摘 要

随着嵌入式技术的日新月异，它在各种领域的应用得到了很大程度的推广。在纺织机械的机电一体化程度不断提高的大前提下，将嵌入式技术引入织机的控制系统来改善性能不失为一个很好的选择。本文针对织机控制系统中提综信息存储模块和人机交互系统两个部分提出了改进方案。通过建立 USB 主机接口来支持对优盘的存储和读写，既省时又省力；嵌入式操作系统+嵌入式 GUI 的人机界面平台，为各种不同的控制系统提供了通用的人机界面设计平台，方便、高效的库函数为界面开发提供了很大的便捷。本文主要包括以下几个部分的内容：

第一，在充分了解 USB 通信协议的基础上，选用以 ISP1160 为 USB 控制芯片的 USB 主机方案。ISP160 是基于 USB2.0 协议的主/从机控制芯片，功能选择通过设置功能寄存器来完成；作为 USB 主机设备，需要实现 ISP1160 的初始化、USB 传输功能以及对 USB 设备的枚举；本设计需要支持优盘，因此还需要添加 Mass Storage 类协议和 FAT16 文件系统。

第二，基于 S3C44B0X 的人机界面硬件系统，它包括了主控制器外围电路、存储器设计、JTAG 接口、液晶接口、触摸屏接口和串口等部分。

第三，选用 μC/OS-II+μC/GUI 的嵌入式操作系统和嵌入式 GUI 作为本人机交互系统的平台，成功移植到 S3C44B0X 的硬件平台上。除了编写液晶驱动和修改其它配置文件外，还添加了针对本控制系统的小汉字库和增加了对触摸屏的支持。

第四，建立了显示任务、触摸屏任务和串口通讯任务，并运用 μC/OS-II 的多任务调度机制和信号量实现了任务间的调度；运用 μC/GUI 提供的接口函数完成了织机控制系统的参数设置、参数显示等 17 个显示画面，并利用回调机制实现画面间的切换。

关键词：嵌入式系统　USB 人机界面 μC/OS-II　μC/GUI　触摸屏

Abstract

With the rapid development of the Embedded Technology,its application was widely promoted in a great many fields. In the condition that the level of mechatronics in textile machinery was enhanced prodigiously, it's a good choice to introduce the Embedded Technology into the primary control system to improve the performance. So the means of seting up USB host to storage the loom data and make a new human-machine interface was put forward. The USB host supports the writing and reading operation on flash, while the human-machine interface platform based on embedded operation system and embedded GUI provideing many library functions to make the screen preparing more easier. This design includes these parts as below:

First, on understanding the agreement of the USB communication, the means based on ISP1160 was choosen. ISP1160 follows the rule of USB 2.0 and has function registers to make it work. As the USB host, the initialization of the ISP1160, the function of USB communication, the enumeration of the USB device are necessary. To support flash device, there should be Mass Storage function and document system of FAT16.

Second, the hardware system of the human-machine interface based on S3C44b0X which includes external circuit of the core, memory circuit, JTAG interface, LCD interface, touchscreen interface and UART interface was setup.

Third, the μC/OS-II and μC/GUI were choosen and transplanted in the hardware system of S3C44b0X. This proceed included programming a LCD driver, a chinese characters store, a touchscreen driver and overwriting some configuration files.

Fourth, three tasks includes displaying, touchscreen, UART communication were created while code flag was provided by the operation system to carry out scheduling of the tasks. Then, using the function interface of the μC/GUI, 17 screens displaying parament watching and configuration were prgrammed and switched in callback rule.

Key Words: Embedd System, USB, HMI, μC/OS-II, μC/GUI, Touchscreen

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：[signature]　　签字日期：2007 年 6 月 14 日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：[signature]　　导师签名：[signature]

签字日期：2007 年 6 月 14 日　　签字日期：2007 年 6 月 14 日

学位论文作者毕业后去向：

工作单位：　　电话：

通讯地址：　　邮编：

第 1 章 绪 论

§1.1 引言

近年来，中国为世界的消费者提供了大量质量精良、价格实惠的纺织产品。同时，中国纺织业促进了各国相关产业的共同发展。过去 5 年，中国累计进口的纺织设备高达 188 亿美元。2005 年，中国进口了 235 亿美元的纺机设备、棉花、羊毛、染化料、化纤等。中国纺织业还吸引了大量的外资进入，目前，外资企业的纺织品出口占中国总出口的 1/3 以上，2005 年取消配额产品的出口增量中有 70%是外资企业完成的。目前，中国已经成为世界纺织品消费第一大国，日益增长的国内消费需求为世界纺织业的发展提供了前所未有的机遇。2005 年，中国人均纤维消费量从 2000 年的 7.5 公斤上升到 14 公斤，衣着类消费金额由 3375 亿元人民币增长至 6826 亿元，年均增幅超过 15%。

问题是在快速发展的同时，我们也看到了这样的数据，在整个纺织生产、销售过程中，中国企业只赚取不到 10%的加工费，许多与品牌、营销等环节相关的市场收益，都被其他国家的企业所分享。导致这个结果的原因首先是研发投入少，缺乏自主创新能力。据 2004 年工业普查数据显示，中国规模以上纺织企业研发投入比例仅为销售收入的 0.25%。其次，中国纺织业利用高新技术及生物资源开发研制的化纤品种和规模也很少。中国纺织业在全球产业链中还处在加工制造阶段，增长方式仍以粗放型为主，出口整体水平较低[1]。

面对机遇与挑战并存的局面，唯有我们抓住目前国内纺织机械水平与国外相比还较落后这个主要矛盾，大力发展具有自主知识产权的高新技术，开发出更多适合于国内形势的纺织机械。

§1.2 纺织机械的机电一体化发展现状[2~3]

近年来，机电一体化技术得到了广泛的应用，促使国内外纺织机械有了较大的发展，如：普遍采用交流变频调速技术，实现人机对话的电子触摸屏应用，计算机控制技术扩大应用并向着智能化的专家系统方向发展，网络和现场总线技术

等，在纺织机械中都有长足的发展。

在化纤机械方面，国内多采用单板机、单片机和步进电机、变频电机等组成的控制系统，而国际先进化纤装置大部分采用了计算机控制技术、交流变频技术、电子触摸屏及现场总线技术形成 CAN 控制系统，国内外技术差距较大。

在棉纺设备方面，国内的梳棉机已采用较理想的前馈-反馈系统，粗纱卷绕机取消了锥轮变速装置，花式纺纱的多单元同步控制均用微机控制器对变频器、直流伺服电机进行控制，与世界先进水平接近。

在织造设备方面，国外基本上采用计算机控制，实现恒张力、恒线速度、电子化、智能化控制，而国内广泛采用的是变频调速、电子寻纬、电子送经、电子卷取、电子选色、电子多臂装置、电子提花、电子计长、伸缩筘自动横移等技术。

在印染后处理设备方面，在国外，纳米技术、生物整理技术、低温等离子处理技术、数码喷射印花技术与印花、染色、整理等工艺紧密结合应用，而国内机电一体化水平也有较大提高，交流变频同步调速技术、计算机控制技术、信息技术与传感技术已普遍应用。

§1.3 嵌入式技术

§1.3.1 嵌入式技术的特点

随着生产和科学技术的发展，特别是微电子技术的迅猛发展，原型技术和设计技术的不断成熟，以及计算机技术迅速地向非计算机领域地渗透，嵌入式系统越来越普遍地得到应用。

嵌入式系统是以应用为中心，以计算机技术为基础，并且软硬件可裁剪，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统。它是一个大系统或电子设备的一部分，工作在一个与外界发生交互并受到时间约束的环境中，在没有人工干预的情况下进行实时控制。其中，软件用于实现有关功能并使其系统具有灵活性和适应性；硬件(处理器、存储器等)用于满足性能甚至安全的需要。嵌入式系统通常具有以下特征[4]：

(1) 完成单一或一组紧密相关的特定功能；

(2) 具有高性能和实时的要求，并且这些要求正不断增加；

(3) 系统作为设备的一部分，其运行一般不需要人工干预；

(4) 系统的电源可靠性和安全性，通常是影响设计的重要因素；

(5) 处理器的选择是嵌入式系统设计的关键部分(包括系统的硬件尺寸、电源以及开发费用)。

§1.3.2 嵌入式技术的发展方向[5]

(1) 对应用开发提供强大支持

应用功能密度的增长是网络时代嵌入式产品的普遍现象。随着因特网技术的成熟、带宽的提高,ICP和ASP在网上提供的信息内容日趋丰富、应用项目多种多样。资源精贵的嵌入式设备,如高性能打印机、贵重实验设备连网共享资源或提供商业性服务;而数量巨大的用户则通过小型电子设备随时随地上网查询消息或取得服务。因此,像电话手机、电话座机及电冰箱、微波炉等嵌入式电子设备的功能不再单一,电气结构也更为复杂。为了满足应用功能的升级,设计师们一方面采用更强大的嵌入式处理器如32位、64位RISC芯片或信号处理器DSP增强处理能力;同时还采用实时多任务编程技术和交叉开发工具来控制功能复杂性,简化应用程序设计、保障软件质量和缩短开发周期。

(2) 为设备网络通信提供标准接口

为适应嵌入式分布处理结构和应用上网需求,面向21世纪的嵌入式系统要求配备标准的一种或多种网络通信接口。针对外部联网要求,嵌入设备必须配Ethenet网口,相应需要TCP/ IP 协议簇软件支持;由于家用电器相互关联(如防盗报警、灯光能源控制、影视设备和信息终端交换信息) 及实验现场仪器的协调工作等要求,新一代嵌入式设备还需具备IEEE1394、USB、CAN、或IrDA通信接口,同时也需要提供相应的通信组网协议软件和物理层驱动软件。为了支持应用软件的特定编程模式,如Web或无线Web编程模式, 还需要相应的协议软件, 如HTTP、WAP等。

(3) 支持小型电子设备，实现小尺寸、微功耗和低成本

为满足这种特性,要求嵌入式产品设计者相应降低处理器的性能,限制内存容量和复用接口芯片,这就相应提高了对嵌入式软件设计的技术要求。如选用最佳的编程模型和不断改进算法,采用Java编程模式,优化编译器性能。因此既要软件

人员有丰富的经验,更需要发展先进嵌入式软件技术,如Java、Web和WAP等。

(4) 提供精巧的多媒体人机界面

嵌入式设备之所以为亿万用户乐于接受,重要因素之一是它们与使用者之间的亲和力,自然的人机交互界面,如司机操纵高度自动化的汽车主要还是通过习惯的方向盘、脚踏板和操纵杆。人们与信息终端交互要求以GUI屏幕为中心的多媒体界面。手写文字输入、语音拨号上网,收发电子邮件以及彩色图形、图像已取得初步成效。目前一些先进的PDA在显示屏幕上已实现汉字写入,短消息语音发布,但离掌式语言同声翻译还有很大距离。

§1.4 人机交互技术[11]

§1.4.1 人机交互技术的发展

人机交互(Human-Computer Interaction,HCI)是研究人、计算机以及它们之间相互影响的技术[7][10]。从人机工程学的界面技术角度讲，人机工程学分为 3 个阶段。第一代人机工程学以人机界面技术(Human/Machine Interface technology)为标志；第二代人机工程学以用户界面技术(User Interface Technology)为标志；第三代人机工程学又称宏观人机工程学(Macro-ergonomics)，是以组织机器界面技术(Organization/Machine Interface Technology)或工作系统设计(Work System Design)的研究、发展和应用为标志[9]。

(1) 人机界面技术

人机界面技术着重研究以人为主题、机械为劳动工具的人机系统，人和机器的合理分工及相互适应的问题。在这个阶段，机器被看作是“人手的延伸”，而人是人机系统中的反应者和控制者。人机界面技术主要是在物理运动层面上研究人和机器各自的特点和功能的匹配问题。在这个阶段，人机关系遵循的是：机器系统应尽量满足使用者的生理、心理、审美以及社会价值观念等条件的要求；在受到机器限制的情况下，要充分 发挥人的可塑性这一特点，使人去适应机器的要求。

(2) 用户界面技术

用户界面技术研究的人机系统已由第一代的人、机电系统转变为用户、计算

机系统。它研究的重点从人机的物理界面转移到人机的认识界面。人机交互技术所要解决的问题就是用户界面技术。

人机界面主要是人-软件系统的界面，简称为用户界面(user interface)。人与机器是两个相对独立且具有智能的认知系统，人和机器的关系既不是完全的控制，也不仅仅是监控，而是互相交换信息，协同完成任务。用户界面技术就是增强人机交互双方的互相了解，从而在一定程度上提高系统的自动化能力。

(3) 组织机器界面技术

随着科技的进一步发展，各种人-机-环境系统也变得越来越复杂。社会技术系统理论认为任何系统都可以看作是一个复杂的社会技术系统，而且系统是开放的、能随人员的变化而作相应变化的灵活系统。在这个理论的基础上，组织机器界面技术就要求从系统各个层次来通盘考虑人的因素，进行系统界面设计。

§1.4.2 嵌入式系统中的人机界面

嵌入式系统中的人机界面设计，不但要遵循一般人机界面设计的设计原则，如用户控制、直接性、可辨性、美观性、反馈性等外，更要考虑到嵌入式系统空间、成本、存储以及通讯等各方面的限制。因此嵌入式系统的人机界面一般为控制板，包括显示器和控制器两部分。

根据人机界面设计的基本原则，结合嵌入式系统的特点，在设计嵌入式系统的人机界面时需要考虑以下几个基本问题：

(1) 信息表现媒体

人类通过视觉接收约83%的信息，它是人类获取信息的主要途径。视觉显示要求用户的随意注意且注意有方向性。一般情况下，嵌入式系统的控制面板显示空间有限、显示能力弱、灵活性差，一次信息显示可选择的媒体种类少且可显示的信息量相当有限。因此较为实用的显示媒体有：图形符号，固定格式的文字(包括字母、数字、有限数量的汉字)。

(2) 信息导航

由于缺乏类似通用系统提供的强大的电子文档和通信带宽的限制，而缺乏联机求助手段，因此常导致许多用户难以使用系统。设法为负责的嵌入式系统提供上下文状态显示，以免用户迷途，甚至造成破坏性结果。

(3) 容错

嵌入式计算机系统的容错性是至关重要的。因为反馈信息和指导信息有限,用户常常不能及时发现操作错误并修正错误,而要等到系统产生动作结果之后才能发现。如果是不可逆的操作,用户当然无法纠正操作错误,于是可能造成破坏性结果。现有的容错措施主要是被动型保护及事后保护,如利用断电保护功能防止信息丢失,利用延时开机防止误启动,利用自动关机以节约能量,利用信息加锁以防止信息误删除,等等。从人机界面设计原则来看,既要允许用户犯错误,又要求系统能识别用户错误并对不合法的操作加以限制等,即要求在人机对话的较低层次上校验用户的输入信息的合法性。

(4) 交互技术和交互设备

交互技术决定了人机界面的外观和感觉。目前嵌入式系统的输出装置的可编程能力较弱，并且空间有限，难以实现通用系统所使用的高级交互技术。一般，用户的交互控制包括发出命令、输入数值和系统状态设置等。

§ 1.4.3 嵌入式 GUI

GUI 是 Graphical User Interface 的简称，即图形用户接口面。对嵌入式环境的多样性，目前国内外已由不少的嵌入式 GUI，比较成熟的嵌入式 GUI 系统有 Qt/Embedded、MicroWindow、MiniGUI、WinCe 等[8]。

(1) MicroWindows

MicroWindows是Century Software设计的用于带小型显示单元的微型设备的项目。MicroWindows体系结构是基于客户机/服务器的，并且具有分层设计。最底层是屏幕和输入设备驱动程序来与实际硬件交互。在中间层,可移植的图形引擎提供对线的绘制、区域的填充、多边形、裁剪以及颜色模型的支持。在最上层，MicroWindows支持两种API：Win32/WinCE API实现，称为MicroWindows；另一种API与GDK非常相似，称为Nano- X。MicroWindows的主要特色是提供了类似X 的客户/服务体系结构，并提供相对完善的图形功能。但MicroWindows，性能不高，特别在图形引擎中有许多低效的算法。

(2) Qt/Embedded

Qt/Embedded是Trolltech公司开发的用于嵌入式Linux的图形用户界面系统。它

以原始Qt为基础，针对嵌入式环境做了一些调整以适应嵌入式环境。Qt Embedded 通过Qt API与Linux I/O设施直接交互。Qt Embedded面向对象的体系结构使代码结构化、可重用。但Qt/Embedded用C++函数库，效率不高。此外，其结构过于复杂，很难进行系统裁减、扩充和移植。

(3) MiniGUI

MiniGUI是一种在嵌入式系统中提供图形及图形用户界面支持的中间件技术，早期由魏永明先生主持和开发，现由北京飞漫软件技术有限公司维护并开展后续开发。MiniGUI是在Linux控制台上运行的、基于SVGALib和LinuxThread库的多窗口图形用户界面支持系统。为适应不同的环境，MinuGUI可以配置成三种不同的运行模式:MiniGUI- Threads、MiniGUILite和MiniGUI- Standalone。MiniGUI在GDI层面上的设计不够模块，由于它将画笔、画刷等GDI对象的属性都整合到DC结构中，缺乏GDI对象的概念，同时画笔、画刷只支持实型,功能不够强大。

(4) WinCE

WinCE 操作系统是微软专门为嵌入式市场专门设计的操作系统。当然它的 GUI 系统也秉承了 Windows 操作系统华丽的窗口界面风格，只要硬件系统支持，用户就可以看到与 PC 上 Windows 类似的界面。而正是这样的特点，它对硬件系统的要求也较为苛刻，一般的应用不推荐使用。

§ 1.5 本课题的研究意义、研究内容和创新点

§ 1.5.1 本课题的研究意义

近年来，各种嵌入式技术的不断进步，凭借其自身高性能、小体积和高性价比的优势，它的应用正得到不断的创新和拓展。在充分分析了原有织机系统控制系统的基础上，本课题提出了改进的方案，包括增加USB主机接口实现对USB设备的支持和设计一个基于嵌入式GUI的人机交互界面来取代原有的商用人机交互界面，以此来获得一个功能更加完善的织机控制系统。

