2000 年第 3 期

高速输出(HSO)是 MCS-96 系列单片机区别于其他型号单片机的一个重要特色。该文就设置 HSO 事件时,若事件的触发时间超过定时器 T1、T2 的溢出间隔的问题进行了讨论,提出了具体的解决方案,对许多文献资料中关于这一问题的讨论进行了补充,具有实用意义。文中指出,当事件的触发时间大于 T1 的溢出间隔时,可利用 T1 的中断设置事件,也可利用软件定时器设置事件,用 T2 作为时间基准时,HSO 事件设置的具体做法与以 T1 为时间基准时的相似。

5.10 MAX3100 与 PIC16C5X 系列单片机的接口设计

摘自《电气自动化》双月刊,2000 年第 4 期

该文简要介绍了通用异步收发器芯片 MAX3100 的功能特点及工作原理,给出了用 MAX3100 实现 PIC16C5X 系列单片机异步通信的接口设计实例。文中给出了用 MAX3100 实现 PIC16C54 与 RS-232 的接口电路及软件编程实例。

5.11 单片 MODEM 芯片在远程数据通信中的应用

摘自《自动化技术与应用》双月刊,2001 年第 6 期

该文通过一个火灾远程报警系统实例,介绍了 51 系列单片机新型单片调制解调芯片 MSM7512B 芯片的通信接口电路设计及软件实现。文中还简要介绍了 MSM7512B 在无线射频和专用通信线路中的应用。

基于芯片 MSM7512B 构成的远程通信系统和其他的远程通信方案比较,具有距离不受限制、可靠性高、接口电路简单等优势,在诸如水、电、煤气自动抄表系统,电力负荷监控仪,IC 卡电话,家庭安防系统,电话计费器,遥控、遥调、遥测仪表远距离数据传输中都有广泛的应用前景。

5.12 MX919 在无线高速 MODEM 中的应用

摘自《电子技术应用》月刊,2001 年第 8 期

MX919 是 MX.COM 公司的无线调制解调芯片,能完成 4 电平 FSK 调制和解调,内嵌有可编程工作寄存器。该文介绍了 MX919 的特点及其在无线高速 MODEM 中的应用。芯片通过音频接口与话音电台相接,通过并口与 CPU 连接,可以实现与传统话音电台进行数据信号的无线传输。文中给出了 MX919 接口电路、MX919 的数据发送流程图和 MX919 的数据接收流程图。用 MX919 设计的无线 MODEM 具有简单方便、实用灵活、开发周期短等优点。目前我们已将其成功应用到油田地理信息系统、银行运钞车、公共汽车的 GPS 车辆监控与调度系统中,效果良好。

5.13 高速串行数据收发器 CY7B923/933 及应用

摘自《电子技术应用》月刊,2001 年第 6 期

CYPRESS 公司的数据收发器 CY7B923/933 使用方便、可靠性好,可广泛应用于长距离高速点对点串行通信。该文介绍了 CY7B923/933 的性能特点、结构原理、工作模式及应用电路。文中还给出了在一 VME 总线系统中,采用该收发器及 UTP 双绞线实现了 400Mbps 的串行数据传输实例。

5.14 双口 RAM 与 FIFO 芯片在数据处理系统中应用的比较

摘自《微型机与应用》月刊,2000 年第 9 期

该文利用双口 RAM 和 FIFO 芯片实现了流水线工作的电路结构,比较了它们各自的优缺点,并指出应用中应注意的问题。文中给出了采用双口 RAM 的电路结构方案,采用 FIFO 芯片的结构方案。作者认为,采用双口 RAM 速度较快,但它的控制逻辑比采用 FIFO 芯片系

统要复杂一些,存取速度也要略低一点,而它对数据的存取是随机的,其备份电池保持数据的能力是一大优势,其存储容量比 FIFO 也要大一些,故在价格方面也稍高于 FIFO 芯片。FIFO 芯片存储容量要小一点,故它的价格也要便宜一些,其数据必须先进先出,才能保证数据的正确性。当采用存储容量较大的 FIFO 芯片时,其存取时间也有所增加。

对于需要交换数据较多的场合,一般采用双口 RAM,对于交换数据较少,速度要求较高的场合一般采用 FIFO 芯片。

5.15 MAX202E 在串行通信中的应用

摘自《电测与仪表》月刊,2001 年第 7 期

该文介绍了使用 MAX202E 和单片机 AT89C51 的 UART 端口构建的串行通信接口。文中给出了 PC 机通信时硬件组成和软件编程。内容包括:MAX202E 驱动电路设计,单片机端串口程序设计,RS-232 接口设计,PC 机端程序设计。AT89C51 与 PC 机之间波特率为 9600bps 的全双工串行通信模式,可以在 30s 内进行 20K 容量的数据交换。

5.16 线性隔离放大器 ISO122 的原理及应用

摘自《电测与仪表》月刊,2001 年第 4 期

美国 BB 公司设计的 ISO122 隔离放大器可以既完成绝缘作用,又完成信号放大作用。该文介绍 ISO122 的基本电路,并结合等离子切割机中弧压取样电路,分析隔离放大器的优点及应用。文中给出了 ISO122 隔离放大器的内部原理图和 ISO122 的应用电路图。

5.17 AD606 对数放大器的研究与应用

摘自《电测与仪表》月刊,2001 年第 7 期

该文研究了 AD 公司 AD606 似对数放大器的设计原理,介绍其技术特点,说明使用时的参数调整方法,并给出了实际测试波形。内容包括:一种似对数放大器的数学模型,AD606 特点简介,载波波形对 AD606 传递函数的影响,AD606 的使用及 AD606 应用实例。

5.18 电流/电压转换芯片 MAX472 在永磁直流电动机虚拟测试系统中的应用

摘自《电子技术应用》月刊,2000 年第 5 期

MAX472 是美国 MAXIM 公司最新生产的电流/电压转换器,其响应时间、线性度、漂移等指标均很理想,且能适应大范围大电流的测量。该文阐述了 MAX472 的工作原理、在永磁直流电动机虚拟测试系统中的具体应用电路及各项参数的计算。文中从不同角度分析了系统的测量精度,从而验证了应用该芯片的可行性。内容包括:电流测量方法及其电路实现,测试曲线的精度分析和系统测试结果精度分析。

MAX472 能很好地实现电动机电流的测量,满足整个测试系统的精度和速度要求,是比较理想的一种电流/电压转换测量方案。

5.19 高精度模数转换器 AD676 的原理及应用

摘自《电测与仪表》月刊,2001 年第 7 期

AD676 是一个多功能的 16 位并行输出模拟数字转换器。它利用电容切换/电荷再分配

结构达到 100ksps 的转换率(总转换时间 10μs)。通过芯片自动校准,即数字式纠正内部的非线性得到最优的性能。该文介绍了 AD676 的基本原理及应用,并结合基于 80C196 单片机的数据采集系统分析了其应用。文中给出了 AD676 转换时序,AD676 的接线原理图,AD676 参考电压电路及包含模拟与数字电路的 A/ D 变换器简化图。AD676 标准的接口特性和优良的性能将使其在高精度测量仪器中得到广泛的应用。

5 .20 DS2450 A/ D 转换器的特性与应用

摘自《电子技术应用》月刊,2000 年第 5 期

DS2450 是美国 Dallas 公司生产的符合单总线协议的四路 A/ D 转换器。每个输入通道有各自的寄存器组来存储输入电压的范围、分辨率和报警门限值以及当输入电压偏离指定范围时的使能标志。该文介绍了 DS2450 的主要特性、工作原理、单总线协议及其设置方法,并具体介绍了 DS2450 在工程应用中与计算机的接口技术和软件编程特点。

5 .21 80C196KC 内部 A/ D 转换器的使用

摘自《仪表技术》双月刊,2001 年第 6 期

80C196KC 是 Intel 公司生产的 CHMOS 型高性能 16 位单片机,片内集成了一个 4 或 8 通道的逐次逼近式 A/ D 转换器,又增加了外设事务服务器(PTS)。该文介绍 80C196KC 内部 A/ D 转换器的控制方法,着重说明 A/ D 转换时间的设置和 PTS 如何工作于 A/ D 模式。内容包括:A/ D 转换时间的设置,HSO 控制和 PTS 控制。

5 .22 一种 16～24 位分辨率 D/ A 转换器的设计

摘自《仪表技术》双月刊,2001 年第 6 期

该文介绍如何由两个 12 位 D/ A 组成廉价的、高分辨率的 16～24 位 D/ A 转换器。在高分辨率 D/ A 转换器设计中,给出了 16～24 位 D/ A 转换器电路,在 D/ A 输出实验中,给出了 D/ A 输入的混沌映射和实验结果。由二片 12 位 ADC7512 组成的 16～24 位 D/ A 转换器在混沌产生器系统中,表现了良好的输出特性。它还可在复杂信号发生器,实验室仪器仪表等场合发挥作用,有一定的实用价值。

5 .23 串行 A/ D 转换器 TLC2543 与 TMS320C25 的接口及编程

摘自《测控技术》月刊,2000 年第 4 期

该文以 TI 公司的 DSP 芯片 TMS320C25 与 11 通道 12 位串行模/ 数转换芯片 TLC2543 为例,介绍该类 ADC 与 DSP 的接口及编程方法。这种方法也适用于其他具有 SPI 串行接口的 A/ D 转换器。文中给出了 TMS320C25 与 TLC2543 硬件接口电路,软件设计和软件流程图。

5 .24 A/ D 转换器 ICL7135 积分特性应用

摘自《电子测量技术》月刊,2001 年第 5 期

A/ D 转换器 ICL7135 是 4 位半双积分型 A/ D 芯片,它具有精度高、价格低、抗干扰性强且与微控制器连接方便等优点。该文介绍如何利用 ICL7135 的积分特性和 51 系列单片机计

数及外部中断功能,实现与 ICL7135 的串行、光隔采样,提高采样速度,保证模拟部分的稳定性。文中给出了硬件实现原理图和初始化及软件设计。

5.25 高精度 A/D 转换器 AD7711A 及应用

摘自《电子技术应用》月刊,2000 年第 2 期

AD7711A 是美国 ANALOG DEVICE 公司推出的低价、高分辨率 A/D 器件,采用 Δ-Σ 原理,可实现高达 24 位的分辨率,特别适合于低频率、高分辨率、宽动态范围的 A/D 转换。该文介绍了其特点和应用,给出了一种用 AD7711A、单片机和 PC 构成的高精度橡胶硫化温控系统。该系统的稳态误差精度达±0.3℃。文中给出了系统简介,AD7711A 与单片机的接口,AD7711A 的读写时序和单片机代码,PID 调节器以及温度控制结果及分析。

5.26 多路 A/D 转换器 AD7714 及其与 M68HC11 单片机接口技术

摘自《测控技术》月刊,2000 年第 11 期

AD7714 是美国 AD 公司推出的一种高分辨率的多路模数转换器件,其采用了 Σ-Δ 技术实现 A/D 转换。该文介绍了 AD7714 的特点、内部寄存器结构和外部接口,并详细阐述了 AD7714 与 M68HC11 系列单片机的接口技术。给出了 AD7714 与 M68HC11 微控制器的接口图,AD7714 与 M68HC11 微控制器的接口图,并介绍了 AD7714 与 M68HC11 单片机接口程序范例。

5.27 用 AD7755 设计的低成本电能表

摘自《电子技术应用》月刊,2000 年第 8 期

该文较详细介绍了应用 AD7755 设计的一种低成本电能表。内容包括严谨的设计原理,精心的电路分析、先进的抗电磁干扰设计方案和详细的 PCB 布线图。本文对国内电子式电能表的生产具有参考价值。文中列有下列内容:设计原理,AD7755 基准源电路,分流器的选择,设计值的计算,校验电表,抗混叠滤波器,电源设计,抗电磁干扰设计,静电放电(ESD),高频电磁场,缩小电路带宽,隔离敏感器件,快速瞬态脉冲群(EFT),S20K275 型 MOV 和 EMC 测试结果。感兴趣的读者可与北京市英赛尔器件集团及其分公司联系,或直接访问英赛尔器件集团的网址:http:// www .incel .cn, com

