摘 要

心血管疾病是当今世界危害人类健康的头号杀手，主要由高血压和动态粥样硬化等病症引起，早期这些病症不明显，但是一些相关的参数都已发生变化。因此通过检测这些参数就可以及早诊断出心血管疾病的潜在危险，也可以评估病人的病况和预示疾病的程度。因此若能及时检查这些参数就可以及早诊断出心血管疾病的潜在危险，为其预防和治疗争取了宝贵的时间。大量的临床实测结果证实，脉搏波的波形特征与心血管疾病密切相关。因此，系统通过检测脉搏信号来检测心血管参数。

便携式医疗仪器具有很大的市场，医疗仪器已从传统的 PC 和工业控制计算机转向嵌入式计算机系统。随着微处理器运算能力的增加，ARM 微处理器及其优越的性能必将成为心血管检测系统的的主要平台。本系统采用三星 ARM920 作为处理器，通过脉搏传感器采集脉搏信号，并基于嵌入式 Linux 操作系统来实现。系统可实时显示脉搏波波形，选择显示心血管参数。本论文详细阐述了如何通过检测脉搏波来计算心血管参数；具体分析了系统的硬件平台；主要论述了软件的实现，包括 bootload 的移植，嵌入式 Linux 系统的移植，驱动程序的移植；应用程序的编写；基于 QT 的图形界面开发。采用高性能的 ARM 处理器作为系统的控制核心，不但能实时检测到脉搏信号，并对信号进行分析处理，而且集成了丰富的外设接口，有利于整个系统的集成。进一步提高通过脉搏波信号计算心血管参数的精度，系统的集成化和小型化，对参数异常处理的进一步处理是今后工作的发展趋势。

随着医疗卫生事业的发展，心血管疾病的预防和治疗急需解决，心血管检测系统具有广阔的市场空间，不仅适合临床使用，也适合普通家庭的应用。

关键词：心血管参数；脉搏波；ARM；嵌入式系统；QT。

Abstract

Cardiovascular disease is one of the most terrible killer in the world,it is caused by hypertension and AS,in the early these symptoms is not obvious,but some of the correlative parameter is changed.So we can diagnose the potential crisis in advance by detecting the cardiovascular parameters,that can also evaluate patients' state and forsee the state of illness.So detect the parameters in time, cardiovasclar would be dignosed as soon as possible,which can save time for prevent and treatment.Plenty of clinical results approve that pulse wave is closely connected to cardiovascular diseases. The system detects the cardiovascular diseases by testing its parameters.

Presently the medical appliance transforms from traditional PC and industrial control computer to embedded system.With development of micro-processor,ARM core embedded system will be the main platform for homely used cardiovascular detecting system.The system chooses ARM920 from SamSung as processor, acquire the pulse signal through the pulse sensor using embedded Linux OS.The system can display the pulse wave real-timely,calculate and display the cardiovascular parameters selectively.The paper expounds how to use the pulse wave to calculate the cardiovascular parameters;analyse the hardware platform of the system;Mainly discuss the software developments,include porting the bootloader,Linux OS and the drivers to the ARM board;the development of applications;the graphical interfaces development based on QT. It is a trend that the cardiovascular parameters measurement system has further more precision,systematic integration and miniaturization,the process of parameter abnormity.

With the medical treatment develops,the prevent and treatment of cardiovascular need to solve immediately.The cardiovascular detecting system has a wide market,use not only in clinic,but also in family health care.

Keywords: Cardiovascular parameter;pulse wave;ARM;Embeddedsystem;QT.