由于USB存储设备具有容量大、体积小、携带方便等优点，使用已经非常普及，使用USB接口作为织机提综信息的存储设备显然是理想的选择。

随着嵌入式技术的成熟，原本相对简单的嵌入式系统也要寻求着自我完善，

而人机交互则是很重要的一个方面。作为专门面向嵌入式系统的图形用户接口，嵌入式GUI也不断地得到完善，本设计正是通过搭建这样的一个嵌入式系统的平台，利用其中丰富的接口函数来完成所需要的人机界面显示和交互功能。

§1.5.2 课题研究内容

本课题涉及嵌入式系统、嵌入式GUI、LCD驱动、触摸屏等内容，主要进行的工作如下：

(1) 了解和比较了各种嵌入式操作系统；

(2) 分析和设计了USB主机接口的硬件接口电路；

(3) 设计了基于ISP1160的USB主设备的软件；

(4) 分析和比较了各种嵌入式GUI的特点及应用场合，并确定适合本课题的方案；

(5) 设计了人机界面的硬件系统，包括主控制器及存储器，LCD接口，触摸屏接口等；

(6) 分析和设计了LCD的驱动程序；

(7) 分析和设计了触摸屏的驱动程序；

(8) 将μC/OS-II移植到人机界面平台；

(9) 将μC/GUI移植到μC/OS-II上，添加LCD和触摸屏的驱动，并编写了小汉字库，提供对系统的中文支持；

(10) 利用μC/GUI的接口函数编写画面；

(11) 设计基于μC/OS-II多任务机制的人机交互系统，包括GUI的显示任务、触摸屏任务和串口通信任务；

(12) 测试整个系统；

(13) 总结课题，提出改进方案。

§1.5.3 本课题的创新点

(1) 将移动存储性能优越的USB技术引入织机控制系统，改变了原先提综信息等织机数据的存储方式。利用能和PC通用的U盘来替代专门设计的EPROM大大地提高了系统的通用性能和效率。

(2) 将嵌入式技术引入织机控制平台的人机界面设计。利用性能优越的嵌入式操作系统和嵌入式GUI，构造出一套可以针对不同的控制系统的人机界面平台。特别是嵌入式GUI提供的接口函数种类齐全、性能优越、使用方便的优点，为人机界面的显示部分提供了强大的支持。

(3) 利用触摸屏技术作为人机交互的输入设备。触摸屏简单明了的输入方式不但能提高人机交互系统的整体水平，而且它并不复杂的电路设计也能简化传统的按键输入等方式的硬件和结构设计。

第 2 章 USB 主机接口设计

作为对原嵌入式织机控制系统的改进，USB 主机的设计能够更方便地实现织机信息的存储。本章主要是介绍一个基于 USB 主机芯片 ISP1160 的硬件、软件设计，以及对一些关键的 USB 通信的概念进行简要的介绍。

§ 2.1 USB 接口电路设计

§ 2.1.1 ISP1160 简介

ISP1160 是 Philip 公司生产的一款嵌入式 USB 主机控制器(HC)，并遵循 USB2.0 规范，支持全速(12Mb/s)及低速(1.5Mb/s)两种数据传输模式。它提供两个下行端口，每一个下行端口都有一个过流(OC)检测输入引脚及电源开关控制输出引脚。HC 的下行端口能与 USB 设备及拥有 USB 上行端口的 USB 集线器连接。

ISP1160 具有如下特征[12]：

- 遵循 USB2.0 规范；
- 支持全速(12Mb/s)及低速(1.5Mb/s)两种数据传输模式；
- 适配器遵循 USB 开放主机控制器接口规范 1.0a 版；
- HC 可选择一个或两个下行端口；
- 高速并行接口支持大部分通用微处理器及 RISC 处理器；
- 微处理器及 HC 之间的数据传输率最大为 15MB/s；
- 支持单周期及 DMA 突发模式操作；
- 主机控制器内嵌 FIFO 式 RAM 缓冲区(4KB)；
- 终端带有双缓冲区，以增加数据传输量，并可稳定实时数据同步传输；
- HC 下行端口可通过软件方式，选择内部 15 欧下拉电阻；
- 5V 或 3.3V 电压源。

§ 2.1.2 ISP1160 接口电路

本设计采用的 ISP1160 为 LQFP64 封装。D0~D15 是 16 位数据总线，与

S3C4510 的 D0~D15 相连；A0 是地址输入；INT1 为 HC 中断引脚，连接 ExINT4[13]；/CS 为 ISP1160 的片选；/WR 和/RD 分别为写使能和读使能；/RESET 为硬件复位引脚；H_WAKEUP，H_SUSPEND 用于控制 ISP1160 唤醒和获取挂起状态；/H_PSW1，/H_PSW2 分别为下行端口 1 和 2 的电源开关控制输出；H_DM1,H_DP1,H_DM2,H_DP2 分别为 USB 数据线；H_OC1 和 H_OC2 为下行端口 1 和 2 的过流探测输入引脚；由于不使用 DMA 模式访问 ISP1160 内部的 FIFO 缓冲区 RAM，DMA 请求引脚 DREQ 悬空，DMA 模式使能引脚 DACK1,DACK2 接高电平。具体连接如图 2.1 所示[12]。

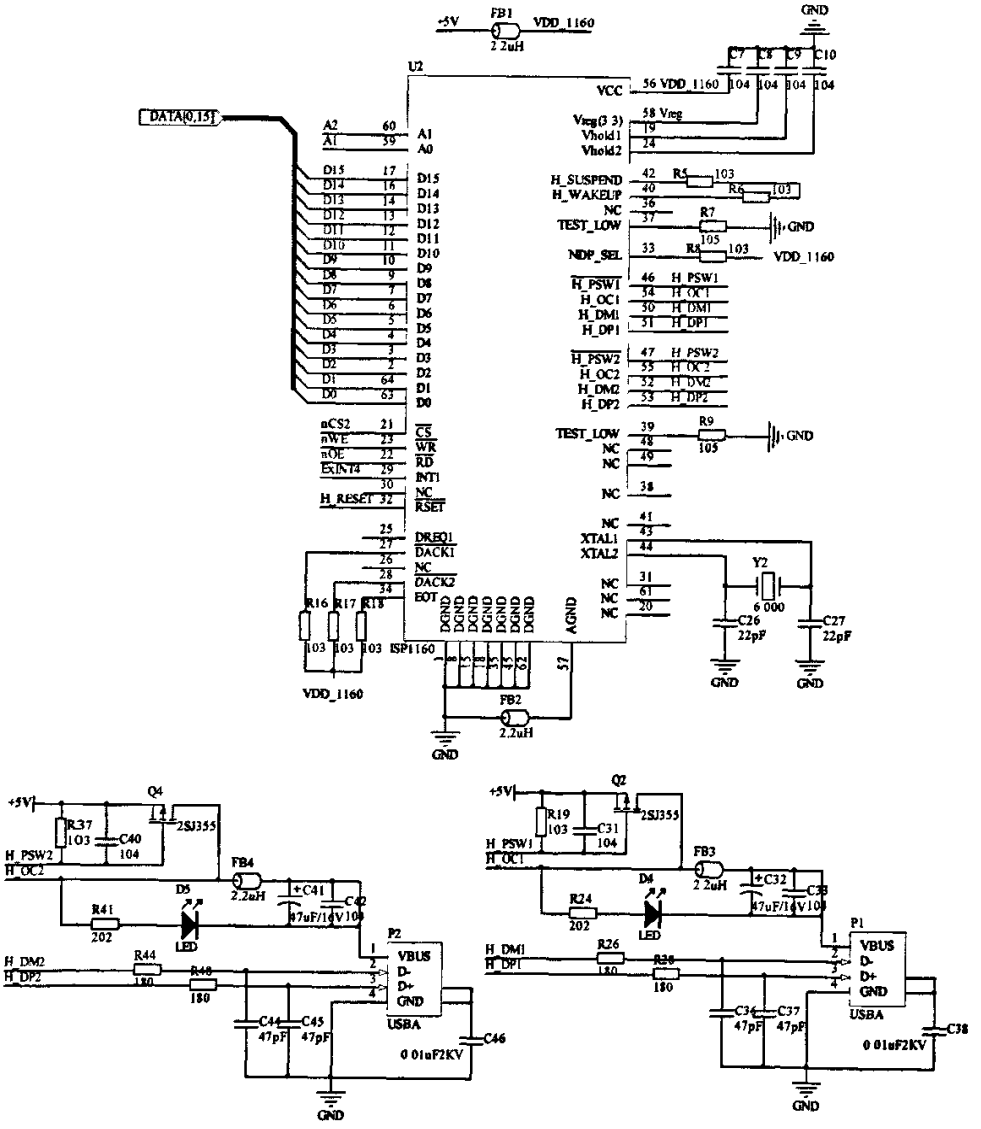

图 2.1 ISP1160 连接图

§ 2.2 USB 通信概述

§ 2.2.1 USB 技术特点

1994 年 11 月，Compaq(康柏), DEC, IBM, Intel, NEC, 微软以及 Northern Telecom(北方电讯)，为了解决 PC 机外围设备拥挤的问题，提高设备的传输速度，提出了通用串行总线 USB 的概念。USB 是一种 PC 机的外挂总线，其目的是在一个 PC 机上，能够挂接鼠标、键盘、声卡、Modem、打印机、扫描仪等更多的设备[14]。

由于 USB 技术较复杂，需要硬件和软件的支持，因此直到 Windows98 推出以后，USB 技术才得到实现，随计算机技术的迅速发展逐渐显现出其市场价值和技术优势，使人们真正领略了 USB 的方便与快捷。

USB 之所以有着巨大的魅力，主要是它具有许多其他总线无法比拟的优点：

(1) 速度快

USB 有高速和低速两种方式，主模式为高速模式，速率可达 12Mbps，另外为了适应一些不需要很大吞吐量和实时性很强的设备，如鼠标等，USB 还提供低速方式，速率为 1.5Mb/s。相比之下，普通串口数据传输率仅有 115kbps~230kbps，标准并口的数据传输率也才达到 1Mbps。新推出的 USB2.0 协议，在理论上其传输率可以达到 480Mbps。

(2) 设备安装和配置容易

USB 设备支持即插即用 PnP (Plug and Play)和热拔插，系统对其进行自动配置，不再占用中断资源或者 DMA 资源，彻底抛弃了过去的跳线和拨码开关设置。USB 为接缆和连接头提供了单一模型，解决了外设越来越多造成的插槽紧张问题。

(3) 易于扩展

用 USB 连接的外围设备数目最多达 127 个，共 5 层。所谓 5 层是指从主装置开始可以经由 4 个集线器进行菊花链接。标准 USB 电缆长度为 3 米(低速可达 5 米)。通过 Hub 或中继器可以使外设距离达到 12 米。

(4) 采用总线供电方式

USB 总线提供最大达 5V/500mA 电流，对于功耗较小的设备来说，不需要为

设备提供专用电源，使用极其方便。当然，也采用自供电方式。

(5) 使用灵活

USB 总线具有四种传输模式：控制传输(control)、同步传输(synchronization)，中断传输(interrupt)和批量传输(bulk)，可适应不同设备传输速率的需要。

(6) 产品成本低

USB 功能模块价格低廉，应用 USB 设计的产品成本低具有市场竞争优势。目前，多家厂家都有USB产品。在国内应用较多的USB的控制器，主要有National Semiconductor 的 USBN9602 系列和 Philips 的 PDIUSBDII 等系列。

§ 2.2.2 USB 通信相关概念[15]

USB 最基本的数据单元是包，每一个包基本上包含了一个完整的 USB 信息。按照包在整个 USB 数据传输中的作用，包可分为 4 类：令牌包、数据包、握手包和特殊包。区分不同的包，就要将它们分解成更小的单元“域”。域又被分为 7 类：同步序列域、包标识域、地址域、端点域、帧号域、数据域和 CRC 校验域。

以包为基础，USB 定义了 4 种数据的传输类型：控制传输、中断传输、批量传输和同步传输。而传输的过程通常是包括多个数据交换的过程，每次传输数据的一部分，这每一次的数据交换叫做“事务”。每种传输方式都由很多个事务来完成，每一笔事务由底层包组成[15]。

USB 通信的主要概念如下：

(1) 域

7 种域各有不同的作用。同步序列域(SYNC)用于本地时钟与输入信号的同步，代表一个包的起始。

包标识域(PID)紧跟在同步域之后，表明包的类型和格式，并作为包的错误检测手段的一种，它是 USB 软件机制最先收到并处理的包的内容。在本文所要实现的 USB1.1 协议中使用了 10 种，如表 2-1 所示。

(2) 包

包是最基本的 USB 数据单元，由一系列的域组成。根据 PID 的不同，令牌包分为输入包(IN)、输出包(OUT)、设置包(SETUP)和起始帧包(SOF)。IN、OUT

和 SETUP 三种包的结构相同，都包括同步域、标识域、地址域、端点域和校验域；SOF 在前三者的基础上增加了 11 位的帧起始域，用于代表帧号，而不包括地址域和端点域。

数据包分为 DATA0 和 DATA1 两种包，两者数据格式一样，用法也相同。当 USB 发送数据包时，如果一次发送的数据长度大于相应端点的容量，就需要把数据分批发送。这样第一个数据定义为 DATA0，第二个为 DATA1，交替完成。

表 2-1 标识域类型

数据包类型	标识域名称	标识符值 PID[3:0]	标识域意义
令牌包	输出(OUT)	0001	启动主机到设备的数据传输，并包含设备地址和端点号
	输入(IN)	1001	启动设备到主机的数据传输，并包含设备地址和端点号
	起始帧(SOF)	0101	一个帧的开始，包含相应帧号
	设置(SETUP)	1101	启动通过控制管道进行设置的数据传输，并包含设备地址和端点号
数据包	数据 0(DATA0)	0011	偶数据包
	数据 1(DATA1)	1011	奇数据包
握手包	确认(ACK)	0010	没有接到错误的数据包
	无效(NAK)	1010	接收端无法接收或发送端无法发送
	错误(STALL)	1110	端点被禁止或不支持控制管道请求
特殊包	前导(PRE)	1100	启动下行端口的低速速设备传输

握手包仅有同步域和标识域组成，用于报告数据的传输状态，3 种握手包各有不同的意义。

确认包 ACK：用于标识数据包被成功接收，具体如下：

- 标识域 PID 被正确接收；
- 并且没有发生数据位错误；
- 没有发生数据域的 CRC 校验错误。

无效包 NAK，主要表示：

- 在接收主机发来的 OUT 命令后，设备无法接收数据；

- 接到主机的 IN 命令，但设备没有数据发送给主机。

错误包 STALL，主要表示：

- 设备无法发送数据；
- 设备无法接收数据；
- 不支持某一种控制管道的命令。

(3) 数据的传输类型

USB 的传输是 USB 面向用户的、最高级的数据结构。USB 定义了 4 种数据传输的类型，即控制传输、中断传输、批量传输和同步传输。运用这 4 种传输方式，可以实现不同类型数据的传输。批量传输通过错误检测和重试的方法保证数据在主机和功能设备之间无错传输，它分为 3 个阶段处理：令牌、数据和握手包；控制传输包括两个处理阶段：建立和状态，也可以在两个阶段之间再包含一个数据阶段；中断处理可以由 IN 或 OUT 传输组成；同步处理有令牌和数据阶段，但没有握手阶段。

(4) 数据流模型

端点(ENDP)实际上是设备硬件上具有的一定大小的数据缓冲器，它主要特性有数据传输方式(用于 IN 事务、OUT 事务和 SETUP 事务的端点等)、总线访问频率、带宽、端点号和数据包最大容量等。

管道(Pipe)不像端点那样具有实在的意义，它是一种逻辑上的概念，是指主机和设备端点之间的连接。管道就是数据传输的管道，代表主机的数据缓冲区与设备端点之间交换数据的能力。

§ 2.3 USB 主机软件设计[17][24][25]

USB 主机软件结构包括主机控制器驱动程序(HCD)、USB 驱动程序(USBD)和客户端软件。HCD 和 USBD 将被载入 USB 主机栈中。在 USB 主机栈中，均需要支持 USBD 和 HCD。HCD 可以访问 USB 主机控制器硬件，并通过设置主机控制器中的可编程寄存器对主机控制器进行驱动。客户端软件可以是应用代码和 USB 类驱动[16]，本 USB 主机接口需要支持 Flash 闪盘的读写，则需要实现 Mass Storage 类协议和 FAT16 文件系统。

§ 2.3.1 主机控制器驱动(HCD)

ISP1160 提供给主机驱动可操作的功能部件主要有：HC 控制和状态寄存器、ATL 缓冲区和 ITL 缓冲区。HC 内部 FIFO 缓冲区的 4KB RAM 可被分为 ATL(应答传输列表)缓冲区和 ITL(同步传输列表)缓冲区，其中 ITL 缓冲区应用于同步传输中，本主机系统不需要实现，因此不对其分配 RAM 和操作。

PTD(PHILIPS Transfer Descriptor)是 HCD 和 ISP1160 之间传递的一种数据格式，它提供 USB 通信的相关信息，如指令、状态、USB 数据包等。ATL 缓冲区中的 PTD 通过 HCD 建立的传输调度机制实现数据传输，具体分为前台和后台操作。

(1) 主机硬件初始化

当 ISP1160 上电后，HCD 必须完成硬件初始化步骤以使主机控制器进入操作状态。硬件初始化的过程，实质上是对主机控制器寄存器初始化的过程。流程如下所示：

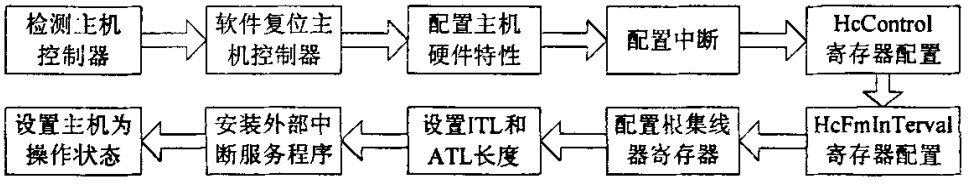

图 2.2 ISP1160 硬件初始化流程

其中各个寄存器的配置值可根据需求和数据手册进行设置；安装的外部中断服务程序用于向主控制芯片 S3C4510B 的 ExINT4 发送中断请求。

(2) 前台操作

前台操作是指程序以用户指定以用户指定的流程运行的操作，包括初始化传输描述符、创建传输描述符、设置传输描述符和关闭传输描述符等，其中传输描述符是用于前后台之间进行通信和交换的数据结构。