5.28 20 位 Σ-Δ 立体声 ADA 电路 TLC320AD75C 的接口电路设计

摘自《电子技术应用》月刊,2000 年第 5 期

TLC320AD75C 是使用四阶 Σ-Δ 技术的高性能 20 位立体声模数和数模转换器(ADA),能同时进行四路 20 位分辨率的模拟到数字(A/D)和数字到模拟(D/A)信号通道的转换。该文介绍了 TLC320AD75C 的特点及构成,并详细论述了 Σ-Δ 型立体声 ADA 电路 TLC320AD75C 的模拟与数字音频数据接口技术、DAC 的串行控制接口技术及该类器件的使用注意事项。文中给出了 TLC320AD75C 的模拟音频数据接口电路,TLC320AD75C 的 ADC 接口电路及 TLC320AD75C 的 DAC 串行控制接口。

5.29 24 位 A/D 转换器 ADS1210/1211 及其应用

摘自《电子技术应用》月刊,2000 年第 3 期

ADS1210/1211 是高精度、宽动态特性的 Δ-Σ 型模拟/数字转换器。它的差动输入端可以直接与传感器或微小的电压信号相连。其内部的 Δ-Σ 结构可确保它的宽动态特性和 24 位的分辨率。该文详细阐述了 ADS1210/1211 的结构、原理及特点,并以数字电压表为例介绍了它的应用,给出了相关电路图和主程序框图。

5.30 模数转换器 AD7705 及其接口电路

摘自《国外电子测量技术》双月刊,2001 年第 2 期

该文介绍了串行 16 位的 ADC 芯片 AD7705 的基本特点及其 Σ-Δ 工作原理,阐述了该芯片与单片机 8051 的接口方法。文中给出了 AD7705 的内部原理框图,AD7705 与 8051 单片机接口电路。

5.31 串行 A/D 转换器 ADS7812 与单片机的接口技术

摘自《仪表技术》双月刊,2001 年第 3 期

ADS7812 是 BB 公司生产的低功耗、12 位 A/D 转换器。该文介绍 ADS7812 的使用方法,以及它与单片机之间的接口—“串并”转换电路的设计。文中还提出了 ADS7812 应用中应注意的一些问题。文中给出了采用内部移位时钟的 A/D 转换电路,ADS7812 和 AT89C51 之间的接口以及“串并”电路 PLD 实现的顶层原理图。

5.32 串行 A/D 转换器 TLC548/549 及其应用

摘自《仪表技术》双月刊,2001 年第 3 期

TLC548/549 是以 8 位开关电容逐次逼近 A/D 转换器为基础而构造的 CMOS A/D 转换器,设计成能通过三态输出与微处理器或外围设备串行接口。该文介绍 TLC548/549 的结构、特性和工作原理,讨论了芯片与 MCU 连接的软件和硬件接口设计。文中给出了 LTC548/549 与 89C51 的连接图及部分程序清单。

5.33 采样率可变 16 通道 16 位隔离 A/D 电路

摘自《计算机自动测量与控制》月刊,2000 年第 6 期

该文介绍一种采样率可变的 16 通道 16 位隔离 A/D 电路,该电路采用串行接口的 16 位 A/D 和数字隔离技术,它能在不影响精度的前提下大量减少隔离器件的数目。文中介绍了电路工作原理并给出了 A/D 转换部分和启动及输出部分电路图。

本电路仅使用 8 个 6N137 数字光电隔离器(包括 4 个用于模拟开关)。实验证明,采用串口隔离技术是解决高精度 A/D 或 D/A 隔离的一个较有效的方法。采用上述方案制作的 16 位 A/D 隔离接口电路的隔离电压为 1500V,A/D 转换时间 < 25μs,A/D 通过率 > 40k,16 通道模拟输入,测量精度为 0.01%,定时触发周期可编程从 24μs 至 5.8 年。

5.34 TLC549 在交流有效值测量中的应用

摘自《微处理机》季刊,2000 年第 4 期

TLC549 是以 8 位开关电容逐次逼近 A/D 转换器为基础构造的 CMOS 模数转换器。该文应用 TLC549 及 MCU AT2051 设计交流有效值测量装置,给出了有效值测量线路原理框图,A/D 转换子函数框图和有效值测量框图,并专门讨论了通道增益,零漂的补偿。

作者建议在精度指标要求不高时,可采用 8 位串行 A/D。在精度要求高的设备中,可采用 12 位或 16 位串行 A/D 转换器。

5.35 温度传感器 DS18B20 的特性及程序设计方法

摘自《电测与仪表》月刊,2001 年第 10 期

DALLAS 公司推出的 DS18B20,具有比 DS1820 更好的性能,在温控系统中已得到广泛应用。该文说明了单总线数字式温度传感器 DS18B20 的性能结构和新特点。文中给出了利用该器件与单片机构成的一个测温系统,给出了程序设计方法。文中还给出了主程序框图和温度检测系统图。

5.36 DS1820 及其高精度温度测量的实现

摘自《电子技术应用》月刊,2000 年第 1 期

DS1820 是美国 DALLAS 公司生产的可组网数字式温度传感器,其内部使用了在板专利技术,将全部传感元件及转换电路集成在形如一只三极管的集成电路内。该文结合 DS1820 在水轮发电机组轴瓦温度测量中的应用经验,提出了一个用 DS1820 实现轴瓦温度高精度、高可靠性测量的可行性方案。文中并就提高 DS1820 测温精度的途径,DS1820 使用中注意事项提出了建议。

5.37 采用 DS1820 的电弧炉炉底温度监测系统

摘自《兵工自动化》双月刊,2000 年第 1 期

DS1820 是美国 DALLAS 公司研制的数字温度计,它用一根 I/O 线传送数据与命令,并可兼作电源线,是新一代单总线智能化数字温度传感器件。该文介绍了该器件的原理和在电弧炉底温度监测系统中的应用。文中给出了系统硬件框图和主程序流程图。

5.38 并行实时时钟芯片 DS12887 及其应用

摘自《电测与仪表》月刊,2001 年第 6 期

DS12887 是 DALLAS 公司推出的 8 位并行实时时钟芯片,由于它的并行控制功能使其在与微处理器接口时能大大提高 CPU 的工作效率,其计时精度非常高,在 25℃的工作环境中误差约为±1 分钟/月。该文介绍了 DS12887 的内部结构及使用方法,并给出了 DS12887 应用于智能电量测量仪设计中的实例。文中给出了 DS12887 应用电路。

5.39 利用实时时钟 X1203 开启单片机系统

摘自《电测与仪表》月刊,2001 年第 10 期

许多便携式记录系统,虽然系统大部分时间处于睡眠状态,但这样仍消耗一定的能量。为了进一步降低系统功耗,该文设计了一个电源开关电路,使单片机系统平时处于关闭状态,定时开启时系统上电,进行数据采集,一次工作结束时单片机关闭开关,系统断电。文中介绍了开关电路工作原理,给出了开关电路原理图,单片机控制软件流程图。本开关电路原理明晰,所选器件特点突出、工作稳定,可靠性好,为整个系统减少功耗。

5.40 时钟芯片 DS1302 及其在数据记录中的应用

摘自《电子技术应用》月刊,2000 年第 3 期

DS1302 是美国 DALLAS 公司推出的一种高性能、低功耗、带 RAM 的实时时钟芯片,它可以对年、月、日、周、天、时、分、秒进行计时,且具有闰年补偿功能。该文介绍了 DS1302 的结构和工作原理及其在测量系统中的应用。文中给出了 DS1302 与 CPU 连接的电路原理图和主程序部分框图。

采用 DS1302 作为记录测控系统中的数据记录,其软硬件设计简单,时间记录准确,既避免了连续记录的大工作量,又避免了定时记录的盲目性,给连续长时间的测量、控制系统的正常运行及检查都来了很大方便,可广泛应用于长时间连续的测控系统中。

5.41 串行显示驱动器 PS7219 及与单片机的接口技术

摘自《电子世界》月刊,2001 年第 7 期

PS7219 是一种新型的串行接口的 8 位数字静态显示芯片。采用流行的同步串行外设接口(SPI),可同时驱动 8 位 LED。该文介绍它与单片机 89C51 的硬件接口和具体的程序设计,实现 LED 的显示和功能控制。文中给出电路设计中的接口电路和写入子程序的程序流程图。

5.42 MAX7219 在 PLC 中的应用

摘自《电子技术应用》月刊,2000 年第 3 期

MAX7219 是 MAXIM 公司新推出的一种新型串行接口的 8 位数字静态显示芯片。该文介绍了 MAX7219 的工作原理,提出了一个适于 PLC 的结构简单、实用的静态显示系统,并以 SIEMENS 公司的 S7-200PLC 为例,讨论了硬件连接和软件编程。文中给出了 CPU214 与 MAX7219 的硬件连接和写入子程序流程图。

5.43 一种实用的 LED 光柱显示器驱动方法

摘自《电子技术应用》月刊,2000 年第 3 期

该文介绍了一种 LED 光柱显示器的驱动方法。文中给出了 LED 光柱显示器内部结构图,LM3914 驱动器内部结构图和驱动电路的实现电路。该电路稍加改动,即可用于 51 线或 41 线光柱,效果很好。

5.44 基于电能测量芯片 ADE7756 的智能电度表设计

摘自《电测与仪表》月刊,2001 年第 9 期

ADE7756 是一个带有串行外围接口(SPI)和脉冲输出的高精度专用电能测量芯片。该文介绍了 ADE7756 芯片特性,并利用该芯片设计了一种可实现远方自动抄表功能智能电度表。文中介绍了智能电度表构成原理并给出电路图。该表的性能指标满足 IEC 687/ 1036 国际标准,通信协议采用 IEC60870-5-102,可广泛应用于高层建筑、居民小区的电能自动抄表系统。

5.45 TSS721A 在自动抄表系统中的应用

摘自《电子技术应用》月刊,2000 年第 8 期

TSS721A 是一种用于仪表总线的收发器集成芯片,其内含的接口电路可以调节仪表总线结构中主从机之间的电平,可通过光电耦合器等隔离器件与总线连接,同时该收发器由总线供电,对从机不增加功率需求。该文介绍 TSS721A 的基本功能以及应用电路连接。这种收发器应用于自动抄表系统中,可使系统的实现大为简化。文中给出了 TSS721A 远程供电方式下电路连接和 TSS721A 在自动抄表系统中的应用。

5.46 电流传感放大器 MAX471/ MAX472 的原理及应用

摘自《电子技术应用》月刊,2000 年第 2 期

MAX471/ MAX72 是美国 MAXIM 公司生产的双向、精密电流传感放大器。MAX471/ MAX472 都可通过一个输出电阻将电流输出转化为对地电压输出。该文对它们的功能、特点、工作原理及应用场合作了介绍,文中并给出了 MAX471 在步进电机驱动器中的一个应用电路原理图。电流传感放大器 MAX471/ MAX472 的应用非常广泛。

5.47 8XC552 模数转换过程及其自动调零机制

摘自《电脑与信息技术》双月刊,2000 年第 3 期

该文介绍了 Philips 8XC552 系列单片机的模拟采样端口的模数转换工作原理、软/ 硬件启动方式及内部实现框图,着重分析了模拟采样的自动调零机制及其对模拟采样精度的影响。文中给出了 80C552 ADC 内部实现的框图,ADC 采样波形图和 ADC 比较单元的采样和自动调零。

5.48 旋转变压器-数字转换器 AD2S83 在伺服系统中的应用

摘自《电子技术应用》月刊,2000 年第 2 期

AD2S83 芯片是 AD 公司生产的跟踪式旋转变压器-数字转换器(R/ D 转换器)。该文介绍了其在伺服系统中的应用,重点介绍了该器件与主控芯片 DSP(TMS320F240)的接口电路设计。文中给出了 AD2S83 芯片外围电路的典型配置图和直接数据读取接口原理示意图。实践证明,这种方案能够实时地读取位置信息,而且接口简单可行。

