摘　要

在工业生产中，二次自动化仪表是构成自动化系统的基本单元之一。我国的单元仪表已基本完成由电动III型仪表向基于八位或十六位单片机为基础设计的数字化仪表的转换。由于常规单片机资源的限制，以单片机为基础设计的单元仪表基本上还是在功能上替代电动III型仪表，并按电动III型功能进行分类。这样造成国内自动化仪表生产厂家生产的二次数字化仪表品种繁杂，标准难以统一，设计随意性大。因此带来如下现实问题：

1. 自动化系统设计单位的仪表选型、系统调试、使用中操作、维修和系统的功能优化及备件的准备非常的不方便；
2. 仪表生产厂家的批量生产困难，产品质量的提高及成本的节约不利；
3. 国内现在自动化仪表厂家数量众多，但都无法形成规模生产，质量不佳，而国外进口的二次仪表往往依附于特定的集散系统，也存在标准不统一，难以灵活替换的问题，且价格昂贵。

自动化系统设计、生产及应用迫切需要一种使用方便、通用性强的智能型二次仪表，以解决上述问题，改变传统设计、生产及应用方式，这将是未来自动化仪表的发展趋势，也就是本课题的努力方向。

本论文正是针对上述问题，以设计出一种可灵活组态的通用智能型二次仪表为研究对象，在深入分析国内主流仪表厂家的仪表操作方式和仪表功能的基础上，合理地进行软硬件设计，为在同一硬件平台下实现多种仪表的功能进行了创新性和探索性研究。主要内容为：

1. 各种常规二次仪表功能、标准、接线、操作习惯及结构方式的归类分析；
2. 多信号多量程的柔性测量方法研究；
3. 系统整机设计以及系统可靠性设计；
4. u-boot 的向 ARM 的移植、uClinux 向 ARM 的移植、uClinux 下的通用组态软件设计。

本文设计了一种以三星公司的 ARM7TDMI 系列处理器 S3C44B0X 为核心，辅以外围电路，实现同一硬件平台下多种仪表的功能，并成功制作了样品系统。

本文所讨论的基于 S3C44B0X 和 uClinux 的智能仪表系统的开发技术同样适用于其它项目的开发，对其它嵌入式的应用系统开发有重要的参考价值。

关键词：ARM,S3C44B0X，u-boot,uClinux，测量，智能仪表

ABSTRACT

In the industrial production, two Process Automation Instrumentation constitutes one of automated system basic units.Our country's unit measuring appliance oneself complete basically by the DDZ-III appliance to based on 8 or 16 MCU into foundation design digitized measuring appliance transformation.As a result of the conventional MCU resources limit, basically substitutes electrically operated take the monolithic integrated circuit as the foundation design unit measuring appliance in the function DDZ-III Measuring appliance, and according to the DDZ-III The function carries on the classification.Like this creates the domestic Process Automation Instrumentation manufacturer production two digitized measuring appliance variety to be numerous and diverse, the standard unifies with difficulty, the design capriciousness is big.Therefore brings the following realistic question:

1. automated system designing department's measuring appliance shaping, the system debugging, the use operates, the service and the system function optimization and the spare parts preparation unusual is not convenient;

2. measuring appliance manufacturer volume production difficulty, the product quality enhancement and the cost save disadvantageously;

3.Domestic present Process Automation Instrumentation factory quantity is multitudinous, but all is unable to form the scale production, the quality is not good, but the overseas import two measuring appliances often attach in the specific collection and distribution system, also has the standard not to be unified, nimbly replaces with difficulty the question, also the price is expensive.

The automated system design, the production and the application urgent need one kind of easy to operate, the versatile strong intelligence two measuring appliances, solve the above problem, the change tradition design, the production and the application way, this will be the future Process Automation Instrumentation trend of development, also is this topic diligently the direction.

The present paper is precisely in view of the above question, take designs one kind to be possible the nimble configuration general intelligence two measuring

appliances as the object of study, in the thorough analysis domestic mainstream measuring appliance factory measuring appliance operating mode and in the measuring appliance function foundation, carries on the software and hardware design reasonably, for realized many kinds of measuring appliance function under the identical hardware platform to carry on the innovation and the exploring research.The primary content is:

1.each kind of conventional two measuring appliance functions, standard, wiring, operation custom and structure way classification analysis;

2.signal multi-measuring range flexibility measuring technique research;

3.system design and the reliable design of system;

4.U-BOOT to ARM transplant, uClinux to ARM transplant, general configuration software design based on uClinux.

This article has designed one kind take samsung Corporation's ARM7TDMI series processor S3C44B0X as a core, auxiliary by periphery electric circuit, under realization identical hardware platform many kinds of measuring appliance function, and we has manufactured the sample system .

This article discusses is suitable similarly based on S3C44B0X and the uClinux intelligent measuring appliance system development technology for other project development, has the important reference value to other embedded application system development.

Keywords: ARM,S3C44B0X,u-boot,uClinux,Measurement, Intelligent measuring appliance

图表目录

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，也不包含为获得电子科技大学或其它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。

签名：刘俊　　日期：2007年 6月7日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、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。本人授权电子科技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应遵守此规定）

签名：刘俊　　导师签名：

日期：2007年 6月7日

第一章 绪论

1.1 智能仪表的发展概况及现状

从 20 世纪 60 年代开始，为满足工业发展的需要，将测量记录和控制功能组合在一起产生了较早的测控仪表，这类仪表称为“基地式”仪表。这种仪表通常是以在带有调节单元的显示记录仪“基地”上，配上测量元件及执行器构成简单控制系统。随着生产规模的扩大，产生了以功能划分的“单元组合式”仪表。根据不同的控制要求，选择相应仪表单元组合起来构成各种不同复杂程度的控制系统。无论是“基地式”仪表还是“单元组合式”仪表，它们的共同特点都是模拟式的，采用的是模拟技术，为模拟仪表时代，控制系统以经典控制理论为基础。

然而，随着计算机技术的迅猛发展，测量仪器仪表产业迎来了一次技术革命，取得了巨大的进步。20 世纪 80 年代，随着计算机技术的发展及其在仪器仪表中的应用， 以微处理器为核心器件的微机化仪表应运而生， 产生了各种数字式变送器、数字式调节器、数字式显示记录仪、可编程控制器和智能仪表。数字化仪表具有模拟式仪表无法比拟的优势，主要特点是采用了数字技术。从而，计算机技术能够应用于仪器仪表和控制领域，计算机控制系统在工业控制中也得到广泛的应用与推广。现代仪器仪表在测量过程自动化、测量结果的数据处理及系统控制等诸多方面取得了很大进展。到 90 年代，测量仪器的高准确度、高性能以及多功能特性大多都得益于微处理器技术的推广。随着信息时代的来临，人们对仪器仪表的观念得到了进一步更新。当前，国际上自动化仪表与系统正进行着一场巨大的变革，即仪表的总线化，智能化，网络化，虚拟化等。

1.2 课题研究的目的与意义

在工业生产中，二次自动化仪表是构成自动化系统的基本单元之一，我国的单元仪表已基本完成由电动III型仪表向基于八位或十六位单片机[1][2][3]为基础设计的数字化仪表的转换，但由于常规单片机资源的限制，基本上还是处于在功能上替代电动III型仪表的阶段，并按电动III型功能进行分类（略有扩充）。造成国内自动化仪表生产厂家生产的二次数字化仪表品种繁杂，标准难以统一，仪表设计随意性大。这给设计单位选型、系统调试、使用中操作、维修、功能的优化及备件的准备带来极大的不便；给仪表生产厂家的批量生产带来困难，不利于产品质量的提高及成本的节约。这也是国内现在自动化仪表厂家数量众多，但都无法形成规模生产，质量不佳的主要原因之一。而国外进口的二次仪表往往依附于特定的集散系统，也存在标准不统一，难以灵活替换的问题，且价格昂贵。

自动化系统设计、生产及应用迫切需要一种使用方便，通用性强的智能型二

次仪表，以解决上述问题，改变传统设计、生产及应用方式，这将是未来自动化仪表的发展趋势，也就是本课题的努力方向。

本项目拟利用32位ARM处理器丰富的可配置硬件资源及强大的数据处理能力设计出一种可灵活组态的通用智能型二次仪表，完成各种常规自动化仪表单元的功能。其作用类似于常规逻辑控制中的PLC(虽然PLC也可以灵活组态，但在操作、显示适用性及成本上无法替代常规过程测控中的二次仪表)。系统设计选用时，用户只需要选择仪表单元AI、AO、DI、DO的数量、具体完成的功能、人机界面设置等。

本仪表设计打破了常规自动化仪表依据持定功能(如流量测量仪、调节器等)要求进行设计生产的模式。通过同一硬件平台下组态软件设置单元仪表功能，方便了设计维护、调试、维修、及备件的订购及贮备。有利于生产厂家的大规模生产及产品质量提高和成本的控制。克服了传统数字化仪表由于批量小，难以支持按设计要求订制专用集成芯片的不足。

二次自动化仪表国内需求很大，据统计仅数显表一个系列品种，国内年需求量就达到几万台。如果能达到规模生产的目的，将会使自动化仪表行业的生产及使用产生巨大的变化，创造可观的经济及社会效益。

1.3 本论文的研究内容和技术难点

1.3.1 本论文的研究内容

本论文以设计出一种可灵活组态的通用智能型二次仪表为研究对象，在深入分析国内主流仪表厂家的仪表操作方式和仪表功能的基础上，合理地进行软硬件设计，为在同一硬件平台下实现多种仪表的功能进行了创新性和探索性研究。主要内容为：

1. 各种常规二次仪表功能、标准、接线、操作习惯及结构方式的归类分析；
2. 多信号多量程的柔性测量方法[4][5]研究；
3. 系统整机设计以及系统可靠性设计；
4. U-BOOT的向ARM[6][7]的移植、UCLINUX[8]向ARM的移植、通用组态软件设计。

1.3.2 本论文的技术难点

本论文所讨论的基于S3C44B0X和UCLINUX的嵌入式智能仪表的主要技术难点在于：

1. 将各种自动化仪表功能进行综合，合理地进行软硬件设计，以得到良好的功能组态界面及组态后形成的各种功能仪表操作界面符合广大用户的使用习惯。在这方面拟参考PLC的组态功能设计组态界面，综合国内主流仪表厂家的仪表操作方式，并征求用户及设计院意见进行操作界面设计，并完

成了一部分调研工作。ARM 丰富的片内资源也为功能强大使用方便的软件设计提供了硬件基础。不同种类的二次仪表显示内容、显示方式差异极大，本设计用整屏液晶显示解决此问题。

2. 由于通用仪表可完成各种自动化仪表功能，简单地把各种常规仪表硬件电路组合在一起将会使系统电路复杂，成本提高并降低产品的可靠性。所以本课题的另一个技术难点就是以尽可能用简单的电路，利用各种先进前端测量及后端控制技术，充分利用软件功能，在保证高精度高可靠性的前提下完成系统硬件设计。
3. 本设计采用嵌入式 LINUX[9][10][11]操作系统，这使得本系统设计时可以更加灵活，软件功能更加强大，但另一方面也因 LINUX 的移植困难、源码繁多而使软件编程更加困难（相比使用 uCOS-II[12]而言），这是本设计的又一大难点。
4. 成本控制及可靠性设计是本项目成功的关键，也是本设计首先应该克服的困难。本项目单机成本主要受显示液晶、ARM 芯片及多信号输入处理硬件设计的影响，其中多信号输入处理硬件设计已找到合理解决方案，显示单元液晶成本快速下降，在达到量产后，320*240 液晶成本可控制在 200 元以内，同时拟采用彩色和黑白二种液晶供选择。ARM 芯片现已开始大量进入市场，其内部带有比一般单片机丰富的硬件资源，成本和一般单片机扩展外部功能器件相当，据估计 5 年内其价格还有大的下降空间。即本项目硬件成本和常规数字仪表设计相当，但由于生产品种规格简单，有利于达到批量生产，有条件订制显示液晶及 ARM 芯片，促使硬件成本的下降。同时使设计选型、调试、维修等各环节软成本大幅降低。ARM 的高集成度和量产的实现也为可靠性提高创造了条件。
5. 技术指标：系统设计结构、电气特性及接线方式满足电动III型仪表国家标准要求。仪表功能满足多种二次仪表功能要求。输入信号包括热电偶[13]、热电阻[14]、各种标准信号、频率信号及现场总线信号等，输出信号包括标准信号、功率输出开关信号、现场总线信号等变送、控制、报警信号。输入输出信号精度达到千分之二。控制精度达到千分之五。功能的保证及精度的要求也成为本设计的难点。

第二章 嵌入式系统概述

2.1 嵌入式系统的定义

嵌入式系统[15][16]是指以应用为中心，以计算机技术为基础，软件硬件可剪裁，适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。它主要由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户应用软件等部分组成。它通常嵌入在主要设备中运行，主要用于实现对其它设备的控制、监视和管理等功能。

2.2 嵌入式系统的特点

与通用型计算机系统相比较而言，嵌入式计算机系统具有以下特点：

1. 嵌入式系统通常采用面向特定应用的嵌入式 CPU

嵌入式 CPU 大多工作在为特定用户群设计的系统中，它们通常都具有低功耗、体积小、集成度高等特点，能够把通用 CPU 中许多由板卡完成的任务集成在芯片内部，从而使嵌入式系统的设计趋于小型化，移动能力大大增强，跟网络的融合也越来越紧密。

2. 对软、硬件的设计要求高

嵌入式系统的硬件和软件都必须高效率地设计，量体裁衣、去除冗余，力争在同样的硅片面积上实现更高的性能，这样在产品设计中嵌入式处理器和嵌入式操作系统的选择成为系统设计的关键所在。

3. 以特定应用为目标

嵌入式系统往往和具体应用有机地结合在一起，它的升级换代也是和具体产品同步进行，因此嵌入式系统产品一旦进入市场，都具有较长的生命周期。

4. 固化存储

为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机本身中，而不是存贮于磁盘等载体中。

5. 开发环境

嵌入式系统本身不具备开发能力，即使设计完成以后用户通常也不能对其中的程序功能进行修改，必须借助于一套开发工具和环境才行。

2.3 嵌入式系统的发展历程

嵌入式系统的出现至今已经有 30 多年的历史，近几年来，计算机、通信、消费电子的一体化趋势日益明显，嵌入式技术已成为一个研究热点。纵观嵌入式技术的发展过程，大致经历了四个阶段。

第一阶段是以单芯片为核心的可编程控制器形式的系统，具有与监测、伺服、指示设备相配合的功能。

第二阶段是以嵌入式微处理器为基础、以简单操作系统为核心的嵌入式系统。主要特点是:CPU 种类繁多，通用性比较弱;系统开销小，效率高;操作系统达到一定的兼容性和扩展性;应用软件较专业化，用户界面不够友好。

第三阶段是以嵌入式操作系统为标志的嵌入式系统。

第四阶段是以 Internet 为标志的嵌入式系统。

2.4 当今嵌入式技术的热点

综上所述，今天的嵌入式技术已经达到了一个新的高度，尤其具有以下三个发展热点:

1. 由 8 位或 16 位嵌入式微处理器向 32 位 RISC 嵌入式微处理器过渡

2. 嵌入式操作系统逐步走向成熟，与嵌入式系统的结合日益紧密

3. 嵌入式 Internet 技术全面应用

可见，在未来嵌入式技术的发展进程中，32/64 位微处理器将在系统中占主导地位，嵌入式操作系统的功能将更加强大，系统与网络、与 Internet 的结合将日益密切，甚至是直接嵌入到 Internet 中去应用，以此为基点推动嵌入式系统向更广泛、更深入的方向发展。

第三章 S3C44B0X 处理器概述

3.1 S3C44B0X 处理器简介

S3C44B0X[17][18]是三星公司为手持设备和工业应用等一般类型应用提供的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，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），PWM 功能的 5 通道定制器，I/O 端口，RTC，8 通道 10 位 ADC，I^2C-BUS 接口，IIS-BUS 接口，同步 SIO 接口和 PLL 倍频器。下图 3-1 是 S3C44B0X 的内部结构框图。

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的全静态设计特别适用于对成本和功耗敏感的应用。同样 S3C44B0X 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）。