传输描述符初始化包括传输调度链表、插入调度链表和 ATL 缓冲区的初始化；创建传输描述符时需要 USBD 提供用于传输的端点描述符结构指针 epi_ptr；设置传输描述符包括了传输描述符结构指针，传输数据缓冲区指针，传输的数据长度和传递方向等。具体实现的函数声明如下：

```
void   Init_transfer_instance(void);
```

```
void    Close_transfer_instance(transfer_instance *tr_instance_ptr);
transfer_instance    *Create_transfer_instance(endpoit_info *epi_ptr);
unsigned char    Set_transfer_instance(transfer_instance *tr_inst_ptr, unsigned char
                                *buffer_ptr,unsigned short length,unsigned char direction);
```

其中 epi_ptr 为端点描述数据结构指针，tr_inst_ptr 是传输描述符数据结构指针，buffer_ptr 是数据缓冲区指针，length 是传输的数据长度，direction 是传输方向。

(3) 后台操作

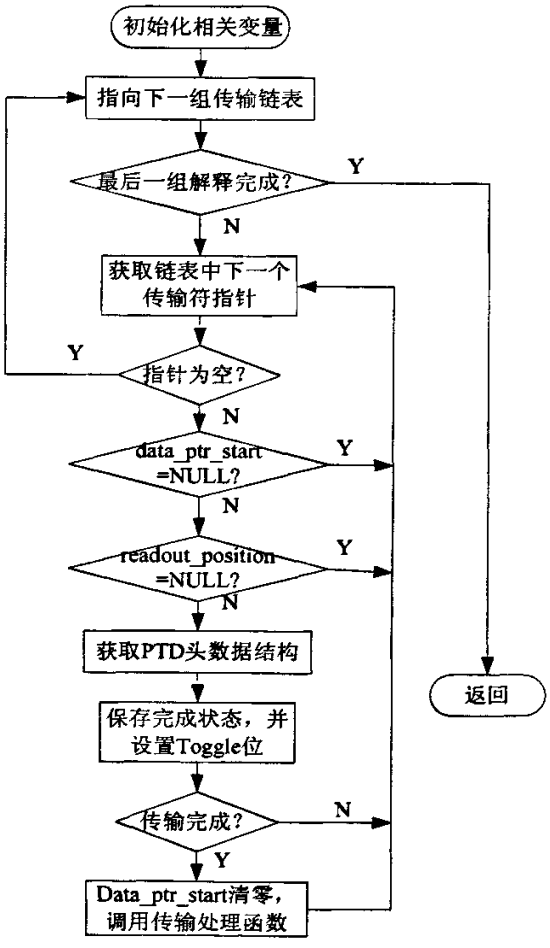

图 2-3 解析 ATL 缓冲区

后台操作是指由中断服务程序处理的操作，主要包括解析 ATL 缓冲区和调度传输描述符。HCD 在调度传输时，是通过传输描述符与 USB 主机进行通信的。在调度传输描述符时，将每个传输描述符数据结构构成 PTD 数据结构，然后将

所有被调度的 PTD 写入主机控制器的 ATL 缓冲区。当传输完成或完成一帧的调度时，HCD 需读出 ATL 缓冲区，并解析被调度的传输描述符的 PTD 数据结构，还要将完成的状态和接收到的数据存放在传输描述符指定的地址中。解析 ATL 缓冲区的流程如图 2-3 所示。

调度传输描述符时，首先扫描调度传输链表中的每一个传输描述符，若满足调度要求，则将传输描述符构造成 PTD 数据结构并存放在 ATL 映像缓冲区准备调度。传输链表中包括中断传输链、控制传输链和批量传输链 3 种指针，扫描次序为：中断传输、控制传输、批量传输。调度传输描述符流程如图 2-4 所示：

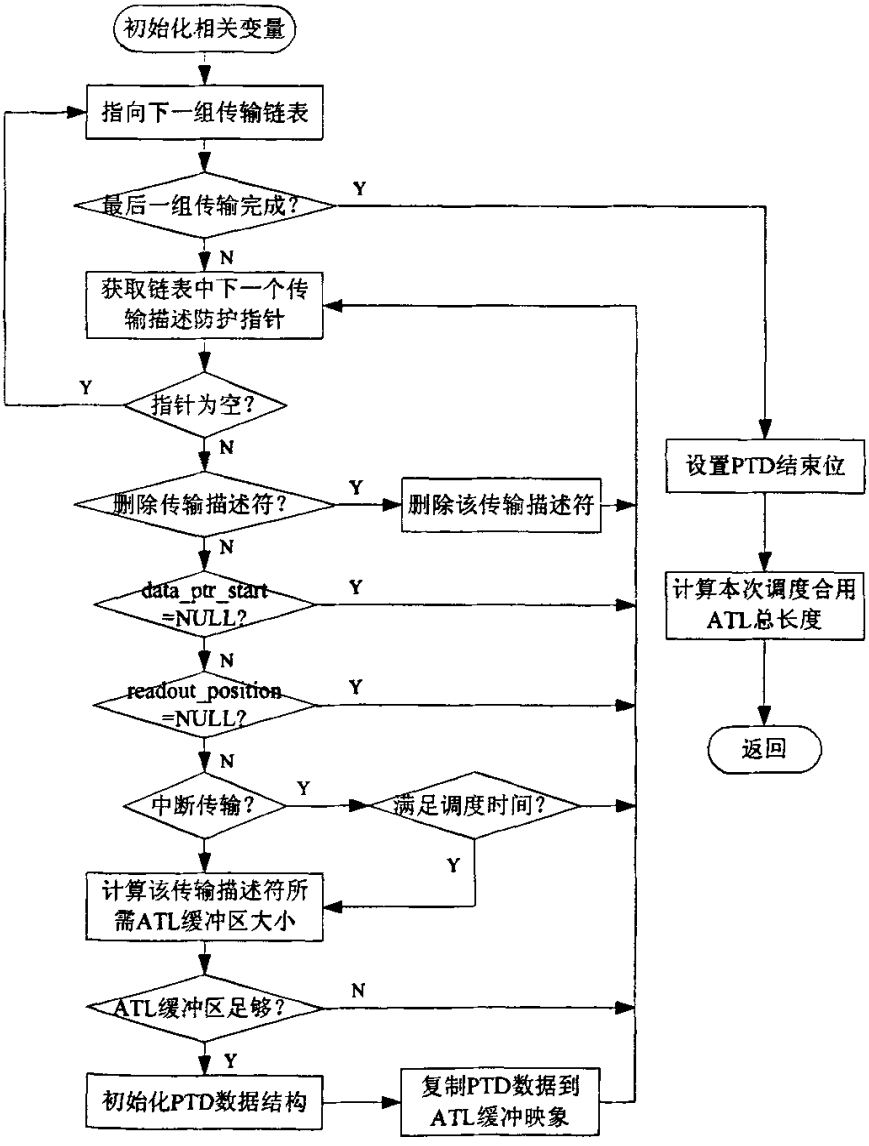

图 2-4 调度传输描述符

§ 2.3.2 USB 驱动(USBD)

USB 驱动程序(USBD)处理客户软件发送命令到设备和客户软件与 USB 外设的数据传输。它主要包括设备的枚举、数据传输的实现等[18]。

(1) 设备枚举

在 USB 设备连接到主机之后，必须通过控制传输来交换信息、设备地址和读取设备的描述符，这样主机才能识别该设备，并且只有在主机对设备进行重新配置后，设备才能正常工作，此过程在 USB 协议中称为枚举。具体的流程和所定义的函数如图 2-5 所示：

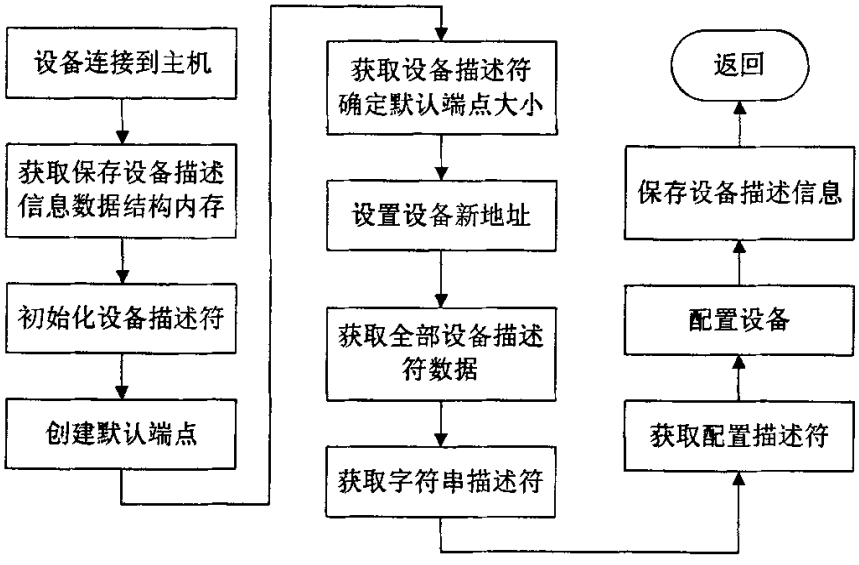

图 2-5 USB 设备枚举

(2) 数据传输

HCD 使用 PTD 处理主机与 USB 外围设备之间的数据交换，并构建传输描述符来实现调度传输，提供了接口函数供 USBD 来处理控制传输、中断传输和批量传输。函数声明如下：

```
unsigned char control_transaction(unsigned char direction, unsigned char *data_ptr,
                                  unsigned short *size_ptr, endpoint_info_ptr epi_ptr);
unsigned char interrupt_transaction(unsigned char direction, unsigned char *data_ptr,
                                  unsigned int *size_ptr, transfer_instance hTrInstance);
unsigned char bulk_transaction(unsigned char direction, unsigned char *data_ptr,
```

unsigned int *size_ptr, transfer_instance *hTrInstance);

其中 direction 是传输方向，data_ptr 是数据缓冲区指针，size_ptr 是数据长度指针,epi_ptr 是端点描述结构指针，hTrInstance 是传输描述符指针。

控制传输与中断传输和批量传输不同,一次控制传输具有 2 或 3 个处理阶段：建立(SETUP)，数据(DATA，可无)，状态(STATUS)。每个阶段的 PTD 不能同时存放 ATL 缓冲区中，这样，控制传输就需要在不同的帧按次序处理建立阶段、数据阶段和状态阶段的 PTD。

§ 2.3.3 Mass Storage 类协议

主机中的类协议设备驱动和用户软件构成主机系统的客户层，直接面向不同的 USB 设备。而不同的设备分属不同的设备类，USB 主机针对不同的设备需要具有不同的类协议设备驱动程序。由于 USB 设备种类繁多,不可能在嵌入式 USB 主机上实现所有的设备类协议，因此，一般只实现某一特定的类协议。本设计中，需要实现对 Flash 闪盘的读写，只要支持 Mass Storage 类协议[22]。

(1) Mass Storage 类概述

Mass Storage 类主要用于为软磁盘接口、ATA 接口、IDE 硬盘接口及 Flash 存储器等设备建立的 USB 接口。它的特点是数据交换量大，有可能直接涉及文件的各种操作，并且支持不同的数据存储载体接口本身的一些操作命令。

Mass Storage 类包含 4 个子类，CBI Transport 传输协议，Bulk-Only Transport 传输协议，ATA Command Block，UFI Command Specifcation。前两个规范定义了数据/命令/状态在 USB 上的传输方法，Bulk-Only 传输规范仅仅使用 Bulk 端点传送数据/命令/状态，CBI 传输规范则使用 Control/Bulk/Interrupt 三种类型的端点进行数据/命令/状态传送；后两个规范定义了存储介质的操作命令。ATA 命令规范用于硬盘，UFI 命令规范是针对 Flash 闪盘[20]。

根据 Mass Storage 四个子类的应用场合，本设计使用的是 Bulk-Only 子类传输协议(子类代码 0x08)和 UFI 命令(子类代码 0x04)。

(2) Bulk-Only 单批量传输协议

Bulk-Only 传输协议没有使用中断和控制端点，仅仅使用 Bulk 批量端点来进行命令、数据和状态的传输。其传输流程图 2-6 所示：

其中 CBW 和 CSW 分别称为命令块封包和命令状态封包。命令块封包用于用户层向 USB 系统层发送命令，用于与 USB 设备建立连接，读取设备的各种信息；命令状态封包则是由系统层向用户层，用于指示发送的命令执行是否成功。

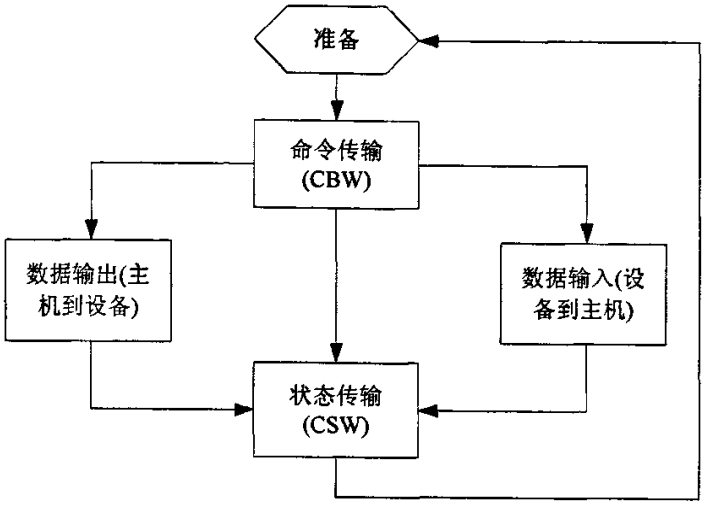

图 2-6 Bulk-Only 流程[21]

CBW 和 CSW 的数据结构如下：

```
typedef struct _COMMAND_BLOCK_WRAPPER{
    unsigned char       dCBW_Signature_0;
    unsigned char       dCBW_Signature_1;
    unsigned char       dCBW_Signature_2;
    unsigned char       dCBW_Signature_3;
    unsigned int        dCBW_Tag;
    unsigned char       dCBW_DataXferLen_0;
    unsigned char       dCBW_DataXferLen_1;
    unsigned char       dCBW_DataXferLen_2;
    unsigned char       dCBW_DataXferLen_3;
    unsigned char       bCBW_Flag;
    unsigned char       bCBW_LUN;
    unsigned char       bCBW_CDBLength;
    unsigned char       cdbRBC[16];
    } COMMAND_BLOCK_WRAPPER, *PCOMMAND_BLOCK_WRAPPER;
```

```
typedef  struct _COMMAND_STATUS_WRAPPER{
    unsigned char       dCSW_Signature_0;     // 0x55
    unsigned char       dCSW_Signature_1;     // 0x53
    unsigned char       dCSW_Signature_2;     // 0x42
    unsigned char       dCSW_Signature_3;     // 0x53
    unsigned int        dCSW_Tag;             // 与 CBW 一致
    unsigned char       dCSW_DataResidue_0;
    unsigned char       dCSW_DataResidue_1;
    unsigned char       dCSW_DataResidue_2;
    unsigned char       dCSW_DataResidue_3;
    unsigned char       bCSW_Status;
    } COMMAND_STATUS_WRAPPER, *PCOMMAND_STATUS_WRAPPER;
```

在进行单批量传输时，命令块包、数据和命令块状态包都是通过批量输入和批量输出端点传输的，因此，命令传输(CBW)和状态传输(CSW)需要调用 USBD 提供的批量传输函数 bulk_transaction()来传输数据。命令传输(CBW)发送预先设置好的 CBW 命令包，状态传输则接收设备返回给主机的 CSW 状态包。整个过程构成了单批量传输处理函数 BulkOnlyComHandle()。

(3) UFI 子类命令

UFI子命令块包含在Bulk-Only传输协议传送的CBW命令封包中的cdbRBC[16]字段。常用的 USB Flash 存储器等大容量设备使用 SCSI-2 子类命令块协议，USB 软驱则使用 UFI 子类命令块协议。由于 UFI 子类的命令块是基于 SCSI-2 的，因此UFI 的常用指令域 SCSI-2 相应的指令相似，包括测试单元准备(Test Unit Ready 0x00)，查询(Inquiry 0x12)，读操作(Read(10) 0x28)，写操作(Write(10) 0x2A)，获取容量(Read Capacity 0x25)。实现的函数声明如下：

```
void Mcom_INQUIRY(uint8 *ComBuffPtr);
void Mcom_Read10(uint8 *ComBuffPtr, uint32 LBA, uint16 TrBlocks);
void Mcom_Write10(uint8 *ComBuffPtr, uint32 LBA, uint16 TrBlocks);
void Mcom_ReadCapacity(uint8 *ComBuffPtr);
```

其中 ComBuffPtr 是命令缓冲区指针，LBA 是逻辑块地址，TrBlocks 是读取或

者写入的逻辑块数。

§ 2.3.4 FAT16 文件系统

大部分的 Flash 闪盘采用 FAT16 文件系统，因此 USB 主机系统还需要建立 FAT16 文件系统并通过 Mass Storage 类协议中的 UFI 命令与 U 盘进行联系，实现对 U 盘的读、写等一系列功能，并提供实现用户功能的文件处理函数[19]。

由于项目时间的限制，本系统的 FAT16 文件系统通过修改周立功的软件开发包实现。

第 3 章 人机界面硬件设计

本人机界面的硬件部分是基于 ARM7TDI 核心的 S3C44B0X 设计的，主要包括了 S3C44B0X 的外围电路、存储器、液晶接口、触摸屏接口、串口以及供电模块等几个部分。

§3.1 S3C44B0X 简介

Samsung 公司推出的 16/32 位 RISC 处理器 S3C440X 为手持设备和一般类型应用提供了高性价比和高性能的微控制器解决方案。为了降低成本，S3C44B0X 提供了丰富的内置部件，包括：8KB cache，内部 SRAM，LCD 控制器，带自动握手的 2 通道 UART，4 通道 DMA，系统管理器(片选逻辑，FP/EDO/SDRAM 控制器)，代用 PWM 功能的 5 通道定制器，I/O 端口，RTC，8 通道 10 位 ADC，IIC-BUS 接口，IIS-BUS 接口，同步 SIO 接口和 PLL 倍频器。