5.49 具有串行接口的 I/O 扩展器 EM83010 及其应用

摘自《电子技术应用》月刊,2000 年第 7 期

该文介绍了具有串行接口的 I/O 扩展器 EM83010 的性能和特点,并利用 EM83010 实现了对 MCS51 单片机的 I/O 扩展。文中给出了 EM83010 应用电路和 MCS51 系列单片机对 EM83010 的控制方法。

5.50 新型 LED 驱动器 TEC9607 及其应用

摘自《兵工自动化》双月刊,2000 年第 1 期

TEC9607 是驱动共阳极 5 位 8 段 LED 显示器或共阳极 5×8 点阵 LED 显示的 ASIC。该文详细介绍了 TEC9607 的工作原理,提出了单片和多片 TEC9607 与单片机接口的使用方法及其显示驱动程序框图与相关程序,文中并介绍了其在电参测量仪表中的应用,内容包括:单片 TEC9607 驱动 5 位 LED 显示器电路和多片 TEC9607 扩展使用。

5.51 新型语音识别电路 AP7003 及其应用

摘自《电子技术应用》月刊,2001 年第 7 期

AP7003 是一种新型、低成本语音识别专用集成电路,内置有麦克风放大器、A/D 转换器、语音处理器和 I/O 控制器,经预处理后可识别 12 组不同的字词,每组 1.5 s 时长,可连词或单词识别。该文介绍了 AP7003 的性能、引脚、指令等内容,并给出了应用电路原理图及详细的应用方法,并指出应用中应注意的问题。应用表明,AP7003 使用简单方便,具有较高的识别率,一般可达 80%以上,是一款较好的语音识别芯片。

六、总线技术

6.1 现场总线技术的发展及应用展望

摘自《仪表技术》双月刊,2001 年第 2 期

该文论述计算机通信技术、智能仪表的发展和现场总线之间的联系以及集散控制系统(DCS)向现场总线控制系统(FCS)的转变。介绍了 PROFIBUS 技术应用,探讨了 PROFIBUS 产品的研究开发,最后展望了现场总线技术给我国自动化行业带来的机遇。

文中提出我国自动化行业不仅应通过引进产品来缩小与国际水平的差距,更应着手进行现场总线产品和全新的现场总线控制系统的研究开发,以抓住机遇赶超世界先进水平。

6.2 CAN 总线点对点通信应用研究

摘自《电测与仪表》月刊,2001 年第 1 期

该文主要介绍应用 CAN 总线实现分布式微机监控系统中,在点对点通信类型存在的问题及解决的办法。文中提出,为了避免数据传送错误,应在接收期间注意进行站号判断,在编写数据发送、数据接收程序时,注意加入针对各个节点的识别信息。

6.3 基于 CAN 总线的数据通信系统研究

摘自《测控技术》月刊,2000 年第 10 期

该文给出了 CAN 总线通信系统的软、硬件设计方案,探讨了应用中需注意的一些问题。

文中包括 CAN 接口电路和软件设计,其中包括:初始化部分,CAN 发送数据部分,CAN 接收中断处理部分。文中并对 CAN 应用中的 SJA1000 工作模式,总线定时寄存器 BTR1 和 BTR0 的初始化,接收码寄存器和接收屏蔽寄存器的初始化和状态寄存器的处理等进行了讨论。

6.4 基于 CAN 总线的分布式数据采集与控制系统

摘自《工业控制计算机》月刊,2000 年第 5 期

该文给出了一种基于 CAN 总线的分布式数据采集与控制系统的设计与应用,介绍了 CAN 总线的硬件接口电路设计,并对系统的整体结构、硬件配置、软件功能及各节点功能分别作了详细说明。运用 CAN 总线技术可极大地提高系统的可靠性、实时性,系统开发较廉价,性能价格比高,安装维护简便,具有广阔的应用前景,也是工业控制领域的一个重要发展方向。

6.5 基于 CAN 总线的分布式铝电解智能系统

摘自《电工技术》月刊,2001 年第 12 期

该文介绍基于 CAN 总线的分布式铝电解智能系统的设计和实现方法,给出了 CAN 总线的硬件接口电路设计和槽控机电路结构图。该系统可实现对电解槽生产的高效控制,保证电

解槽在新型工艺技术条件下的平衡运行。系统结构简洁,集成度、智能化程度和可靠性高,结构与功能可扩展性好,且易于操作与维护,有推广应用价值。

6.6 CAN 总线在通信电源监控系统中的应用

摘自《电工技术》月刊,2001 年第 12 期

该文介绍了 CAN 总线的性能,论述了 CAN 总线在通信电源监控系统中的应用。文中提出了一个基于 CAN 总线的通信电源监控系统的原理框图。CAN 现场总线在通信电源监控系统中的应用,可大大提高监控系统和整个通信电源系统的可靠性。

6.7 CAN 总线在弧焊机器人控制系统中的应用

摘自《微计算机应用》月刊,2000 年第 11 期

该文设计并实现了基于 CAN 总线的弧焊机器人控制系统,文中对该系统的设计方案、整体构成以及系统的硬件、软件设计等方面进行了较为详细的分析与讨论。

CAN 总线在弧焊机器人控制系统中的应用,提高了焊接过程控制系统的实时性、数据通信的可靠性,显示了 CAN 总线在工业控制领域中的优势,实现了集散控制系统向数字化系统的过渡,为焊接过程控制网络打下了一定基础。

6.8 CAN 总线及其在喷浆机器人中的应用

摘自《测控技术》月刊,2000 年第 3 期

该文介绍了 CAN 总线及其特性,论述了它在喷浆机器人中的应用,包括节点硬件电路设计和节点软件设计以及在机器人中的实际应用。

6.9 基于 CAN 控制器的单片机农业温室控制系统的设计

摘自《测控技术》月刊,2000 年第 12 期

该文以农业温室为背景,从低成本角度出发,介绍了一种基于 CAN 控制器的单片机农业温室控制系统,并就其软硬件设计作了较详细分析。包括系统组成原理及接口电路,工作过程,监控软件的设计和系统通信软件设计等。该系统具有通信质量高、系统性能稳定、价格适中等优点,适合于现阶段我国低成本自动化领域的应用,特别是农业领域。

6.10 现场总线国际标准与 LonWorks 在智能电器中的应用

摘自《电工技术》月刊,2001 年第 6 期

基于现场总线的智能电器是当今智能电器的发展方向。该文介绍了 IEC 现场总线国际标准的内容,结合 LonWorks 现场总线在变电站自动化的应用实例,讨论了 LonWorks 在智能电器中的应用。

6.11 基于 LON 总线技术的暖通空调控制系统

摘自《测控技术》月刊,2000 年第 11 期

该文基于 LonWorks 技术设计了一套较为通用的暖通空调控制系统。文中介绍了底层硬件及软件的设计、系统的组建、PC 机的通信与组态管理。内容包括:LONBAC-3000 暖通空调

控制系统的硬件结构;使用 LONBAC0-3000 构成空调控制系统;LONBAC-3000 暖通空调控制系统的软件设计。系统具有方案简单,层次清晰;便于安装与维护;便于用户网络化管理;系统及信号传输可靠性高;便于将来系统的扩充等优点。

6.12 通用串行总线(USB)及其芯片的使用

摘自《测控技术》月刊,2000 年第 10 期

通用串行总线(USB)作为一种即插即用的新型接口,成为电脑系统与外围设备连接的实用选择。该文介绍 USB 的特点及其专用芯片 USBN9602 的使用。

作者提示,目前对于 USBN9602 与单片机的连接,虽然能取得较好的效果,但是设计和调试比较麻烦,一些芯片厂商推出了具备 USB 通信接口的单片机,内部集成了 USB 专用芯片,这些单片机处理能力强,有的还配备 Windows 驱动程序以及 Windows 开发库,构成系统的电路简单,调试方便。

6.13 USB 在数据采集系统中的应用

摘自《电子技术应用》月刊,2000 年第 4 期

通用串行总线 USB 可为多点数据采集提供很大便利。该文介绍了如何利用 USB 接口来实现多点数据采集。内容包括:采用 USB 传输的数据采集设备;软件构成;实现 USB 远距离采集数据传输及综合式采集数据传输系统的实现。即将出台的 USB2.0 协议,数据传输速率高达 480 Mbps,USB 必将在更广阔的领域得到更深层次的应用。

6.14 用 MC68HC05JB4 开发 USB 外设

摘自《电子技术应用》月刊,2000 年第 5 期

该文结合 USB 手写板,介绍了用 MC68HC05JB4 等 MOTOROLA 单片机开发 USB 设备的一般思路和具体方法。内容包括:USB 总线系统硬件连接;USB 总线系统软件设计和 USB 设备的测试。USB 总线及设备是计算机技术高速发展的必然产物,国外市场上的 USB 产品已经很多了,但目前国内大都局限在少数几类设备上。USB 设备应该是国内硬件产品发展的重要方向之一。

6.15 8x930Ax/Hx USB 控制器芯片及其在数字音频中的应用

摘自《电子技术应用》月刊,2000 年第 1 期

该文介绍了 INTEL 公司的符合 USB1.0 规范的 USB 控制器芯片 8x930Ax/Hx,给出了该芯片的特性和功能描述,最后介绍了 USB 音频的特点及 8x930Ax/Hx 在数字音频中的应用。USB 协议和 8x930Ax/Hx 控制器的功能能很好地传输音频数据,应进一步开发。

6.16 基于 MC68HC(9)08JB8 芯片的 USB 产品——键盘设计

摘自《测控技术》月刊,2000 年第 12 期

该文简单介绍 MC68HC(9)08JB8 MCU 芯片的基本结构和 USB 协议,并系统介绍了 MC68HC(9)08JB8 在 USB 产品——键盘设计中的应用,其在传统键盘设计的基础上实现了即插即用的功能。文中给出了 USB 总线系统硬件连接图和软件设计图。

6.17 I^2C 总线在 LonWorks 网络节点上的应用

摘自《工业控制计算机》月刊,2000 年第 3 期

该文提出了一种仅用两个 I/O 口就在 LonWorks 节点上扩展出一个 I^2C 总线子系的方法,而将 I^2C 总线与 LonWorks 网络结合,并给出了一个应用实例。将 I^2C 总线应用于 LON 节点,使 I^2C 总线与 LonWorks 网络相结合,为我们今后 LonWorks 网络的开发应用提供了新方法。

6.18 Neuron3150 的并行 I/O 接口对象及其应用

摘自《测控技术》月刊,2000 年第 6 期

该文介绍 Neuron 芯片 MC143150 的并行 I/O 接口对象,着重说明 Neuron 3150 工作在从 B 方式下的令牌传送协议,并给出 Neuron 3150 工作在从 B 模式下与微处理器 M68HC11A1 的接口电路。利用 Neuron 芯片和 M68HC11A1 并行接口方式可实现 M68HC11A1 对数据的采集和测控,既弥补了 Neuron 3150 没有硬件中断系统、对 I/O 口反应慢的不足,又可发挥 Neuron 3150 强大网络通信功能这一优势。

6.19 新型串行 E^2 PROM 24LC65 在 LonWorks 节点中的应用

摘自《测控技术》月刊,2000 年第 2 期

该文介绍了 MicroChip 公司的串行 E^2 PROM 24LC65 的功能特点及工作原理,并具体给出了这种具有 I^2C 接口的存储器在 LonWorks 节点中的应用实例。文中给出了 MC143150 (Neuron Chip)与 24LC65 组成的 8KB 串行存储器电路图及控制软件。

6.20 利用 I^2C 总线实现 DSP 对 CMOS 图像传感器的控制

摘自《国外电子测量技术》双月刊,2001 年第 4 期

OV7100 是美国 OmniVision 公司推出的单片 CMOS 图像传感器,TMS320C3X 是 TI 公司的第三代产品,也是第一代浮点 DSP 芯片。该文分别介绍了它们的性能和特点。提出了一种利用 DSP 的多功能口实现 I^2C 总线控制方法。文中给出了用汇编语言开发的 I^2C 总线的发送程序。