厦门大学学位论文原创性声明

兹呈交的学位论文，是本人在导师指导下独立完成的研究成果。本人在论文写作中参考的其他个人或集体的研究成果，均在文中以明确方式标明。本人依法享有和承担由此论文产生的权利和责任。

声明人（签名）：杨创萍

2007年 6月 2日

厦门大学学位论文著作权使用声明

本人完全了解厦门大学有关保留、使用学位论文的规定。厦门大学有权保留并向国家主管部门或其指定机构送交论文的纸质版和电子版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆被查阅，有权将学位论文的内容编入有关数据库进行检索，有权将学位论文的标题和摘要汇编出版。保密的学位论文在解密后适用本规定。

本学位论文属于

1. 保密（ ），在年解密后适用本授权书。

2. 不保密（√）

（请在以上相应括号内打“√”）

作者签名：杨剑萍 日期：2008 年 6 月 2 日

导师签名：达力 日期：2008 年 6 月 2 日

第一章 绪 论

1.1 心血管参数检测意义与方法

人体心血管动脉系统是一个几何特性和物理特性都相当复杂的弹性管系。在动脉中流动的血液具有多种有形颗粒的悬浮液体，具有异常粘度的非牛顿液体。动脉系统各部分的流动特征差异很大，情况十分复杂，这就给其参数准确测量带来困难。因此，建立一个科学、准确的心血管动力学参数无创检测原理和新方法，为广大医生对心血管疾病的检测、诊断、监护及确定治疗措施提供一个简捷方便和准确可靠的重要诊断手段，无疑对生物医学工程的发展具有深远的意义[1]。

1.1.1 心血管参数检测意义

心血管疾病是当今发达国家死亡率占第一的重要疾病，在我国也是死亡率最高的一类疾病，世界卫生组织已将其列为 21 世纪危害人类健康的头号杀手。因此，如何治疗这类疾病，已成为世界各国迫切需要解决的一项重大课题。对心血管血流动力学参数进行科学合理的检测、诊断、分析，这对临床医学的发展是极有意义的。

高血压和动脉粥样硬化是引起各种心血管病的主要原因。但这些危险因素早期症状不明显，一般经过多年才在临床上有所表现，因而往往很容易会被人们所忽视而错过早期治疗的机会。如何能在发病初期方便、准确、舒适的检查和诊断出这种无明显症状的病症来是一个极大的难题。研究表明，高血压和动脉粥样硬化的初期阶段，虽然没有自觉症状，但血压、血流、血管阻力等一系列心血管参数实际上已开始发生变化。如果能及时检查出这些参数的变化，并对其特征进行分析，就可能在还没有自觉症状的情况下及早诊断出高血压和动脉粥样硬化这两个心血管病的潜在危险因素，为心血管病的预防和治疗争取到宝贵的时间[2]。

1.1.2 心血管参数检测方法

随着心脏的间歇性的收缩和舒张，血液压力、血流速度和血流量的脉动以及血管壁的变形和振动在血管系统中的传播，统称为脉搏波或脉搏波在血管中的传

播[1]。

脉搏波压力及波形特征变化是评价人体心血管系统生理病理状态的重要依据。无论是中医切脉或是西医心血管疾病检查，都试图从脉搏波的压力与波形变化中提取各种生理病理信息。这是因为当脉搏波出心脏开始向动脉系统传播时，不仅要受到心脏本身的影响，同时也会受到流经各级动脉及分支中各种生理因素如血管阻力、血管壁弹性等的影响，使脉搏波中包含有极丰富的心血管系统生理病理信息。脉搏波所呈现出的形态（波形）、强度（波幅）、速率（波速）和节律（周期）等方面的综合信息，很大程度上反映了人体心血管系统中许多生理病理的特征[1]。

脉搏波的传播特性等与心血管系统中的力学参数变化密切相关的。大量实验和计算结果证明，在动脉血管弯曲、分叉和狭窄部位最容易引起动脉粥样硬化等心血管疾病。对心血管血流动力学参数进行科学合理的检测、诊断和分析，对临床医学的发展有极其重要的意义。另外，中医脉象和心血管系统血液运动、血管壁的运动及脉搏波的传播规律有密切的关系。因此，对脉搏波传播规律进行研究，并将其与传统中医的脉象诊断相结合，以求利用无创检测方法对人体心血管疾病进行早期的诊断治疗，具有实用价值[4]。