S3C44B0X 采用了 ARM7TDMI 内核，0.25um 工艺的 CMOS 标准宏单元和存储编译器。它的低功耗精简和出色全静态设计特别适用于对成本和功耗敏感的应用。同样 S3C440BX 还采用了一种新的总线结构，即 SAMBAII(三星 ARM CPU 嵌入式微处理器总线结构)。

S3C44B0X 的杰出特性是它的 CPU 核，是由 ARM 公司设计的 16/32 位 ARM7TDMI RISC 处理器(66MHZ)。ARM7TDMI 体系结构的特点是它集成了 Thumb 代码压缩器，片上的 ICE 断电调试支持，和一个 32 位的硬件乘法器。它还通过全面的、通用的片上外设，大大减少了系统电路中除处理器以外的元件配置，从而最小化系统成本。各种片上功能如下：

- 2.5V ARM7TDMI 内核，带有 8K 高速缓存器(SAMBAII 总线体系机构，主频高至 66MHZ)；
- 外部存储器控制器(FP/EDO/SDRAM 控制，片选逻辑)；
- LCD 控制器(最大支持 256 色 STN，LCD 具有专门 DMA)；
- 2 通道通用 DMA、2 通道外设 DMA 并具有外部请求引脚；
- 2 通道 UART，带有握手协议(支持 IrDA1.0，具有 16-byte FIFO)/1 通道 SIO；

- 1 通道多主 IIC-BUS 控制器；
- 1 通道 IIS-BIS 控制器；
- 看门狗定时器；
- 71 个通用 I/O 8 通道外部中断源；
- 功耗控制：具有普通，慢速，空闲和停止模式；
- 8 通道 10 位 ADC；
- 具有日历功能的 RTC；
- 具有 PLL 的片上时钟发生器。

其内部功能框图如图 3-1 所示。

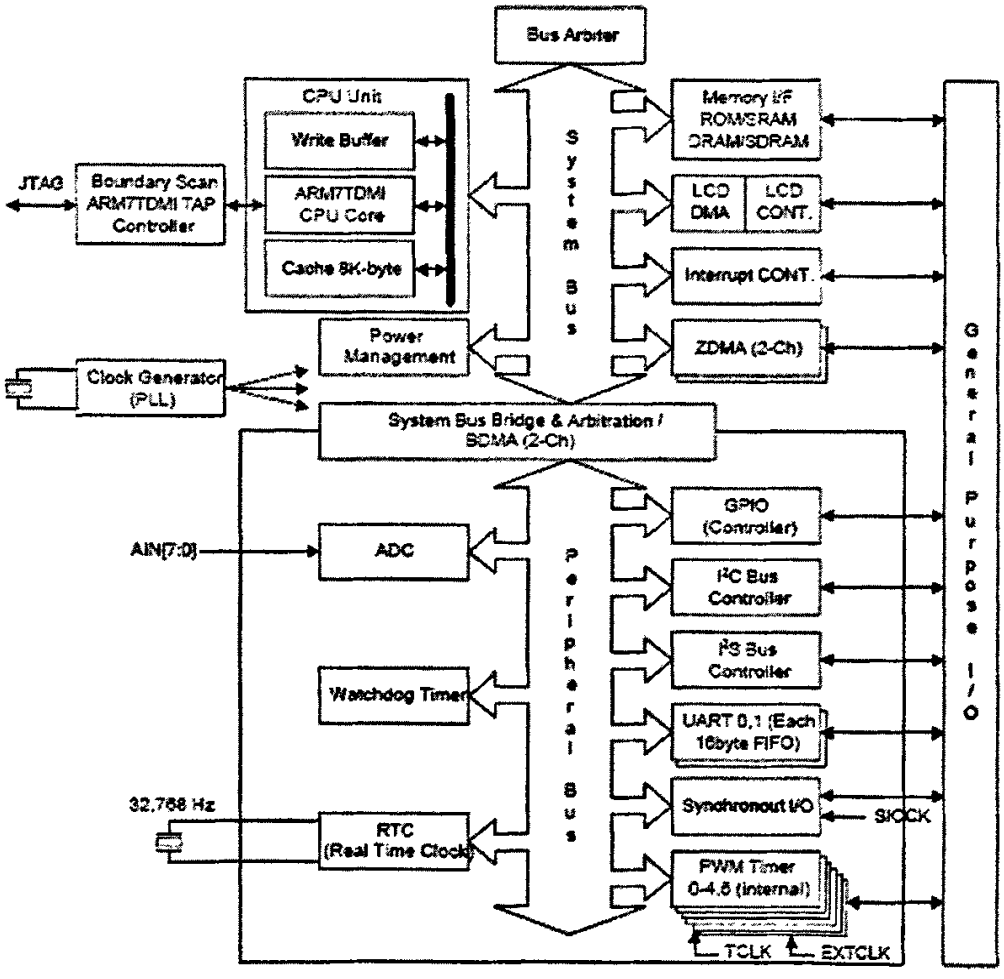

图 3-1 S3C44B0X 内部功能结构[26]

§3.2 人机界面平台的硬件结构

§3.2.1 硬件平台总体结构

在充分利用 S3C44B0X 片内资源和考虑到人机界面平台需要，本设计除了需要运行系统必要的电源和外部晶振电路，片外存储器 8M 的 SDRAM、8M 的 NORFLASH 和调试系统用的 JTAG 电路以外，还需要设计与 258 色 STN LCD 相连的 20 脚液晶接口，与 4 线电阻式触摸屏相连的 4 脚触摸屏接口，与织机主控制器通讯用的 UART 接口以及一些其它的辅助电路[33]。基本结构如图 3-2 所示。

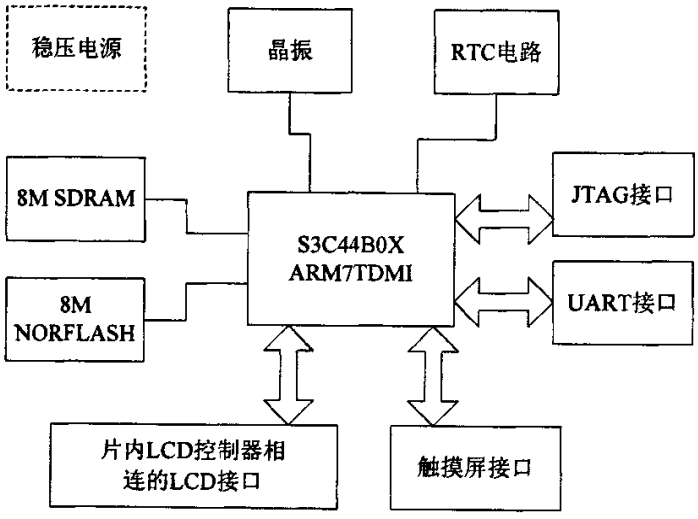

图 3-2 人机界面硬件结构

§3.2.2 存储器设计

(1) 存储器控制器

S3C44B0X 内部的存储器控制器为外部存储器操作提供两套必要的存储器控制信号。它具有以下特性：

- 小/大端控制(通过外部引脚选择)
- 地址空间：32M 字节控制每 bank
- 所有 bank 都有可编程的总线宽度
- 8 个存储器 banks，6 个可用作 ROM，RAM 映射空间的存储器 bank
- 具有可编程的操作周期

◆ 专用 DRAM/SDRAM 接口支持自刷新模式

S3C44B0X 具有一个输入引脚 ENDIAN，处理器通过它的输入逻辑电平来确定数据结构是小端还是大端：0-小端，1-大端。BOOT ROM 在地址上位于 ARM 处理器的 Bank0 区，它具有多种数据总线宽度，这个宽度是通过硬件引脚 OM[1:0] 设定的，如表 3-1 所示。

表 3-1 ROM bank0 总线宽度设置

OM[1:0]	数据总线宽度
00	8-bit(byte)
01	16-bit(half-word)
10	32-bit(word)
11	Test Mode

(2) NORFLASH 电路

作为一种非易失性存储器，FLASH 在系统中通常用于存放程序代码、常量表以及一些在系统掉电后需要保存的用户数据等。本系统采用的 SST39V160 是一种 16 位数据宽度，编程电压为 3.3V 的 NORFLASH[27]。

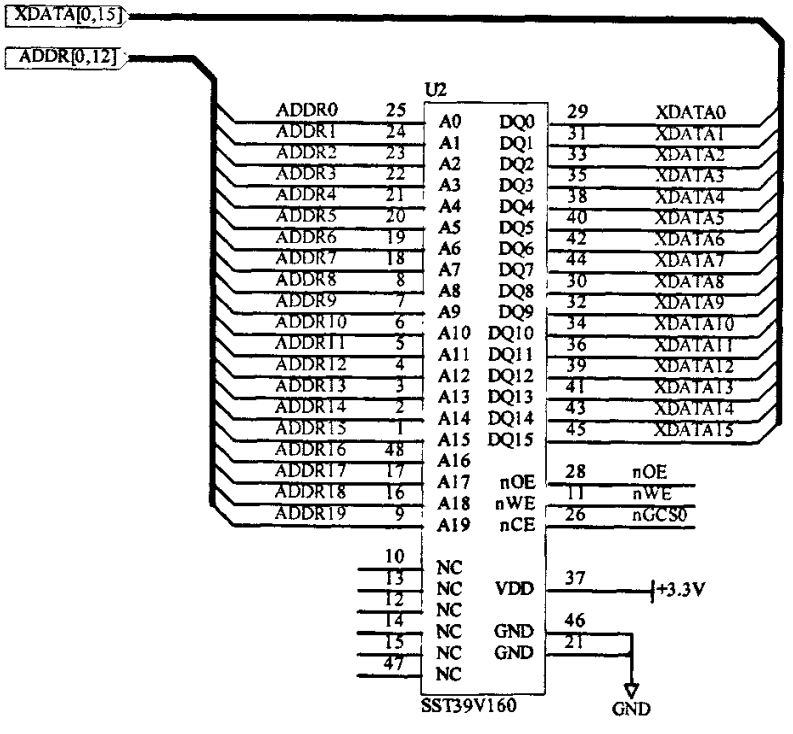

图 3-3 NORFLASH 连接图[32]

需要注意的是，S3C44B0 的写字节使能 nWBE[3:0]四个信号的使用。它必须与存储器的数据总线宽度、处理器的大端或小端模式综合起来考虑。当处理器工作在小端模式时，如果外接 4 片 8 位数据宽度的存储器，则 nWBE[3:0]分别接到这四片存储器上作为写使能来区分从高位到低位的四字节数据。如果外接的是 16 位数据宽度的存储器，则 nWBE[1:0]用于选通存放低 16 位数据的存储器，而 nWBE[3:2]则用于选通存放高 16 位数据的存储器。本文只使用了一片 16 位的存储器，所以只需要将 nWBE0 接到写使能上就可以了。具体的连接如图 3-3 所示。

(3) SDRAM 电路

与 Flash 存储器相比，SDRAM 不具有掉电保持数据的特性，但它存取速度大大高于 Flash 存储器，它适合用作程序的运行空间、数据及堆栈区。一般系统运行时最终都将代码调入 SDRAM 中运行，以提高系统运行速度。本设计采用的是 K4S641632 的 8M SDRAM[28]。具体的连接如图 3-4 所示。

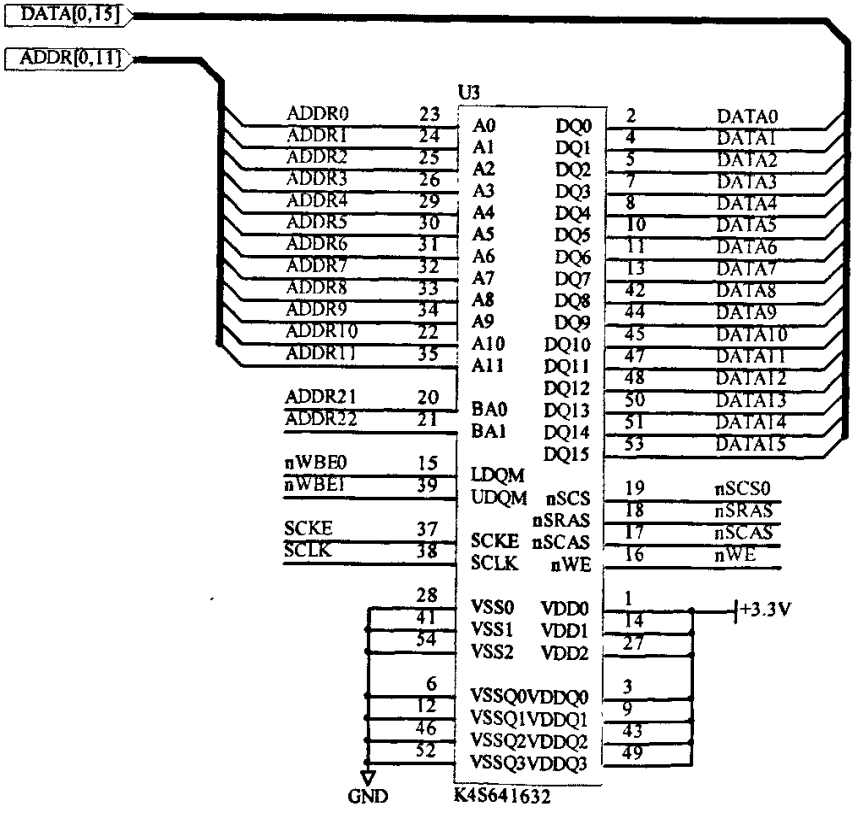

图 3-4 SDRAM 连接图

§3.2.3 液晶接口

S3C44B0X 内置的 LCD 控制器既可以支持 4 级灰度、16 级灰度灰白屏，也可以支持 256 色的 STN 彩色屏。它通过编程实现对不同种类 LCD 要求的支持，包括行和列象素数，数据总线宽度，接口时序和刷新频率等。

LCD 控制器的主要工作是将定位在系统存储器显示缓冲区中的 LCD 图像数据传送到外部 LCD 驱动器。它外部的接口信号包括：

- VFRAME：LCD 控制器和 LCD 驱动器之间的帧同步信号。LCD 驱动器在一个完整的帧显示完成后立即插入一个 VFRAME 信号，开始新一帧的显示。
- VLINE：LCD 控制器和 LCD 驱动器之间的线同步脉冲信号，该信号用于 LCD 驱动器将行移位寄存器的内容传送给 LCD 显示。LCD 控制器在整行数据移入 LCD 驱动器后，插入一个 VLINE 信号。
- VCLK：LCD 控制器和 LCD 驱动器之间的象素时钟信号，由 LCD 控制器送出的数据在 VCLK 上升延处送出，在 VCLK 下降沿处被 LCD 驱动器采样。
- VM：LCD 驱动器的 AC 信号。VM 信号被 LCD 驱动器用于改变行和列的电压极性，从而控制象素点的显示或熄灭。VM 信号可以通过与每个帧同步，也可以和可变数量的 VLINE 信号同步。
- VD[7:0]：LCD 象素点数据输出口。

除了这些 LCD 控制器的各种信号外，还需要给液晶模块提供相应的电源和使用 I/O 来实现的控制液晶的背光开关。具体的接口连接如图 3-5 所示[34]。

§3.2.4 触摸屏接口

(1) 实现方案

触摸屏输入系统由触摸屏、触摸屏控制器和微控制器三部分组成。本设计中触摸屏采用四线电阻式触摸屏，触摸屏控制器是 ADS7843，微控制器就是系统的主控芯片 S3C44B0。

四线电阻式触摸屏，是由两个透明电阻膜构成，在它的水平和垂直电阻网上施加电压，当触摸屏上发生触摸事件时，触摸屏就会产生相应的电压变化，然后通过触摸屏控制器 ADS7843 得到触摸点的水平和垂直坐标，并发送到主控制器

进行坐标校准等处理。

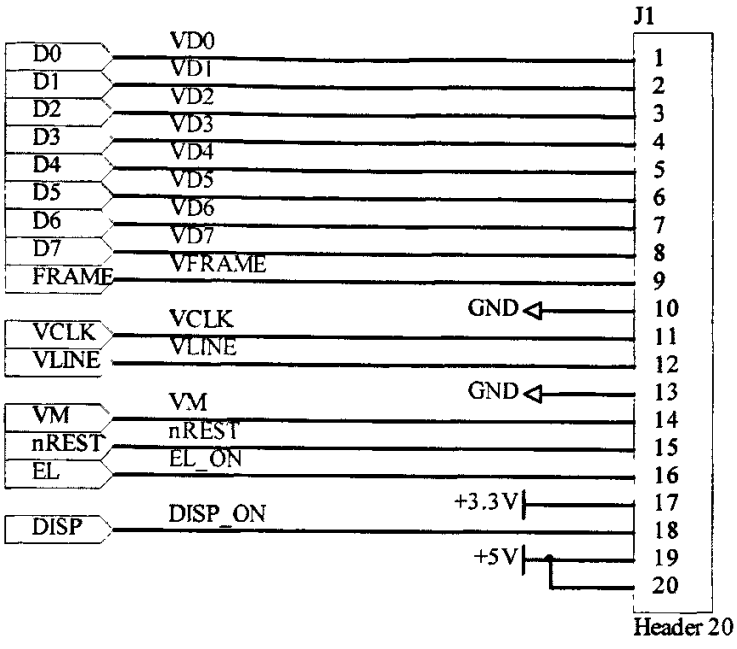

图 3-5 LCD 接口

(2) ADS7843 的特点

ADS7843 是 BB 公司生产的四线电阻触摸屏转换芯片。它是一款具有同步串行接口的 12 位取样模数转换器。在 125kHz 吞吐速率和 2.7V 电压下的功耗为 750uW。因此，ADS7843 以其低功耗和高效率等特性，被广泛应用于各种嵌入式设备上。本系统使用的 ADS7843 采用 SSOP-16 封装形式[28]。各引脚功能如表 3-2 所示：