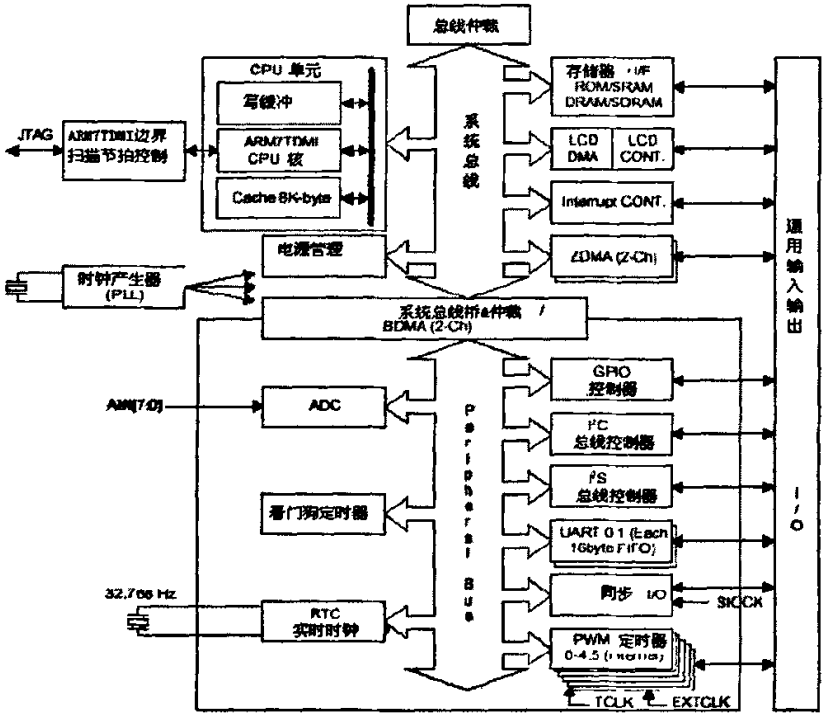


图 3-1 S3C44B0X 内部结构框图

3.2 S3C44B0X 存储器空间划分

下图 3-2 为 S3C44B0X 复位后的存储器地址分配图。

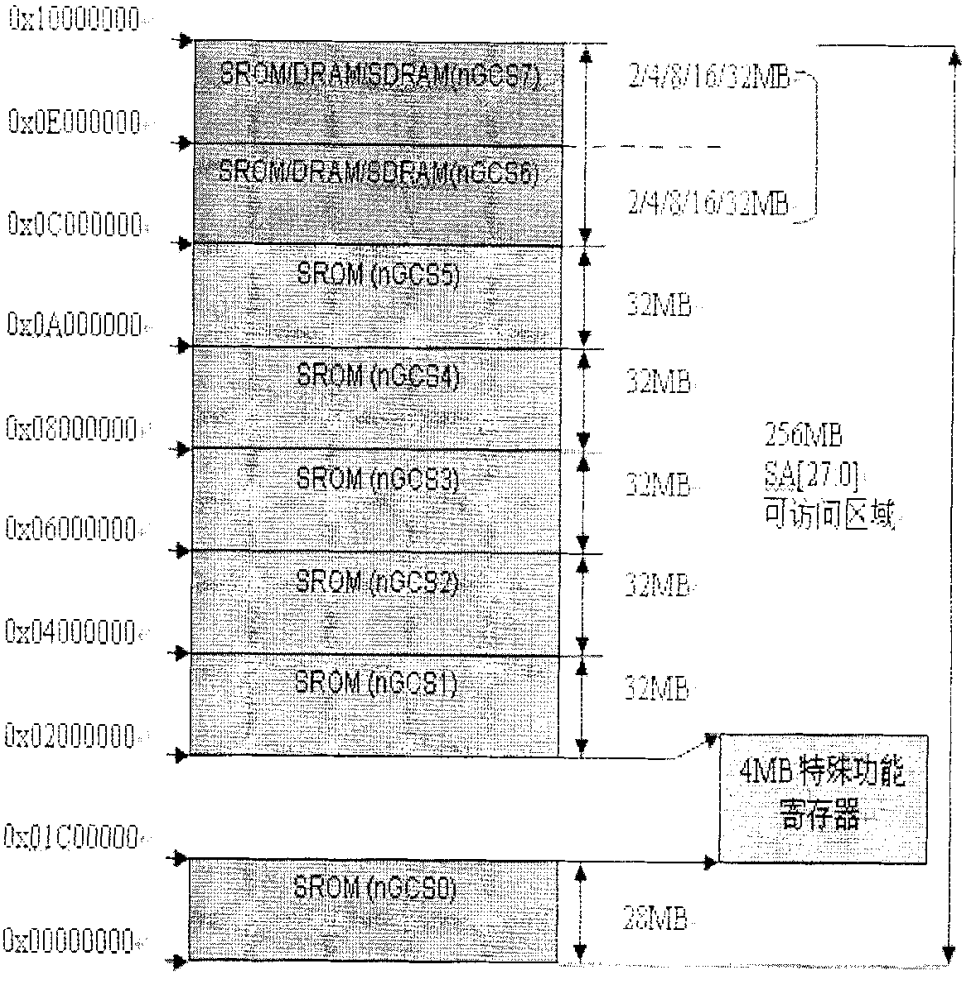


图 3-2 S3C44B0X 复位后的存储器地址分配图

从图中可以看出：

★特殊功能寄存器位于 0x01C00000～0x02000000 的 4MB 空间内

★Bank0～Bank5 的起始地址和空间大小是固定的

★Bank6 的的起始地址固定的，空间可配置为 2/4/8/32MB

★Bank7 的空间大小和 Bank6 一样可变。Bank6 和 Bank7 的详细地址和空间大小可参考 3-1 表。

表 3-1 Bank6 与 Bank7 的详细地址

地址/大小	2MB	4MB	8MB	16MB	32MB
Bank6					
起始地址	0x0C000000	0x0C000000	0x0C000000	0x0C000000	0x0C000000
结束地址	0x0C1FFFFF	0x0C3FFFFF	0x0C7FFFFF	0x0CFFFFFF	0x0DFFFFFF
Bank7					
起始地址	0x0C200000	0x0C400000	0x0C800000	0x0D000000	0x0E000000
结束地址	0x0C3FFFFF	0x0C7FFFFF	0x0CFFFFFF	0x0DFFFFF	0x0FFFFFFF

不难从上图看出，对外围器件的访问地址都是通过 nGCS 来控制的。而加载镜象的起始地址是 0x0C000000。

3.3 S3C44B0X I/O 功能配置

S3C44B0X 有 71 个通用可编程多功能输入/输出引脚，可分为以下 7 类端口：

◆两个 9 位输入/输出端口(PortE 和 PortF)

◆两个 8 位输入/输出端口(PortD 和 PortG)

◆一个 16 位输入/输出端口(PortC)

◆一个 10 位输出端口(PortA)

◆一个 11 位输出端口(PortB)

除了 PortA 和 PortB 外每个端口都包含三组寄存器，分别是：端口配置寄存器(PCONA-G)；端口数据寄存器(PDATA-G)；端口上拉设置寄存器(PUPC-G)。

◆端口配置寄存器(PCONA-G)

由于多数端口都是多功能口，因此，需要用“端口配置寄存器 PCONn”来设置每个引脚工作在哪一个功能模式下。

◆端口数据寄存器(PDATA-G)

当端口被设置为输出脚时，输出数据的方法就是将数据写入到 PDATAn 的相应位中；当端口被设置为输入脚时，读入数据的方法就是将 PDATAn 中的相应位读出。

◆端口上拉设置寄存器(PUPC-G)

端口上拉寄存器用来设定 PC-PG 这几组端口是否具有内部上拉。当 PUPn 的对

应位为 0 时，该引脚上的上拉使能，当为 1 时，该引脚上的上拉禁能。

每个端口都可以通过软件设置来满足各种系统设置和设计要求。

第四章 uClinux 嵌入式操作系统概述

4.1 uCliux 的系统简介

Linux 是 1991 年 10 月由芬兰学生 Linus Torvalds 开发的并发布的，是目前常见和使用比较多的操作系统。

Linux 最初设计为桌面系统，现在广泛应用于服务器领域。它支持多种体系结构，支持大量外设，网络功能完善，与 UNIX 系统兼容，开放源代码并有丰富的软件资源，内核稳定而高效，大小及功能均可定制。正是由于 Linux 自身的一些优良特性，能在很大程度上满足嵌入式操作系统的特殊要求，催生了一些嵌入式 Linux 系统，其中就包括 uClinux。

uClinux 是 Linux2.0 版本的一个分支，被设计用来微型控制应用领域。uClinux 单词中 u 代表 Micro，C 代表 Control，uClinux 的含义就是 Micro-Control-Linux，意思是"针对微控制领域而设计的 Linux 系统"，单词发音"you see linux"。

uClinux 主要针对无内存管理单元 MMU（Memory Management Unit）的处理器设计，支持多任务，具有完备的 TCP/IP 协议栈并支持多种网络协议。uClinux 还支持多种文件系统，如 ROMFS，NFS，EXT2，FAT16/32。实际上，uClinux 已经成功应用于路由器、网络摄像机、机顶盒、PDA 等诸多领域。uClinux 首先被移植到 MOTOROLA（现在的 FreeScale）的 MC68328 龙珠微处理器上。之后，它越来越被业界所青睐，被移植到了更多的无 MMU 的处理器上。uClinux 发行至今已经历多个版本，最新的版本为 uClinux-dist-20051103，目前的 uClinux 不仅支持 NOMMU 处理器，同时也支持 MMU 处理器，包括 ARM、MIPS、sh、68K、x86、SPARC，甚至 blackfin 等架构的高性能处理器，并在 60 个以上基于这些架构处理器的开发平台成功实现移植。

4.2 uCliux 的组成结构

4.2.1 uClinux 的系统组成

一个基于 uClinux 的完整的嵌入式系统由三个部分组成：BootLoader、uClinux 操作系统内核和文件系统。如下图 4-1 所示。

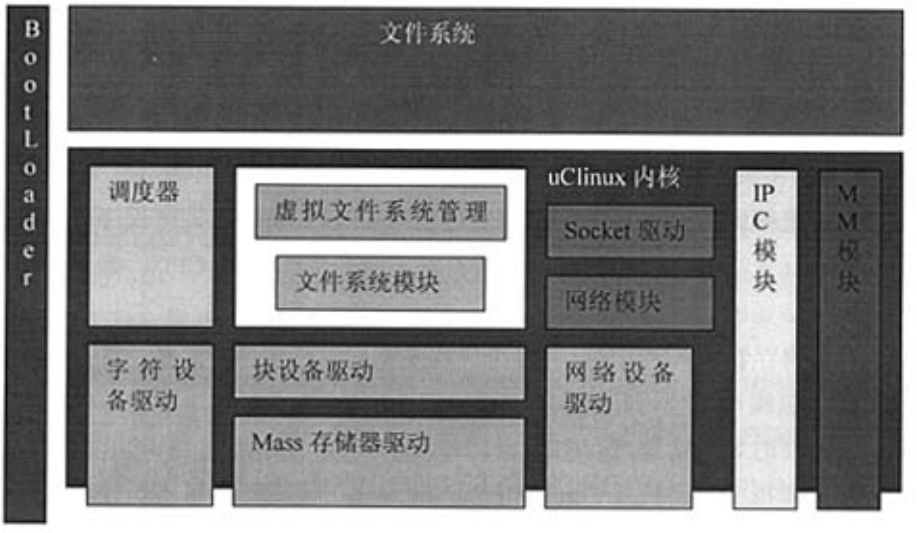


图 4-1 uClinux 系统组成

BootLoader：系统引导程序，主要负责系统初始化系统资源，包括 SDRAM，然后将系统控制权交给操作系统，控制权交给操作系统后，系统的运行和 Boot Loader 再无任何关系。uClinux 的发行包中不包含 Boot Loader，Boot Loader 必须由用户自己设计，但用户可以直接使用或参考一些开源的 Boot Loader 软件工程，如 u-boot，本论文会在第七章中将讲述 u-boot 在开发板上的移植。

uClinux 内核：完成的功能和大家熟悉的 Linux 内核相同，包括设备驱动、进程调度、内存管理（MM）、文件系统管理、网络接口和进程间通信（IPC）。使用未压缩的系统内核，一般要占用 400K 字节到 900K 字节空间，如果是压缩后的系统内核，则占用空间一般在 300K 字节到 500K 字节之间。

文件系统主要存储用户应用程序，同时包括系统配置文件、系统程序和必需的驱动程序，其占用空间由用户的应用程序规模决定，从 200K 字节到 1M 字节不等。Boot Loader、uClinux 内核和文件系统固化在 Flash 中，分为三个区存放，其中 Boot Loader 的首地址为系统复位的入口地址，系统内核和文件系统可以根据用户的需要指定首地址，各部分存储的空间只要保证不冲突，可以固定一段存储区域，也可以后一部分紧接前一部分，采用前一种存储方式的优点是方便内核的加载和文件系统的挂载，利于系统内核的和用户程序的调试，缺点是会浪费一部分空间，如下图 4.2 所示。

图 4-2 常见的 Flash 空间分配

4.2.2 uClinux 的内核结构

Linux 的内核主要包括五个子系统如图 4-3 所示:

（1） 进程调度（SCHED）控制着进程对 CPU 的访问，调度程序使用一种较简单的基于优先级的进程调度算法使所有进程能公平地访问 CPU，确保内核在任意时刻执行必要的硬件操作。

（2） 内存管理（MM）允许多个进程安全地共享主内存区域。并且支持虚拟内存，使得进程可以使用超过实际内存大小的地址内存空间，不用的块保留在磁盘上，在需要时导入物理内存中。

（3） 虚拟文件系统（Virtul File System,VFS）隐藏了各种不同硬件的具体细节，为所有设备提供了统一的接口。VFS 可分为逻辑文件系统和设备驱动程序，逻辑文件系统指 Linux 所支持的文件系统如 ext2,FAT 等，设备驱动程序指的是为每一种硬件控制器所编写的驱动程序。

（4） 网络接口（NET）提供对各种网络标准协议的存取和各种网络硬件的支持。网络接口可分为网络协议和网络驱动程序两部分，前者负责实现每一种可能的网络传输协议，后者负责与硬件设备进行通信，每一种可能的硬件设备都有相应的设备驱动程序。

（5） 进程间通信（IPC）支持系统内部进程间通信机制。

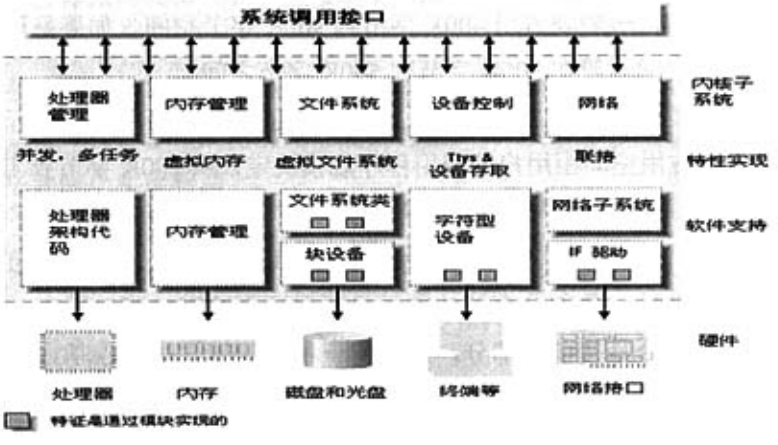


图 4-3 Linux 内核结构

4.2.3 uClinux 的文件目录结构

uClinux 的文件目录结构如图 4-4 所示。

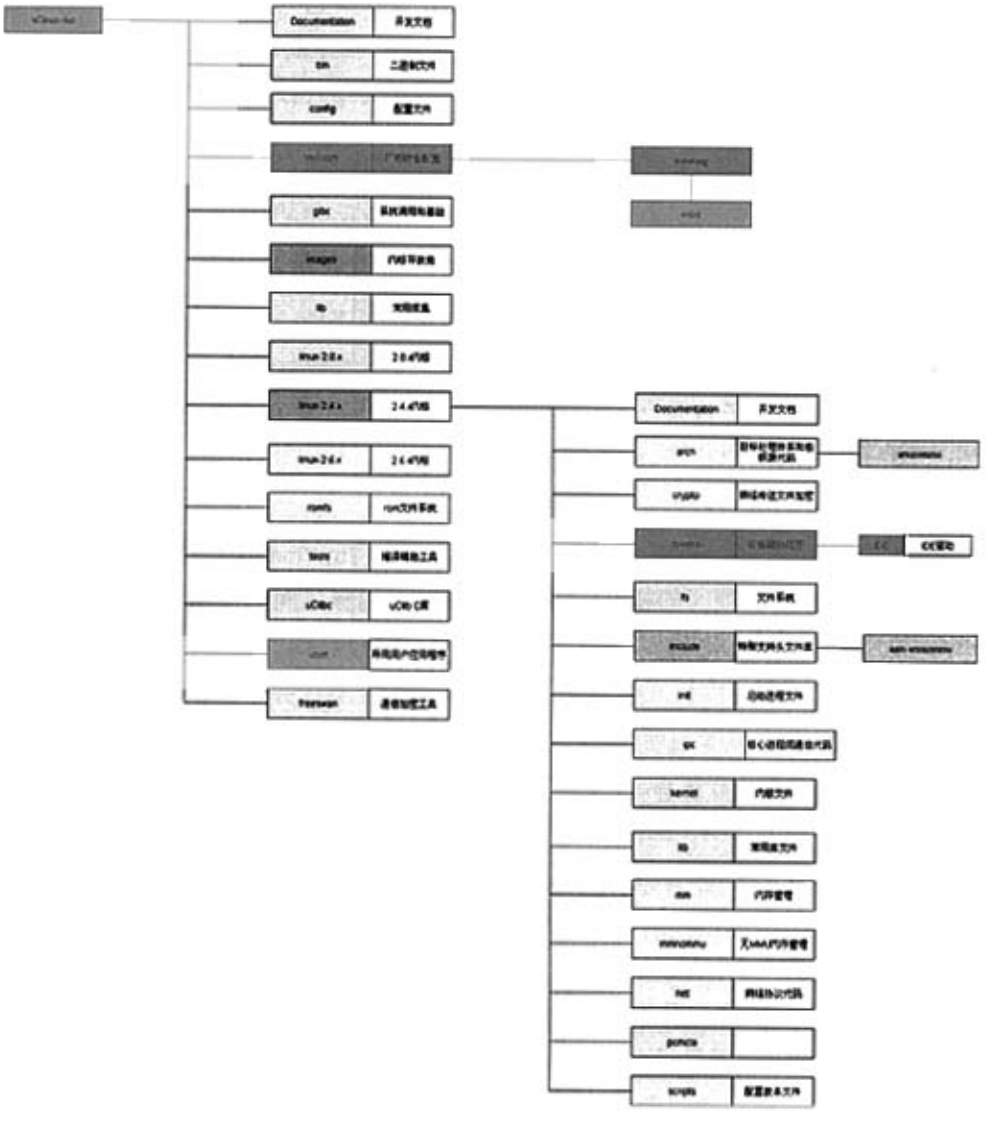


图 4-4 uClinux 目录结构

注：图中用特殊颜色标记项在日常开发中经常用到。

①. arch 目录：结构/平台源专用文件放在子目录“arch”下。在 ARM/uClinux 状态下，其子目录 “armnommu”用来区分由 uClinux 开发者所写的程序和 Linux 的源