心血管参数主要是指收缩压 P_s，舒张压 P_d、脉压差（P_s-P_d）、平均动脉压 Pm、心率 HR、心搏出量 SV、心输出量 CO、心搏指数 SI、心脏指数 CI、外周阻力 TPR、血管顺应性 AC、血流的半更新率 ALK、血流的半更新时间 ALT 以及血流的平均滞留时间 T_m等十余项指标。这些参数都可根据血压和血流分别求出。其中，心输出量 CO 或心搏出量 SV 是评估心血管功能最为重要的治疗效果，从而确定病人状况是否已得到改善或进一步恶化[3]。

1.2 技术背景与研究现状

1.2.1 脉搏波理论基础

人体的循环系统的原动力是心脏的射血，其中心室起主要作用，所以通常所谓的心动周期实际上是指心室活动周期。左心室的收缩将血液射入动脉，左心室的舒张又将新鲜血液吸入心室，在此过程中脉搏波也形成了一个周期。心脏射血

时，心室肌收缩所释放的能量，一部分用于推动血液流动，成为血流的动能，另一部分则形成对血管壁的侧压，成为作用于血管壁的势能[4]。

动脉血管系统，按其结构与机能特点可分为两部分：一是主动脉和大动脉部分，二是小动脉和微动脉部分。主动脉与大动脉部分的血管壁坚厚，含有丰富的弹性纤维和胶原纤维，但平滑肌成分较少。这些弹性成分使得主动脉和大动脉血管在左心室射血时，能被动扩大其容量，暂时贮存一部分血液，以缓冲压力过度升高。这时心室收缩驱动血液的一部分能量，便以势能的形式贮存在弹性动脉管壁中。当心室舒张时，被扩张的血管发生弹性回缩，将射血期多容纳的那部分血液继续向外周推进，从而使间断的射血变为连续性流动。从主动脉和大动脉血管分支出来的各动脉以并联的形式向全身各器官的毛细血管供血。这些小动脉和微动脉管壁的结构特点是，弹性纤维和胶原纤维成分减少，而环形平滑肌成分明显增多。每当血管平滑肌收缩时，可直接使血管口径缩小，增加血流阻力，从而使向心端大动脉血管内压力升高[4]。

动脉血压就是当血液流经动脉系统时对血管壁施加的侧压。动脉血压是推动血液流动的驱动力，它必须达到一定的高度才能保证向全身各器官的血液供应。在血管系统内有足够的血液充盈是形成血压的前提。在此基础上心脏射血所作的功，一部分形成流速，一部分产生侧压。但是如果不存在主要由阻力血管所构成的外周阻力，则心脏射出的血液将迅速流向外周，致使心室收缩释放的能量全部或大部分转为动能，而形不成侧压。只有在外周阻力配合下，心脏射出的血液不能迅速流走，暂时存留在阻力血管向心端的较大动脉的血管内，这时心室收缩的能量才能大部分以侧压形式表现出来，形成较高的血压水平。所以，动脉血压的形成是心脏射血和外周阻力相互作用的结果。在心室舒张期停止射血时，则由大动脉回弹作用与外周阻力相配合，以维持一定的血压水平[4]。

动脉压从主动脉和大动脉传播到小血管和微血管的过程就形成了动脉脉搏波，它随心动周期成周期性变化。不同动脉段的血压数值有所不同。与主动脉内的动脉压相比，外周动脉的收缩压较高、舒张压较低。桡动脉脉搏波容易从体表测量，其典型波形如图 1-1 所示，它可以很好地反映心血管的信息，如果人体动脉系统产生异常（如出现动脉粥样硬化等），动脉血管的各项性质会发生改变，从而桡动脉脉搏波形也会发生一定的改变[4]。