表 3-2 ADS7843 引脚功能

引脚	名称	作用
1	+Vcc	电源
2	X+	X+输入端
3	Y+	Y+输入端
4	X-	X-输入端
5	Y-	Y-输入端
6	GND	地
7	IN3	ADC 输入通道 3

8	IN4	ADC 输入通道 4
9	Vref	参考电压输入
10	+Vcc	供电电源
11	IRQ	中断输出，须外接上拉电阻
12	DOUT	串行数据输出端。CS 低时，DCLK 的下降延触发
13	BUSY	忙指示，低电平有效
14	DIN	串行数据输入端。CS 低时，DCLK 的上升延触发
15	CS	片选
16	DCLK	外部时钟输入端

(3) 接口电路

ADS7843 采用 3.3V 供电，为了保证 AD 转换器内部参考电压的稳定，在 Vref 端通过 10uf 的电容接地。X+,Y+,X-,Y-分别与触摸屏的 4 个输出端相连，IN3，IN4 不需要使用，接地。与 S3C44B0 的通信则时通过 3 线的同步串行通信实现，GPB4 作为时钟信号，GPB6 作为数据输入脚，GPB8 作为数据输出脚。片选信号由 GPB5 控制，GPB7 连接 BUSY 输出端，中断的输出脚与 S3C44B0 的外部中断 3 输入引脚相连。具体连接如图 3-6 所示。

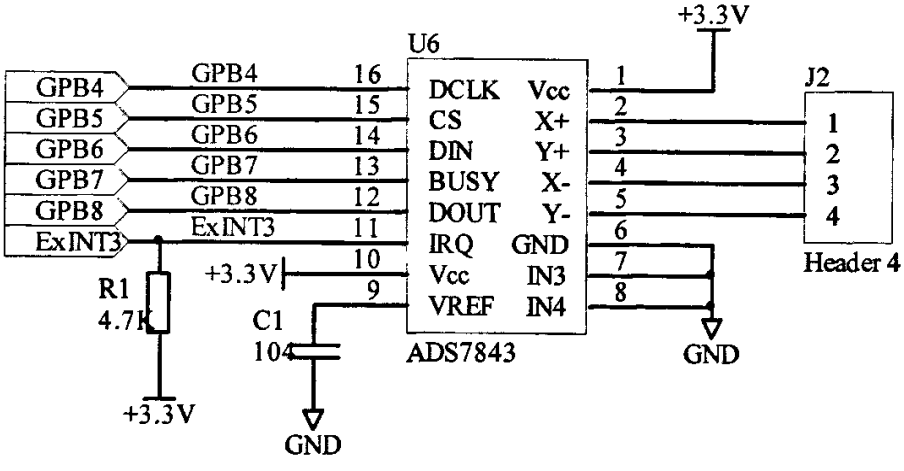

图 3-6 触摸屏接口

§3.2.5 UART 接口

S3C44B0X 的 UART 单元提供两个独立的异步串行 I/O 端口，每个都可以在中断和 DMA 两种模式下工作。它们支持的最高波特率为 115.2Kbps。每个通道包含 2 个 16 位 FIFO 分别提供给接收和发送[30]。连接如图 3-7 所示。

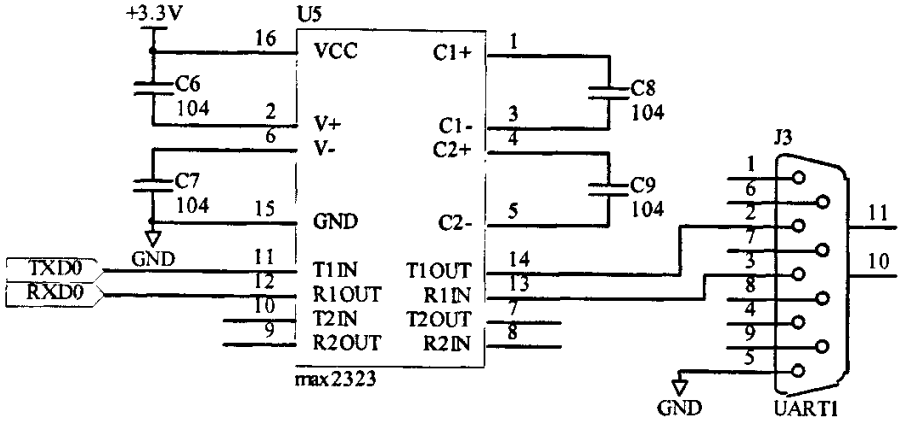

图 3-7 UART 接口

§3.2.6 电源

由于本设计采用+5V 的直流电源供电，而部分器件的工作电压在 3.3V，需要进行转换。另外，人机界面内显示的日期需要使用 S3C44B0 的 RTC 时钟，这就需要一颗电池来给 RTC 提供电源[31]。具体原理如图 3-8 所示。

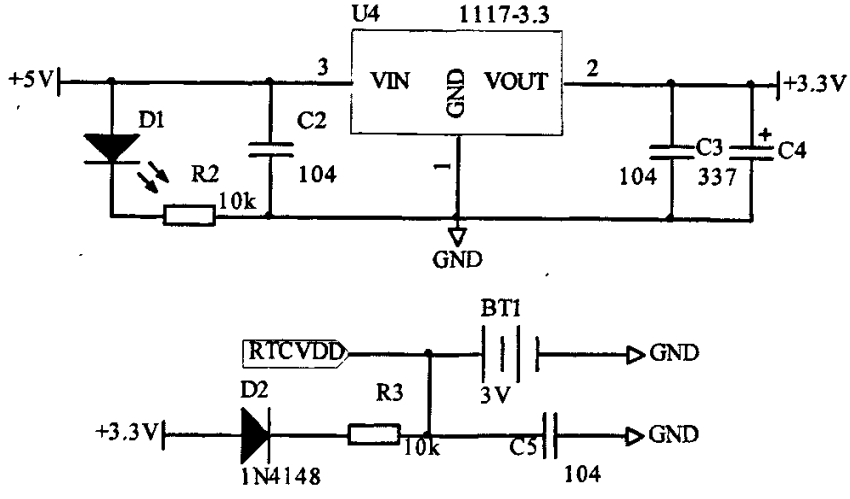

图 3-8 电源

第 4 章 μC/GUI 在 μC/OS-II 上的实现

对于选定的人机界面软件平台 μC/OS-II+μC/GUI，具有高移植性的特点，本章主要展开它们在 S3C44B0X 这个硬件平台上移植，包括核心的移植、液晶驱动的编写、添加中文的支持以及触摸屏驱动的编写等。

§ 4.1 μC/OS-II 的移植[35][37][38]

§ 4.1.1 移植的条件

μC/OS-II 是一个源码公开的、基于优先级的、占先式的、可裁减的多任务实时内核、其绝大部分源码是用 ANSIC 编写的，代码可读性强，可移植性好，对处理器及资源要求不高。目前，它支持 x86、ARM、PowerPC、MIPS 等众多体系结构，并有很多商业应用实例。μC/OS-II 是优秀的轻量级实时操作系统，适用于本文嵌入式系统[36]。

一般来说，能移植 μC/OS-II 的 CPU 必须满足以下条件[37]：

(1) 处理器的 C 编译器能够产生可重入代码。

(2) 用 C 语言就可以打开和关闭中断。

(3) 处理器支持中断，并能产生定时中断。

(4) 处理器支持能够容纳一定量数据的硬件堆栈。

(5) 处理器有堆栈指针和其它 CPU 寄存器读出和存储导堆栈或内存中的指令。

本系统采用 C 编译器为 ADS1.2，可以生成可重入代码；开关中断可以通过设置寄存器来实现；μC/OS-II 通过对处理器产生的定时器中断来实现多任务之间的调度；在 ARM7TDMI 的处理器上可以产生定时器中断。另外，ARM7TDMI 核的处理器具有硬件堆栈和有关的堆栈指令。因此，本系统完成符合 μC/OS-II 的移植条件。

§ 4.1.2 OS_CPU.H

包括数据类型定义，堆栈单位定义，堆栈增长方向定义，开关中断函数和进行任务切换函数的声明。

其中堆栈的单位长度与 CPU 的寄存器宽度相等，为 32 位。绝大多数处理器的堆栈是从上向下生长的。μC/OS-II 采用兼容性设计，只要设定 OS_STK_GROWTH 的值即可，1 为从上往下，0 为从下往上。

§ 4.1.3 OS_CPU_A.ASM

μC/OS-II 的移植需要编写几个简单的汇编语言函数：前面在 OS_CPU.H 中声明过的函数、OSCtxSw()、OSIntCtxSw()、OSTickISR()、OSStartHighRdy()。

(1) OS_ENTER_CRITICAL 和 OS_EXIT_CRITICAL

在 S3C44B0 处理器中，由于没有直接的汇编指令及 C 语句支持中断的开关，所以只能用一段汇编程序来模拟。通过修改微处理器 S3C44B0 的程序状态寄存器 CPSR 的中断屏蔽位来实现 μC/OS-II 中的禁止和允许中断。OS_ENTER_CRITICAL 的代码如下：

```
OS_ENTER_CRITICAL
        STMFD   SP!,        {R0,LR}
        MRS     R0,         CPSR
        ORR     R0,         R0, #(1<<7)
        MSR     CPSR_c,     R0
        LDR     R0,         =OsEnterSum
        LDR     R1,         [R0]
        ADD     R1,         R1, #1
        STR     R1,         [R0]
        LDMFD   SP!,        {R0,PC}
```

(2) OSCtxSw()

任务级切换是通过 SWI 或者 TRAP 人为制造的中断来实现的。ISR 的向量地址必须指向 OSCtxSw()。这一中断完成的功能是：保存任务的环境变量；

将当前的 SP 存入任务 TCB 中；载入就绪最高优先级任务的 SP；恢复就绪最高优先级的环境变量；中断返回。

```
OSCtxSw
        STMFD   SP!,        {R0-R12}      ; 保存 R12-R0.
        LDR     R0,         =OsEnterSum
        LDR     R0,         [R0]
        MRS     R1,         CPSR
        STMFD   SP!,        {R0-R1}       ; 保存 CPSR,OsEnterSum
        LDR     R1,         =OSTCBCur
        LDR     R1,         [R1]
        STR     SP,  [R1]     ; OSTCBCur->OSTCBStkPtr = SP.
        BL      OSTaskSwHook              ; 调用钩子函数
        LDR     R0,  =OSPrioCur ; OSPrioCur = OSPrioHighRdy.
        LDR     R1,  =OSPrioHighRdy
        LDRB    R1,  [R1]
        STRB    R1,   [R0]
        LDR     R0,  =OSTCBCur   ; OSTCBCur   = OSTCBHighRdy.
        LDR     R1,  =OSTCBHighRdy
        LDR     R1,   [R1]
        STR     R1,   [R0]
```

(3) OSIntCtxSw()

中断级任务切换在中断服务函数 OSIntExit()中调用，实现中断级任务切换。在时钟中断 ISR 中，当发现有高优先级任务等待的时钟信号到来时，在中断退出后并不返回被中断的任务，而是直接调度就绪的最高优先级任务执行，从而能够尽快地让高优先级任务得到响应。代码如下：

```
__OSIntCtxSw
    STMFD   SP!,        {R0}
    MOV     R0,         LR
    MSR     CPSR_c,     #(NoInt | SYS32Mode) ; 切换到系统模式.
```

```
STMFD   SP!,        {R0}                    ; 保存 PC.
STMFD   SP!,        {LR}                    ; 保存 LR.
MSR     CPSR_c,     #(NoInt | IRQ32Mode)    ; 切回 IRQ 模式.
LDMFD   SP!,        {R0}
LDR     LR,         =OSCtxSw_01             ; 任务级切换
STMFD   SP!,        {LR}
LDMFD   SP!,        {PC}^
```

(4) OSTickISR()

系统时钟节拍中断函数是一个周期性中断，为内核提供时钟节拍。频率越高，系统负荷越重。该函数地具体内容：保存寄存器；调用 OSIntEnter()；调用 OSTimeTick()；调用 OSIntExit()；恢复寄存器；中断返回。

(5) OSStartHighRdy()

在 μC/OS-II 初始化时用来使就绪态任务中优先级最高的任务开始运行。移植时，通过执行汇编指令，将就绪的最高优先级任务堆栈中保存的寄存器值按指定的顺序恢复到微处理器 S3C44B0 相应的寄存器中。并且执行中断的返回，即可运行最高优先级任务。代码如下：

```
OSStartHighRdy
        LDR     R0,     =OSRunning；告诉 μC/OS-II 自身已经运行.
        MOV     R1,     #1
        STRB    R1,     [R0]
        LDR     R1,     =OSTCBHighRdy
        LDR     R1,     [R1]
        LDR     SP,     [R1, #0]
                ;获取新任务堆栈指针 OSTCBHighRdy->OSTCBStkPtr.
        LDMFD   SP!,    {R0-R1}     ; 恢复 CPSR,OsEnterSum.
        LDR     R2,     =OsEnterSum
        STR     R0,     [R2]
        MSR     CPSR_cxsf,  R1
        LDMFD   SP!, {R0-R12,LR,PC}  ; 恢复 R12-R0,LR,PC.
```

§ 4.1.4 OS_CPU_C.C

函数 OSTaskStkInit()用于初始化任务的堆栈结构，不同的处理器在进行函数调用时需保存的寄存器不同及用户设计的堆栈结构不同，所以必须要改写。具体做法参照微处理器 S3C44B0 的寄存器结构特点进行。

在 uC/OS-II 中规定了一些 Hook 函数：OSTaskCreateHook()，OSTaskDelHook()，OSTaskSwHook()，OSTaskStatHook()，OSTimeTickHook()。这些函数必须声明，但无特殊需求，可以将它们设为空函数。

经过上述 3 个系统文件的修改，μC/OS-II 就能在 S3C44B0 上正常运行了。

§ 4.2 μCGUI 的移植[42][43]

§ 4.2.1 嵌入式 GUI 概述

目前，国内外已经推出了多种嵌入式 GUI 系统。比如基于 Linux 的 MicroWindows，OpenGUI 以及国产的 MiniGui 等，另外众多的嵌入式操作系统，比如 μC/OS,VxWorks 等，也提供了相应的 GUI。 嵌入式 GUI 一般具有以下特点[36]：

(1) 可配置性。嵌入式系统往往是一种特制设备。各种系统对 GUI 的要求各不相同，因此，嵌入式 GUI 必须呈模块结构，以便于配置和定制。

(2) 嵌入式微处理器普遍存在运算速度相对慢，内存容量小的特点，因而要求图形算法简洁、快速，占用资源少。

(3) 硬件无关性。嵌入式系统应用的领域不同，所采用的人机设备也不同，如果 GUI 没有实现平台无关性，将出现每更换一种硬件设备都要编写 GUI 系统函数的问题。

(4) 嵌入式 GUI 必须具有高性能、高可靠性。

在考虑了织机控制系统的人机交互界面简洁、美观的要求，比较各种嵌入式 GUI 的优缺点之后，本设计选择了 μC/GUI 作为人机界面的平台。

μC/GUI 作为图形系统，它的库函数提供了丰富的文本、字体、二维绘图库、可扩充字符集、Unicode、位图显示，多级 RGB 及灰度调整、动画优化显示、具

有 Windows 风格的对话框和预定义控件(按钮、编辑框、列表框、进度调、单选多选框等)，以及对键盘、鼠标、触摸屏等输入设备的支持。各个子目录的路径及功能支持如表 4-1 所示。

表 4-1　GUI 文件目录

文件目录	内容
Config	配置文件
GUI\AntiAlias	抗齿功能，优化 LCD 边缘模糊效果
GUI\ConvertMono	单色及灰度显示的颜色转换程序
GUI\ConverColor	彩色显示的颜色转换程序
GUI\Core	μC/GUI 内核
GUI\Font	多种字体文件
GUI\LCDDriver	LCD 硬件驱动文件
GUI\MenDev	内存设备支持，防止在画图时产生的抖动
GUI\Touch	触摸屏(只支持模拟的触摸屏)
GUI\Winget	窗口控件类
GUI\WM	窗口管理器

§ 4.2.2 μC/GUI 配置文件

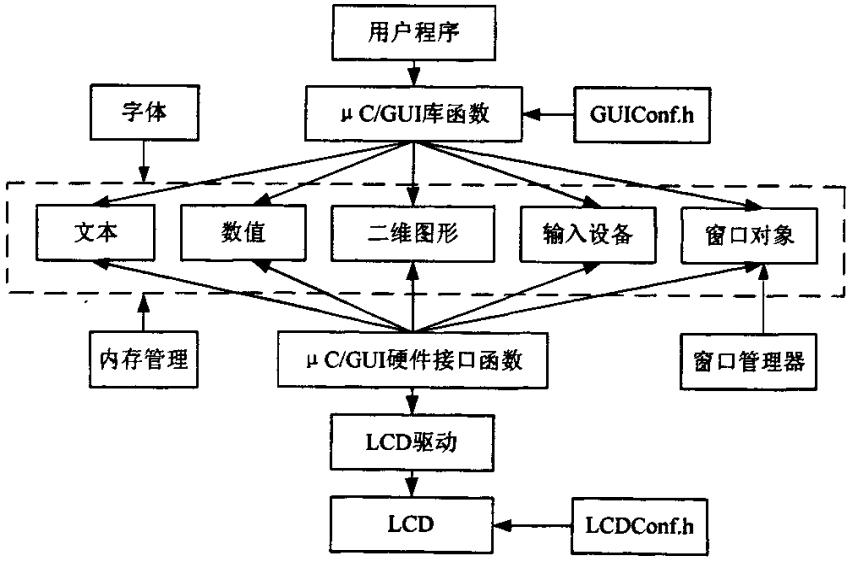

图 4-1 μC/GUI 结构图

嵌入式 GUI 是一种类似于操作系统的基础软件，这种软件系统遵循一定的标准。μC/GUI 系统采取分层设计的结构，对不同层次进行封装。封装的结构如图 4-1。

(1) GUIConf.h

GUIConf.h 文件主要是对一些高级选项进行控制和定义。定义了对操作系统的有关配置、动态内存控件、默认字体、是否使用触摸屏、窗口管理和存储器管理等选项。

本设计中采用了 μC/OS-II 操作系统；由于是一个多窗口的界面设计，须将窗口管理打开；为了获得给好的显示效果，支持存储器管理功能；采用触摸屏作为本人机界面的输入设备，则支持触摸屏。具体的配置代码如下：

```
#define GUI_OS                    (1)
#define GUI_WINSUPPORT            (1)
#define GUI_SUPPORT_MEMDEV        (1)
#define GUI_SUPPORT_TOUCH         (1)
#define GUI_SUPPORT_UNICODE       (1)
#define GUI_MAXTASK               (5)    // 多任务数
#define GUI_DEBUG_LEVEL   GUI_DEBUG_LEVEL_LOG_ALL
#define GUI_ALLOC_SIZE            12500    /* Size of dynamic memory */
#define GUI_DEFAULT_FONT      &GUI_Font8x16    // 缺省字体.
```