程序。

②. driver 目录：该子目录包括所有的设备驱动程序。这些驱动进一步被分为块(block)设备、字符 (char)设备、网络 (net)设备和其它一些类型设备。块设备指的是那些只能以块为单位进行读或写数据的设备。典型的块大小为 512, 1024 或 4096 字节。块设备通过缓存来随机地进行访问。而串行口、并行口、触摸屏等都是字符设备，它们是直接存取而不通过缓存。

③. fs 目录：操作系统的一个目的就是向用户隐藏硬件设备的细节。Linux 中的虚拟文件系统 (VFS)就是向用户提供同一种格式的文件系统而不管硬件是否相同。文件系统一般位于块设备中。

④. include 目录：C 语言头文件就包含在这个目录中。它们进一步分为与硬件有关的文件和通用文件。子目录 “asm-$(ARCH)”下存放与硬件有关的文件，这些文件又分别存放在子目录 “arch-$(MACHINE)”、“pro-$(PROCESSOR)”下。其中“arch-$(MACHINE)”专门用来存放硬件专用头文件，“pro-$(PROCESSOR)”用来存放进程专用头文件。

⑤. init 目录：Linux 入口函数“strat-kernel()”在 “init/main.c”中定义。

⑥.ipc 目录：进程间的相互通讯，以及进程与内核之间的通讯，以便协调它们的运行。Linux 支持信号量、线程、系统 VIPC 等进程通讯机制。

⑦. kernel 目录：内核文件所在的目录。Linux 是多任务操作系统，在同一时间会有多个进程等待响应，调度程序会找出最需要执行的一个进程进行执行。这个子目录包含着调度程序的源代码。

⑧. mmnommu 目录：操作系统的最重要特征之一就是它支持虚拟内存的技术，这个技术会使用户感觉可用内存大于实际的物理内存。然而虚拟内存技术需要 MMU 的支持。对于没有 MMU 的处理器，这个子目录会被子目录 “mm”代替。“mm” 中包含简单内存页面管理源代码。

⑨. net 目录：网络几乎就是 Linux 的同义词。网络协议就包含在这个子目录中。

⑩. lib 目录：该子目录包含 Linux 自身的源代码，不需要标准的 C 程序库。

⑪. scripts 目录：该目录包含配置和编译支持脚本文件。

4.3 建立 uClinux 的开发环境

4.3.1 建立交叉编译环境

基于 uClinux 操作系统的应用开发环境一般是由目标开发板(S3C44B0X 开发板)和宿主机(PC)所构成。目标硬件开发板用于运行操作系统和系统应用软件，而目标板所用到的操作系统的内核编译、应用程序的开发和调试则需要通过宿主机来完成。双方之间一般通过串口，并口或以太网接口建立连接调试关系。如下图 4-5 所示。

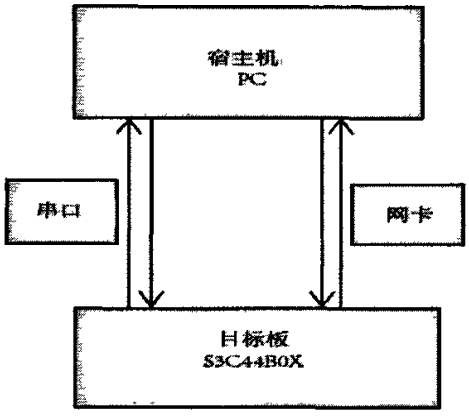


图 4-5 开发环境

一般我们在宿主机上建立的开发和调试环境有两种途径：一种是在虚拟机（如：Vmware,Cygwin）上安装 Linux 并建立，另一种是直接在宿主 Linux 环境下建立。

本文采用的是在宿主机上安装 Linux ubuntu 发行版建立的。

第一步，我们需要得到交叉编译工具链(Cross-Platform Development Toolchain)，我们得到交叉编译工具的途径有两种：一种是直接到下面的地址下载:http://www.uClinux.org/pub/uClinux/m68k-elf-tools/arm-elf-tools-20030314.sh（最新的打包的交叉编译工具链）。如果要编译 2.6 的内核则需要下面的地址下载：http://opensrc.sec.samsung.com/download/arm-elf-tools-20040427.sh(交叉编译工具链包)关于 2.6 内核的移植可以参见 Hyok S. Choi 的《Getting Familiar with uClinux/ARM 2.6》）。

接下来就需要解压工具链包。

sudo sh arm-elf-tools*.sh

再接着就是测试交叉编译环境是否正常工作，在这里我们用一个简单的 test.c 程序进行测试。

```
#include <stdio.h>
int main(void)
{
    printf("Arm-elf-gcc is installed successfully!\n");
    while(1);
}
```

编译程序：

#arm-elf-gcc test.c –o test

#file test

如果能看见 ELF 格式的文件信息说明交叉编译环境成功建立。

4.3.2 建立调试环境

我们在进行应用程序和内核调试时一般采用下面三种连接方式：a.串口 b.并口 c.以太网接口。下面我们来介绍在 Linux 下面建立 a,c 连接的方法。因为本文所用的是 ubuntu 操作系统，所以在下面介绍都是基于它的。

串口连接的方式有 3 种。

1．cu 命令，a.安装 uucp 包，可以从"新立得包安装管理器"里得到；b.配置文件(这里以*/dev/ttyS0*, 115200 bps, 8N1 为例)见下面：

- */etc/uucp/sys*:

```
#
# /dev/ttyS0 at 115200 bps:
#
        system          S0@115200
        port            serial0_115200
        time            any
```

/etc/uucp/port:

```
#
# /dev/ttyS0 at 115200 bps:
#
        port            serial0_115200
        type            direct
        device          /dev/ttyS0
        speed           115200
        hardflow        false
```

c.连接串口。如下面，退出用~.即可。

```
$ cu S0@115200
Connected.
```

2. 使用 kermit 命令，先到 http://www.columbia.edu/kermit/ 下载并安装 gkermit 包。kermit 是通过在用户主目录下的*.kermrc* 文件的执行来连接的，所以我们需要对其进行配置，如下所示，并可以通过$ kermit -c 连接。

- *~/.kermrc*:

```
set line /dev/ttyS0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
robust
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
set window 5
```

3. 使用 minicom，在这里 u-boot 的开发文档上说这个程序在 u-boot 与 Linux 系统连接传输 image 文件时有错误，所以本文没有采用这种方式。

4. 设置 tftp 服务器（一般我们传输大的文件或者 uClinux 映像文件）。我们通过以下步骤来完成。

1) 安装 tftp 程序，在这里 *usr/sbin/in.tftpd* 可以看到。

2) 启动 tftp 服务，编辑*/etc/xinetd.d/tftp* 文件，如下：

```
# default: off
# description: The tftp server serves files using the trivial file #transferprotocol.  The
tftp protocol is often used to boot diskless
#workstations download configuration files to network-aware printers,
#and to start the installation process for some operating systems.
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot
#       disable                 = yes
```

```
        per_source                  = 11
        cps                         = 100 2
}
```

3) 检查/tftpboot 文件夹是否存在，并将其权限设置为可读写。

至此，我们的串口连接和 tftp 服务已经建立。至于还有 u-boot 还支持的更多网络传输协议可以参见 u-boot 的文挡《The DENX U-boot and Linux Guide (DULG) for TQM8xxL》。

另言：因主要考虑的课题具体实施，所以对于建立仿真环境 GDB-ARMulater 或 SkyEye 没有详述。如果有需要可以参考下面的网站提供的信息：

GDB-ARMulater:http://www.uClinux.org/pub/uClinux/utilities/armulator/

SkyEye: http://www.skyeye.org

第五章 S3C44B0X 开发板设计

5.1 S3C44B0X 开发板框图

本文设计的 S3C44B0X 开发板主要由：

CPU(S3C44B0X)，SDRAM，Flash，RTL8019，RS232，GPIO，USB 和系统总线等几个部分构成，见图 5-1 所示。

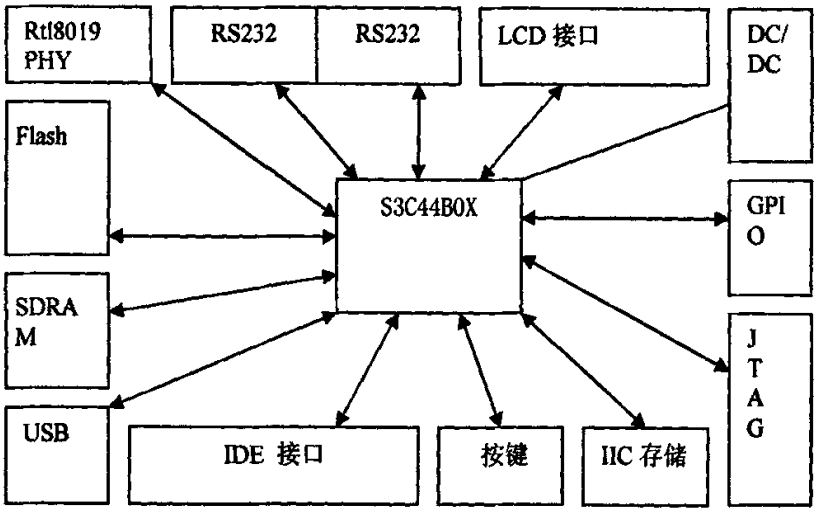


图 5-1 S3C44B0X 开发板框图

5.2 存储接口设计

ROM 接口：S3C44B0X 复位的时候访问的地址是 0x00000000 并执行存放在此处代码作一些系统变量的配置，所以我们必须将存放启动代码的 ROM(Flash)连接在 nGCS0 的位置选择 Bank0(见图 5-2)。而连接在 Bank0 位置的数据总线宽度是通过 OM 配置的(见表 5-1)，采用的是 16 位模式(即半字模式)。

表 5-1 OM[1:0]引脚描述

信号	类型	描述
OM[1:0]	输入	OM[1:0]设置 S3C44B0X 在测试模式和确定 nGCS0 的总线宽度，逻辑电平在复位期间由这些管脚的上拉下拉电阻确定。 00:8-bit 01:16-bit 10:32-bit 11:Test mode

本文采用的是 AMD 公司的 AM29LV160（大小为 2M）作为 BootROM 的，由于 Flash 是 16 位的所以连接如图 5-2 所示，Flash 的地址为 0x00000000～0x00200000。

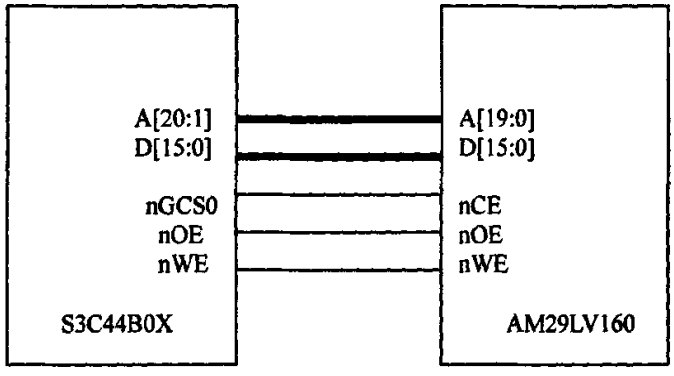


图 5-2 ROM 连接框图

RAM 接口：从第三章图 3-2 中可以看出 S3C44B0X 的内存大小是由 Bank6 和 Bank7 的大小决定的,而且其通过 S/W 和 BSWCON 来选择数据总线的宽度。本文选用的 HY57V64120(8M)片选 nGCS6 选择 Bank6 空间作为内存空间。如图 5.3 所示,内存地址为 0x0c000000～0x0c7FFFFF。

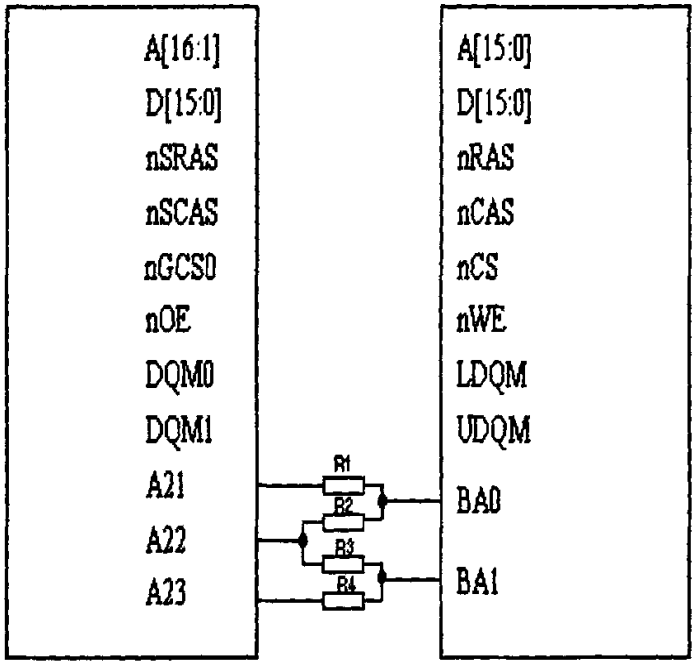


图 5-3 RAM 连接框图

5.3 其他部分设计

网络接口：本文选用的是全双工 RTL8019AS 作为网络控制器具体见附录 I 的 RTL8019 原理图。片选为 nGCS3。

IDE 接口：根据 ATA-3 协议，本文画出了 IDE 接口可作为 IDE 硬盘和 CF 卡外扩。原理图可见附录 I 的 IDE 部分，片选为 nGCS2。

以上接口主要是为后期产品功能扩充提供硬件支持。

由于篇幅有限，其他部分电路设计见附录 I。

第六章 信号处理电路的设计

6.1 信号处理电路设计概述

在自动化系统中，各种规格仪表的输入信号类别大多包含在下表 6-1 中。由表 6-1 可归纳出如表 6-2 所示的从传感器及变送器传送到测控仪表的输入信号类型。传统的自动化仪表产品设计对不同输入信号采用不同的测量电路，形成规格众多的产品序列，给生产厂家的设计、生产、售后服务、用户的设备选型、系统的调试维护带来诸多不便。本文介绍的基于 ARM 的嵌入式智能仪表系统与传统单一信号采集系统区别在于通过尽可能简单的硬件电路，尽可能少的接线端子，完成各种常用自动化信号的柔性测量工作，用户通过友好的人机交互界面设定仪表的工作状态让该智能仪表工作在某一种仪表状态，如智能流量积算控制仪、智能多路巡检显示控制仪等等。而要达到如此高度统一的硬件平台依赖于良好的多功能前端信号处理电路。

表 6-1 输入信号类别

Sn	输入类型	测量范围	Sn	输入类型	测量范围	Sn	输入类型	测量范围
01	0.2 ～ 1V	-1999 ～ 9999 字	11	0 ～ 100Ω	-1999 ～ 9999 字	21	B	0～1800℃
02	0～5V	-1999 ～ 9999 字	12	30 ～ 350Ω	-1999 ～ 9999 字	22	R	0～1700℃
03	1～5V	-1999 ～ 9999 字	13	0 ～ 20mV	-1999 ～ 9999 字	23	T	-200～350℃
04	0.2 ～ 1V 开方	-1999 ～ 9999 字	14	0 ～ 60mV	-1999 ～ 9999 字	24	J	0～1000℃
05	0～5V 开方	-1999 ～ 9999 字	15	0 ～ 100mV	-1999 ～ 9999 字	25	WRE	0～2300℃
06	1～5V 开方	-1999 ～ 9999 字	16	0 ～ 10mA	-1999 ～ 9999 字	26	Cu50	-50～150℃
07	0 ～ 20mA	-1999 ～ 9999 字	17	0 ～ 10mA 开方	-1999 ～ 9999 字	27	Cu100	-50～150℃
08	4 ～ 20mA	-1999 ～ 9999 字	18	K	-50 ～ 1300℃	28	Pt100	-200～650℃
09	0 ～ 20Ma 开方	-1999 ～ 9999 字	19	S	-50 ～ 1700℃	29	BA_1	-50～500℃