6.21 在 I^2C 总线系统中扩展 LCD 显示器

摘自《电测与仪表》月刊,2001 年第 9 期

该文介绍如何利用 Philips 的 P87LPC764 单片机作为 I^2C 总线控制器与 I^2C 总线显示器件 PCF8577C 构成 LCD 显示器电路,并给出相应的程序清单。参考本文思路,也可实现动态 LCD 显示器的扩展。该方法在 I^2C 总线系统中进行人机接口电路设计时具有较好的使用价值。

6.22 基于 Windows 环境的 GPIB 接口设计实现

摘自《测控技术》月刊,2000 年第 8 期

通用接口总线(GPIB)的问世使得自动化测试中仪器的互联有了统一标准,推动了仪器制

造业的技术发展。该文作者开发出了 Windows 环境下 GPIB 接口及完善的软件开发工具包。该文全面介绍 GPIB 接口的软硬件设计方法,并着重论述 Windows 环境下 GPIB 接口驱动软件的设计开发方法及采用的关键技术,包括 VISA 控制技术的应用和多语言环境中动态链接库(DLL)函数的调用。

6.23 微机 PCI 总线接口的研究与设计

摘自《航空计算技术》季刊,2000 年第 2 期

该文介绍了目前实现微机 PCI 总线接口可采用的一些方法以及这些方法各自的优缺点。包括采用现有的芯片设计 PCI 接口;采用专用软件进行设计和用 CPLD 实现 PCI 总线接口。作者还采用 AMD 公司的 MACH4 系列 CPLD 芯片及其开发工具对 PCI 总线接口进行了设计、仿真和实现。文中介绍了设计中的一些体会。

6.24 通用串行总线(USB)原理及接口设计

摘自《电子技术应用》月刊,2000 年第 12 期

该文以 USB1.1 为基础,讨论了 USB 的基本原理、工作流程、通信协议和相应的关键技术,并介绍了一种 USB 接口的 10M 以太网卡的设计方案。文末还论述了 USB2.0 的改进和优点。内容包括:USB 的结构与工作原理;Windows USB 驱动程序接口和 USB 接口 10M 以太网卡的实现。

6.25 CAN 总线与 1553B 总线性能分析比较

摘自《测控技术》月刊,2000 年第 6 期

该文介绍了 CAN 总线与 1553B 总线及特点,并对二者响应时间和可靠性进行了分析。CAN 总线和 1553B 总线由于其具有传输速率高、可靠性高、实时性好等优点,深受许多工业应用部门的青睐,但由于 CAN 总线的性能不比 1553B 总线的性能差,并且价格低廉,故 CAN 总线的应用范围更广。

6.26 利用 USB 接口实现双机互联通信

摘自《计算机应用研究》月刊,2001 年第 6 期

该文介绍利用 USB 接口实现双机互联的方法,并与其他接口方法进行了比较。作者认为,与红外线接口、并行接口和串行接口相比,其具备传输速率高、有热插拔功能、不受电磁干扰、无方向性等明显优点。目前,USB 接口还存在一些不足。

6.27 一种带 USB 接口的便携式语音采集卡的设计

摘自《计算机应用研究》月刊,2001 年第 5 期

该文以一种带 USB 接口的语音采集卡设计为实例,重点介绍了如何结合 USB 协议规范,利用 USB 控制器 USBN9603 进行 USB 设备软、硬件的设计。作者认为,设计关键在硬件方面,对 USB 接口要加入匹配滤波网络和静电保护措施。另外,采集卡是数模电路共存的系统,还需注意供电的防干扰,即数字地、模拟地一点相连。

七、可靠性技术

7.1 电磁干扰与电磁兼容设计

摘自《电脑开发与应用》月刊,2000 年第 1 期

该文介绍了在 PC 板设计中走线、物理布局及元器件处理的 12 条抗干扰措施和计算机系统设计中屏蔽、滤波和接地的 4 条建议,最后对电磁兼容设计给出了 5 条提示,包括铁氧体珠可用来滤除连接到接插件、开关或电源的引线上的噪声,光电隔离器可用于数字和模拟电路的隔离,以减小传导 EMI 等。

7.2 计算机的防电磁泄漏技术

摘自《微型机与应用》月刊,2000 年第 10 期

该文分析了信息的泄漏方式、TEMPEST 技术、计算机和外部设备中的 TEMPEST 技术,提出了一些简易防泄漏技术的方法以及发展我国 TEMPEST 技术的措施,其中包括:利用噪声干扰源,采用屏蔽技术,“红”/“黑”隔离,滤波技术以及低辐射设备,距离防护,使用铁氧体环,选用合适的电缆和注意计算机的位置等。

7.3 低辐射计算机系统的设计实现

摘自《电气与控制》季刊,2000 年第 3 期

该文给出了一个军用低辐射计算机系统的设计方案,包括电路板的低辐射设计和计算机系统泄漏防护,并给出了低辐射计算机系统的电磁发射测量与结果分析。

经过对 486 型微机系统采取低辐射电路板设计,降低机内源辐射,再对系统整体和局部进行屏蔽、滤波等多种防护技术处理,其传导发射强度和辐射发射强度较防护前都大大降低,电磁发射指标均比标准极限值低。

7.4 静电测量及其程序设计

摘自《宇航计测技术》双月刊,2000 年第 1 期

近年来,静电的应用、静电的危害及防止研究迅速开展起来。该文设计了一种新型静电传感器,在此基础上研制了一种智能化便携式静电测量仪。文中讨论了程序设计并就被测量正负判别提出了一种通用的简便有效的方法。文中除讨论静电测量原理,还给出了系统硬件框图以及利用软件进行正负判别及程序流图。

7.5 电子产品生产中的静电防护技术

摘自《自动驾驶仪与红外技术》季刊,2000 年第 4 期

该文介绍了电子产品生产过程中的静电危害性和静电的防护措施等,对保障安全生产,提高产品质量,降低生产成本,增长经济效益,有较大参考价值。文章内容包括:静电接地,化学

涂层,改变湿度,配备离子产生器,加强管理,保障静电控制措施的有效性等。

7.6 电子测控系统中的屏蔽与接地技术

摘自《航空计测技术》双月刊,2000 年第 1 期

本文主要讨论电子测控系统中的屏蔽与接地技术,包括:电场屏蔽;磁场屏蔽;双层屏蔽;各种电缆屏蔽;金属网屏蔽室等。测控系统所遇到的屏蔽和接地问题千差万别,本文只讨论了一般性问题,对具体问题还须作具体分析。

7.7 微机控制系统的抗干扰技术

摘自《仪表技术》月刊,2001 年第 4 期

该文讨论由微机组成的系统在实时控制过程中所遇到的干扰及其产生的影响,提出了排除干扰的软、硬件措施。硬件抗干扰措施方面包括:变压器隔离;光电隔离;浮地屏蔽滤波。软件抗干扰措施方面包括:初始化引导程序;反复置栈顶与反复开中断;自动复位重新启动(即看门狗)以及采用参数表表决纠错。

7.8 如何提高单片机应用产品的抗干扰能力

摘自《电子世界》月刊,2000 年第 8 期

该文就如何提高单片机应用产品抗干扰能力问题分别从硬件设计方面和软件设计方面进行了讨论。

7.9 工业控制计算机系统中的常见干扰及处理措施

摘自《工业控制计算机》月刊,2000 年第 5 期

该文介绍微型计算机在应用于工业控制装置时的干扰问题,并从干扰类型、产生原因和处理方法进行了讨论,包括对来自输入/输出通道的干扰解决方法,电源系统与接地抗干扰措施以及软件抗干扰措施,包括:限幅滤波,限速滤波和平滑滤波等。

7.10 GPS 用于军用导航中的抗干扰和干扰对抗研究

摘自《电气与控制》季刊,2000 年第 4 期

GPS 卫星导航系统已由民用扩展到了军用,导致新一代更高级别的 GPS 电子导航战。GPS 系统的抗干扰能力和干扰对抗已成为各国军方关注的焦点。该文论述导航战场作用和军用范围,进攻导航战的目标和防御战的措施以及 GPS 系统中 C/A 码的扩展和加密应用,新码的发展和抗干扰能力。文中还通过应用实例进一步论述对 GPS 接收机的干扰原理和干扰效果分析以及美、俄两国发展 GPS 干扰机的技术水平。

7.11 基于开放式体系结构的数控机床可靠性及抗干扰设计

摘自《制造业自动化》月刊,2000 年第 4 期

该文针对具有开放体系结构的数控机床,详细分析了结构体系的设计、内部各单元之间的联接以及外部电源系统等方面对数控机床可靠性的影响,并从电源的处理、接地与布线、隔离、滤波、屏蔽等硬件以及软件设计提出了一系列抗干扰的措施,内容包括:开放式数控机床结构

体系的可靠性设计;数控机床系统内部可靠性与抗干扰设计;外部干扰与抗干扰设计;软件可靠性与抗干扰设计等。

7.12 变频器应用技术中的抗干扰问题

摘自《电工技术》月刊,2001 年第 11 期

该文通过变频器应用过程中遇到的干扰问题,分析了产生干扰的根源和传播途径,提出了针对不同类型干扰的抗干扰方法,包括:切断传导干扰的方法和切断诱导干扰的方法。

7.13 单片机的软件可靠性编程

摘自《电工技术》月刊,2001 年第 7 期

单片机系统的可靠性分为硬件可靠性和软件可靠性。该文主要探讨软件可靠性,包括:如何防止数据丢失;判断程序进入死循环并进行处理;判断程序从第一句开始执行的原因类型。

7.14 单片微机的软件抑噪方案

摘自《电工技术》月刊,2001 年第 1 期

该文针对单片微机控制系统提出了几种有效的软件抗干扰对策,基于各部件的抑噪性能试验,阐述了抗干扰程序的原理及方法,内容包括:主动初始化;数据冗余法;重复执行;重要数据的保护和恢复;片内软 WDT 监控。

实践表明,本文针对微处理机易受干扰的部件提出的软件抗干扰的一些方案,具有较高的经济性和可靠性,很容易将其结合进工业控制程序之中,不失为一类简便而有效的抗扰手段。

7.15 SmartLock 并口单片机软件狗加密技术

摘自《电子技术应用》月刊,2001 年第 7 期

该文介绍一套单片机软件狗加密系统——SmartLock。它安装在微机并行口上,通过并口与加密软件进行通信,主要使用代码移植的方法实现软件加密。该系统具有加密可靠、使用方便、兼容性好等特点。文中给出了 SmartLock 软件加密系统结构图,包括:底层 I/O 模块;用户接口模块;全自动操作模块;反跟踪模块和异常情况处理模块。文中还阐述了加密软件的使用方法和代码移植技术。

7.16 单片机系统中复位电路可靠性设计

摘自《电子测量技术》双月刊,2001 年第 3 期

单片机应用系统中的复位电路能否可靠工作对整个系统至关重要。该文对常用复位电路中存在的一些问题进行了分析,并提出了解决方法,包括:电路结构;复位电路的抗干扰措施和系统中接口芯片的可靠复位等。

7.17 测控系统中实现数据安全存储的实用技术

摘自《电测与仪表》月刊,2001 年第 2 期

该文介绍了测控系统中对于非易失性存储的重要信息,如何实现数据安全存储的实用技术,包括芯片的选型,相应的硬件、软件措施以及数据破坏后的校验恢复手段。文章内容包括:

数据存储芯片的选择;数据安全的硬件措施;数据安全的软件措施。文中所述各种实用技术已在开发的多项产品中得到了实际应用,取得了良好应用效果。要想全面提高系统的可靠性和数据安全性应将以上技术结合系统整体设计综合考虑。

7.18 高精度仪表信号隔离电路设计

摘自《传感器技术》月刊,2000 年第 2 期

该文给出了一种新型结构的以模拟电路来实现的信号隔离电路,同时给出了理论分析和计算。这种新型隔离电路具有精度高、调试简单、稳定性好、可靠性高等特点,已在仪表的实际使用中获得了满意效果。文章内容包括:基本电路原理图;调制电路运放输出波形;理论计算及实测结果。