(2) LCDConf.h

LCDConf.h 中主要是一些与 LCD 硬件相关的配置，包括 LCD 的尺寸、控制器的型号、液晶总线的宽度以及其它的一些配置选项，其中最关键的是 GUI 用于读写显存的函数的定义。具体的配置代码如下：

```
#define LCD_XSIZE 320
#define LCD_YSIZE 240
#define SCR_XSIZE 320
#define SCR_YSIZE 240
extern unsigned short frameBuffer[];  //LCDDriver 中显存缓冲区地址
#define LCD_BITSPERPIXEL  (8)
#define LCD_CONTROLLER    1375
```

```
#define LCD_FIXEDPALETTE    (323)
#define LCD_BUSWIDTH    (16)
#define LCD_READ_MEM(Off)
                    *((unsigned short*)(frameBuffer+(((unsigned int)(Off)))))
#define LCD_WRITE_MEM(Off,data)
                *((unsigned short*)(frameBuffer+(((unsigned int)(Off)))))=data
#define LCD_SWAP_BYTE_ORDER (1)              //字节转换
#define LCD_SWAP_RB              (1)              // 红绿位置交换.
#define LCD_INIT_CONTROLLER()   LCD256_Init()
```

§ 4.2.3 LCD 驱动的实现

一般情况下 LCDDriver.c 主要是定义一些 μC/GUI 与 LCD 的硬件接口函数，包括画点、线、矩形、多边形、位图等二维图形的函数，以及 LCD 的初始化函数等。而本设计所采用的做法是利用 μC/GUI 自带驱动的 buffer 型 LCD 控制器 EPSON1375，通过对原驱动中显存地址初始化的修改，就可以利用驱动中的其它二维画图函数。具体的做法已经在 LCDConf.h 中给出。而 LCD 初始化的函数，与采用的 LCD 有关，通过参看 S3C44B0 数据手册的 LCD 驱动器的配置来进行编写。具体代码如下：

```
void LCD256_Init(){
    rLCDCON1=(0)|(2<<5)|(MVAL_USED<<7)
                        |(0x3<<8)|(0x3<<10)|(CLKVAL_COLOR<<12);
    rLCDCON2=       (LINEVAL)|(HOZVAL_COLOR<<10)|(10<<21);
    rLCDSADDR1=    (0x3<<27) | ( ((unsigned long)frameBuffer>>22)<<21 )
                        | M5D((unsigned long)frameBuffer>>1);
    rLCDSADDR2=    M5D((((unsigned long)frameBuffer+
                       (SCR_XSIZE*LCD_YSIZE))>>1)) | (MVAL<<21) | (1<<29);
    rLCDSADDR3=    (LCD_XSIZE/2) | ( ((SCR_XSIZE-LCD_XSIZE)/2)<<9 );
    rREDLUT=0xfdb96420;
    rGREENLUT=0xfdb96420;
```

```
    rBLUELUT =0xfb40;
    rDITHMODE=0x0;
    rDP1_2 =0xa5a5;
    rDP4_7 =0xba5da65;
    rDP3_5 =0xa5a5f;
    rDP2_3 =0xd6b;
    rDP5_7 =0xeb7b5ed;
    rDP3_4 =0x7dbe;
    rDP4_5 =0x7ebdf;
    rDP6_7 =0x7fdfbfe;
    rLCDCON1=(1)|(2<<5)|(MVAL_USED<<7)|(0x3<<8)|(0x3<<10)
                    |(CLKVAL_COLOR<<12);
    }
```

LCD 的背光控制引脚是直接与 S3C44BO 的多功能 I/O 口相连，因此，背光的开关也就十分简单。开背光代码如下：

```
void LCD256_DispON(){
    rPCONE = rPCONE & (~(0xf<<6)) | (0x5<<6) ;
    rPDATE = ( rPDATE | (3<<3) );
}
```

§ 4.2.4 中文的实现[48]

μC/GUI 本身缺乏对中文汉字的支持，但它采用 Unicode 编码，即每个字符的显示都是先通过一个编码查找它的字模的位置，然后完成该字符的显示的。字体库中的字母字模通过 GUI_FONT 和 GUI_FONT_PROP 两个结构体来进行统一管理。GUI_FONT 结构体中定义了该字母的点阵大小(比如 12x12 或 8x8)和 GUI_FONT_PROP 的入口地址。GUI_FONT_PROP 这个结构体建立了字库中字母编码(比如 A 在 ASCII 中的字母编码为 33)和字模数据存放地址的映像。

特别是GUI_FONT_PROP结构体中pNext指针指向下一个GUI_FONT_PROP数据的入口地址，这为解决在字母编码不连续的情况下，保证字模数据在程序段

的存储连续问题提供了解决方案。GUI_FONT_PROP 结构如下：

```
Typedef  struct {
  U16P First;                              /* first character */
  U16P Last;                               /* last character */
  const GUI_CHARINFO  *paCharInfo;         /* address of first character*/
  void  * pNext;                           /* pointer to next */
} GUI_FONT_PROP;
```

运用上述方法，本设计先对人机界面需要用到的汉字进行收集整理，然后通过字模提取工具提取所要字体的字模，建立供本设计所用的小汉字库，为了方便使用时查找，在字库文件里列出汉字的编码即可。经过整理，建立了 24x24 和 12x12 两个小汉字库，其中 24x24 有汉字 12 个，12x12 有汉字 136 个。

使用汉字库的方法，则是先从字库中挑选出所需要的汉字，将汉字编码存放在一个数组内，如要显示中文“浙江大学”，它们的编码分别为 0xa1a1,0xa1a2,0xa1a3,0xa1a4，定义数组 ZJU 存放编码，如下：

```
Const unsigned char ZJU[]={0xa1,0xa1,0xa1,0xa2,0xa1,0xa3,0xa1,0xa4,0x00};
```

需要显示时，先选择字库 GUI_SetFont(GUI_FontHZ12),然后使用字库显示函数输出编码地址数组即可。

§ 4.3 μC/GUI 与 μC/OS-II 的结合

由于设计采用了 μC/OS-II 实时操作系统，因此还必须改写以下与操作性系统有关的函数：系统时间函数、包括取系统时间函数 GUI_X_GetTime()和延时函数 GUI_X_Delay()；任务调度函数，包括任务初始化函数 GUI_X_InitOS()、任务锁定函数 GUI_X_Lock()和任务解除函数 GUI_X_Unlock()。

(1) GUI_X_GetTime()

GUI_X_GetTime()使用了 μC/OS-II 的函数 OSTimeGet()，返回当前的系统时间。函数代码如下：

```
int  GUI_X_GetTime(void) {
     return OSTimeGet();
   }
```

(2) GUI_X_Delay()

函数原型为 void GUI_X_Delay(int ms)，它使用了 μC/OS-II 的函数 OSTimeDlyHSM()来延迟一个指定的时间段，代码如下：

```
void GUI_X_Delay(int Period) {
    OSTimeDly(Period);
    }
```

(3) GUI_X_InitOS()

任务初始化函数用于生成可供函数 GUI_X_Lock()与 GUI_X_Unlock()使用的互斥型资源信号量。代码如下：

```
void GUI_X_InitOS (void){
    DispSem = OSSemCreate(1);
    }
```

(4) GUI_X_Lock()

任务锁定函数用于GUI访问显示区域或关键数据结构前调用以阻止其它任务进入这些资源，知道 GUI 调用 GUI_X_Unlock 释放这些资源位置。函数代码如下：

```
void GUI_X_Lock (void) {
INT8U err;
 OSSemPend(DispSem, 0, &err);
}
```

(5) GUI_X_Unlock()

任务解除函数用于解锁在 GUI 访问显示区域或关键数据结构前由 GUI_X_Lock()锁定的独占资源。函数代码如下：

```
void GUI_X_Unlock (void){
    OSSemPost(DispSem);
    }
```

(6) 其它操作系统接口函数

除了以上介绍的这些函数以外，下面几个操作系统中用到，但实际没有特殊要求可以省略的函数，我定义为空函数。

```
void GUI_X_Warn(const char *s) {}
```

```
void GUI_X_ErrorOut(const char *s)  {}
void GUI_X_Log(const char *s)  {}
void GUI_X_Init(void)  {}
```

经过上面几个步骤的移植，μC/GUI 就可以以单任务或者多任务的形式在 μC/OS-II 上正常运行了。

§ 4.4 触摸屏驱动的实现[46][47]

§ 4.4.1 ADS7843 的初始化

在完成了触摸屏、ADS7843、S3C44B0 的硬件连接后，首先要对与 ADS7843 的 IO 进行配置，具体操作是：GPB4、GPB5 设置为 OUTPUT，GPB6、GPB7、GPB8 设置为 INPUT，使能外部中断 3[44]。

§ 4.4.2 坐标转换的实现

触摸屏控制器启动后能够自动完成扫描，并且将物理坐标的 AD 值存入特定寄存器后产生触摸屏扫描 AD 中断，而要读取并实现与 LCD 坐标值的对应，则需要先向 ADS7843 写特定的控制字，然后再通过 SPI 通信得到寄存器中存储的物理坐标，最后通过简单的坐标转换实现与 LCD 坐标的匹配[41]。

(1) ADS7843 控制字

ADS7843 的控制字如表 4-2 所示，其中 S 为数据传输起始标志位，该位必为“1”。A2~A0 进行通道选择，见表 4-3。MODE 用来选择 A/DS 转换精度。“1”选择 8 位，“0”选择 12 位。SER/DFR 选择参考电压的输入模式，见表 4-3。PD1，PD0 选择省电模式：“00”省电模式允许，在两次 A/D 转换之间掉点，且中断允许；“01”同“00”，只是不允许中断；“10”保留；“11”禁止省电模式[27]。

表 4-2 ADS7843 的控制字

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
S	A2	A1	A0	MODE	SER/DFR	PD1	PD0

表 4-3 差动输入模式

A2	A1	A0	X+	Y+	IN3	IN4	X 开关	Y 开关
0	0	1	+IN				OFF	ON
1	0	1		+IN			ON	OFF
0	1	0			+IN		OFF	OFF
1	1	0				+IN	OFF	OFF

为了使得到的物理坐标准确、稳定，根据以上选择方式，并经过多次测试，确定通道 X 选择的控制字为 0x90，通道 Y 选择的控制字为 0xD0。

(2) SPI 时序

确定了的控制字，需要 S3C44B0 通过 SPI 发送给 ADS7843 来实现物理坐标的读取，而 S3C44B0 自身不带 SPI 口，通过 3 个 I/O 口来模拟时序获得。写控制字和读取坐标值的时序如图 4-2 所示：

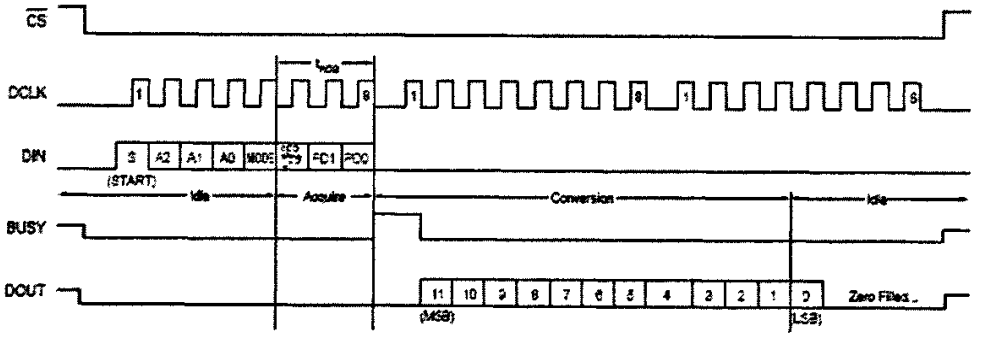

图 4-2 A/D 转换时序图

通过对以上时序的分析，使用 S3C44B0 的 I/O 口模拟同步串行通信，只是简单的写 I/O 操作，本设计中，编写了 ADS7843_Read()和 ADS7843_Write()两个函数来完成对 SPI 的读和写操作。

(3) 坐标转换的基本流程

对于本系统的触摸屏设计，在触摸事件发生后，ADS7843 输出的 A/D 转换值都会有一个不稳定的抖动期，这个时间大概需要 10ms，因为为了得到准确的物理坐标，在主控芯片接收到触摸屏发来的中断后，先进行 10ms 的延时，然后再由 SPI 口读出 A/D 转换值。

通过 A/D 转换得到的是物理坐标，而我们需要的是与 LCD 尺寸相对应的坐标，因此还需要进行数值上的处理。ADS7843 采用的是 12 位的 A/D 转换，

即满值为 4095，而 LCD 尺寸为 320X240，所以，转换实际上是把 0～4095 的坐标按比例缩小到 320 的 X 坐标和 240 的 Y 坐标。具体计算如下：

LCD_X=320－(320*X/4095);

LCD_Y=240*X/4095;

整个触摸屏功能的实现流程如图 4-3 所示：

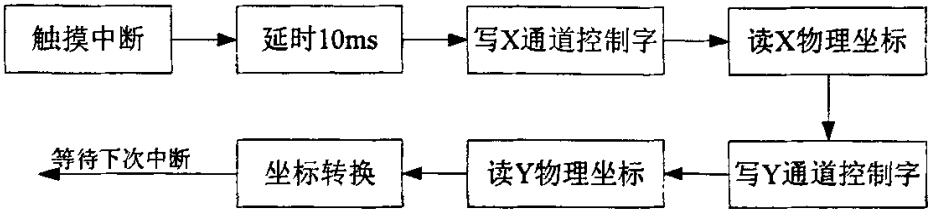

图 4-3 触摸屏流程图

§ 4.4.3 其它配置

μC/GUI 本身就包括了对键盘、鼠标、触摸屏等外设的支持，并在源码级对其消息进行了响应，为了使得触摸屏正常工作，用户除了要实现触摸屏底层驱动以外，还需要对其相应的配置文件进行修改和添加一些相关的函数。

在功能块配置文件 GUIConf.h 中的宏 GUI_SUPPORT_TOUCH 设置为 1 的情况下，μC/GUI 就增加了对触摸屏的支持。GUI_TouchConf.h 文件包括了触摸屏矫正时使用的一些宏定义。包括触摸屏的校准信息和其它的一些开关量。

理论上来说，触摸屏边缘输出的物理坐标应该是 12 位 A/D 转换的最大值 4095 和最小值 0，但是在实际中，触摸屏输出的物理坐标并非是理想状态下预期的值，因此就需要通过校准来得到更加准确的坐标。本设计通过多次点击触摸屏左上角和右下脚，大致确定其坐标输出为(30,40)，(4065,4075)。故可确定相应的宏定义如下：

```
#define GUI_TOUCH_AD_LEFT       30
#define GUI_TOUCH_AD_RIGHT      4065
#define GUI_TOUCH_AD_TOP        40
#define GUI_TOUCH_AD_BOTTOM     4075
#define GUI_TOUCH_SWAP_XY       1
#define GUI_TOUCH_MIRROR_X      1
```

```
#define GUI_TOUCH_MIRROR_Y   1
```

另外，得到的触摸屏坐标要传递给 μC/GUI，还需要相应地增加几个功能函数的定义。它们定义在 GUI_X_Touch.c 文件中，分别是：

```
void   GUI_TOUCH_X_ActivateX(void){}
void   GUI_TOUCH_X_ActivateY(void) {}
int    GUI_TOUCH_X_MeasureX(void) {}
int   GUI_TOUCH_X_MeasureY(void){}
```

其中，GUI_X_MeasureX 和 GUI_X_MeasureY 就是用来传递得到的触摸屏坐标值，而 GUI_TOUCH_X_ActivateX 和 GUI_TOUCH_X_ActivateY 则在一般情况下定义为空即可。

第 5 章 人机界面软件

本章主要介绍在上一章建立的软件平台上所作的应用开发，包括了对各个任务的创建和管理，显示任务，基于触摸屏的人机交互任务以及通信任务等。

§5.1 多任务调度机制

μC/OS-II 是专门为计算机的嵌入式应用而设计的实时操作系统，是基于静态优先级的抢占式多任务实时内核。内核是操作系统的核心部分，内核不能建立新的任务，它提供的基本服务是任务切换，此外还负责进程调度，消息处理，任务状态的转换等操作。

一个任务，也称作一个线程，是一个简单的程序，该程序可以认为 CPU 完全只属于该程序自己。实时应用程序的设计过程包括如何把问题分割成多个任务。每个任务都是整个应用的一部分，都被赋予一定的优先级，有自己的一套 CPU 寄存器和栈控件。每个任务都是一个无限的循环，都可以出于 5 种状态之一：休眠态、就绪态、运行态、挂起态及被中断态。它们之间具体的切换过程如图 5-1 所示[35]：