10	4 ～ 20mA 开方	-1999 ～ 9999 字	20	E	0 ～ 1000℃	30	BA_2	-50～500℃

6-2 输入信号类型

1、标准热电偶（B、S、K、E、R、T、J、WRE 等）
2、标准热电阻（Pt100、Cu50、Cu100、BA_1、BA_2 等）
3、电流（0~10mA、4~20mA、0~20mA。输入电阻≤250Ω）
4、电压（0.2~1V、0~5V、1~5V、mV。输入电阻≥250kΩ）
5、电阻

注："Sn"为输入信号类别

多功能信号处理电路主要是由前端通道选择电路和信号调理电路两部分组成。前端通道选择电路主要由两个 4052 和一个 4051 组成，该电路依据仪表工作状态打开相应的通道来进行数据采集。信号调理电路由选通、程控放大等部分电路组成，该电路仅通过五个端子（EXT1、EXT2、EXT3、EXT4、EXT5）就可以测量各种常用信号，其中 EXT1 为模拟地。实际使用时用户根据输入信号类型设定输入选择参数，并从以下介绍的输入端输入被测信号。电路框图见下图 6-1。

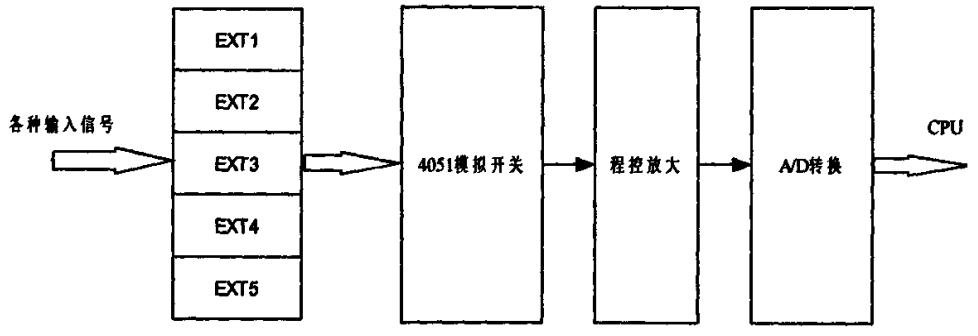


图 6-1 信号调理电路框图

当采集电阻信号，从 EXT1、EXT2 与 EXT3 输入端输入；当采集热电偶及 mV 信号，从 EXT1 与 EXT2 端输入；当采集大电压信号，从 EXT1 与 EXT4 端输入。当采集标准电流信号，从 EXT1 与 EXT5 端输入，先通过 EXT5 端对地 50Ω

采样电阻把电流信号转化为电压信号，再将 EXT4、EXT5 端短路，转化的电压信号最终从 EXT4 端输入。具体电路见下图 6-2 所示。

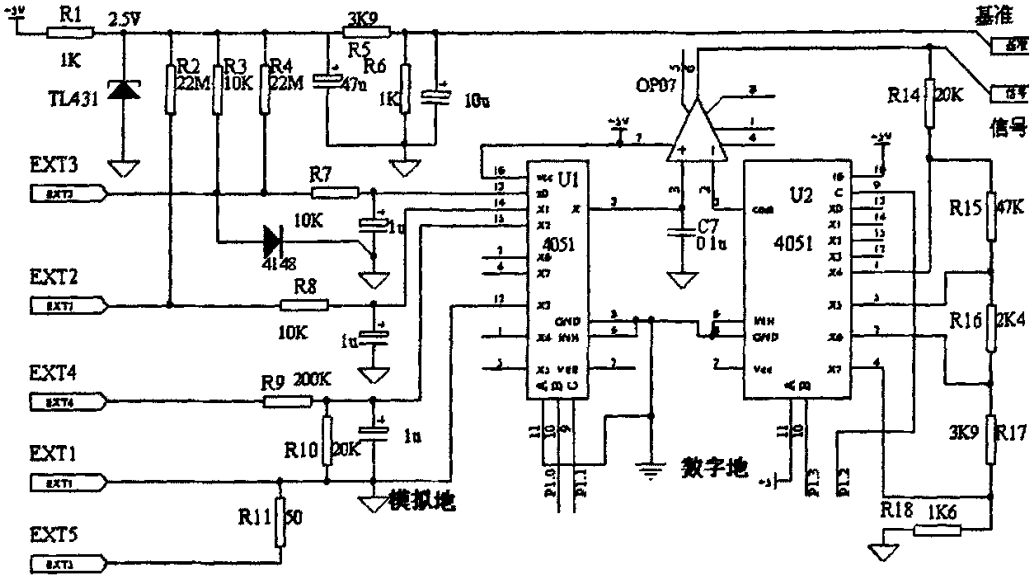


图 6-2 信号调理电路具体电路

6.2 信号调理电路工作原理分析

6.2.1 输入信号的选通

信号选通采用 U1(CD4051)作模拟信号选择，4051 是单 8 通道数字控制模拟开关，有三个二进制通道选择输入端.A0、A1、A2 和片选 INH 输入。电路只用了四路选通，A 端接低电平，B、C 端由 ARM 芯片的通用输出 I/O 口控制。ARM 芯片通用输出口输出不同控制选通信号(00、01、10、11)分别选择 X0、X1、X2 及 X3 不同的选通端口电压信号送到程控放大 OP07 的正向输入端。它们分别对应信号输入端的 EXT3、EXT2、EXT4 及 EXT1。电路中 TL431 产生 2.5V 基准电压，R2、R4 两个电阻完成热电阻热电偶测量的断线检测。

6.2.2 程控放大

该部分电路由一个 4051(U2)和放大器 OP07 及处围电阻组成，多路模拟开关 4051 用来改变放大器的增益；根据输入信号的大小，由 ARM 控制 4051 的选通，改变其反馈电阻的大小，从而达到改变放大器增益的目的,实现量程的自动切换。各种信号由采样端采集后，首先通过 RC 滤波并经过 U1(4051)通道选择，进入程控

放大，程控放大采用四档。输入信号类型不同，则转化的电压信号也不同,采用不同的放大倍数,使这些放大后的信号最大值接近 A/D 的最大充许值。以充分利用 A/D 资源，保证测量精度。各种信号的放大倍数的确定和后面 A/D 器件的模拟输入有关，本电路采用的 A/D 芯片为 ICL7135(四位半)，该芯片基准电压为 0.5V，因此 ICL7135 模拟输入电压范围为 0～1V。

程控放大等效电路见图 6-3。

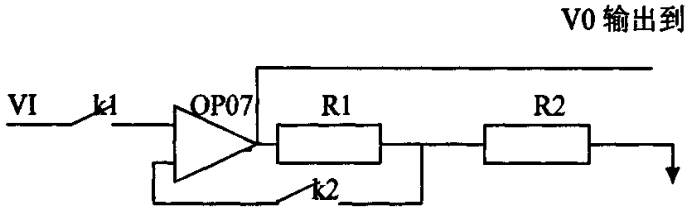


图 6-3 程控放大等效电路

图 6-3 中 k1 和 k2 表示两个 4051，由于运放输入阻抗很高，模拟开关 K1、K2 的导通电阻对放大没有影响。K1 选择不同输入通道（衰减或不衰减通道），k2 选择放大倍数 A=1+R1/R2。当 k2 采用不同的选通端口，R1 和 R2 都会变化,在程控放大电路中， R14、R15、R16、R17、R18 五个串联电阻由模拟开关 k2 分成 R1 和 R2 两部分，如在电路中通过 k2 选通 X4，则：R1=R14=20K,R2=R14+R15+R16+R17+R18=54.9K，放大倍数 A1=1+20/54.9≈1.3。同样道理 K2 选通 X5、X6、X7 时可以分别求得对应的放大倍数分别为 A2=1+67/7.9≈10、A3=1+69.4/5.4≈14 、A4=1+73.3/1.6≈47。

所有输入信号转化电压信号经模拟开关到达程控放大器的正向输入端，其最大值(小于 0.7V)经放大后应小于 1V,各放大倍数可适应的输入信号范围为：1、放大 1.3 倍的信号有：输入信号最大值 100mV 到 0.7V 的电压信号；2、放大 10 倍的信号有：输入信号最大值 60mV 到 100mvV 的电压信号；3、放大 14 倍的信号有：输入信号最大值 20mV 到 60mV 的电压信号。4、放大 47 倍的信号有：输入信号最大值≤20mV 的电压信号。对于不同的输入类型信号，转换为电压信号后，就可以按上述的放大倍数进行处理。实际应用本电路时，可以根据 A/D 模拟输入的要求调整 R14~R18 的阻值。

6.3 信号测量原理分析

为满足多信号输入的要求，软件设计必须能根据用户通过参数设定选择的输入信号类型自动调整放大倍数及数据处理方法。对各种输入信号处理方法如下：

6.3.1 电阻信号的测量

电阻信号的测量等效电路如下所示，Rt 为待测电阻，一般小于 500Ω。电阻输入采用三线制，从 EXT1、EXT2 与 EXT3 输入端输入。由于实际应用中连接特定某个电阻式传感器到测量仪表的三根连线粗细相同，长度一样，所以引线电阻 r 相同，Rt 表示待测电阻，其等效电路见图 6-4。2.5V 电压经过 10K 电阻及两个引线电阻 r 和被测电阻 Rt 到地，产生 EXT2 和 EXT3 电压信号，经 RC 滤波后分时通过模拟开关 U1 送入 OP07 运放，由于运放输入阻抗很高，EXT2 和 EXT3 端不会产生分流。仪表通过采集 EXT2 和 EXT3 端的信号，通过处理，排除 r 对测量的影响，得出被测电阻 Rt 的阻值。

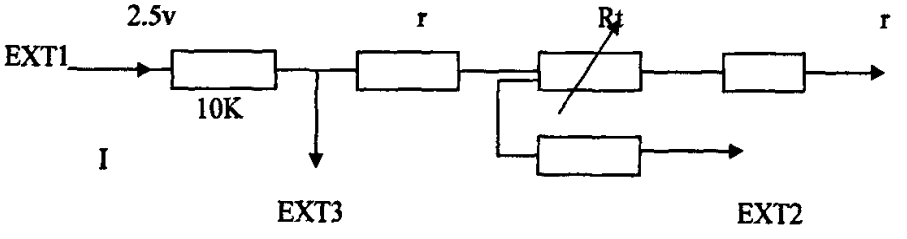


图 6-4 电阻测量等效电路

在测量电阻时，由于 10k 远远大于 Rt+2r,我们近似认为 I 为恒流源 I≈2.5V/10K，则可以采用了以下的计算处理过程计算出被测电阻 Rt 阻值。

V_{EXT3}=I（2r+Rt） 对应 A/D 采集值为 D1。

V_{EXT2}=I（r+Rt） 对应 A/D 采集值为 D2 。

△D=2D2-D1 实际上就是在恒流源 I 作用下被测电阻 Rt 两端的电压，对应输入电路中：

I*Rt =2 I（r+Rt）—I（2r+Rt）=2V_{EXT2}-V_{EXT3}

可以看出，在最后的结果中，没有出现 r，这样就消除了导线电阻的影响，即在假设恒流源不变的情况下，通过对 EXT2、EXT3 的测量计算可以求出待测电阻 Rt。同时不同热电阻输入时，转化的电压值不同，故应选择不同的放大倍数，如

Cu50 的放大倍数为 47，Pt100、Cu100 选取放大倍数为 10。对 30Ω 至 360Ω 电位器式位置传感器信号选取放大倍数为 1.3。但实际上，测量回路电流并非真正的恒流源，而是由稳压管 TL431 的 2.5v 电压和 10K 电阻热 Rt、2r 构成，此恒流源的内阻应该是≈10K+Rt，而不是无穷大。随着被测电阻的变化，电流 I 也会发生一定程度的变化。当把 I 作恒流源来处理时 I1=V/10K。 但在实际的计算中，因为 Rt 对回路电流影响不能忽略，故 I 应采用公式 I2=V/（10K+Rt+2r）计算。由于 r 一般小于 10Ω 忽略不计，由此可得电流的误差：

$\Delta I=I1-I2=V/10k-V/(Rt+2r+10K)$　　$\Delta I/I1\approx(Rt/10k)$

当 R=400Ω 时，恒流源误差可达 4%。由此产生的误差是本系统不能容忍的。然而通过硬件手段来解决此问题比较麻烦，但可以采用软件算法来进行校正消除随着被测电阻上升使电流下降带来的误差。

根据公式 I2=V/（10K+Rt+2r），可以得出被测电阻和对应的电压是一条曲线。可以通过曲线转化为折线段来进行非线性修正处理，如图 6-5 示。这样简单而且能够满足精度，精度的高低直接与分的段数的多少直接相关。

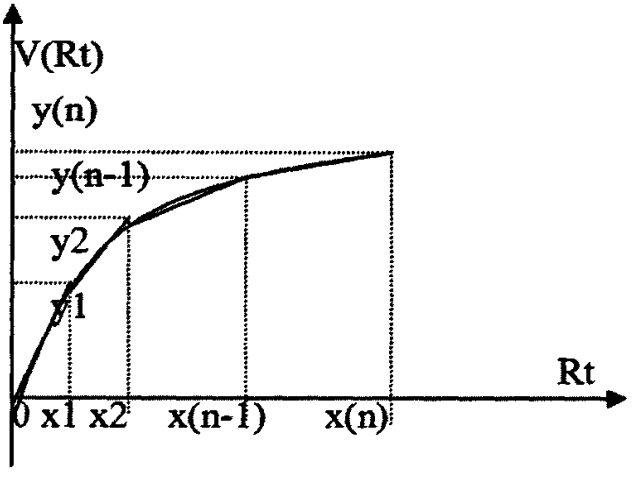


图 6-5 非线性修正处理示意图

每条曲线段方程为 Y=(x - x(n-1)) * (y(n) - y(n-1))/(x(n) - x(n-1))+y(n-1),我们把电阻测量范围分为 0～100Ω 和 0～400Ω 两种(主要考虑 CU50 和 PT100 的测量)，可以分别为 10 段和 20 段进行处理，测量的精度可完全满足要求。同时如果待测电阻为热电阻信号，还要根据分度号表进行二次非线性修正把电阻测量值转化为温度值。同时测量电路中 10K 电阻的误差通过仪表出厂调校时标准电阻信号标定加以解决。由于电阻测量时被测电阻小于 500 欧，EXT3 端电压小于 0.2V，故二极管 1N4148 没有作用。

6.3.2 热电偶的测量

当采集热电偶信号时，mV 信号从 EXT1(-)与 EXT2(+)端输入，EXT3 作为冷端补偿输入；当信号采集端采集到信号后，经过通道选择，信号进入程控放大电路，信号分度号不同则 mV 值的高低也不同，通过软件选择不同的放大倍数，设放大倍数为 A，则从程控放大出来的信号即为 AX，对于 B、S 标准热电偶，放大倍数应选择 47。对于 WRE、K、E、J 标准热电偶，放大倍数应选择 14。而由于热电偶本身的原因，其 mV 温度间关系是非线性的，在软件设计中可以根据分度号表进行分段折线非线性校正算出 CX，加上通过测量冷端补偿二极管电压得出的冷端补偿温度 C0，就得到该路的实际测量温度 C，即 C=CX+C0。

在实际应用中，在接线端(冷端) 的温度是变化的，由于热电偶补偿导线和接线端材质不同，热电偶输入 mV 值会被冷端抵消掉一部分。这种情况造成的误差不能忽略。必须对它进行 0-50℃范围的冷端补偿。因为二极管在温度变化时，其正向导通电压变化稳定(-2mV/℃) ，因此可以根据 EXT3 端 1N4148 二极管正向压降变化量的大小推算出冷端温度变化的大小，EXT3 端电压为 0.65V 左右，放大倍数选择为 1.3 倍。如果冷补问题通过现场冷端补偿器解决，也可以选择不需要冷补。采用二极管冷端补偿的具体做法如下：

第一步：在冷端补偿输入端 EXT3 输入一标准电压 0.7V 得到一个 AD 采样值 D0，然后我们再输入一标准电压 0.6V，得到一个 AD 采样值 D1。两者相减得到一个值 ΔD，根据二极管的特性，每 1℃电压变化 2mV，我们输入的第 1 个标准信号和第 2 个标准信号相差为 100mV，相当于实际二极管变化电压 100mV，冷端温度变化 50℃，就可以求出温度每变化 1℃时其 AD 值变化多少的系数 K=ΔD/50，由于冷端温度变化范围小(0-50℃)，相对精度要求不高，因此设计产品时把该系数直接固化于程序中。

第二步：把冷补二极管 1N4148 接入 EXT3 后，我们在仪表设置状态输入当前环境温度 Ta，设计的软件马上自动测出二极管 1N4148 所在 EXT3 端电压经 1.3 倍放大及 AD 转换后的值 Da，并将 Ta、Da 值存储到 EEPROM 里面，以后仪表处于工作状态时实时地测出二极管经 1.3 倍放大及 AD 转换后的值 Db，再把两者相减得 ΔDab= Da- Db，ΔDab 除以 K(代表每一个 1℃的 AD 采样值的大小)得到一个温度值差 Y。Y 加上设置环境温度初值 Ta 得到实际环境温度 C0=Y+Ta。当环境温度变化时，所测的实际冷端温度 C0 将会跟随变化，在一定时期内环境温度的变化不大,因此它存在的误差和热电偶高温测量相比十分的小,可以忽略。但当环境变化较大时,比如从冬天到夏天几十℃的变化,冷补误差大于 1 度时 ,我们可以重新进入设