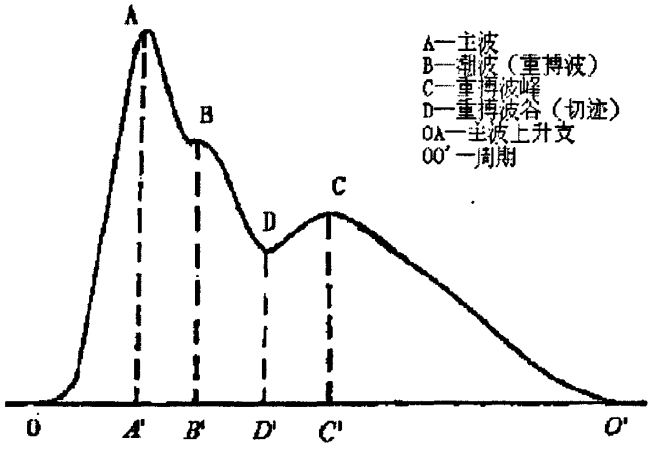

图 1-1 桡动脉脉搏波波形图

脉搏波形包括一个升支和降支。升支是左心室射血时动脉壁突然扩张所引起的。在心室快速射血期，动脉血压迅速上升，管壁被扩张，形成脉搏波形中的上升支。上升支的斜率和幅度受射血速度、心输出量以及射血所遇阻力的影响，若射血时遇到的阻力大，心输出量小，射血的速度慢，则脉搏波形中上升支的斜率小，幅度也低；反之，若射血所遇到的阻力小，心输出量大，射血的速度快，则上升支较陡，幅度也较大。大动脉的可扩张性减小时，弹性贮器作用减弱，动脉血压的波动幅度增大，脉搏波上升支的斜率和幅度也加大。主动脉瓣狭窄时，射血阻力高，脉搏波上升支的斜率和幅度都较小[1]。

心室射血的后期，射血的速度减慢，进入主动脉的血量少于由主动脉流向外周的血量，故被扩张的大动脉开始回缩，动脉血压逐渐降低，形成脉搏波形中下降支的前段，即潮波。随后，心室舒张，动脉血压继续下降，形成下降支的其余部分。在降支中段出现的小波称为降中波，又称重搏波。降中波前面的下凹部分称降中峡。降中峡的产生是由于左心室舒张，主动脉内血液倒流，血压突然下降，管壁回缩所形成。降中波则是由于主动脉瓣关闭血流冲击在主动脉瓣上而弹回，动脉压再次稍有升高，管壁又稍有扩张而形成。动脉脉搏波形中下降支的形状可大致反映外周阻力的高低。外周阻力高时，脉搏波降支的下降速率较慢，切迹的位置较高。如果外周阻力较低，则下降支的下降速率较快，切迹的位置较低，切迹以后下降支的坡度小，较为平坦。主动脉瓣关闭不全时，心舒期有部分血液倒流入心室。故下降支很陡，降中波不明显或者消失[1]。

综上所述，脉搏波的幅值与波形变化，反映出在一个心动周期中动脉血压随时间的脉动变化。脉搏波中所包含的高血压和动脉硬化等信息主要反映在脉搏波的幅值与波形变化之中，并通过血压、血流、血管阻力和血管壁弹性等血流参数的变化表示出来。因此，对脉搏波形特征进行研究，利用无创检测方法对人体心血管疾病进行早期的诊断治疗，这对临床医学的发展是很有价值的。

1.2.2 基于脉搏波的主要心血管功能参数

①心率

指每分钟心脏所经历的心动周期的个数，反映心脏跳动的频率。心率对动脉脉搏波的一个明显影响就是周期，心率加快周期变短。另外，心率对动脉压也会产生影响：室每次收缩射入主动脉的血液只有一部分在收缩期内流走，其余部分则需要在舒张期内流向外周。如果心率突然增快，则因舒张期变短，流向外周的血量减少，致使舒张末期主动脉内存留血量增多、舒张压升高。在此基础上收缩压也会升高，但动脉压升高可使血流速度加快，在收缩期内有较多的血液流向外周，致使收缩压的升高不如舒张压升高显著。因此当心率增加时，动脉脉搏波的波形就会“变低”。

②心输出量、每搏输出量、心脏指数与心搏指数

心脏的基本生理机能在于有节律地将血液射入动脉系统中。心室每分钟射出的血量称为心输出量，每次射出的血量称为每搏输出量。机体静息时的心输出量同体内代谢过程相适应，与体表面积成正比关系。为了便于在个体之间进行比较，将单位体表面积的心输出量称为心脏指数，相应的单位体表面积的每搏输出量称为心搏指数。当心室收缩加强输出量增加时，动脉血压升高。这时血流速度加快，血液向外周流走的量增多，因而动脉系统舒张末期总容量虽然有所增加但程度不大。因此收缩压的升高要大于舒张压的升高，从而使得动脉脉搏波的波形“变高”。