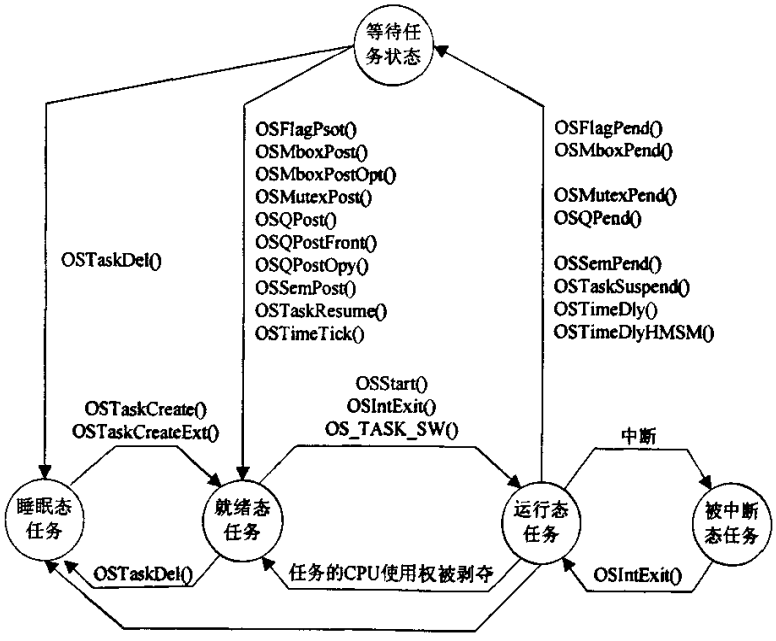

图 5-1 多任务状态图

对于 μC/OS-II 这样的可剥夺性内核，最高优先级的任务一旦就绪，总能得到 CPU 的使用权。当一个运行着的任务使一个比它优先级高的任务进入了就绪态时，当前任务的 CPU 使用权就被剥夺了，或者说被挂起了，最高优先级的任务立刻得到 CPU 使用权。任务的调度主要包括两部分：最高优先级任务的寻找和任务切换。最高优先级任务的寻找用过函数 OSSched()实现，宏调用 OS_TASK_SW()完成任务的切换[53][54]。切换流程如图 5-2 所示：

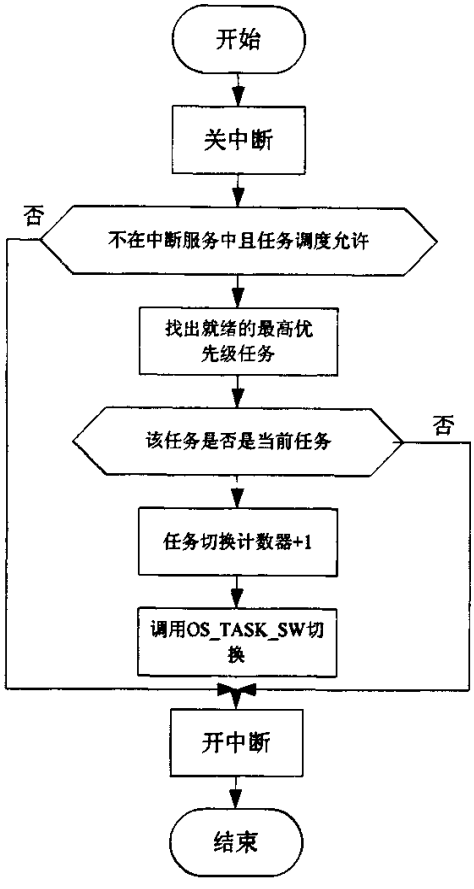

图 5-2 任务切换流程图

§ 5.2 任务的创建

μC/OS-II 可以管理多达 64 个任务，除去建议保留的 4 个最高优先级和 4 个最低优先级任务，还有 56 个任务可供应用程序使用。

本设计需要使用 3 个任务，分别为 μC/GUI 显示任务，触摸屏任务，串口通讯任务。优先级分配为显示任务为 3 个任务中最低，设为 13，触摸屏任务设为

最高的 5，串口通讯设为 9，此外任务的创建还需要进行堆栈的分配，创建任务的代码如下[52]：

```
OSTaskCreate (GUI_ExecTask,(void *)0, &GUI_ExecTaskStk[1023], 13);
OSTaskCreate (UART_Task,(void *)0, & UART_TaskStk[1023], 9);
OSTaskCreate (Touch_Task,(void *)0, & Touch_TaskStk[1023], 5);
```

为了更好的协调各个任务，还需要通过设置信号量等来实现必须的任务切换，而这些工作都需要在系统运行前完成。因此 μC/OS-II 的启动过程包括初始化变量和设定系统堆栈，创建任务间通信对象并初始化，创建用户任务(分配优先级和设定任务堆栈)，流程图 5-3 所示：

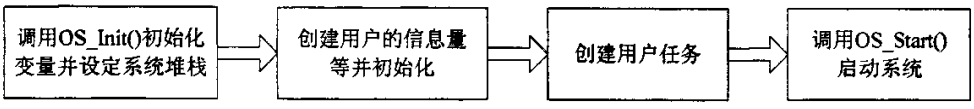

图 5-3 系统启动流程

§ 5.3 μC/GUI 显示任务

作为 μC/OS-II 多任务系统中的一个任务，μC/GUI 显示任务相对于其它任务独立，通过系统启动前创建的信息和邮箱来实现任务之间的通信，本任务的职能则是完成织机控制系统的人机界面显示。

§ 5.3.1 界面组织结构

为了得到一个能够直观显示织机各种性能参数，操作简单的人机交互界面，整个任务设有 17 个界面。人机界面的主界面，包括欢迎页面和主功能菜单。通过主功能菜单，就可以进入各个功能菜单选项：参数查看包括 4 个页面，显示织机目前的各种性能状态；单班统计和四班统计包括 3 个班次统计画面，使织造过程的管理变得更加简单；参数设置需要由工程师来完成，需要通过输入密码才能进入；参数设置包括了织机各种功能参数的设定，共有 6 个页面；说明书和留言板功能，作为本系统的辅助功能，使本人机界面的功能更加完善。各个页面的结构如图 5-4 所示[55]：

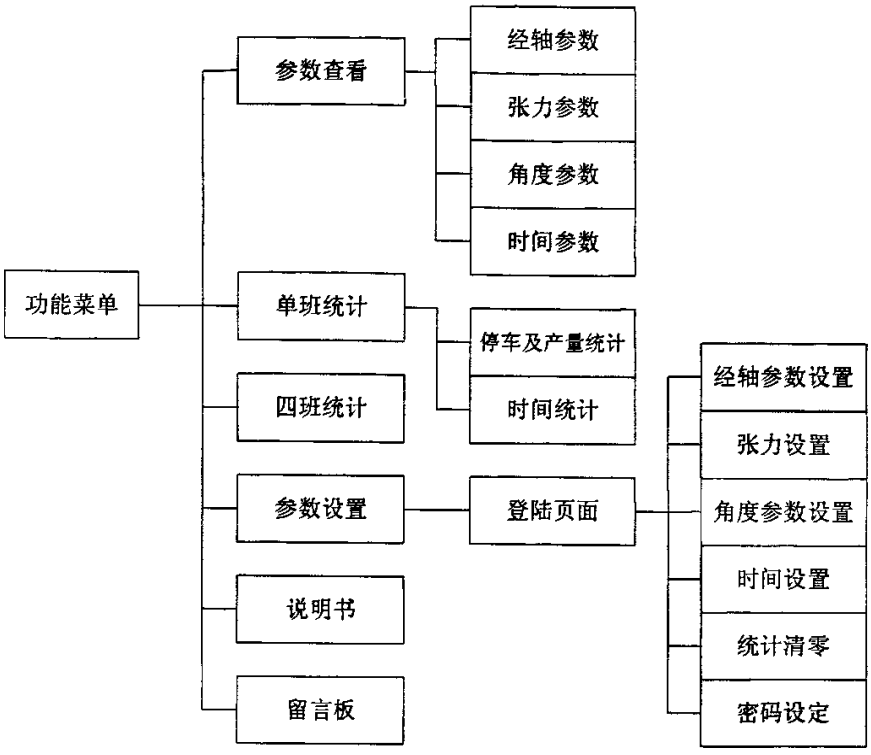

图 5-4 界面结构图

§5.3.2 人机交互界面的实现

(1) 显示接口函数

μC/GUI 为用户提供了完善的画图接口函数，界面的编写只需调用相应的接口函数即可，十分方便。对于本界面并不复杂的图形要求而言，只需几个常用的显示函数即可，功能如表 5-1 所示：

表 5-1 μC/GUI 显示接口函数

GUI_DispString	显示字符串，包括汉字
GUI_DispDec	显示数字
GUI_DrawBitmap	显示位图

利用这些接口函数，编写了包括参数查看、参数设置、单班统计、四班统计等 17 个界面。

(2) 回调机制

对于以上所编写的界面，目前还都是各自孤立的，没有实现互相的联系。但

μC/GUI 的窗口管理器为窗口和窗口对象(控件)提供的回调机制实质是一个消息的事件驱动系统。正如大多数视窗系统一样，原则是控制流程不只是从用户程序到图形系统，而且可以从用户程序到图形系统，同时也从图形系统回到用户程序。利用这种回调机制，就可以很好地把 17 个界面很好的联系起来，实现真正的人机显示界面。

μC/GUI 中的消息其实与 WINDOWS 类似，几种基本的消息是一样的，但它的更简单且消息更少，对于一些消息的处理也更简化，没有 WINDOWS 那样的复杂。μC/GU 中消息所包含的信息如下所示：

```
typedef  struct {
  int MsgId;                    // 消息的类型
  WM_HWIN hWin;                 // 目标窗口
  WM_HWIN hWinSrc;              // 源窗口
  union {
    void* p;                    //数据指针
    int v;                      // 数据数值
    GUI_COLOR Color;
   } Data;                      // 传递的数据
} WM_MESSAGE;
```

回调机制的实现依赖于消息的类型，μC/GUI 中主要的消息类型如表 5-2 所示：

表 5-2 μC/GUI 消息类型

WM_PAINT	重绘窗口
WM_CREATE	一个窗口创建后即发送
WM_DELETE	告诉窗口释放它的数据结构，然后将它删除
WM_MOVE	当一个窗口移动后发送到它
WM_SHOW	当一个窗口收到显示命令后发送到它
WM_HIDE	当一个窗口收到隐藏命令后发送到它
WM_TOUCH	触摸屏消息
WM_KEY	按键消息

WM_SETFOCUS	设置焦点窗口消息
WM_NOTIFY_PARENT	子窗体改变后发送到父窗体

(3) 控件

控件是具有对象性质的窗口，是构造用户接口的元素。它们能自动对某些事件起反应。控件需要建立，具有能在它们存在的任何事件修改的特性，在它们不在需要是能被删除。与窗口类似，一个控件通过它的建立函数返回的句柄而引用。控件需要使用视窗管理器。一旦控件被建立，它被处理成与其它窗口一样，WM保证它在需要的时候能正确的显示。控件的使用能是编程更容易。一些常用的控件如表 5-3 所示：

表 5-3 常用控件

BUTTON	可以按下的按钮，在按钮上可以显示文本或位图
CHECKBOX	复选框，提供多个选项
EDIT	单行文本编辑框，提示用户输入数字或文本
LISTBOX	列表框，被选中的项目会高亮显示
PROBAR	进度条，用于观察
TEXT	文本控件，典型应用在对话框中

控件在创建的同时，就会建立相应的资源表，其定义如下：

```
Typedef  struct
{
    Const char*  pName;          //文本(并非所有控件都需要)
    I16  Id;                     //控件窗口 Id
    I16  x0,y0,xSize,,ySize;     //控件的大小和位置
    I16  Flags;                  //控件专用标识
    I32  Para;                   //控件专用参数
}GUI_WIDGET_CREATE_INFO;
```

由于在消息响应的过程中，GUI 是根据控件的窗口 Id 来判断应该对谁作出响应的，因此，必须保证控件创建时 Id 的唯一性，除了系统已经给出的 Id 以外，用户也可以根据自己的需要自定义 Id。

(4) 整体流程

在 μC/GUI 回调机制的基础上，利用调用各种控件和创建相应的回调函数，就可以比较自如的实现各个界面之间的切换、参数输入和显示等功能。整个界面的基本流程如图 5-5 所示：

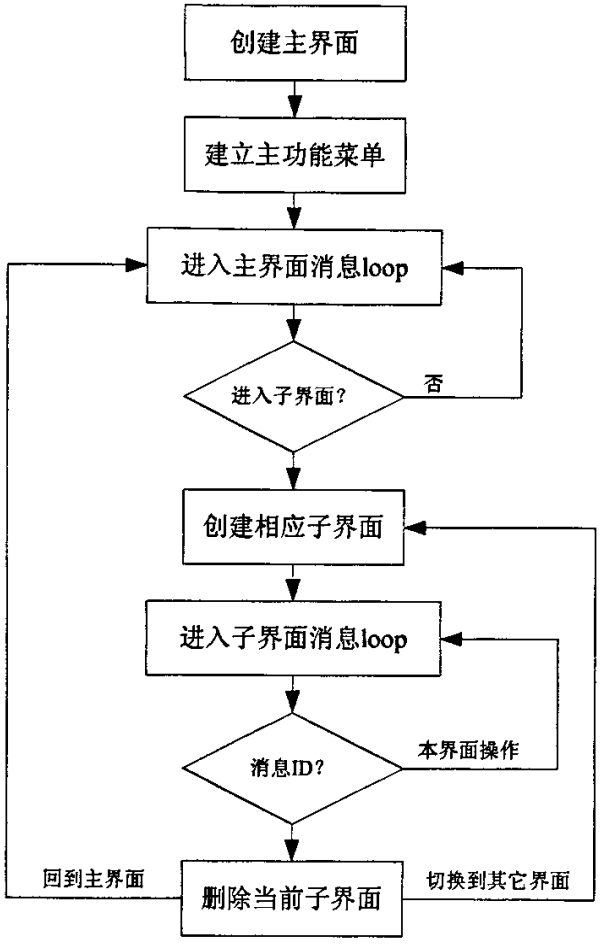

图 5-5 显示界面流程

§ 5.4 触摸屏任务

(1) 信号量机制

信号量实际上是一种约定机制，在多任务内核中普遍使用。它就像一把钥匙，任务要运行下去，需要先拿到这把钥匙。图 5-6 描述了任务、中断服务程序和信号量之间的关系。图中用钥匙或旗帜表示信号量，如果信号量用于对共享资源的访问，用钥匙符号，如果信号量用于表示某事件的发生，就用旗帜表示。

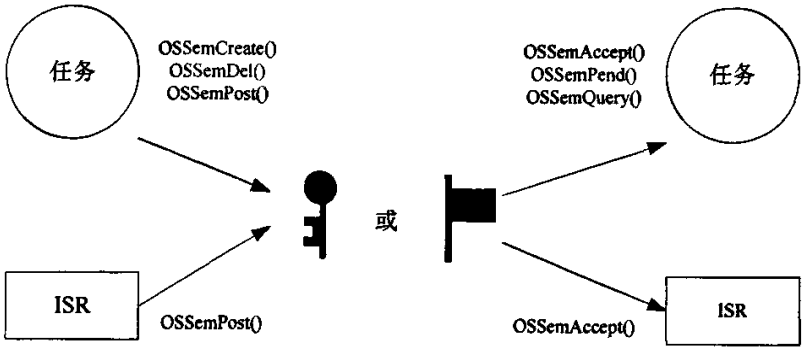

图 5-6 任务、中断和信号量的关系[32]

(2) 任务流程

μC/GUI 本身提供了对鼠标和触摸屏输入的支持，上一章节中，也已经完成了对所采用的四线电阻式触摸屏的驱动的编写。利用 μC/OS-II 内的信号量机制，采用了只在中断中置位触摸屏信号量(TOUCHSem)，而把读取坐标放到高优先级的任务中，处理完后，再将触摸屏任务挂起，等待下一次中断的触发。任务流程如图 5-7 所示[56][58]：

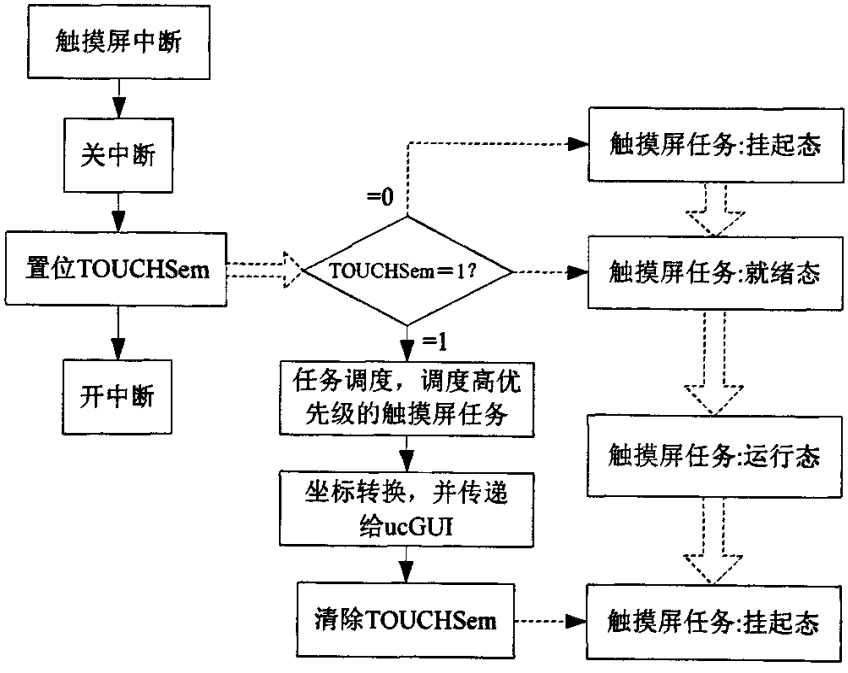

图 5-7 触摸屏任务流程

§5.5 串口通讯任务

(1) Modbus 协议

本设计中人机界面与织机主控制器之间通过传统的 RS-232 连接，使用工业领域流行的 Modbus 协议进行通讯。Modbus 协议中，用于串口的支持 ASCII，RTU 两种模式。ASCII 协议拥有开始和结束标志，在处理程序时更加方便，而且传输的是可见的 ASCII 字符，所以调试能更加直观，但是因为它传输的是 ASCII 字符，RTU 传输的数据每一个字节 ASCII 都要用两个字节来传输，为了提高通信的传输效率，本设计采用 RTU 模式。

Modbus 协议模式下协议格式如下：

[设备地址][功能码][地址高 8 位][地址低 8 位][操作个数高 8 位][操作个数低 8 位][CRC 校验低 8 位][CRC 校验高 8 位]