定状态输入基准 Ta 校正。

6.3.3 电压信号的测量

当被测电压信号小于 0.7 V 时，处理方法与热电偶相似，只不过不需要冷端补偿及非线性校正。当被测电压信号大于 0.7V 时(0-5V、0.2-1V、1-5V 等信号类型)，信号从 EXT1(-)与 EXT4(+)端输入，并经 R9、R10 衰减 10 倍后输入。电压输入时也需要根据输入信号的大小选择合适的放大倍数。

6.3.4 电流信号的测量

当采集标准电流信号时，信号从 EXT1(-)与 EXT5(+)端输入，EXT4、EXT5 端短路，通过 EXT5 端对地 R11(50Ω)采样电阻把电流信号转化为电压信号从 EXT4 端输入，之后的处理方法和大于 0.7V 电压时相似。

6.4 零点满度自校正

本电路设计适用于多种输入信号类型，在用一般器件的情况下保证系统的测量精度非常重要，设计的基本思路就是在仪表的设定参数状态通过标准信号源输入一系列精确的标准信号 INm，并根据标准信号的大小选择合适的放大倍数，最后把各标准信号通过前端电路及数据处理后得出的值 ADm 存储于 E^2PROM 中；仪表实际测量工作时，通过测量获得输入信号 INx 及输入端模拟地(EXT1) 经同一放大倍数通道下 A/D 转换值 ADx 和 AD_0，结合同一放大倍数通道下同种类型信号的 ADm 值实现零点满度的自校正。假设某一时刻信号采集值为 ADx，则 $(ADx-AD_0)/(ADm-D0)=INx/INm$，其中 AD_0 为当前时刻零点实际 A/D 转换值，ADm 是仪表出厂校定时通过标定功能菜单的操作,存放于 EEPROM 中的调试参数，INm 值可以约定固化于程序中。则可求出 $INx=INm*(ADx-AD_0)/(ADm-AD_0)$ 。由于 INm、INx、EXT1 三个信号经过同样的硬件输入通道,硬件的离散性误差及零点的不确定对三者的影响相同，通过此公式可以校正这些误差。这里的关键是确定合适的 INm，确定并存储校正模型的参数。在正式测量时，根据测量结果和校正模型参数求取最终测量值，从而消除误差。经计算实验，采用下表 6.3 所列的基准 INm 值可满足合种信号的校正需要。

表 6-3 信号基准

电阻型输入信号基准	80Ω	200Ω	300Ω
小电压型输入信号基准	20mV	60mV	100mV
大电压型输入信号基准	0.5V	2.5V	5V
电流型输入信号基准	10mA	15mA	20mA

在软件的配合下整个工作过程是这样的，首先在参数设定状态下，依次进入各标定参数菜单，在对应输入端子上输入上表中各标准信号，软件自动选择相应的放大倍数，采集该标准信号的 A/D 转换值，并确认为 ADm 值存放于串行 E^2PROM 中，对应十二种标准信号生成十二个 ADm 数据。仪表使用时，用户通过编程菜单选择输入信号类型；当测量工作开始后，软件把适合于用户使用信号类型相应的 ADm 值从 E^2PROM 中取出，并选择相应的放大倍数实时采集 ADx 值及 AD_0 值，通过上述零点满度自校正公式算出 INx 值(其中确定的 INm 值固化于程序中)，最后经过线性化修正或工程量标度变换得出最终测量工程量。

在校正过程中 AD_0 是实时采集的，零漂可以得到及时修正，而增益的调校是以仪表标定时的各 ADm 值为准，所以电路中应选用温漂小的器件(主要是电阻)，反而对器件的精度要求不高。

第七章 基于44B0X开发板的u-boot和uClinux移植

7.1 u-boot移植

7.1.1 u-boot简介

u-boot是由德国的工程师Wolfgang Denk从8XXROM代码发展而来的，它支持很多处理器，比如PowerPC、ARM、MIPS和x86。目前，u-boot源代码在sourceforge网站的社区服务器中，Internet上有一群自由开发人员对其进行维护和开发，它的项目主页是 http://sourceforge.net/projects/u-boot。u-boot 的最新版本源代码可以在Sourceforge的CVS服务器中匿名获得。

#cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/u-boot login

#cvs -z6 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/u-boot \ co -P u-boot

源码目录结构

◆ board：和一些已有开发板有关的文件，比如Makefile和u-boot.lds等都和具体开发板的硬件和地址分配有关。

◆ common：与体系结构无关的文件，实现各种命令的C文件。

◆ cpu：CPU相关文件，其中的子目录都是以U-BOOT所支持的CPU为名，比如有子目录arm926ejs、mips、mpc8260和nios等，每个特定的子目录中都包括cpu.c和interrupt.c，start.S。其中cpu.c初始化CPU、设置指令Cache和数据Cache等；interrupt.c设置系统的各种中断和异常，比如快速中断、开关中断、时钟中断、软件中断、预取中止和未定义指令等；start.S是U-BOOT启动时执行的第一个文件，它主要是设置系统堆栈和工作方式，为进入C程序奠定基础。

◆ disk：disk驱动的分区处理代码。

◆ doc：文档。

◆ drivers：通用设备驱动程序，比如各种网卡、支持CFI的Flash、串口和USB总线等。

◆ fs:支持文件系统的文件，U-BOOT现在支持cramfs、fat、fdos、jffs2和registerfs。

◆ include：头文件，还有对各种硬件平台支持的汇编文件，系统的配置文件和对文件系统支持的文件。

◆ net：与网络有关的代码，BOOTP协议、TFTP协议、RARP协议和NFS文

件系统的实现。

◆ lib_arm：与 ARM 体系结构相关的代码。

◆ tools：创建 S-Record 格式文件 和 U-BOOT images 的工具。

7.1.2 u-boot 移植过程

因为 u-boot 已经支持 S3C44B0X 处理器，所以我们的移植主要是针对开发板的板级的支持。本文是在 u-boot 已经支持的 B2 板为模板基础上进行了移植，具体步骤如下。

首先，我们需要增加对开发板的支持，做如下配置。

a. 在 board 文件夹中新建开发板目录:

```
#sudo cd board
#sudo cp -r dave cuit
```

b. 拷贝 b2 开发板的配置:

```
#sudo cd cuit
#sudo mv B2 cuit01
#sudo cd cuit01
#sudo mv B2.c cuit01.c
#sudo cd include/configs
#sudo cp B2.h cuit01.h
```

c. 修改 Makefile,cuit01.c,flash.c,cuit01.h 将其中的 B2 替换为 cuit01，并在 Makefile 中增加 cuit01_config 编译项。

```
cuit01_config : unconfig
@./mkconfig $(@:_config=) arm s3c44b0 cuit01 cuit
```

其次，需要对开发板的相关参数进行配置。在配置参数前对了解 u-boot 的启动流程是相当必要的。u-boot 的运行流程图，如下图 7.1 所示。

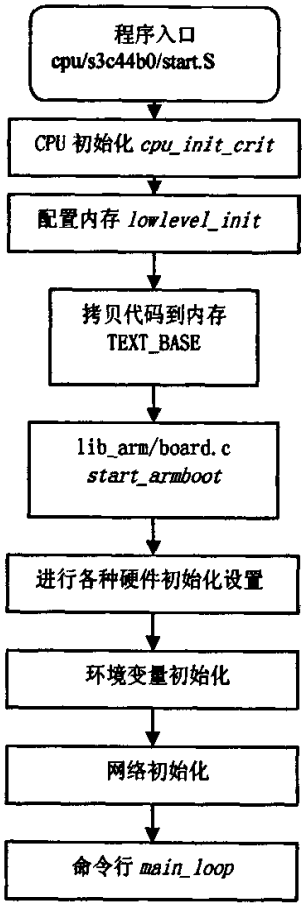


图 7-1 u-boot 的运行流程图

上图中的硬件相关的初始化设置可见表 7-1。

表 7-1 相关的硬件初始化设置

初始化设置函数	相关功能	所在文件
cpu_init	CPU 相关的设置	/cpu/s3c44b0/cpu.c
Board_init	板子相关的设置	/board/cuit/cuit01/cuit01.c
interrupt_init	中断设置	/cpu/s3c44b0/interrrupts.c
env_init	初始化环境变量的设置	如果是 Flash 存储，则在 common/env_flash.c
Serial_init	串口初始化设置	/cpu/s3c44b0/serial.c
Console_init_f	控制台配置	/common/console.c
Dram_init	内存配置	/board/cuit/cuit01/cuit01.c
Flash_init	Flash 初始化	/drivers/cfi_flash.c

★配置 configs/cuit01.h 相关参数：

a. CPU 主频配置，因为本文选用的外频是 10Mhz，所以 CPU 设置为 60Mhz:

```
#define CONFIG_S3C44B0_CLOCK_SPEED 60
```

b. 配置 Flash 和内存:

```
#define CONFIG_NR_DRAM_BANKS    1    //我们只用了 Bank6 作为内存
#define PHYS_SDRAM_1          0x0c000000 /* SDRAM Bank #1 */
#define PHYS_SDRAM_1_SIZE  0x00800000 /*  内存大小 8MB */
#define PHYS_FLASH_1      0x00000000 /* Flash Bank #1 */
#define PHYS_FLASH_SIZE      0x00200000 /* Flash 大小 2MB */
#define CFG_FLASH_BASE       PHYS_FLASH_1
#define  CFG_LOAD_ADDR       0x0c008000 /*默认载入的内存地址*/
```

c. 设置 Flash 存储环境变量：

```
#define CFG_ENV_IS_IN_FLASH       1
#define CFG_ENV_ADDR    (PHYS_FLASH_1+0x40000)//参数存储地址
#define CFG_ENV_OFFSET 0x40000
```

d. 设置网卡控制器驱动：

```
#define CONFIG_DRIVER_RTL8019
#define RTL8019_BASE       0x06000300 //基址
#define RTL8019_BUS32      0
#define CONFIG_SMC_USE_16_BIT
```

e. 配置串口和网络参数：

```
//串口
#define CONFIG_SERIAL1          1
#define CONFIG_BAUDRATE          115200   //波特率
//网络
#define CONFIG_NETMASK   255.255.255.0
#define CONFIG_IPADDR     192.168.0.30
#define CONFIG_SERVERIP  192.168.0.10
#define CONFIG_BOOTFILE "uClinux_rom.bin"//内核文件名
#define CONFIG_BOOTCOMMAND  "bootm 0x50000"
/*0x50000 为内核在 Flash 中地址*/
```

f. 配置提示符:

```
#define    CFG_PROMPT  "CUIT=>"    /* Monitor Command Prompt */
```

★配置 cpu/s3c44b0/serial.c 相关参数,将 *serial_setbrg* 中分频纠正:

```
    case 1200:
#if CONFIG_S3C44B0_CLOCK_SPEED==60
        divisor =   3124;
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif
        break;
```

```
    case 9600:
#if CONFIG_S3C44B0_CLOCK_SPEED==60
        divisor = 390;
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif
        break;
    case 19200:
#if CONFIG_S3C44B0_CLOCK_SPEED==60
        divisor = 194;
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif
        break;
    case 38400:
#if CONFIG_S3C44B0_CLOCK_SPEED==60
        divisor = 97;
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif      break;
    case 57600:
#if CONFIG_S3C44B0_CLOCK_SPEED==60
        divisor = 64;
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif      break;
    case 115200:
#if CONFIG_S3C44B0_CLOCK_SPEED==60
        divisor = 32;
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif
```

★ 修改 cpu/s3c44b0/start.S 中的 *cpu_init_crit* 中频率相关的代码

(ldr　r0, =0xac042//75Mhz)为:

```
#if CONFIG_S3C44B0_CLOCK_SPEED==60
    ldr  r0, =0x88042 /* 60MHz */
#else
# error CONFIG_S3C44B0_CLOCK_SPEED undefined
#endif
```

★ 修改/board/cuit/cuit01/cuit01.c 代码,根据开发板硬件对端口进行需要的初始化:

```
//配置复位为高
PCONF = 0x9256A;
PDATF   = 0xff;
 //配置端口 G
```

```
    PUPG = 0xf;
    PCONG = 0xFFFF; //配制 EINT0 为 linux 内核做准备
    INTMSK = 0x03fffeff;
    INTCON = 0x05;
   //为网卡控制器中断置高,我们的开发板上为 EINT3 与 RTL8019AS 中断相
连
    temp = EXTINT;
    temp &= ~(0x7<<4);
    temp |= (0x4<<4);
```

★修改/board/cuit/cuit01/config.mk,将其中的 TEXT_BASE 改为 0xc700000

★修改/board/cuit/cuit01/memsetup.s, 这是 u-boot 移植中最重要的一个环节，这里设置了 bootloader 中各个 bank 区的大小，总线宽度等等十分重要的参数，修改的部分源代码如附录 I

接下来，我们需要在交叉编译环境下编译 u-boot。在 ubuntu 下：

```
#cd u-boot
#make cuit01_config
#make
```

可能会遇到 hello_world.srec 找不到的问题，可以通过修改其 Makefile 的方式注释掉对它的检查。

最后，将 u-boot 烧写入 Flash 并连接宿主机调试成功，如下图 7-2 所示。

```
U-Boot 1.1.1 --Paul.CUIT (Jun  1 2006 - 20:41:39)

U-Boot code: 0C700000 -> 0C7212F0  BSS: -> 0C755C3C
RAM Configuration:
Bank #0: 0c000000  8 MB
Flash:  2 MB
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
CUIT=>printenv
bootargs=devfs=mount root=ramfs console=ttyS0,9600
bootcmd=bootm 0x50000
bootdelay=3
baudrate=115200
ethaddr=00:50:c2:1e:af:fb
ipaddr=192.168.0.30
serverip=192.168.0.10
netmask=255.255.255.0
bootfile="uclinux_ram.bin"
stdin=serial
stdout=serial
stderr=serial

Environment size: 259/65532 bytes
CUIT=>_
```

图 7-2 u-boot 启动回送到串口信息

7.2 uClinux裁减及移植

7.2.2 uClinux裁减及编译

uClinux是一个可配置、可裁减的系统。使用者可以自由修改uClinux以便进行移植。uClinux大量采用了条件编译，通过条件编译，在内核中选择需要的模块，去掉不参与编译的不必要模块，这样就能将内核的大小控制在一定的范围内，以适合嵌入式应用的需求。在影响内核大小的因素中，以下几个方面起着比较重要的作用。

(1)选择的内核版本。一般 2.4 的内核会比 2.6 的内核小，虽然2.6支持的功能更多，但稳定性不如2.4。

(2)是否选择网络支持。uClinux 支持的网络类型有 ARCnet、Ethernet 1OM/100M/1000M、WirelessLAN、Token Ring WAN 等

(3)文件系统的支持类型。uClinux 支持文件系统的类型包括 Ext2、ROMFS、DOS FAT、JFFS、 ISO 9660. /proc 等。

(4)设备驱动的支持 包括块设备驱动、字符设备驱动、USB 设备、ISDN 支持、SCSI 设备支持等。

通过选择以上的这些内核功能支持，根据产品特定的需求和硬件情况做相应的定制.达到实现嵌入式的功能，提高系统性能，确保安全。

从下面的网站下载uClinux的源代码包：

http://www.uClinux.org/pub/uClinux/dist/uClinux-dist-20070130.tar.gz

再通过tar -zxvf uClinux-dist*.tar.gz解压文件包。进入uClinux-dist目录，使用make menuconfig或者make xconfig配置uClinux的内核和用户应用程序的。如下:

```
#sudo make menuconfig
```

进入uClinux配置，选中” Kernel/Library/Defaults Selection->”用空格进入，如图7-3，其中有两个选项，定制内核设置和定制用户选项设置。

图 7-3 uClinux 配置

选中它们，按 ESC 退出，并保存。终端将首先进入内核设置，这样我们就可以选择和取消选项来设定内核所具有的功能。这也是我们裁剪内核的基本方法。同样我们可以对应用程序进行裁剪。(内核选项的意义可以参见附录 II)

这当中的每个选项都对应了一个宏定义，在 make menuconfig 结束后，自动将配置结果保存为.config，而前一次的备份为.config.old 文件。

对于编译 uClinux 2.4 内核 ，不能简单地通过 make 来实现。我们需要有一些特定的步骤才能保证编译的正确。这是因为 uClinux 所需要支持的硬件平台太多了，不能考虑的很周到。

为了编译最后得到的镜像文件，我们需要 linux 的内核以及 romfs 。对于我们的 S3C44B0X 的移植来说，romfs 是被编译到内核里面去的。因此，在编译内核前需要一个 romfs。为了得到 romfs 的 image ，我们又需要编译用户的应用程序。而为了编译用户的应用程序，我们又需要编译 C 运行库，这里我们用的 C 运行库是 uClibc 。