7.19 基于 AT89C2051 单片机的防误操作智能锁

摘自《电气自动化》双月刊,2000 年第 5 期

该文叙述了由 AT89C2051 单片机组成的智能锁的结构、特点、作用和工作原理,并详细介绍智能锁 IC 卡钥匙的硬件设计和软件编程方法。文中给出了硬件电路结构图;主程序框图和开锁操作程序框图。

7.20 E-mail 的安全问题与保护措施

摘自《航空计算技术》季刊,2000 年第 2 期

该文全面分析了 E-mail 的安全问题,提出了应该采用的有效方法,以保护 E-mail 系统安全运行。文中提出的主要问题包括:E-mail 的安全漏洞;E-mail 的安全措施;E-mail 有关的病毒;E-mail 垃圾;E-mail 炸弹和防范;防火墙技术;E-mail 的乱码及解决方法。

7.21 双机容错系统的一种实现途径

摘自《计算机工程》月刊,2000 年第 9 期

该文介绍一种基于 Windows 多线程思想设计的双机容错系统的一种途径,主要是利用串行口进行纯软件双机热备份。双机热备份系统的主要功能是确保系统的不间断运行,主机和副机可以通过网络、串口、SCSI 等通道相互监视各自的运行情况,一旦某台机器发生故障,另一台机器将迅速自动接管它的全部资源,从而保证系统的不间断性,也保证系统数据的完整性。这种双机容错系统中引入多线程思想,大大地降低了应用成本,并且串口操作简单方便,易于维护。

7.22 单片机应用系统抗干扰设计综述

摘自《计算机自动测量与控制》月刊,2000 年第 1 期

该文综合分析了单片机系统干扰的现象和原因,提出了一些在实际应用中取得了良好效果的提高抗干扰能力的方法,包括单片机应用系统的硬件抗干扰设计和单片机应用系统的软件抗干扰设计。这些方法有效可行,已在很多系统设计中采用,收到了良好的效果。

7.23 微机控制系统中的干扰及其抑制方法

摘自《电脑开发与应用》月刊,2000 年第 1 期

微机工业应用大致受到三方面的干扰:交流电源、信号通道和空间辐射。本文着重探讨信号通道和交流电源干扰的抑制措施。通过对模拟量的隔离和对开关量的隔离可以抑制来自信号通道的干扰。避开干扰,采用清洁电源、开关电源和隔离变压器的方法,可以克服来自交流电源的干扰。

7.24 智能仪表的抗干扰和故障诊断

摘自《制造业自动化》月刊,2000 年第 7 期

基于作者多年的智能仪器仪表开发经验,该文介绍了一些基于单片机的智能仪器仪表的软硬件抗干扰和故障诊断方法,内容包括:硬件抗干扰方法;软件抗干扰方法以及硬件故障诊断和软件故障诊断。

八、应用实践

8.1 AT89C51 在银行利率显示屏中的应用

摘自《兵工自动化》双月刊,2000 年第 1 期

该文介绍基于单片机 AT89C51 软硬件资源实现银行利率显示屏中人民币利率的设置、显示和时钟指示。文中给出了系统原理和结构配置,利率显示屏控制原理图和软件设计。该显示屏系统置数灵活、数据显示亮度高、数据可靠、功能齐全,易于扩展,维护方便,对应用环境要求不高。

8.2 基于 8xC196MC 实现的磁链轨迹跟踪控制

摘自《电气自动化》双月刊,2000 年第 2 期

该文通过分析磁链轨迹跟踪控制的原理,提出一种均匀劈零磁链轨迹跟踪 PWM 方法,并用 Intel8xC196MC 单片机特有的波形发生器功能实现了上述方法。文中给出了 8xC196MC/MD 的波形发生器及 PWM 波形实现及分析。对 380V 400W 异步电动机和 380V 7.5 kW 稀土永磁同步电机的实验验证了该方法的有效性。

8.3 基于 80C196KC 的开关磁阻电机测试系统

摘自《电子测量技术》双月刊,2000 年第 2 期

该文介绍了利用 80C196KC 单片机、高速 14 位 A/D 转换器(MAX125)及数字扭矩仪构成的开关磁阻电机测试系统,给出了硬件设计方案、软件模块构成以及提高系统抗干扰性的措施。文中给出了硬件设计中的电流、电压传感器,信号调理电路,隔直和比较电路,测转矩和转速电路,单片机系统及 A/D 转换器及软件设计中的单片机程序设计,PC 机软件设计。本系统已成功运用于三相 30KW 开关磁阻电动机的参数测量,实现了电压、电流、转矩、转速的实时监测。

8.4 80C196KB 单片机在绕线式异步电动机启动控制中的应用

摘自《电工技术》双月刊,2001 年第 4 期

该文提出了一种采用 80C196KB 单片机作为控制器,采用 IGBT(绝缘栅双极晶体管)作为斩波器控制转子电阻的新型启动方法。文中给出了系统总体框图、启动过程控制算法和控制软件。实验表明,系统工作稳定可靠,启动时间可以进行调节,大大减少对设备的冲击,具有一定推广价值。

8.5 GPS 时钟系统

摘自《电子世界》月刊,2000 年第 5 期

全球定位系统(GPS)是一种先进、完善的卫星定位系统,既具有全球实时、连续的高精度

三维定位能力,也具有精密的授时能力。该文利用 GPS 接收板,通过单片机技术的应用,开发高精度的 GPS 时钟。文中给出了 GPS 时钟的硬件设计。整个 GPS 时钟系统小巧、紧凑,硬件软件大大简化,并且提高了可靠性。

8.6 一种由 AT89C2051 单片微机实现的功率因数补偿装置

摘自《计算机自动测量与控制》月刊,2000 年第 8 期

功率因数补偿装置用于电力配电系统旨在提高功率因数。该装置以 AT89C2051 为核心,可自动完成电力供电线路的参数测定、运算及功率因数自动补偿控制。本文提出以可控硅全控桥作为执行部件,实现无级平滑容性补偿的方案,一改传统的机械触点式多组开关进行电容器分组切换投入进行补偿的方法,具有一定的实用性。文中给出了硬件电路设计的系统原理图,软件流程。功率因数补偿装置实现了智能运算和控制,电路方案简洁,硬件线路简单,由于采用新颖的可控硅全控桥模块作为执行部件,实现了功率因数补偿的平滑连续控制。

8.7 数据采集系统芯片 ADμC812 及其在温度监测系统中的应用

摘自《国外电子测量技术》双月刊,2001 年第 4 期

ADμC812 是全集成的 12 位数据采集系统,可组成单片数据采集系统,应用于工业控制、家用电器、通信、自动化和军事等领域。该文详细介绍了 ADμC812 的性能、特点及其在温度监测系统中应用的实例。文中给出了 ADμC812、温度传感器和显示电路的接口电路。这种数据采集芯片的高转换速度和精度,具有很强的竞争力。

8.8 用 AVR 单片机实现蓄电池剩余电量的测量

摘自《电工技术》月刊,2001 年第 5 期

该文提出了利用新一代 AVR 单片机(AT90S8515)实现蓄电池剩余电量在线测量的一种方法。文中给出了测试仪硬件框图和主要程序流程图。本法不仅适用各种容量的蓄电池剩余电量的测量,也可用于 Ni-Mh、Ni-Cd 及 Li 电池,具有良好的通用性和实用性。

8.9 基于 SA9604 的多功能电度表

摘自《电测与仪表》月刊,2001 年第 2 期

该文介绍一种以 SA9604 为计量芯片的多功能电度表,工作原理,同时给出功能框图,计算机自动校表的原理和修正公式。本电度表适用于城乡一般计量场合,是一款性价比较高的三相电能计量器具。

8.10 数字正交上变频器 AD9856 的原理及其应用

摘自《电子技术应用》月刊,2000 年第 12 期

AD9856 是 AD 公司生产的一种通用、高性能的数字正交上变频器件,具有集成度高、性能好、体积小、功耗低等特点,使用该器件很容易实现信号的数字正交调制。该文介绍了 AD9856 的工作原理及使用方法,并给出了其在数字音频广播(DAB)发射系统中的具体应用。该中频正交调制器结构简单,易于调试,且调制参数可根据需要进行设置,在实际应用中效果良好。

8.11 基于 MC628 的可变参数 PID 控制方法的实现

摘自《制造业自动化》月刊,2000 年第 7 期

该文介绍 MC628 运动控制器实现全数字交流伺服的体系结构,论述了系统参数对系统的影响,并给出了一种工程实际中简单的 PID 参数在线调节方法,内容包括:控制系统结构,系统参数对系统性能的影响和系统参数整定方法。

8.12 Windows 98 下远程数据采集系统设计

摘自《测控技术》月刊,2000 年第 1 期

在 Windows 98 下,进行串行通信用户需要编写相应软件,可通过调用 API 提供的函数来完成;也可用 Visual C++ 6.0 中的 ActiveX 控件 MSComm。本文采用后者实现了 Windows 98下 PC 机与 8098 的串行通信,编程任务小,并使系统具有更强的通用性和可移植性。文中介绍了系统的硬件构成;8098 单片机程序的编制和 PC 机通信程序。该方法也可用于类似的工业场合。

8.13 一种新式微流量计的研究

摘自《测控技术》月刊,2000 年第 6 期

该文提出了一种新式微流量计的设计方案。该方案采用层流式微流量传感器,使用电子线路对微流量传感器的信号进行放大、转换,用单片机实现非线性补偿与标度转换,进而研制成数字显示的高精度微流量计。文中介绍了流量计工作原理,流量计的硬件组成及软件设计。小型高精度传感器,不仅精度高、体积小,而且易于与单片机结合实现数字显示;但也存在缺点:由于节流管径小,如果被测气体中有微小颗粒,使节流管道堵塞,影响测量精度,如测非洁净气体时,需外加过滤器。

8.14 一种便携式多通道精密测温仪

摘自《航空计测技术》双月刊,2000 年第 5 期

该文介绍了一种利用数字式温度传感器 DS1624 构成的多路温度测量仪器。它具有线路简单,测量精度高的优点(在 -55～+125℃内,精度优于 0.05℃);带有计算机接口,能够实现长时间连续测量。文中给出了多路测温仪的原理框图,软件框图,测温仪的校准与测量实验。

8.15 一种高精度定时器的设计及其应用

摘自《电测与仪表》月刊,2001 年第 9 期

该文介绍了一种基于 Pentium CPU 的高精度定时方法,在不同主频的 CPU 下,可获得不同的定时精度,最高可达到 10 微秒量级。同时,介绍了这种定时方法在步进电机控制中的应用。文中给出了定时器的设计,不同主频的 CPU 的计数精度及定时线程流程图。定时器可以应用于大多数计算机实时控制系统。

8.16 智能湿度仪设计

摘自《电子测量技术》双月刊,2001 年第 2 期

该文利用电容式湿敏传感器和 MCS-51 单片机设计了具有高精度的智能湿度仪。该湿度仪具有体积小,线性好,滞后小,重复性好,响应快,能在 -10℃～70℃环境中使用,并能测量全湿范围的湿度的特点。文中介绍了高分子电容式湿敏传感器的结构,单片机对电容量的测量原理,并给出了智能湿度仪的整体逻辑框图。

8.17 固态数字语音记录仪的设计与实现

摘自《电子技术应用》月刊,2000 年第 11 期

该文介绍一种固态数字语音记录仪的实现方案。它主要由 DSP 最小系统、液晶显示模块、串行通信接口、键盘电路和片外闪烁存储器电路构成,实现语音信息的采集与播放,并且能够和笔记本电脑通过 RS232 串行口进行通信。文中给出了系统硬件结构,软件设计及其实现。由于采用了高速 DSP 技术,能够实时实现包括 G.723.1 在内的大部分低速率语音编解码算法,减小了数据量,降低了生产成本。