功能代码用于标识协议的功能，如读取线圈状态等。功能代码如表 5-4 所示。

表 5-4 Modbus 功能代码

功能码	名称	作用
01	读取线圈状态	取得一组逻辑线圈的当前状态(ON/OFF)
02	读取输入状态	取得一组开关输入的当前状态(ON/OFF)
03	读取保持寄存器	在一个或多个保持寄存器中取得当前的二进制值
04	读取输入寄存器	在一个或多个输入寄存器中取得当前的二进制值
05	强置单线圈	强置一个逻辑线圈的通断状态
06	预置单寄存器	把具体二进制值装入一个保持寄存器

地址高 8 位和低 8 位是指要读取或者写入的起始地址，操作个数高 8 位和低 8 位用于表示从起始地址开始要操作的内容的个数。协议格式最后的是 CRC 校验代码。

对于读取操作，如功能码 01～04 的命令，其响应格式为：

[设备地址][命令号][返回的字节个数][数据 1][数据 2]...[数据 n][CRC 校验低 8 位][CRC 校验高 8 位]

对于写入操作，如功能码 05～06 的命令，其响应的格式为将要收到的命令原样返回。

(2) CRC 校验

CRC 校验即是循环冗余校验，Modbus 协议中规定了 CRC 校验的生成规则：CRC 初始化为 HFFFF(CRC_L=HFF,CRC_H=HFF，H 表示后面的数是 16 进制)，将 CRC_L 与传输的第一个字节进行异或运算，然后将 CRC 进行右移(不循环)并判断：如一处的位是 1，则 CRC 再与 HA001 进行一次异或运算；如移出的是 0，则 CRC 不变。如此右移 8 词即可完成第一个字节的校验，重复上述操作直至全部字节校验完毕，所生成的 CRC16 位码即为传输校验码。

(3) 串口任务设置

本设计中的人机交互系统对于织机主控制器是主机，Modbus 协议设置为主机模式，按需要首先向主控器发送 Modbus 协议包，然后等待返回的数据包，接收到数据包后，根据 Modbus 的协议格式判断读取到的数据种类(线圈或寄存器)并刷新相应的数据值[52]。

发送 Modbus 数据包的工作只是 UART 发送的过程，所以不单独设置任务，在中断服务程序中完成。而在向主控制器发送 Modbus 数据包请求后，接收过程包括等待数据包的返回和对数据包的处理，因此设置了 UART 任务，并定义了信号量(UARTSem)，在 UART 接收中断中置位 UARTSem，将 UART 任务置为就绪状态，等完成相应的数据包处理后，再将 UARTSem 清除，挂起 UART 任务。

§5.6 效果演示和测试

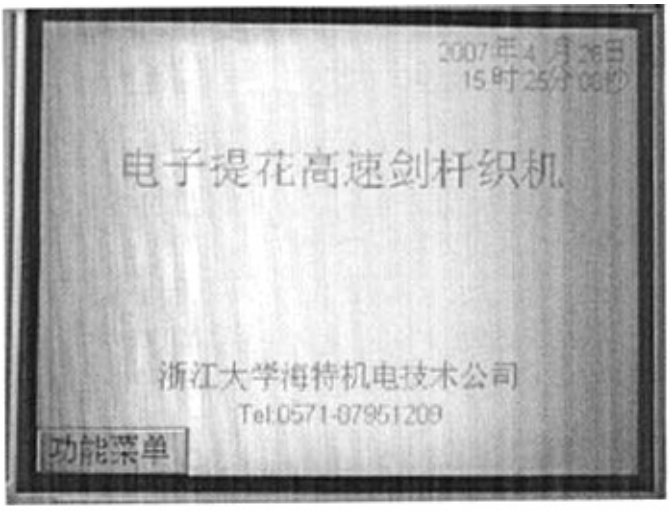

图 5-8 人机界面欢迎界面

通过以上各个任务的创建和实现，本人机交互系统已能够达到与织机主控制器之间的交互和显示实时的织机状态的功能。

人机交互系统上电后，首先进入欢迎界面，如图 5-8 所示，它包含了产品及公司名等基本信息。主功能菜单为弹出式，它是选择各种功能的入口。

参数查看包含了所有织机实时性能，图 5-9 为其中参数查看页面中的一页，它能够使操作工人更好地了解织机运行的状态。

图 5-9 参数查看页面

为了提高织机运行的安全性，织机参数的设置必须加以严格地管理，因此需要设置密码，以防止操作工人的误操作，登陆界面如图 5-10 所示；密码输入正确后，即可进行各项参数的设置，例如图 5-11 所示。

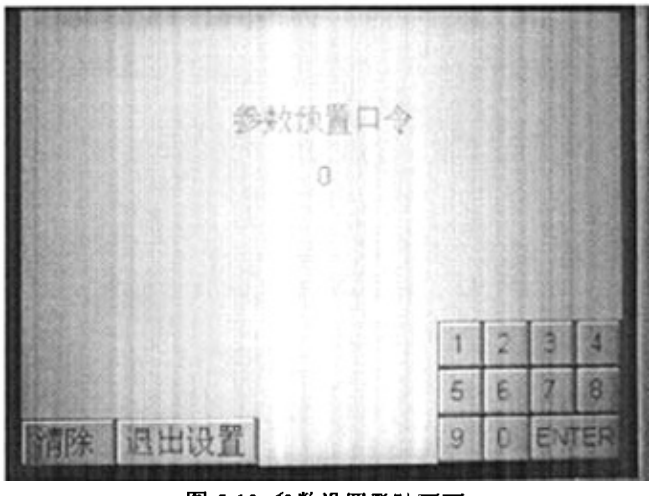

图 5-10 参数设置登陆页面

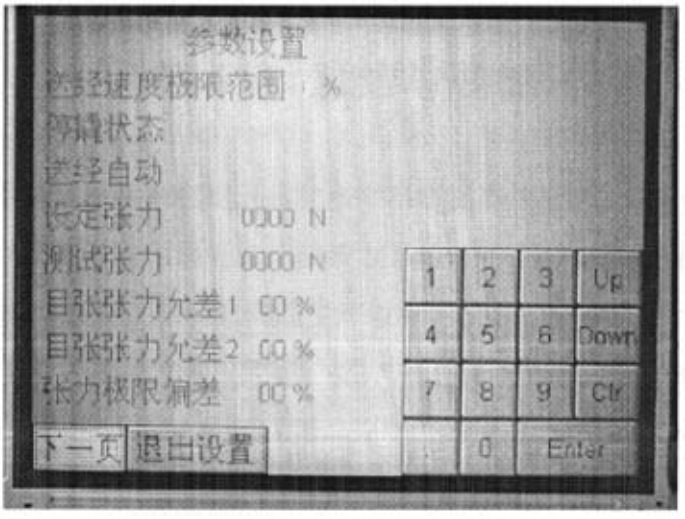

图 5-11 参数设置页面

以上演示的是人机交互系统的部分功能和界面，其它功能都在实验室的环境下通过测试，但是，由于实际生产环境中各种干扰非常复杂，因此，进一步的测试和改进还需要以后来进行完成。

第6章 总结与展望

本文针对目前纺织机械机电一体化水平不断提高的形势，适时地利用了不断趋于成熟的嵌入式技术，对原有的织机控制系统进行了必要和合适的改进，以改善控制系统的整体性能，实现更高的生产效率。USB 主机接口的设计解决了原有存储方法效率低、通用性差的弊端；由 ARM 硬件平台上移植嵌入式系统+嵌入式 GUI 构建的人机系统，性能优越，绘制界面便捷，并且适合不同的控制系统。

总结而言，本文完成了预期的改进方案：设计了 USB 主机接口和人机界面的软硬件系统，并进行了软硬件的测试。具体完成的内容如下：

(1) 了解和比较了当前流行的各种嵌入式系统和嵌入式 GUI，从中选择了最适合本设计的方案。

(2) 分析了 USB 通信协议，建立了基于 ISP1160 的 USB 主机硬件；并就本设计支持优盘的要求进行软件协议开发，主要包括与硬件相关的底层驱动，USB 主机软件，支持 Flash 盘的 Mass Storage 类协议和 FAT16 文件系统。

(3) 熟悉了 ARM7 芯片 S3C44B0X，针对人机界面的需要设计出了硬件平台，主要包括 ARM 外围电路，存储器，LCD 接口，触摸屏接口，JTAG 接口，串口和电源部分。

(3) 通过分析嵌入式操作系统 μC/OS-II 和嵌入式 GUI μC/GUI 的源代码，成功地实现在 S3C44B0X 平台上的移植。

(4) 利用 μC/GUI 本身对 UNICODE 的支持自建小汉字库，实现对中文的支持；编写以 ADS7843 作为控制器的触摸屏驱动，并成功移植到 μC/GUI。

(5) 编写了织机控制系统人机界面，包括参数设置、参数查看等 17 个画面，并利用控件和 μC/GUI 提供的回调机制实现整个人机界面。

(6) 利用 μC/OS-II 的多任务机制和信号量等，实现了对人机交互系统显示任务、触摸屏任务和串口任务之间的协调工作。

由于时间和实验条件的限制，本设计有些方面没有展开更深入的研究，本系统的设计仍有一些缺憾和不足，还有待进一步的完善，主要有：

(1) 本设计基于ISP1160的USB主机接口针对的是本系统所需的对优盘的支持，实现的是Mass Storage类协议和FAT16文件系统，而对其它USB设备类的驱动和协议有待进一步地研究。

(2) μC/GUI提供了强大的绘图和控件接口函数支持，但由于作者缺乏美工功底，编写人机界面的风格只能偏向简洁、朴实，以实现功能为主要目的，界面的显示效果有待进一步提高。

(3) 尽管本人机系统实现了基本的显示和人机交互功能，但是还有其它功能有待扩展。比如，给μC/OS-II添加对Tcp/ip协议以实现对以太网口的支持，这样用户就可以通过Internet对控制系统实现管理和信息的实时交互。

参 考 文 献

[1] 王辉．中国纺织业的方向是什么．www.texindex.com.cn.

[2] 黄凤根,吴文英,陈瑞琪．机电一体化在纺织机械中的应用．国外纺织技术，2002,12：1~2

[3] 周奉磊.纺织机械的机电一体化现状与发展趋势.中国纺织,2005,3:154

[4] 蔡建平．关于嵌入式应用开发技术．单片机与嵌入式系统应用，2001，3：5

[5] 熊光泽，罗蕾．嵌入式软件技术的现状和发展动向．计算机应用 20007，7：2~3

[6] 马义德，刘映杰，张新国．嵌入式系统的现状及发展前景．信息技术，2001,12：5~7

[7] 毛敏，喻翔．人机工程学中界面技术的发展分析．人类工效学，2003，1(3)：43

[8] 方志刚，刘加海，马卫娟．嵌入式计算机系统人机界面设计．人类工效学，1999，9(3)：19~21

[9] 方丰平，陈纯，卜佳俊．嵌入式环境下高性能可配置 GUI 系统设计．计算机工程与应用，2006，30：114~115

[10] 吴瑜．人机交互设计界面问题研究．2004：6

[11] 董建明．人机交互：以用户为中心的设计和评估．第一版．清华大学出版社，2003

[12] ISP1160 Embedded Universal Serial Bus Host Controller Datasheet．Philips Semiconductor Corporation．2003

[13] The Datasheet of S3C4510B. Sumsung Electronics CO LT.2000

[14] 陈启美，王刚，丁传锁等．USB 技术概况．电力自动化设备，2001，2：55~57

[15] Jan Axelson 著，陈逸译．USB 大全．北京：中国电力出版社，2001．46~78

[16] 常青，金科，张其善．基于 USB 的数据存储设备开发方案．遥测遥控，2006，11：51~54

[17] 陶鋆．手持式移动终端设备中 USB 模块的设计与实现．华中科技大学，2005

[18] 刘少峰，韦克平．USB 软件系统的开发．计算机应用研究，2002，3：102~104

[19] 罗悦泽，沈建华．嵌入式系统的硬盘文件系统．计算机工程，2004，1：176~177

[20] 孙骏，王晓蔚．嵌入式系统上的 USB-HOST 设计．现代电子技术，2004，24：89~92

[21] 兰文武，付桂翠，高泽溪等．基于 USB 接口的数据采集系统设计．电子技术应用，2004，2：20~22

[22] Bill Stanley, Dave Gilbert, Eric Luttman et al. Unversal Serial Bus Mass Storage Class Bulk-Only Transport. 1.0,1999.7~12

[23] Stuart Allman. Using the HID class eases the job of writing USB device drivers. EDN,2002,47(20):83~84

[24] 周立功等．USB 2.0 与 OTG 规范及开发指南．北京：北京航空航天大学出版社，2004

[25] 周立功等．ARM 嵌入式系统软件开发实例(二)．北京：北京航空航天大学出版社，2006

[26] S3C44B0X RISC MICROPROCESSOR Datasheet．Sumsang electronics．2000

[27] 16Mbit Multi-Purpose Flash SST39VF160 Datasheet．Silicon Storage Technology,Inc．2002

[28] 64Mb H-die SDRAM K4S641632 Datasheet．Sumsang electronics．2004

[29] Touchscreen Controller ADS7843 Datasheet．Burr-Brown Co．1998

[30] Max3232 Datasheet．Maxim Co．2001

[31] 800mA Low Dropout Voltage Regulator SPX1117 Datasheet．Sipex Co.2004

[32] 马海红，何嘉斌．基于 ARM 的嵌入式系统 FLASH 接口设计和编程．仪表技术与传感器，2005，1：39~42

[33] 江俊辉．居于 ARM 的嵌入式系统硬件设计．微机算机信息，2005，7-2：10~13

[34] 朱伟．S3C44B0X 控制 LCD 的设计与实现．工艺与应用，2006，3：53~56

[35] Jean J.Labrosse 著，邵贝贝等译.嵌入式实时操作系统 μC/OS-II(第 2 版)(MicroC/OS-II The Real-Time Kernel Second Edition).北京：北京航空航天大学出版社，2003

[36] 王铁永，候明善，吴盘龙．嵌入式操作系统 μC/OS-II 的特点及应用．控制工程，2003，1：74~76

[37] 赵宁，陈明，何鹏举．嵌入式操作系统 μC/OS-II 在 ARM 上的移植与应用．计算机技术与应用，2004，4：30~34

[38] 李明．μC/OS-II 在 ARM 上的移植．电子设计应用，2003，4：46~49

[39] 罗强,游大海,何红艳．基于嵌入式 GUI 的电力自动化设备人机界面设计．电力自动化设备,2004,9：63~65

[40] 刘滨，刘兵，赵艳华．基于 μC/GUI 的嵌入式图形界面设计．液晶与显示，2005，6：559~561

[41] 祁献鹏，郭玉东．MiniGUI-面向嵌入式系统的 GUI 系统．信息工程大学学报，2001，9：8~10

[42] 周进．μC/GUI 在基于 Nios 的嵌入式系统中的研究与实现．南京理工大学，2006

[43] 李周勇．DSP 嵌入式系统的图形用户界面(GUI)设计和实现．上海交通大学，2006

[44] 罗雪莲，宋树祥．基于 ARM 的嵌入式系统触摸屏设计．电气时代，2004，12：72~73

[45] 陈世利，孙墨杰，栗大超等. 触摸屏的工作原理及典型应用. 单片机与嵌入式系统应用，2002，2：10~13

[46] 赵芝璞，金小俊. 触摸屏控制器 ADS7846 的原理及应用. 国外电子元器件，2002，5：46~48

[47] 胡冰，吴升艳. ADS7843 触摸屏接口. 国外电子元器件，2002，7：27~29

[48] 李向阳，奚大顺. 在 μC/GUI 中实现汉字显示. 单片机与嵌入式系统应用，2005，5：75~76

[49] Swaminathan, Vishnu. Real time task scheduling for energy-aware embedded system[J]. Journal of the Franklin Institute, September, 2001

[50] L.Maillet, C.Fraboul. Scheduling complex real-time tasks in an embedded distributed system. IEE 7th Euromicro Workshop on Real-Time Systems. Mar 14-16,1995,p6

[51] Graunke P T, Krishnamurthi S. Advanced control flows for flexible graphical user interfaces or growing GUIs on trees or bookmarking GUIs. Software Engineering, 2002. ICSE 2002,Proceedings of the 24rd International Conference on,2002,277~287

[52] 冉明，邢汉承. 基于 μC/OS-II 的嵌入式系统的设计. 微机发展，2005，1：91~93

[53] 沈金荣，刘翔. μC/OS-II 内核结构分析及多任务调度实现. 计算机工程，2006，12：85~87

[54] 陈媛，黄贤英，李卫东. μC/OS-II 在任务调度与中断处理的实现机理. 自动化与仪器仪表，2004，1：46~47

[55] 孟健，刘建辉. 基于 μC/OS-II 的嵌入式数控系统分析与设计. 微计算机信息，2005，11-2：65~67

[56] 胡俐蕊. 基于信号量痛惜的 μC/OS-II 应用程序设计方法. 微型计算机信息，2005，10-2：40~42

[57] 闵联营，杨进勇. 基于 μC/OS-II 的串口驱动程序设计与实现. 武汉理工大学学报，2005，4：115~117

[58] 殷惠莉，刘少君，黄道平. 基于 μClinux 触摸屏的设计. 电子工程师，2004，2：17~19

致 谢

两年的研究生生活即将过去，回首这段时光，我过得忙碌而充实，学到了很多知识。这与老师的指导，同学的帮助和鼓励密不可分。没有他们，我不可能顺利地完成自己的项目，也不会顺利地完成这篇毕业论文，在此我想表示对他们的感谢和敬意。

首先感谢我的导师陈宗农教授。陈老师知识渊博，无论是在技术上还是在管理上都具有很高的水平；对人尤其是对学生态度和蔼、亲切，在学习、工作和生活各方面都给予了我很大的帮助；最让我敬佩的是陈老师严谨的治学态度和踏实的工作作风，潜移默化中改变了我对事的态度，必将让我受益终生。

在这里还要感谢机械设计研究所的詹建潮副教授和王庆九老师等，他们也给了我很多的帮助，感谢他们的耐心指导和悉心教诲；同样也要感谢实验室的师兄朱新杰、蒋振磊、曾科、师姐臧国杰及谢真、王炉意、刘刚，正是他们的帮助和陪伴，才让我的 2 年研究生生涯顺利而精彩。

最后，感谢我的家人和好友，正是他们的关心和支持，让我一步一步地成长。

姚冬冬

二零零七年五月于求是园