根据上面的分析，我们编译 uClinux 的步骤如下：

■#sudo make dep

这个仅仅是在第一次编译的时候需要，以后就不用了，为的是在编译的时候知道文件之间的依赖关系，在进行了多次得编译后，make 会根据这个依赖关系来确定哪些文件需要重新编译、哪些文件可以跳过)。

■#sudo make lib_only

编译 uClibc 。以后我们编译用户程序的时候需要这个运行库。

■#sudo make user_only

编译用户的应用程序，包括初始化进程 init ，和用户交互的 bash，以及集成了很多程序的 busybox （这样对一个嵌入式系统来说可以减少存放的空间，因为不同的程序共用了一套 C 运行库），还有一些服务，如 boa（一个在嵌入式领域用的很多的 Web 服务器）和 telnetd （telnet 服务器，我们可以通过网络来登录我们的 uClinux 而不一定使用串口）。

■#sudo make romfs

在用户程序编译结束后，因为我们用到的是 romfs（一种轻量的、只读的文件系统）作为 uClinux 的根文件系统，所以首先需要把上一步编译的很多应用程序以 uClinux 所需要的目录格式存放起来。原来的程序是分散在 user 目录下，现在例如可执行文件需要放到 bin 目录、配置文件放在 etc 目录下。这些事就是 make romfs 所做的。它会在 uClinux 的目录下生成一个 romfs 目录并且把 user 目录下的文件、以及 vendors 目录下特定系统所需要的文件（我们的 vendors 目录是 vendors/Samsung/44B0X ）组织起来，以便下面生成 romfs 的单个镜像所用。

■#sudo make image

它的作用有 2 个，一个是生成 romfs 的镜像文件，另一个是生成 Linux 的镜像。因为原来的 Linux 编译出来是 elf 格式的，不能直接用于下载或者编译（不过那个文件也是需要的，因为如果你需要，那个 elf 格式的内核文件里面可以包含调试的信息）。因此在这个时候由于还没有编译过 Linux，因此在执行这一步的时候会报错。但是没有关系，因为我们在这里需要的仅仅是 romfs 的镜像，以便在下面编译 Linxu 内核的时候使用。

■#sudo make

有了 romfs 的镜像我们就可以编译 Linux 了。因为我们的 romfs 是嵌入到 linux 内核中去了，所以在编译 Linux 内核的时候就要一个 romfs.o 文件。这个文件是由上面的 make image 生成的。

之后，就会在 images 目录下找到 3 个文件：uClinux_rom.bin，uClinux_ram.bin.gz，romfs.img。将 uClinux_ram.bin.gz 解压为 uClinux_ram.bin 文件后，可以将其下载到目标板 RAM 中进行程序调试。

7.2.2 uClinux 的运行

在完成编译后，将宿主机的 IP 地址设置和板上的 IP 地址为一样(可以通过在 u-boot 下输入 printenv 查看)，开启 tftp 服务，连接串口。接下来使用#tftpboot 命

令来下载 uClinux_ram.bin 到内存中，下载完毕后可以通过 go 0x0c000800 来执行程序。在这里要注意的是，u-boot 环境变量中的文件名要和映像名一样。可以参见下面的 setenv 命令来设置。

查看，设置和保存环境变量如下：

1）查看环境变量：#printenv

2）设置环境变量：#setenv bootfile "uClinux_ram.bin"

3）保存环境变量：#saveenv

下载，运行和烧写程序如下：

1）下载程序：#tftp 或者 #tftpboot 或者 loadb ADDR（串口下载）

2）运行程序：#go ADDR　如：#go 0x0c008000

3）烧写程序：#cp Flash 地址 内存地址 大小(字节) 如：cp 0x50000 0x0c008000 20000

系统复位后就可以直接在目标板运行了，运行结果如下图 7-4 所示。

图 7-4 uClinux 启动后串口截图

第八章 通用组态软件设计

8.1 软件系统总体设计

该仪表可以工作在两种状态：运行态和设定态。当仪表工作在设定态的时候，研发人员可以为仪表用户定义2种常规仪表状态（如1号仪表流量计功能，其中用了1、3、5通道分别采集如表6.1中的2号、3号、7号信号；2号仪表温度计功能，其中用了2、4、8通道分别采集如表6.1中的26号、28号、30号信号等等）或仪表用户根据实际需要自定义一种仪表状态； 当仪表工作在运行态的时候，仪表按照用户设定的状态（可以是研发人员定义的，也可以是用户自定义的）工作在某一种仪表状态，如流量计或压力计等，这样就可以在同一硬件平台下重组出多种仪表功能。

根据以上分析整个程序被明确的分为运行态程序和设定态程序两个部分。整个主程序流程图如下图8-1示。

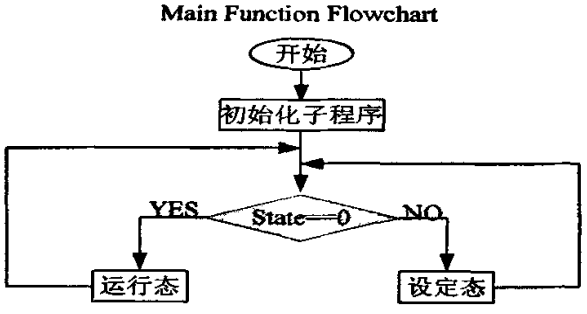


图8-1 主程序流程图

本章将分运行态和设定态两大模块来介绍系统软件设计与实现。

8.2 运行态程序简介

运行态程序主要涉及对某一仪表状态下的八路或其中几路数据的循环采集、存储、光柱显示和对具体某一路数据的波形显示，此外还涉及对采集到的八路或其中几路数据定点打印、时间间隔打印。运行态程序流程图如下图8-2所示。

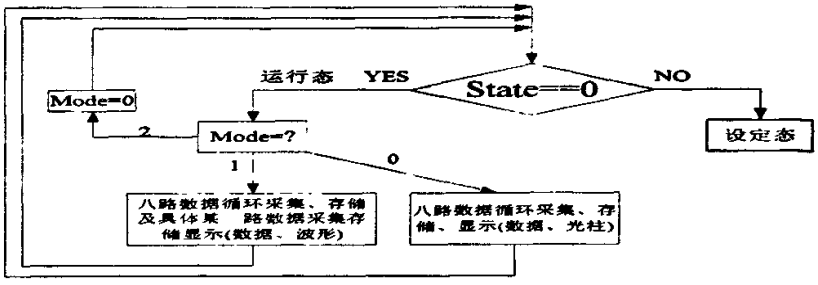


图 8-2 运行态程序流程图

当系统上电复位后先进行初始化，系统默认标志 State=0 进入运行态程序。进入运行态程序后程序根据标志 Mode 的取值分别进入数据循环采集、存储、光柱显示程序和数据循环采集、存储、波形显示程序。下面将分数据循环采集、存储、光柱显示程序和数据循环采集、存储、波形显示程序进一步对运行态程序分析介绍。

(1). 数据循环采集、存储、光柱显示程序流程图如下图 8-3 所示：

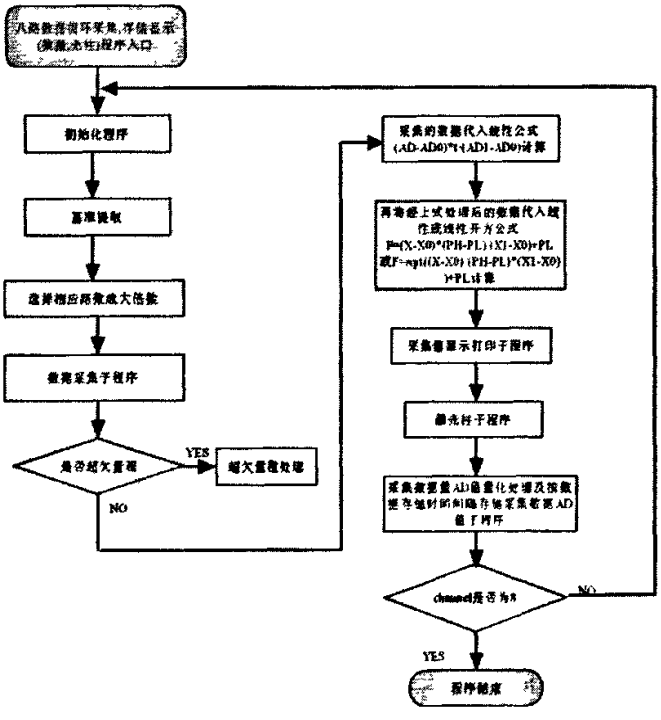


图 8-3 八路数据循环采集、存储、光柱显示程序流程图

进入数据循环采集、存储、光柱显示程序后先运行初始化程序即根据当前的

通道号从相应的 I^2C 存储空间读出测量及便送输出上、下限值，报警上、下限值，报警回差值，零点 AD 值，基准 AD 值，信号类型值，单位编码值等，而后根据相应路数对应的信号类型值选择适当的放大倍数并开始数据采集，再将采集到的数据 AD 值及数据解压后零点 AD 值、基准 AD 值、测量及便送输出上、下限值代入线性公式：x=(AD-AD0)*t / (AD1-AD0)、F=(x-x0)*(PH-PL) / (x1-x0)+PL 或线性开方公式 F=sqrt((x-x0) / (x1-x0))*(PH-PL)+PL，经过该公式计算、处理成所需要的物理数据并送入 M320240-6A2 型液晶显示及 TPuP-A 型微打打印输出，最后将采集到的数据的量化处理 AD 值送入液晶形成光柱图显示，待 8 个通道循检完后退出该程序。

(2). 数据循环采集、存储、波形显示程序流程图如下图 8-4 所示：

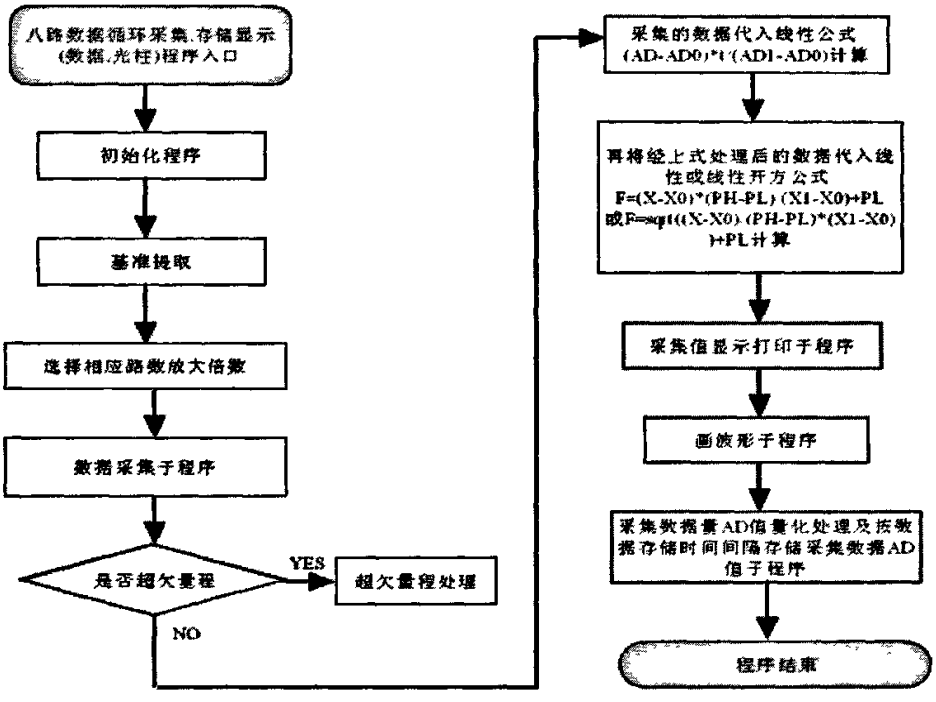


图 8-4 数据循环采集、存储、波形显示程序流程图

该部分程序的设计思路与数据循环采集、存储、光柱显示程序的设计思路基本相同、最主要的区别在于初始化程序中指定 channel 的值即只涉及对具体某一路数据、波形的显示及数据打印输出。而数据循环采集、存储、光柱显示程序则根据系统设定对八路或其中几路进行数据采集、存储、光柱显示及数据打印。

8.3 设定态程序简介

设定态程序主要涉及对系统参数的设定、存储。本系统要求能够对参数设定，而且参数设定值断电后永久保存，所以我们采用了 AT24C64 芯片。整个设定态程序的编写都是以 I^2C 芯片的存贮空间分配为基础，因此在介绍具体程序流程之前有必要将用户要设定的参数与 AT24C64 内部存储空间的对应关系列出，具体对应关系如表 8-1 到表 8-4 所示。

表 8-1 仪表号、通道选择、通道采集信号种类存储地址

存储类别	页内地址
1、仪表号存储地址	00H+10H*(X-1)
2、通道选择存储地址	01H+10H*(X-1)
3、1 号通道信号种类	02H+10H*(X-1)
4、2 号通道信号种类	03H+10H*(X-1)
5、3 号通道信号种类	04H+10H*(X-1)
6、4 号通道信号种类	05H+10H*(X-1)
7、5 号通道信号种类	06H+10H*(X-1)
8、6 号通道信号种类	07H+10H*(X-1)
9、7 号通道信号种类	08H+10H*(X-1)
10、8 号通道信号种类	09H+10H*(X-1)

表 8-2 公共参数存储地址

存储类别	页内地址
1、400 欧电阻信号基准设定	00H---02H
2、100 欧电阻信号基准设定	03H---05H
3、5V 基准设定	06H---08H
4、1V 基准设定	09H---0BH
5、100mV 基准设定	0CH---0EH
6、60mV 基准设定	10H---12H

7、20mV 基准设定	13H---15H
8、20mA 基准设定	16H---18H
9、10mA 基准设定	19H---1BH
10、1.3 倍零点设定	1CH---1EH
11、10 倍零点设定	1FH---21H
12、14 倍零点设定	22H---24H
13、47 倍零点设定	25H---27H
14、冷端补偿参数	29H
15、抗干扰模式	2BH
16、数据时间存储间隔	2DH、2EH
17、波特率设定	30H
18、时间设定	32H---37H
19、打印时间设定	39H、3AH（存放打印时间间隔） 3BH(存放定点打印小时时间)
20、密码设定	3DH---3FH

表 8-3 标准信号参数存储地址

存储类别	页内地址
1、输入信号类别	00H+20H*(Y-1)
2、报警方式设定	03H+20H*(Y-1)
3、模拟输出方式设定	06H+20H*(Y-1)
4、报警回差设定	09H+20H*(Y-1)
5、零点校正	0CH+20H*(Y-1)---0EH+20H*(Y-1)
6、单位设定	10H+20H*(Y-1)
7、测量上下限设定	13H+20H*(Y-1)---18H+20H*(Y-1)
8、上下限报警设定	19H+20H*(Y-1)---1EH+20H*(Y-1)

表 8-4 非标准信号参数存储地址

存储类别	页内地址
1、输入信号类别	00H+20H*(Y-1)
2、报警方式设定	03H+20H*(Y-1)
3、模拟输出方式设定	06H+20H*(Y-1)
4、报警回差设定	09H+20H*(Y-1)
5、零点校正	0CH+20H*(Y-1)---0EH+20H*(Y-1)
6、变送输出上、下限设定	13H+20H*(Y-1)---18H+20H*(Y-1)
7、测量上、下限设定	19H+20H*(Y-1)---1EH+20H*(Y-1)

从以上四个表可以看出除了表 8-2 的参数是所有仪表公用的参数外，其余的表格中的参数是每种仪表都单独占有一段地址空间来存储。表 8-1 中的 X 表示仪表号，每一个仪表号对应一种仪表功能，表 8-3 和 8-4 中的 Y 是某一仪表状态下的信号类别号取值范围与表 6-1 中 Sn 相同，即 1 到 30。

在了解用户要设定的参数与 AT24C64 内部存储空间的对应关系的基础上我们来简要分析一下设定态程序流程图，设定态程序流程图如下图 8-5 所示。

系统共有四个按键，KEY1 是增加键，KEY2 是减键，KEY3 是确定键，KEY4 是取消键。程序进入设定态后，由于默认 Mode=0，程序进入密码判断子程序。待密码正确输入后，程序会从各个 I^2C 读取需要设定的参数值，然后判断是否需要重新选择仪表，如果需要重新选择的话，则判断用户选择的仪表是否是曾经使用过的，如果是，那么就会询问用户是否要对曾经配置过的仪表重新配置。