8.18 多功能语音电话答录器的设计

摘自《计算技术与自动化》季刊,2000 年第 3 期

该文介绍一种在单片机(AT89C51)的控制下,利用数字语音录放芯片(ISD4004)实现的多功能语音电话答录器。这种装置可与普通电话机相连,实现高质量的全数字录音,远程提取留言信息等功能,并可通过与家用电器开关电路相连,利用电话对电器进行远程控制。文中介绍多功能语音电话答录器的特点和功能,给出了硬件设计,多功能语音电话答录器原理图和主程序流程图。

8.19 白炽灯色温测量装置电路设计

摘自《航空计测技术》双月刊,2000 年第 1 期

“色温”是一个表示光源颜色的量,是反映光源特性的一个重要指标。该文介绍如何利用单片单板机接口电路完成对 6 V、30 W 白炽灯光源与其对应的标准白炽灯光源的色温比对标定工作。文中给出了测量原理与方法以及电路设计。

8.20 交直流供电无缝连接电源控制系统设计

摘自《电子测量技术》双月刊,2001 年第 2 期

电源供电系统设计的基本原则是交直流供电能无缝连接,电池供电时效率高,具有电池过放电保护,电池充电和充电保护等。该文介绍一种交直流供电无缝连接电源控制系统的设计。该系统具有电池过放电保护,电池充电和充电保护等功能,能输出 3A(+5V)电流,可用于使用大电流的便携式设备和仪器控制等。文中介绍了交直流转换控制电路设计,电池过放电保护电路设计,输出控制电路设计和电池充电与充电保护电路设计。

8.21 小型电磁辐射敏感度自动测试系统的设计

摘自《航空计测技术》双月刊,2000 年第 4 期

该文介绍一种电磁辐射敏感度自动测试系统组成、工作原理和设计方法,叙述了硬件配置、系统控制、数据采集处理等软件设计,其中较详细介绍了光纤式场强数据采集部分的设计。本系统造价比进口的 EMS(电磁辐射敏感度)测试系统要小得多,而且准确度和灵敏度高,可满足军标和民标的辐射敏感度试验要求。

8.22 生物电极微电流动态检测装置

摘自《测控技术》月刊,2000 年第 6 期

该文介绍一种适用于生物电极输出的微电流动态检测装置,其通过对基底电流的采样和动态补偿,以及多种抗干扰和降噪设计,测量精度达到 0.1 nA,使生物电极反应的定量分析成为可能。文中介绍了工作原理,特别是测量系统的噪声抑制与抗干扰设计。对装置进行了静态、动态特性的测试,并实际测量了葡萄糖氧化酶传感器的输出特性。

8.23 二种铂电阻 4～20 mA 电流变送器电路

摘自《传感器技术》月刊,2000 年第 1 期

该文介绍两种实用的铂电阻 4～20 mA 电流变送器电路,它们分别采用等效负电阻和专用电阻温度传感器信号调理芯片 XTR105 对铂电阻进行非线性校正。文中分别给出了利用负电阻校正铂电阻非线性的 4～20 mA 变送器电路和集成芯片 XTR105 组成的铂电阻 4～20 mA 变送器电路。两种带非线性校正的 4～20 mA 电流变送器电路,大大降低了被测温度与输出电流之间的非线性程度,提高了测量精度,设计应用取得了良好效果。

8.24 基于单片机的智能型光电编码器计数器

摘自《电子测量技术》双月刊,2001 年第 3 期

该文介绍了一种基于单片机的对光电编码器的输出脉冲进行处理的方法,给出了集倍频、鉴向、计数于一体的智能型计数原理及其硬件电路和软件的设计。该计数器结构简单,计数性能和精度高,计数器输出形式及输出量均可根据用户需求改变。

8.25 嵌入式系统中利用 RS－232C 串口扩展矩阵式键盘

摘自《微型机与应用》月刊,2000 年第 3 期

该文介绍一种基于 RS－232C 串口的矩阵式键盘,给出了其硬件电路和部分软件设计源程序清单。本文所介绍的由 RS－232C 构成的键盘所需元件少、结构紧凑,充分利用系统资源,并具有在 15 m 外进行键控的功能,在嵌入式工控系统中具有一定实用价值。

8.26 电压矢量控制 PWM 波的一种实时生成方法

摘自《电气自动化》双月刊,2000 年第 2 期

该文从电压矢量控制的基本原理出发,提出了一种两段式逼近的控制算法。基于 TMS320C52DPS 数字信号处理器实现了同步电压矢量控制 PWM 波形输出。实验结果表明,

该方法具有软件编程容易,控制精度高等优点。文中给出了电压矢量控制的算法确定,硬件结构,软件编制和实验结果与结论。

8.27 便携式电能表校验装置现场使用分析

摘自《电子测量技术》双月刊,2001 年第 4 期

便携式电能表检验装置的工作条件是复杂和多变的。现场使用时装置运行在非标准条件下,所检出的误差失去了客观可比性,不具有法律地位。该文分析了便携式电能表校验装置用于两种电能计量现场检验中的实际问题,包括:实际负荷小,功率因数低,现场环境温度的影响问题,目前国内能够完全适合于实负荷下现场检验的便携式装置还不多。

8.28 用单片机实现大型电动机的在线监测

摘自《电工技术》月刊,2001 年第 7 期

目前,大型电动机监测系统大多数都是独立的单机系统,没有考虑电动机投入运行时的生产系统,更没有考虑生产工艺参数的变化。该文提出用单片机将需要调整的生产工艺参数输入原有的监测系统中进行加权分析,即首先对电动机所处的运行状态进行判断,然后再进行常规监测。文章内容包括:系统组成,单片机的硬件设计,单片机的软件设计,单片机的操作顺序。本系统可以有效减少监测系统因生产工艺参数的改变造成的误报警现象。

8.29 PLC 在 L 型管弯曲机电控系统中的应用

摘自《电工技术》月刊,2001 年第 12 期

该文在分析 L 型管弯曲机传统的继电器控制系统缺点的基础上,从 PLC 的选型、I/O 端子分配、系统程序设计等几方面介绍了 PLC 在 L 型管弯曲机电控系统中的应用。文章内容包括:硬件电路构成,工作顺序确定,设计梯形图,编程及调试。

8.30 用 EPROM 实现步进电机的控制

摘自《仪表技术》双月刊,2001 年第 3 期

该文介绍一种 EPROM 为核心的步进电机驱动电源,这种电源适用于各种相数的步进电机。文中介绍了用 EPROM 控制步进电机的基本原理,环形脉冲分配器的电路构成,并给出环形脉冲分配器的电路图。

8.31 一种手持设备的智能卡实现技术

摘自《计算机工程与科学》双月刊,2000 年第 3 期

该文给出了一台手持设备中的接触式智能卡的读写电路。该电路避免了智能卡的带电插拔操作,能有效地延长智能卡的使用寿命。同时本文还探讨了智能卡的类型识别问题,提出了一种对 ATMEL 公司各型存储器卡的识别办法。本智能卡读写电路能兼容国内目前市面上各种各样的智能卡芯片。

8.32 钞票颜色识别系统的设计

摘自《微处理机》季刊,2000 年第 4 期

通过对钞票颜色的测定可用于钞票的识别和检伪。本文介绍运用 MCS-51 单片机开发设计钞票颜色识别系统中“单片机颜色检测”的过程。着重分析钞票颜色识别系统的传感器的选择和使用。该系统功能完备,实用性强。

8.33 数字锁相环在位置检测中的应用

摘自《制造业自动化》月刊,2000 年第 3 期

该文以数字锁相环为动态测量核心设计了新型数显装置,该装置充分利用了数字锁相环的相位误差累积作用,提高了数显装置的测量精度和测量速度。文中介绍了系统原理,调相电路的设计,数字锁相环的设计及单片机系统和系统软件设计。研制成功的光栅数显装置采用 50 线/mm 的光栅传感器,细分数为 40,8254 计数器的计数频率为 8 MHz,其分辨力为 0.5 μm,速度达到 12 m/min。在实际应用中可以方便地改变系统的分辨力,应用灵活,具有一定的推广价值,并为位置检测装置的设计开辟了新途径。

九、DSP 及其应用技术

9.1 数字信号处理器 DSPs 的发展

摘自《电子技术应用》月刊,2000 年第 5 期

该文介绍了数字信号处理器 DSPs 的发展状况,特点、性能及其评价体系以及 DSPs 的发展趋势。阐述了现代 DSPs 的五种结构,即增强型 DSP,超长指令字(VLIW)结构,超标量体系结构,单指令多数据流(SIMD)结构和 DSP/ 微控制器的混合结构。

9.2 用 TMS320C6201 实现多路 ITU-T G.728 语音编码标准

摘自《电子技术应用》月刊,2000 年第 11 期

ITU-T G.728 标准是国际电信联盟于 1992 年制定的比特率为 16 kbit/s 的低延时 CELP 类语音编码器。该文在扼要介绍 G.728 编解码算法原理和 TMS320C6201 定点 DSP 芯片的基础上,详细讨论了 G.728 算法在 TMS320C6201 上实时实现的硬件设计和软件开发及优化的关键技术,包括资源分配技巧和数据关联性简化技巧。以男生、女生、音乐等多种音源输入的实验表明,系统具有良好的适应性。非正式试听测试表明,恢复语音保留了较好的讲话人特征,具有较高的自然度和可懂度。

9.3 采用 DSP 内核技术进行语音压缩开发

摘自《电子技术应用》月刊,2000 年第 10 期

该文介绍了一种采用 DSP 内核芯片设计开发的手持式语音设备。该设备无需开发系统支持,语音播放时间长达 200 min,压缩比达到 46∶1,仅使用一片 32 兆位的闪速存储器就可保存全部数据。文中给出了语音处理系统构成图,描述了系统开发过程。

9.4 TMS320C80 与存储器接口分析

摘自《电子技术应用》月刊,2000 年第 1 期

该文讨论了 TMS320C80 通过传输控制器与几种常用存储器的接口方式,包括 SRAM 接口,双口 RAM 接口,EPROM 接口和 DRAM 接口,描述了存储器识别信息的产生和时序分析。TMS320C80 的 TC 可支持多种外设,如 FIFO,VRAM,SDRAM,本文未进行讨论,但其接口原理和分析方法类似。

9.5 TMS320C32 浮点 DSP 存储器接口设计

摘自《电子技术应用》月刊,2000 年第 11 期

TMS320C32 是美国 TI 公司第三代数字信号处理器的新产品,广泛应用于实时数据采集和信号处理系统中。该文介绍了 TMS320C32 存储器结构及存储器接口的设计方法,并给出了一个存储器接口设计实例。与 TMS320C30 和 C31 相比,TMS320C32 的存储器接口更灵

活,功能也更强大,在实际中如能灵活运用,就能设计出高效、稳定的系统。

9.6 TMS320VC5402 DSP 的并行 I/O 引导装载方法研究

摘自《电子技术应用》月刊,2000 年第 8 期

TMS320VC5402 是 TI 公司推出的性价比极高的定点数字信号处理器。该文介绍 TMS320VC5402 DSP 芯片的性能,着重分析它的片内 ROM 结构及并行 I/O 引导装载程序,并给出利用 8031 单片机实现 8 位并行 I/O 引导装载的方法,包括硬件电路分析和程序设计。利用廉价、通用的 8031 单片机实现 C5402 8 位并行 I/O Bootloader 功能,一方面,可以很好地解决快速 DSP 与慢速 I/O(EPROM)之间的数据传输问题;另一方面,C5402 Bootloader 成功后,8031 还可作为"协助理器"使用,从而极大提高了系统的灵活性和实用性。

9.7 TMS320C30 系统与 PC104 进行双向并行通信的方法

摘自《电子技术应用》月刊,2000 年第 4 期

TMS320C30 是 TI 公司的通用 DSP 芯片,它有很强的浮点/定点数据运算能力和很高的处理速度,特别适合于进行实时的数据采集及运算处理(如 FFT、FIR、IIR 滤波等)。该文给出一种 TMS320C30/PLD 系统与 PC104 通过标准并行接口进行双向通信时扩展并口的方法,给出了硬件电路框图,分析了通信过程中握手信号的时序关系,并列出了通信测试程序的流程图。