③外周阻力与血液粘度

血液在血管中以一定的流量流动，由于血管半径有限、血液具有一定粘度使得血液流经一段长度后压力减小，也就是收到了阻力的作用，称为外周阻力。设血管两端的压力差为 1mmHg（毫米汞柱），血流量为 1mL/s，则此时的血流阻力称为 1 个外周阻力单位（PRU）。外周阻力增加时，由于血液向外周流走的速

度减慢，致使舒张末期主动脉内存留的血量增多，舒张压增高，此时收缩压也将增高。但由于动脉血压升高使血流速度加快，因此收缩压升高不如舒张压升高明显，从而使得动脉脉搏波的波形“变低”。外周阻力与血液粘度成正比，血液粘度增加就会加大外周阻力、血压升高，而增加心脏的负担。

④主动脉与大动脉的弹性

主动脉和大动脉的弹性贮器作用，可缓冲血压波动的幅度，即防止收缩压过高和舒张压过低，因而使脉压差减小。大动脉弹性随年龄的增大而减小，因而年龄大的人动脉血管壁硬度高、脉搏波形会“变高”。

1.2.3 基于脉搏波的心血管功能研究现状

脉搏波的波形特征信息与生理因素存在着必然的联系。为了研究两者的关系，国内外许多研究者以临床实验或模型计算脉搏波图为根据，然后分别在时域或频域中对脉搏波图提取特征信息。在时域中提取信息，一般是在脉图中提取一些具有明确生理意义的点（如主波波峰、重搏波高度等）。将这些特征点与对应的生理因素结合起来就可能得到许多有临床医学价值的结果。这种方法比较直观、易被临床医生接受，但它是建立在脉搏波曲线某些点的特征上的，没有把曲线所包含的全部信息利用起来。此外，某些与生理以因素密切相关的特征点很难被准确地检测，需要凭经验估计，随机误差很大。在频域中提取信息，一般是利用傅立叶变换从脉搏波频谱中提取与人体生理病理有关的信息。将特征信息以脉搏波所具有的全部频率分量的集合形式表示，保留了脉搏波中的全部信息。但由于计算复杂、结果抽象，难以被广大医务工作者接受[4]。

为解决这个问题，有一些研究者利用仿真模型逼近实测的脉搏波得到不同的模型参数，根据参数的不同来判断人的生理状况。实践证明，这种研究方法是较为有效的，但仿真模型与所提取的特征参数必须得当，才能有效区分病理状态。在许多研究中，由于提取的参数过于繁杂而使得对脉搏波的区分度不高，经常出现误判现象。因此脉搏波参数的提取成为研究的关键[4]。

北京工业大学生物医学工程中心罗志昌教授等人利用已有的双弹性腔模型提取一个代表脉搏波波图面积变化的特征量 K 值（称之为波形系数）。通过模型理论分析、动物实验以及数千例的不同年龄健康人和心血管疾病患者的临床检

测，证实出心血管生理和病理上的变化将会引起脉搏波波图特征和面积的相应变化，进而反映在 K 值的变化上。用 K 值判别人体的生理状况，虽然不能做到精确的定量分析，但具有计算简便、区分度和敏感度高的优点，因而在临床上具有重要的引用价值，是心血管临床检查的一个重要生理指标。

通过桡动脉脉搏波波形可以算出波形系数 K 值，再根据 K 值、收缩压和舒张压可以计算出大部分的心血管参数。

1.3 嵌入式 Linux 系统概述

1.3.1 嵌入式系统概述

1.3.1.1 嵌入式系统定义

虽然嵌入式系统是近几年才风靡起来的，但其历史可追溯到加 20 世纪 70 年代。经过 30 多年的发展，在硬件和软件交替发展的支撑下，嵌入式技术逐渐趋于稳定和成熟，已被广泛应用于工业控制、交通管理、信息家电、家庭智能管理系统、POS 网络及电子商务、环境检测、机器人等各个领域。毫不夸张的说，嵌入式系统已经无所不在。所以，研究和开发嵌入式系统有着十分重要的意义[5]。