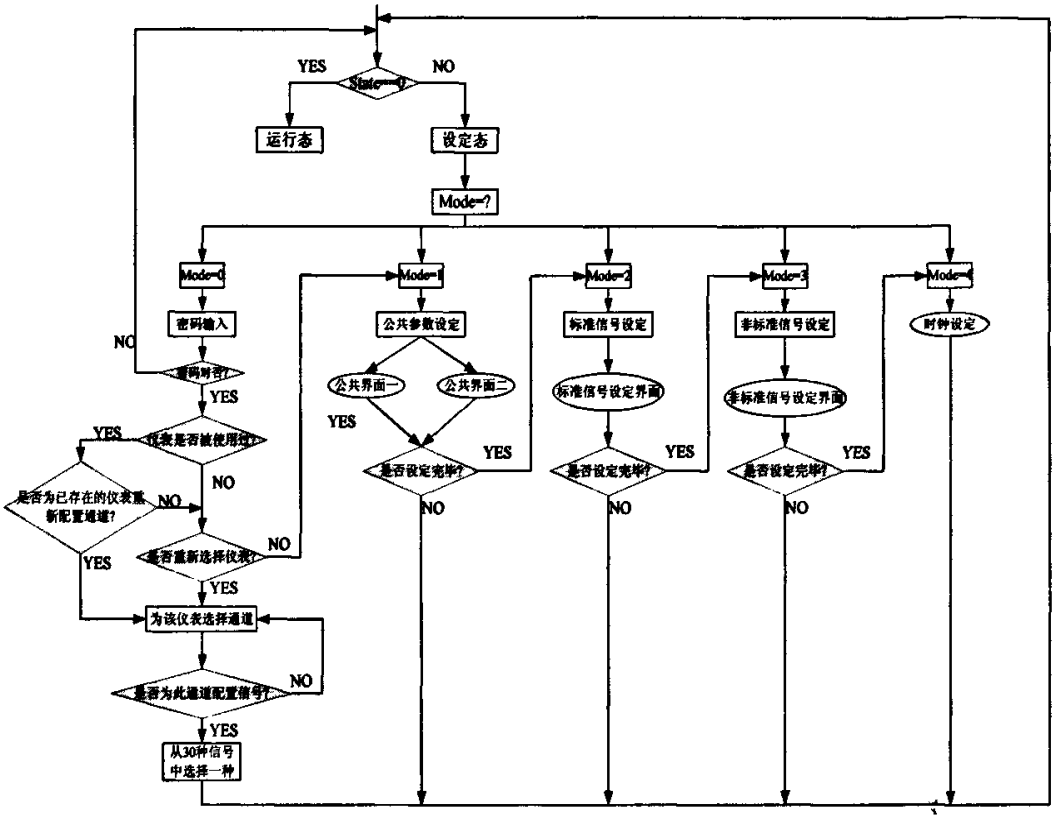


图 8-5 设定态程序流程图

如果不想重新选择仪表，则按一下 KEY1 键就进入公共参数设定界面。

在每次用户需要进行选择的时候，按下 KEY3 键就是选择，按下 KEY4 键就是退出；在 Mode=1 到 Mode=4 之间，如果程序处于该层的最外面，即刚进入该层，此时如果按下 KEY1 或 KEY2 键，程序就会跳到下一层设置，这样方便用户有选择的设定参数。待仪表设定的参数确定后，存入相应的 I^2C 存储芯片的地址空间。

以上是本系统软件设计主要流程的简要分析。下面再列出 I^2C 读、写子程序流程图，数据采集子程序流程图，打印子程序流程图供读者参阅：

1、24C08 用来存储每次用户设定后的数据（例如基准值、零点值等），以免 S3C44B0X 复位以后还要再次重新调整。程序结构主要包括初始化子程序、停止子程序、时钟信号子程序、字节读子程序、字节写子程序。

a. 写数据流程图如图 8-6 所示。

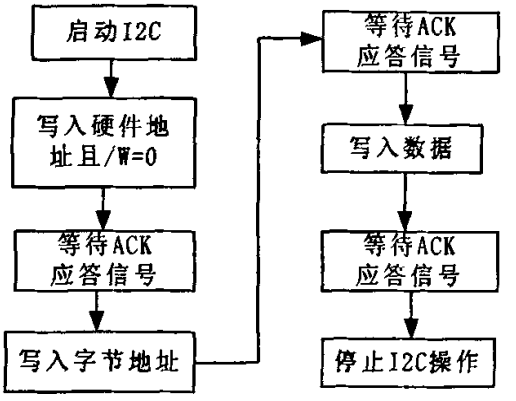


图 8-6 I^2C 写数据流程图

b. 读数据流程图如下图 8-7 所示。

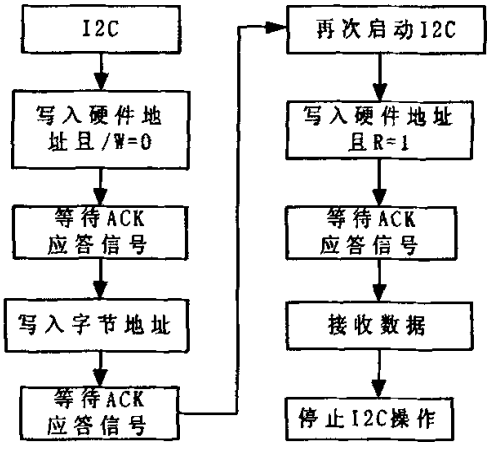


图 8-7 I^2C 读数据流程图

2、数据采集子程序流程图如下图 8-8 所示。

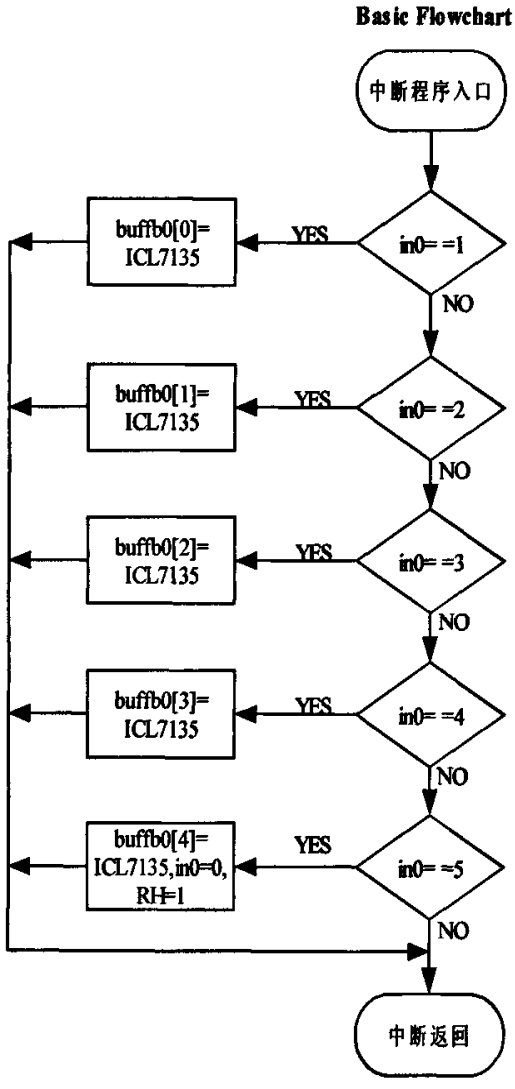


图 8-8 数据采集子程序流程图

3、打印机是用来打印主界面上的数据的，配置它是为了让用户能够把重要数据打印出来和存储下来，以备用户当前和以后分析数据用。在打印程序中，我们采用的是波特率可在 1200～9600 之间选择，无奇、偶校验和 X-ON/X-OFF 握手方式。打印程序流程图如下图 8-9 所示。

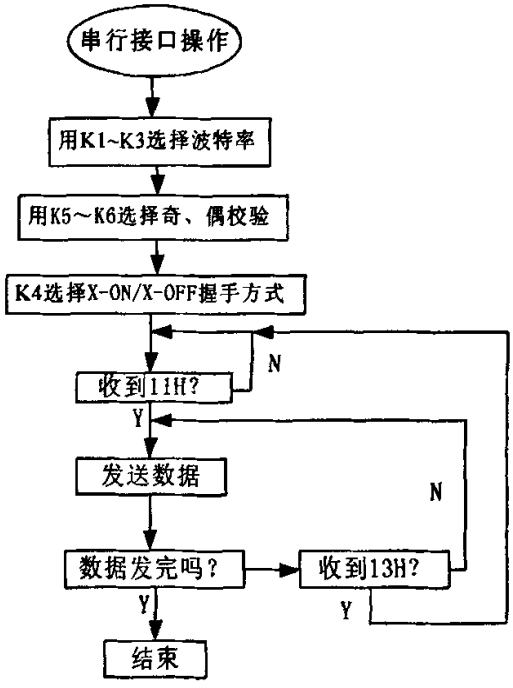

串行接口操作
用K1~K3选择波特率
用K5～K6选择奇、偶校验
K4选择X-ON/X-OFF握手方式
收到11H?
N
Y
发送数据
N
数据发完吗?
收到13H?
Y
Y
结束

图 8-9　打印程序流程图

第九章 测试与验证

测试与验证对于一台设计完成的产品而言是非常重要的。产品的性能指标是否能够达到设计之初的效果，这都需要通过测试来进行验证。整个系统的设计不仅仅完成功能的实现，对于系统各项指标参数的测试是同样重要的。因此，我们在产品调试完成后，对其进行了必要的测试工作来验证我们是否达到初期的设计要求。

基于ARM及uClinux的嵌入式测控仪表的设计项目已经完成了设计调试工作，并且经过鉴定验收，基本达到要求。通过本课题的研究，使我对嵌入式测控仪表有了更加深入的认识，增加了大量的硬件、软件系统设计经验，并且对软硬件测试方法有了更深入的了解。

测试仪器为北京康斯特科技有限责任公司的 CST3003 直流标准信号源(六位显示)，其主要技术指如下表 9-1 所示。

表 9-1 CST3003 直流标准信号源技术指标

输出不确定度表达式为:±(读数的 a%+满度的 b%)			
功能	输出范围	分辨率	不确定度
电压输出	0～100.000V	1μV	0.02%+0.003%
	0～10.0000V	0.μV	
电流输出	0～25.0000mA	0.1μA	0.02%+0.003%
电阻输出	0～500.000Ω	1mΩ	0.03%+0.003%
直流供电输出	24VDC		±1%

整个测试系统是由 CST3003 直流标准信号源和被测试仪表所组成，CST3003 直流标准信号源输出的电信号作为被测试仪表的输入，通过观察并记录信号源和被测试仪表的显示数据，可以对比两组数据得到测量精度。

通过对该仪表的抽样测试鉴定，得到下组测试数据，如表 9-2 所示。

表 9-2 测试数据

输入类型	标准信号源输出	本仪表测量结果	测量精度
0.2～1V	0.5000V	0.4997V	0.06%
0～5V	2.5000V	2.5020V	0.08%
1～5V	2.5000V	2.5032V	0.128%
0～20mA	10.0000mA	9.9817 mA	0.183%
4～20mA	10.0000mA	9.9872 mA	0.128%
0～100Ω	50.000Ω	49.830Ω	0.34%
30～350Ω	200.000Ω	199.270Ω	0.365%

0～20mV	10.0000mV	9.9858mV	0.142%
0～60mV	30.0000mV	29.9477mV	0.172%
0～100mV	50.0000mV	50.0712mV	0.142%
0～10mA	5.0000mA	4.9890mA	0.2%

以上数据为未经过软件处理的实测数据，除电阻测量数据略超出精度外其他测试数据说明本系统已经基本达到设计的精度要求。

第十章 结论和展望

10.1 本论文研究总结

本论文以在 32 位 ARM 微处理器 S3C44B0X 上移植嵌入式操作系统 uClinux 以实现在同一硬件平台下构建具有多种仪表功能的智能二次仪表的研制过程为主要内容，阐述了其具体技术及相关实现方法，实现了嵌入式智能仪表在此 ARM 平台上的应用。

在课题设计中，所做的主要工作和实现的功能如下：

1. 进行了 S3C44B0X 开发板的设计；
2. 进行了前端信号处理电路的设计；
3. 在 S3C44B0X 开发板上移植了 u-boot；
4. 将裁剪后的 uClinux 移植到 S3C44B0X 开发板上，并且可以稳定的运行；
5. 通用组态软件设计与实现。

该系统已经移植了多任务的操作系统，可以方便的加入其它的任务。也就是说，在此平台上，还可以进行二次开发，如可以内嵌 TCP/IP 协议实现对仪表的远程控制和数据传输。

10.2 前景展望

从测试结果可以看出本仪表在电阻测量上还存在一定的误差，这主要是由于系统硬件设计时为简化系统设计和降低系统成本采取简易的恒流源所造成。因此，在未来的系统改进中恒流源的设计成为一个需要解决的问题。另外，本仪表可以配上以太网，CAN，RS485[30]等通信协议，用户除了可以在本地记录仪表测试数据和仪表设定以外，还可以通过各种通信协议对该仪表进行远程工作状态设定、远程桌面显示及远程数据传输等；本仪表还可以在外扩的 IDE 接口上挂载硬盘实现大容量、常时间的数据存储等等。

现代自动化仪表的智能化技术不仅改善了仪表本身的性能，还影响到了控制

网络的体系结构，而且其自身的适应性越来越强，功能也日渐丰富。相信新一代的智能化仪器仪表将在计算机网络技术支持下，在各行各业将会得到更加广泛的应用。

致　谢

本课题不论是最初的选题、方案论证，中期的电路设计、调试，还是最后系统功能指标的实现、测试以及本论文的定稿都得到了我身边的老师、亲人和朋友们的大力帮助与支持，故在此向他们表示深深的谢意。

首先要衷心感谢我的导师李广军教授！从本课题最初的选题到方案确定到实际系统实现、调试都得到了他无微不至的关怀、指导和帮助。导师深厚的理论功底、极其丰富的系统实践经验以及严谨的科研态度与作风等都给我留下了深刻的印象，并对我课题的完成起到了至关重要的作用。导师的言传身教为我今后的科研工作与学习奠定了坚实的基础！

其次，我要特别向明瑞电子公司的赵红武工程师表达我深深的谢意，赵工的丰富工程经验和高效的工作风范给我留下了极为深刻的印象。感谢赵工在本课题的实施中给予我的帮助与指导！感谢教研室的老师和同学，他们为我项目的研发和论文的完成提供了许多无私的帮助。感谢所有曾经关心和帮助过我的朋友们！

最后，感谢我的父母和我的哥哥，他们是我学习和工作中克服困难的动力和源泉，感谢他们默默地支持和关怀！

参考文献

[1] 张培仁.基于 C 语言编程 MCS-51 单片机原理与应用[M].北京：清华大学出版社, 2003

[2] 王福瑞.单片微机测控系统设计大全[M].北航出版社, 2001

[3] 徐惠民，安德宁.单片微型计算机原理、接口及应用[M].北京：北京邮电大学出版社, 2000.10

[4] 丁劲松,李行善.下一代自动测试系统体系结构与关键技术[J].计算机测量与控制 2005,13(1):1-3

[5] 林占江，张乃国.电子测量技术[M].北京：电子工业出版社，2003.9

[6] 李驹光编著.ARM 应用系统开发详解[M].北京：清华大学出版社, 2004

[7] 季显，林俊超，宋飞编著.ARM 嵌入式应用系统开发典型实例[M].北京：中国电力出版社, 2005

[8] Building Embedded UClinux Systems by Karim Yaghmour (January 2001) Publisher: O'Reilly Media

[9] Embedded Linux System Design and Development by P.Raghavan,Amol Lad,Sriram Neelakandan (December 21, 2005) Publisher: Auerbach Publications

[10] Linux for Embedded and Real-time Applications, Second Edition (Embedded Technology) by Doug Abbott (April 3, 2006) Publisher: Newnes

[11] 孙琼.嵌入式 Linux 应用程序开发详解[M].北京：人民邮电出版社, 2006

[12] (美) Jean J. Labrosse 著;邵贝贝等译. 嵌入式实时操作系统--μC/OS-II［M］.北京:北京航空航天大学出版社, 2003

[13] 沙占友.智能化集成温度传感器原理与应用[M].北京：机械工业出版社, 2002

[14] 黄贤武.传感器原理及应用[M].北京:北京航空航天大学出版社, 2000

[15] 陈渝, 李明, 杨晔等编著. 源码开放的嵌入式系统软件分析与实践[M].北京：北京航空航天大学出版社，2004

[16] 田泽编著.嵌入式系统开发与应用[M].北京 ：北京航空航天大学出版社, 2005

[17] 李岩等.基于 S3C44B0X 嵌入式 uClinux 系统原理及应用[M].北京：清华大学出版社, 2005

[18] 廖日坤.ARM 嵌入式应用开发技术白金手册[M].北京：中国电力出版社, 2005

[19] (美)鲁比尼(Alessandro Rubini),Jonathan Corbet 著. LINUX 设备驱动程序[M]. 北京:中国电力出版社, 2002.11

[20] 孙天泽等编著.嵌入式设计及 Linux 驱动开发指南[M].北京：电子工业出版社, 2005

[21] (美)斯肯克(Schenk,T.)等著；熊志辉、胡季红、叶子等.Red Hat Linux 系统管理大全 [M]. 北京:机械工业出版社, 2001.4

[22] ARM Architecture Reference Manual by David Seal (Dec 27 2000) Publisher: Addison -Wesley Professional

[23] ARM System Developer's Guide : Designing and Optimizing System Software by Andrew Sloss,

Dominic Symes, and Chris Wright Publisher: Morgan Kaufmann(Mar 25 2004)
[24] Real-Time Embedded Multithreading : Using ThreadX and ARM by Edward L. Lamie (Jan 1 2005) Publisher: CMP; Bk&CD-Rom edition
[25] Real-Time Systems Development by Rob Williams (Dec 3, 2005) Publisher: Butterworth-Heinemann
[26] Embedded System Design on a Shoestring (Embedded Technology) by Lewin Edwards (Jun 2003) Publisher: Newnes
[27] The Digital Consumer Technology Handbook: A Comprehensive Guide to Devices, Standards, Future Directions, and Programmable Logic Solutions by Amit Dhir (Feb 13, 2004) Publisher: Newnes
[28]Open Source: The Unauthorized White Papers by Donald K. Rosenberg (Jan 15, 2000)Publisher: John Wiley & Sons Canada, Ltd
[29] http://www.nuedc.cn(全国大学生电子设计技术网).