9.8 基于 TMS320C6201 的 G.723.1 多通道语音编解码的实现

摘自《电子技术应用》月刊,2000 年第 12 期

该文介绍了一种基于 TMS320C6201 的 ITU-T G.723.1 全双工实时多通道语音编解码的实现。文中提出了利用 C 语言和汇编语言的各种优化方法以降低计算量,最后给出了各个主要模块的性能指标。本文根据 C6201 芯片的特点,作了大量针对 G.723.1 标准本身的优化,满足了多路信号的实时实现,达到能在 200 MHz 的 C6201 DSP 上实现 16 路语音信号的实时编解码,完全符合 ITU-T G.723.1 标准的定点算法,通过了 ITU-T 的所有测试矢量。

9.9 基于 TMS320C6201 的多通道信号处理平台

摘自《电子技术应用》月刊,2000 年第 11 期

若采用高速 DSP 芯片充当信号处理核心,一片高速芯片可以替代以前的几路,使原有的多通道处理系统可以大大地简化。该文结合在 Smart Gateway 方面的工作,提出了一个基于 TMS320C6201 的多通道信号处理平台的结构,并详细阐述了软硬件的组成。在多媒体和因特网技术日新月异的今天,研究多通道信号处理平台有着很大的实际意义,它将成为新一代智能化多媒体终端的雏形。

9.10 基于两片 TMS320C40 的高速数据采集系统

摘自《数据采集与处理》季刊,2000 年第 1 期

该文采用高速 DSP 芯片 TMS320C40 和乒乓缓存技术设计的嵌入式高速采集系统,实现了数据的高速采集传输存储,文中给出了系统的部分硬件设计和监控软件。该系统在"高频回

波测试系统”的工程应用中,数据采集存盘速率达 2 .5 Mbps;光纤隔离技术的采用使得系统误码率低于 0 .001‰;双 CPU 结构使系统具备了对采集数据选择存储的能力,大大节省了存储空间,有效地提高了系统工作效率,同时,使采集系统增加了数据处理功能;模块化设计和大规模可编程逻辑器件的应用使系统具备良好的扩展性和可维护性。

9 .11 使用 TMS320C542 构成数据采集处理系统

摘自《电子技术应用》月刊,2000 年第 9 期

该文使用 TI 公司 C5000 系列 DSP 中的 TMS320C542 开发了一个数据采集处理系统。文中介绍了系统结构、性能、工作流程及设计的注意事项。文章内容包括:存储器控制,FIFO 控制,A/ D 和 D/ A 控制,CPLDA 和 CPLDB 的应用以及 HPI 接口。在本系统中,由于既有 A/ D,又有 D/ A,构成了一个闭环,自发自收,为算法设计和调试带来很大方便。

9 .12 基于 TMS320C32 的视觉图像处理系统

摘自《探测与控制学报》季刊,2000 年第 1 期

该文作者结合研制视觉跟踪系统的核心部分——基于 TMS320C32 的视觉图像获取与算法处理系统板,阐述了视觉图像处理系统构成,I^2C 总线及 OV7610 工作参数的设置,图像采集控制电路与 DMA 存取及系统的编程及跟踪算法。图像窗口大小设定为 320×240 时,本系统可以 15 帧/ s 的速度获取帧图像,跟踪算法处理速度基本满足系统要求。

9 .13 用 ADSP-2181 和 MC68302 实现 MPEG-2 传送复用器

摘自《电子技术应用》月刊,2000 年第 9 期

MPEG-2 编解码传输系统是目前解决演播室节目制作和传输的主流产品。复用器是整个传输系统的关键设备之一。该文给出了用 ADSP-2181 数字信号处理器和 MC68302 微控制器实现传送复用器的设计方案和实现技术。该方法是在 MC68302 的控制下,用 DSP 查询复用器各输入 FIFO 的状态,根据各 FIFO 的状态读入数据,完成音频、视频的均匀打包。文中较详细描述了复用器的设计、ADSP-2181 的性能、ADSP-2181 的 IDMA 接口与 MC68302 的连接及 ADSP-2181 完成传输流打包。

9 .14 基于 DSP 的 PC 加密卡

摘自《电子技术应用》月刊,2000 年第 9 期

该文对基于 DSP 技术的 PC 加密卡进行了综述,对目前软件和硬件实现网络与信息安全的方法作了比较。文中提出了一种采用 TMS320C54x 和 PCI 接口芯片实现高速加密卡的设计方案。加密卡所实现的主要功能有:数据加密解密功能、数字签名与认证功能、密钥管理功能。功能的实现依赖于加密卡各模块间的协调工作,文中对各模块主要功能进行了描述。

9 .15 TMS320C2XX 及其在宽带恒定束宽波束形成器中的应用

摘自《电子技术应用》月刊,2000 年第 9 期

与 TMS320 系列相比, TMS320C2XX 在速度、片内资源和性价比等方面均有了很大提高,片内资源更加丰富,开发环境更加便利。TMS320C2XX 系列将取代 TMS320C2X 系列,得

到推广和应用。该文介绍了 TMS320C2XX 系列微处理芯片的结构特点及其开发环境和开发过程,并给出了 TMS320F206 在宽带恒定束宽波束形成器中的应用实例,给出了电路设计和软件设计。由于 TMS320F206 高的性价比及良好的开发环境,使该系统具有速度高、可扩展、易调度、便于修正等特点。

9.16 DS80C320 单片机在无人机测控数据采编器中的应用

摘自《遥测遥控》双月刊,2000 年第 4 期

DS80C320 是一种 80C31/80C32 兼容型单片机,使用标准 8051 指令集,引脚与 80C31/80C32 也兼容,可以直接用 DS80C320 替代原单片机系统中的 80C31/80C32,而原程序仍可正常运行。该文介绍以 DS80C320 单片机为核心开发的测控数据采编器,其简化了硬件设计和编程,减小了体积,减轻了质量,提高了系统可靠性,目前该测控数据采编器在某共轴式无人直升机试飞中应用工作正常,达到了各项性能指标的要求。

9.17 基于 TMS320F206 DSP 的图像采集卡设计

摘自《电子技术应用》月刊,2001 年第 11 期

该文提出了一种使用视频 A/D 芯片 TLC5510 与低档 DSP 芯片 TMS320F206 实现图像采集的接口设计方案。文中给出了硬件接口电路设计和接口程序设计。这一应用为 TMS320F206 用于图像处理提供了一种思路,从而为低比特率多媒体通信开辟了一条廉价的途径。

9.18 基于定点 DSP 的实时语音命令识别模块

摘自《电子技术应用》月刊,2000 年第 7 期

该文介绍一种基于定点数字信号处理器 ADSP2181 的实时语音识别系统。文中较为详细地描述了系统软件,包括:语音信号的端点检测,语音参数的选择和计算,语音的编码与回放,参数模板的管理,参数模板的匹配以及进一步提高识别率的措施。经过大量试验和应用表明,该系统稳定可靠,正确识别率超过 97%,系统平均响应时间小于 0.5 s,硬件成本低,是一种实用的语音识别装置。

9.19 基于 TMS320C50 的语音频谱分析仪

摘自《兵工自动化》双月刊,2000 年第 3 期

该文介绍了语音频谱分析仪的硬件结构和软件设计。该仪器从机使用专用数字信号处理芯片 TMS320C50 完成 FFT 运算,主机 PC 完成数据分析和结果显示,C50 通过 RS-323C 串口实现与 PC 通信。利用 PC 机强大灵活的数据处理及编程能力,构造了语音频谱分析仪的虚拟面板和显示视窗。

9.20 利用 DSP 实现的专用数字录音机

摘自《计算机工程》月刊,2000 年第 4 期

该文介绍一种使用 DSP 芯片完成语音压缩的专用自动数字录音机的软、硬件设计方法,并较为详细讨论了语音压缩算法的选择和实现、数据库的功能以及可靠性设计的问题。该录

音机具有经济实用、安全可靠等特点。将来可进一步采用 ITU-T G .729 等高压缩比算法,以求达到更佳的经济性能。

9 .21 基于 DSP 的全数字交流传动系统硬件平台设计

摘自《电气自动化》月刊,2000 年第 3 期

本文以 DSP TMS320F240 为核心,设计了一个基于磁场控制的全数字交流传动控制系统,与较先进的驱动电路及功率电子器件一起,构成全数字交流传动系统,为高性能的算法提供了硬件平台。文中描述了系统的主电路结构及原理,控制系统结构及与 PC 机的通信。

9 .22 ADSP2106x 中 DMA 的应用

摘自《电子技术应用》月刊,2000 年第 6 期

ADSP2106x 为 120MFLOPS(每秒百万次浮点运算),但该速度是以存储在芯片内部存储器中的程序和数据为前提的。为把程序和数据传输到芯片的内存中,通常需要 DMA 操作来实现。该文对 ADSP2106x 中的 DMA 的应用进行了详细介绍,包括:内存与外存间的 DMA,内存与主机间的 DMA,内存与外设间的 DMA 以及内存与 Link 口间的 DMA,同时给出了一些应用实例。ADSP2106x 中 DMA 操作功能强大,形式多样,通过对各种 DMA 的应用,可以使数据进出芯片变得更加流畅,也可以使其核心处理单元的运算能力发挥到极致。

9 .23 软件无线电中 DSP 应用模式的分析

摘自《电子技术应用》月刊,2000 年第 6 期

软件无线电是基于同一硬件平台上,安装不同的软件来灵活实现多通信功能多频段的无线电台,它是未来移动通信的一个重要研究方向。该文系统地介绍了软件无线电的特点及其体系结构,并结合 DSP 芯片 TMS320C541 分析了它的一种应用模式。DSP 技术的优化应用直接决定了软件无线电系统的性能价格比。在目前的 DSP 中,‘C541 是一种较好选择,尤其是其灵活的低功耗指令设置,使软件无线电的应用更加广泛。

9 .24 快速小波变换在 DSP 中的实现方法

摘自《数据采集与处理》季刊,2000 年第 2 期

小波分析是分析非稳定信号的一种非常有效的方法。该文简单介绍了快速小波变换(FWT),详细阐述了在 DSP 上 FWT 的周期性扩展的实现,其中特别介绍了 DSP 的循环寻址。最后,并给出了针对 TI 公司的 TMS320C3X 系列 DSP 的相应的汇编代码。可以预见,随着小波变换理论的进一步深入研究和 DSPs 技术的发展,两者的结合将在数字信号处理领域产生巨大的应用价值。

十、PLD 及 EDA 技术应用

10 .1 可编程器件实现片上系统

摘自《测控技术》月刊,2000 年第 10 期

为满足快出样机和快速上市的要求,现场可编程门阵列(FPGA)和复杂可编程逻辑器件(CPLD)已成为首选方案。本文介绍如何利用可编程器件,特别是 FPGA 来实现片上系统。文中介绍新一代 FPGA 如何在系统集成、系统存储、系统时钟和系统接口等几方面满足实现片上系统的要求,并预测其今后的发展趋势。

10 .2 VHDL 语言在现代数字系统中的应用

摘自《电测与仪表》月刊,2001 年第 6 期

电子设计自动化(EDA)关键技术之一是要求用形式化方法来描述数字系统的硬件电路,硬件描述语言及相关的仿真、综合等技术的研究是当今电子设计自动化领域的一个重要课题。该文介绍了 VHDL 硬件描述语言在一个简单数字系统设计中的具体应用过程,论述了 VHDL 语言在基本结构和基本特征以及在现代数字系统中的重要地位和作用。VHDL 将成为数字系统设计领域中所有技术人员必须掌握的一种语言。

10 .3 用 VHDL 设计有限状态机的方法

摘自《电子技术应用》月刊,2001 年第 7 期

该文以离心机定时顺序控制器的设计为例,阐述了运用 VHDL 设计有限状态机的方法。文中讨论了如何消除状态机输出信号“毛刺”的三种方案,即在状态机输出信号较少的情况下,直接把状态作为输出信号;对于顺序迁移的状态机,选择雷格码作为状态编码;在 Moore 型或 Mealy 型状态机基础上,用时钟同步输出信号。