攻硕期间取得的研究成果

[1] 刘俊、赵建、杨明欣、李广军.柔性测量模块的芯片化设计.传感器与微系统,2006.10:47-50

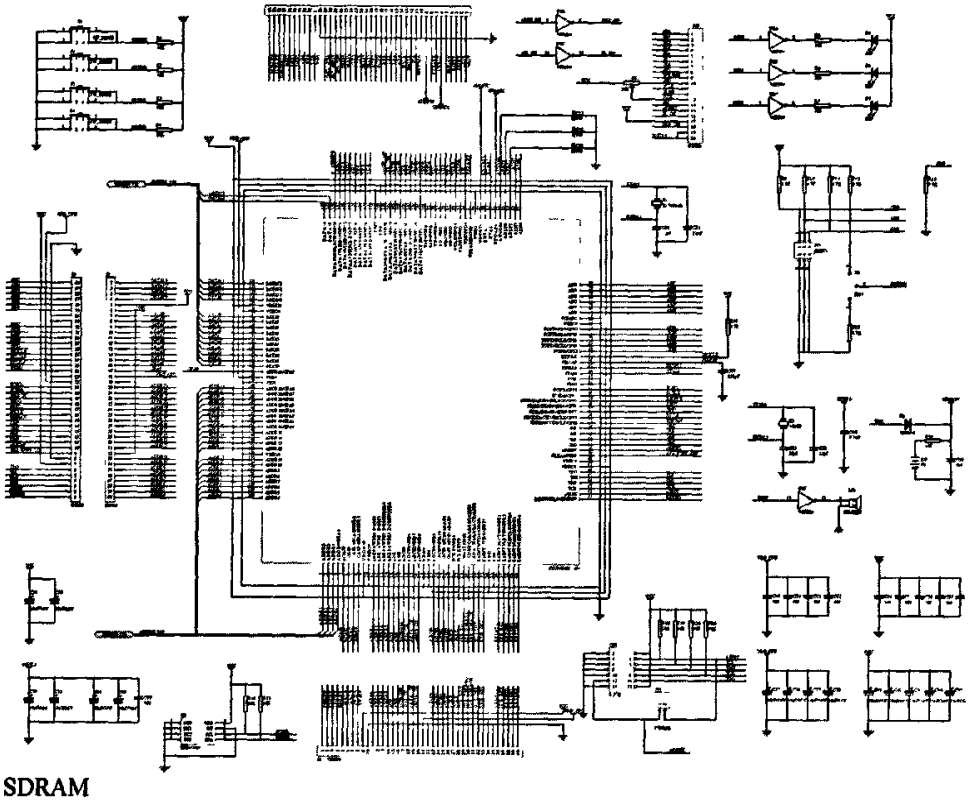
[2] 刘俊、王建波、杨明欣、李广军.基于ARM的嵌入式智能仪表信号处理电路的设计.四川师范大学学报（自然科学版）,2006.11:276-279

[3] 赵建、刘俊、杨明欣、邓娜.基于GSM手机短信的无人值守自动抽水控制仪.微计算机信息,2006.10:112-114

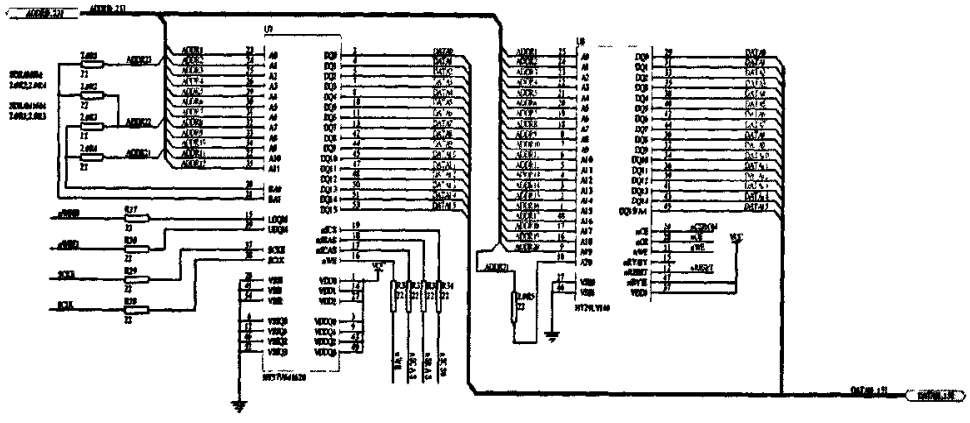
[4] 陈永强，杨明欣，刘俊.基于克隆选择粒子群算法的多用户检测，数据采集与处理,2007.4

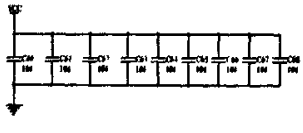
附录 I S3C44B0X 开发板原理图

S3C44B0X Core

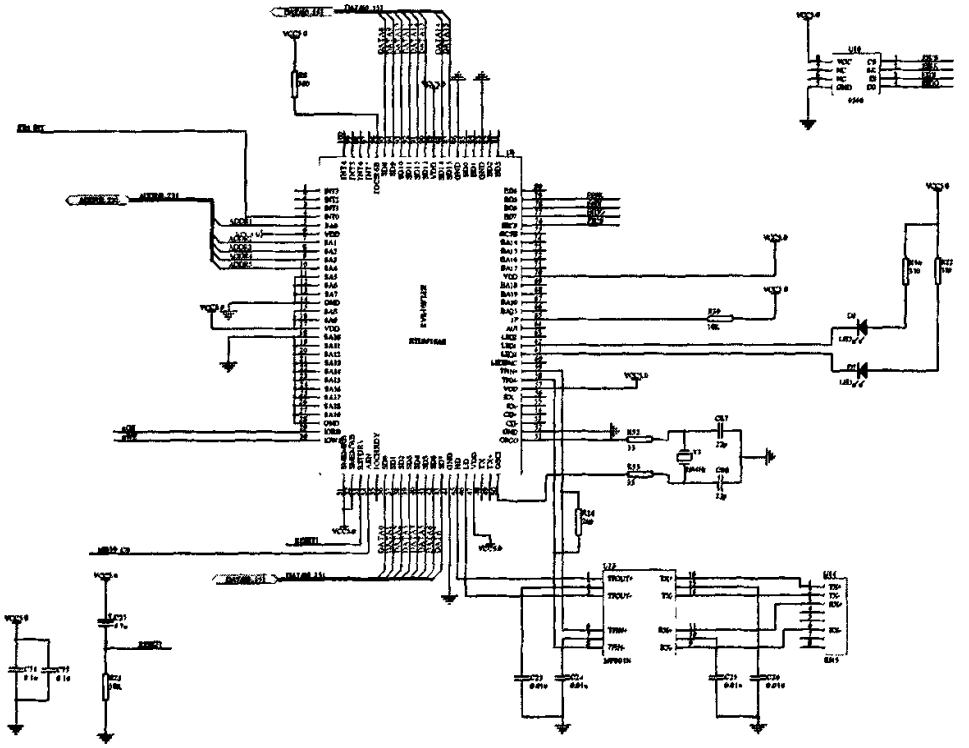


SDRAM

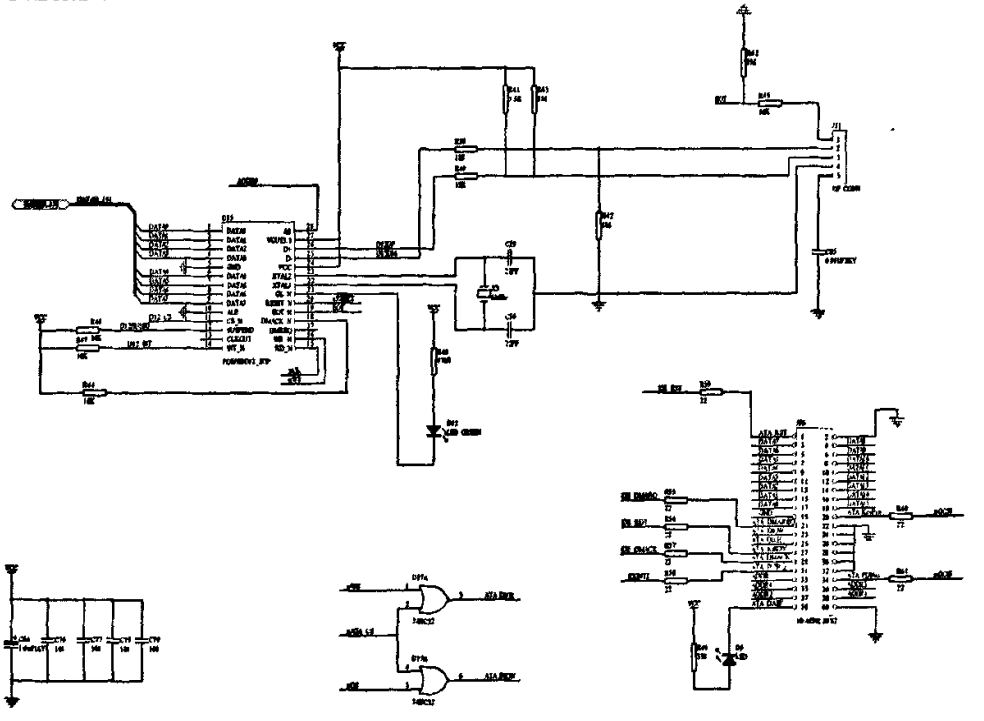




RTL8019



USB&IDE



附录 II memsetup.s 的修改代码

```
MEMORY_CONFIG:    /* 内存配置 long 伪操作，用于分配字节对其的内存单元*/
    .long  0x11000102     /* BWSCON   */
    .long  0x600          /* BANKCON0 */
    .long  0x7ffc         /* BANKCON1 */
    .long  0x7ffc         /* BANKCON2 */
    .long  0x7ffc         /* BANKCON3 */
    .long  0x7ffc         /* BANKCON4 */
    .long  0x7ffc         /* BANKCON5 */
    .long  0x18001        /* BANKCON6 */
    .long  0x18000        /* BANKCON7 */
    .long  0xA60459       /* REFRESH  */
    .long  0x07           /* BANKSIZE */
    .long  0x20           /* MRSRB6   */
    .long  0x20           /* MRSRB7   */
    .globl memsetup
    memsetup:
    adr    r0, MEMORY_CONFIG /*地址读取指令 将MEMORY_CONFIG地址赋给r0*/
    ldmia r0, {r1-r13}       /**/
    ldr    r0, =0x01c80000   /*简单赋值语句*/
    stmia r0, {r1-r13}   /*保存数据到 r0 指向的地址、即由 0x01C80000 开始的地址*/
    mov    pc, lr            /*函数返回*/
```

该段代码的作用是对寄存器赋值，比如对 BANKCON6 与 REFRESH 寄存器的设置等效于：

```
    /* Bank 6 parameter */
    .equ     B6_MT,         0x3 /* RAM类型是同步动态RAM */
    .equ     B6_Trcd,       0x1 /* 2clk */
    .equ     B6_SCAN,       0x1 /* 9bit */
    .equ     B7_MT,         0x3 /* SDRAM */
```

```
.equ    B7_Trcd,        0x1 /* 2clk */
.equ    B7_SCAN,        0x1 /* 9bit */
/* REFRESH parameter */
.equ    REFEN,          0x1   /* 刷新使能 */
.equ    TREFMD,         0x0   /* 自动刷新 */
.equ    Trp,            0x0   /* 2clk */
.equ    Trc,            0x3   /* 0x1=5clk 0x3=11clk*/
.equ    Tchr,           0x0   /* 0x2=3clk 0x0=0clks  */
.equ    REFCNT,         1550
```

附录 III uClinux 内核选项列表

配置选项		描述	说明
Code maturity level options		内核代码成熟等级	
	Prompt for development and/or incomplete code/drivers	显示还在开发或者还没有完成的代码和驱动	有些设备依赖于该项
	Prompt for obsolete code/drivers	显示废弃的代码或者驱动	基本不用
Loadable module support		模块加载支持	
	Enable loadable module support	使能模块加载支持	建议选上
System Type		系统类型	
	ARM system type	ARM 系统	这里选择 CPU 类型
	Generate big endian code	生成大端格式代码	
	Set flash/sram size and base addr	设置 Flash 和 sram 的大小和地址	
	Kernel excutes from	内核执行起点	RAM
General setup		常规设置	
	Support hot-pluggable device	支持热插拔设备	
	Net working support	网络支持	建议选上
	System V IPC	支持 system V 的进程间通讯	建议选上
	Reduced memory footprint		
	BSD Process Accounting		
	Sysctl support	支持在不重启的情况下直接改变内核参数	建议选上
	NWFPE math emulation	模拟数学协处理器	建议选上
	Kernel core(/proc/kcore) format	内核文件格式	ELF/A.OUT 可选
	Support uClinux FLAT format binaries	支持 FLAT 格式文件	选中
	Power Management support (NEW)	电源管理	一般不选
ATA/IDE/MFM/RLL support			
	ATA/IDE/MFM/RLL support	ATA/IDE/MFM/RLL 支持	选中
SCSI support			
	SCSI support	SCSI 支持	
ISDN subsystem			
	ISDN support	ISDN 支持	
Parallel port support			
	Parallel port support	并口支持	
Memory Technology Devices (MTD)			
	Memory Technology Devices (MTD) support	MTD 支持	
Plug and Play configuratoin			
	Plug and Play support	即插即用设备支持	
Block devices		块设备	
	Normal floppy disk support	普通软盘设备支持	

	XT hard disk support	XT 硬盘支持	
	Compaq SMART2 support	Compaq SMART2 支持	
	Loopback device support	Loopback 设备支持	
	RAM disk support	RAM 支持	选中
	Default RAM disk size	默认 RAM 大小	
	Initial RAM disk (initrd) support	初始化 RAM 设备	选中
	RAM disk data block compiled in		
	ROM disk memory block device (blkmem)		
File systems		文件系统	
	Quota support	份额分配支持	
	Kernel automounter support	自动加载支持	在有 NFS 文件系统时建议选上
	Kernel automounter version 4 support (also supports v3)	自动加载支持，V3 版本的升级	同上
	Reiserfs support	Reiserfs 文件系统支持	
	Ext3 journalling file system support	EXT3 文件系统支持	
	DOS FAT fs support	DOS FAT 文件系统支持	选中
	Compressed ROM file system support	Compressed ROM 文件系统支持	
	Virtual memory file system support (former shm fs)	虚拟存储器支持	
	ISO 9660 CDROM file system support	ISO 9660 CDROM 支持	
	JFS filesystem support	JFS 文件系统支持	
	Minix fs support	Minix 文件系统支持	
	FreeVxFS file system support (VERITAS VxFS(TM) compatible)	FreeVxFS 文件系统支持	
	NTFS file system support (read only)	NTFS 文件系统支持	
	OS/2 HPFS file system support	OS/2 HPFS 文件系统支持	
	/proc file system support	/proc 虚拟文件系统支持	
	QNX4 file system support (read only)	QNX4 文件系统支持	
	ROM file system support	ROM 文件系统支持	
	Second extended fs support	EXT2 文件系统支持	
	UDF file system support (read only)	UDF 文件系统支持	
	UFS file system support (read only)	UFS 文件系统支持	
	Partition Types	分区类型	选中
Character devices		字符型设备	
	DS1302 Real Time Clock support	DS1302 实施时钟支持	
	Virtual terminal	虚拟终端	选中
	Standard/generic (8250/16550 and compatible UARTs) serial support	标准/（8250/16C550 兼容串口）支持	选中
	Support for console on serial port	控制台	选中
	Non-standard serial port support	非标准串口支持	

	Unix98 PTY support	PTY 伪终端	
	I2C support	I^2C 支持	
	Mice	鼠标	
	Joysticks	游戏柄	
	QIC-02 tape support	非 SCSI 的磁带支持	
	Controller Area Network Cards/Chips		
	Watchdog Cards	看门狗定时设备	
	NatSemi SCx200 GPIO Support		
	/dev/nvram support	NVRAM 支持	
	Enhanced Real Time Clock Support	增强型实时时钟支持	
	Siemens R3964 line discipline	R3964 协议通讯	
	/dev/agpgart (AGP Support)	AGP 支持	
	Direct Rendering Manager (XFree86 DRI support)	XFree86 DRI 支持	
USB support			
	Support for USB	USB 支持	
I2O device support			
	I2O support	I2O 设备支持	
Kernel hacking			
	Verbose kernel error messages	详细的内核错误信息	一般不选
	Verbose user fault messages	详细的用户出错信息	一般不选
	Include debugging information in kernel binary	在内核库中包含调试信息	一般不选
Library routines			
	CRC32 functions	CRC32 函数	
	zlib compression support		

注：本表来源是参考<内核定制选项>一文并做相应调整而成