10 .4 ISP-PLD 在数字系统设计中的应用

摘自《电子测量技术》双月刊,2001 年第 3 期

该文在简要介绍在系统编程(ISP)技术与具有 ISP 能力的 PLD(ISP-PLD)的基础上,以 Altera 公司的 ISP-PLD 为例,介绍了 ISP-PLD 的设计开发基本过程,最后给出了以 Altera 公司的 Flex 系列 EPF10K10LC84-4 芯片实现一可进行 24 小时循环计时的数字钟的设计实例。ISP-PLD 简化了系统结构,减小了设计单元的规模,单片 ISP-PLD 既可完成传统设计中多片中规模集成电路才能实现的电路功能。随着 ISP 技术的不断成熟,ISP-PLD 正以其高集成度、高速、高可靠性以及开发便捷规范等优点得到越来越广泛的应用。

10.5 基于 FPGA 技术的新型高速图像采集

摘自《电子技术应用》月刊,2001 年第 9 期

该文介绍了一个以 FPGA 为核心芯片的采集频率可达 13.5 MHz 高速图形采集系统。在该系统中,还采用了 PHILIPS 公司最新推出的视频 A/D 芯片 SAA7111,将电视信号转换成数字信号,并由 FPGA 作为控制器将数字信号存放 256K RAM,以便 DSP 芯片根据需要进行预处理,提取有用数据。本系统采用 DSP 芯片实现图像的模式识别,提高了处理速度,解决了图像处理过程中由于图像识别速度慢而影响整个图像的处理流程,解决了实际问题,收到了良好效果。

10.6 Protel 99SE 电路仿真

摘自《计算机自动测量与控制》月刊,2000 年第 5 期

在 EDA 领域中,Protel 已成为家喻户晓的电路设计工具。Protel 99SE 版更是开创了桌面 EDA 的新纪元!它不但使绘制原理图、PCB 版布局面线等功能更加完备,而且还为用户提供了功能强大、使用方便的电路仿真器。该文从实际应用的角度出发,全面介绍了 Protel 99SE 电路仿真功能,包括:强大的分析工具、丰富的信号源、全面的仿真模型库、友好的操作界面等。文末并将 Protel 99SE 与 SMARTSPICE1.55 仿真结果进行了比较。

10.7 可编程逻辑器件(PLD)在电路设计中的应用

摘自《电工技术》月刊,2001 年第 10 期

该文阐述了可编程逻辑器件(PLD)在电路设计中的应用,介绍了在线可编程(ISP)的原理。文中结合电力系统通用控制器的设计,介绍了 Altera 公司的 MAX7000 系列的 EPLD 和开发工具软件 MaxplusII 的使用。

10.8 基于 FPGA 的全数字锁相环路的设计

摘自《电子技术应用》月刊,2001 年第 9 期

该文介绍了应用 VHDL 技术设计嵌入式全数字锁相环路的方法。文中详细叙述了其工作原理和设计思想,并用可编程逻辑器件 FPGA 予以实现。采用 VHDL 设计全数字锁相环路,具有设计灵活、修改方便和易于实现的优点,并能够制成嵌入式片内锁相环。此类数字锁相环路中计数器的模数可以随意修改,这样,就能根据不同情况最大限度地、灵活设计环路。

10.9 基于 EPLD 器件的一对多打印机控制器的研制

摘自《测控技术》月刊,2000 年第 2 期

该文以汽车客运站检票系统实时打印结算单的实际需要为背景,设计开发了打印任务的实时驱动与控制器,由一台计算机利用控制器就可驱动多台打印机的实时打印。文中给出了打印机控制器的系统方案设计,硬件设计和软件设计。该打印机控制器接口简单,使用方便,可靠性高,可广泛适用于车站调度结算、收费票据、统计报表、财务单据等打印任务重、实时性强,而计算机处理任务较少的场合,具有一定推广价值。

10 .10 一种 VHDL 设计实现的有线电视机顶盒信源发生方案

摘自《电子技术应用》月刊,2000 年第 7 期

该文介绍了一种有线电视机顶盒的信源发生方案。该方案采用可编程逻辑器件来完成计算机 EISA 总线输出数据的格式转换,从而提高数据输出速率,满足信源要求。该方案采用了开放式结构,可以通过软件修改来实现功能扩充。文中还详细叙述了采用 VHDL 来进行可编程逻辑器件功能设计过程。

10 .11 一种并行存储器系统的 FPGA 实现

摘自《电子技术应用》月刊,2000 年第 7 期

该文介绍一种可在现代小卫星上应用的高(低)位交叉并行存储系统,并给出了该存储系统控制器的 FPGA 实现。文中提出了一个用 ASIC 设计一个共享总线开关网络,组合成 Omega 网络的方案,以消除对某一组内存的总线竞争,实现多 CPU 对共享分组存储系统的低位交叉并行访问。该系统的应用将极大地增强星上计算机的数据通信和图像处理能力,并提高整个系统的可靠性。

10 .12 SDRAM 接口的 VHDL 设计

摘自《电子技术应用》月刊,2000 年第 6 期

同步动态 RAM(SDRAM)以其高速和大容量的优点获得了广泛的应用,但其接口与目前广泛应用的微处理器系统不兼容。该文介绍了用 VHDL 语言实现的 SDRAM 与 RAM 之间的接口控制电路,从而将 SDRAM 应用到微处理器系统中。文中给出了 SDRAM 接口状态机设计,包括:SDRAM 读操作时序设计,SDRAM 写操作时序设计及其 VHDL 实现。该电路硬件实现和微处理器系统已通过验证,证明可将 SDRAM 作为高速、大容量存储器应用在简单电子系统中。

10 .13 采用 ISP 器件设计可变格式和可变速率的通信数字信号源

摘自《电子技术应用》月刊,2000 年第 10 期

作者采用 Lattice 公司在系统可编程器件 pLS/ ispLSI1016,设计出一个具有可变格式、可变长度和可变速率的通信数字信号源,从而可产生出符合设计者要求的数字码流。文中介绍了系统设计原理及框图,系统程序设计和 isp 编程设计。有关 isp 器件的开发及应用的一些问题还有待进一步探索。

10 .14 利用 FPGA 技术实现数字通信中的交织器和解交织器

摘自《电子技术应用》月刊,2002 年第 10 期

该文介绍利用 FPGA 实现数字通信中的交、解交织器的一种比较通用的方案,详细说明了设计中的一些问题及解决办法,包括:FPGA 的选用及实现的原理框图,读写地址的产生问题,交织、解交织器的最小时延问题及 FPGA 中延时功能的实现。大部分的交织器读、写地址都可用本文介绍的类似方法实现,这样既节省芯片、缩小电路体积,速度又快,调试也大大简化。文中还介绍了一种可供大家参考的实现 FPGA 中信号粗略延时的方法。

10.15 XC9500 系列 CPLD 遥控编程的实现

摘自《电子技术应用》月刊,2000 年第 1 期

该文阐述了 XC9500 系列 CPLD 器件遥控编程的实现方法,并重点介绍了在遥控编程系统中应用微控制器(Intel 8031)实现嵌入式 ISP 的软硬件设计,包括:硬件设计,XSVRF 文件的生成和嵌入式 ISP 编程流程。

10.16 PLD 器件在红外遥控解码中的应用

摘自《电子技术应用》月刊,2000 年第 1 期

本文探讨了如何借鉴家电红外遥控系统的原理,自行设计解码电路,使智能仪器具有遥控功能。文中提出了一种可编程逻辑器件(PLD)进行解码的方案,并分别用 EPROM 和 GAL 实现,给出了红外遥控解码电路原理图。该解码方案基于纯硬件原理,不占用 CPU 软、硬件资源,直接并在键盘扫描矩阵上,对 CPU 是透明的。本文适用于键盘扫描输入的智能仪器系统的红外遥控,对其他红外遥控系统的设计应用亦有借鉴意义。

10.17 利用 XCS40 实现小型声纳的片上系统集成

摘自《电子技术应用》月刊,2000 年第 10 期

该文介绍大规模、高速度的 FPGA 在小型渔用声纳系统设计中的应用。在该系统设计中,采用了 Xilinx 公司的 FPGA 芯片 XCS40 作为主要器件,基本上将整个系统的功能集成在一片芯片上。文中给出了由 XCS40 构成的声纳系统的原理,分布式计算与内置 RAM,集成的显示控制和 MCU 模块,并给出了微处理器核的资源使用状况和内部结构框图。由于采用高度集成的片上设计方法,该声纳信号处理板的体积大大缩小,整个系统仅由模拟电路、XCS40 及存储器构成,设计和调试都非常简便,整机工作性能也十分稳定。

10.18 可编程逻辑器件的 VHDL 设计技术及其在航空火控电子设备中的应用

摘自《电光与控制》季刊,2000 年第 2 期

该文简要介绍了可编程逻辑器件 CPLD 和 FPGA 的结构和特点,着重介绍了 VHDL 语言的特点及选择 VHDL 的理由,并通过几个实际应用中碰到的问题,介绍了使用 VHDL 的一点体会,即在参照和修改一些现成电路时,编写一个优化的 VHDL 代码的关键在于要依照硬件的内在要求去思考,而不是采用传统的设计思路。文末还给出了可编程逻辑器件的 VHDL 设计技术在一个火控产品中的应用。

10.19 DSP + FPGA 实时信号处理系统

摘自《电子技术应用》月刊,2000 年第 9 期

随着大规模可编程器件的发展,采用 DSP + ASIC 结构的信号处理系统正显示出其优越性,逐步得到重视。FPGA 是在专用 ASIC 的基础上发展出来的,成为解决系统级设计的重要选择方案之一。DSP + FPGA 结构最大的特点是结构灵活,有较强的通用性,适于模块化设计,从而能够提高算法效率,同时其开发周期较短,系统易于维护和扩展,适合于实时信号处理。该文介绍了正逐步得到广泛应用的 DSP + FPGA 处理机结构,在此基础上提出了一种实

时信号处理的线性流水阵列,并举例说明了该结构的具体实现,最后分析说明了此结构的优越性是具有灵活的处理结构,对不同结构的算法都有较强的适应能力,尤其适合实时信号处理任务。

10.20 CPLD 在 IGBT 驱动设计中的应用

摘自《电子技术应用》月刊,2000 年第 10 期

该文介绍了一个实用 IGBT 驱动信号转换电路的 CPLD 设计。文中给出了总体设计、通信方案及 CPLD 的实现,最后,还给出了仿真与实验结果。整个 PWM 信号与故障信号传输通路只需 3 片 CPLD 芯片。系统电路的体积大为缩小,从而提高了系统的可靠性。CPLD 的应用缩短了系统的设计周期,降低了开发成本。

10.21 基于 FPGA 的 FIR 滤波器的实现

摘自《电子技术应用》月刊,2000 年第 5 期

该文提出了一种采用现场可编程门阵列器件(FPGA)和窗函数法实现线性 FIR 数字滤波器硬件电路的方案,并以一个十六阶低通 FIR 数字滤波器电路的实现为例,说明了利用 Xilinx 公司 XC4000 系列芯片的设计过程。设计的电路通过软件程序进行了验证和硬件仿真,结果表明,电路工作正确可靠,能满足设计要求。本文设计的低通滤波器可以通过简单地重组滤波器特性参数,而得到高通或带通滤波器。

10.22 用可编程逻辑器件取代 BCD-二进制转换器的设计方法

摘自《测控技术》月刊,2000 年第 3 期

该文阐述了用 PLD 取代 BCD-二进制转换器——74LS184 的设计方法,同时介绍了美国 DATA I/O 公司开发的一种逻辑设计软件——ABEL 语言。文中着重介绍了作者如何解决在编程设计过程中遇到的逻辑方程的乘积项过多问题的经验,具有一定借鉴意义。