根据 IEEE（国际电气和电子工程师协会）的定义，嵌入式系统是“控制、监视或辅助设备、机器和车间运行的装置”，这主要是从应用上加以定义的。不过，上述定义并不能充分体现出嵌入式系统的精髓。目前，国内一个普遍被认同的定义是：以应用为中心、以计算机为基础，软、硬件可裁减，适应应用系统对功能、可靠性、成本、体积、功耗等严格要求的专用计算机系统[5]。

广义地讲，凡是不用于通用目的的可编程计算机设备，就可以算是嵌入式计算机系统。狭义上而言，嵌入式系统是指以应用为核心，以计算机技术为基础，软硬件可裁剪，适于应用系统对功能、可靠性、成本、体积和功耗严格要求的专用计算机系统。

总的来说，嵌入式系统是一个外延极广的概念，凡是与产品结合在一起的、具有嵌入式系统特点的系统都可以称为嵌入式系统。可以从以下几个方面来理解嵌入式系统的含义：

①嵌入式系统是面向用户、面向产品、面向应用的，必须与具体应用相结合

才会具有生命力。正因为与具体应用的紧密结合，嵌入式系统才具有很强的专用性。

②嵌入式系统将先进的半导体技术、计算机技术和电子技术，以及各个行业的具体应用相结合，是一个技术密集、资金密集、学科交叉和不断创新的知识集成系统。

③嵌入式系统必须根据应用需要对硬件和软件进行裁剪，以满足应用系统对功能、可靠性、成本、体积和功耗的要求。

1.3.1.2 嵌入式系统的组成

通常来说，嵌入式系统可以划分成硬件和软件两部分。嵌入式硬件由嵌入式微处理器、片内周边电路和外部设备三部分组成。嵌入式微处理器是嵌入式硬件系统的核心，直接影响嵌入式产品的应用范围和开发复杂度。典型的嵌入式微处理器有 Motorola 公司的 Power PC 系列、Intel 公司的 Strong Arm 系列、AMD 公司的 X86 系列以及 EPSON 公司的 SIC33 系列等。嵌入式软件一般由连接硬件和应用程序的嵌入式实时操作系统（Real-time operating System，简称 RTOS）和在其上运行的应用软件构成[5]。

① 嵌入式处理器

嵌入式处理器是嵌入式系统的核心部件，是控制、辅助系统运行的硬件单元，其功能和性能影响着整个系统的功能、性能和设计。嵌入式处理器的选择也制约着其配套的外围器件及操作系统的选择[7]。

嵌入式处理器的种类繁多、数量庞大。目前世界上嵌入式处理器的种类数量已经超过 1000 种，流行的体系结构有 30 多个系列。嵌入式微处理器具有体积小、重量轻、成本低、可靠性高等优点，因此得到了广泛应用。目前这样的嵌入式处理器类型有 ARM、MIPS、MC6800、386EX、PowerPC 等系列。其中以 ARM 的应用最为广泛。

嵌入式处理器可以分为四类，即

- 嵌入式微处理器（Embedded Microprocessor Unit，EMPU），如 ARM
- 嵌入式微控制器（Microcontroller Unit，MCU）
- 嵌入式 DSP 处理器（Embedded Digital Signal Processor，EDSP）

- 嵌入式片上系统（System On Chip）

② 嵌入式外围设备

这里所说的嵌入式外围设备，指在一个嵌入式硬件系统中，除了中心控制部件（MCU, DSP, EMPU, SOC）以外的完成存储、通信、保护、调试、显示等辅助功能的其他部件[7]。根据外围设备的功能可分为以下三类，即：

- 存储器类型：静态易失型存储器（RAM，SRAM）、动态存储器（DRAM）、非易失型存储器（ROM，EPROM，EEPROM，FLASH）。其中，因为 FLASH 可以擦写多次，存储速度快，容量大，价格便宜在嵌入式领域应用广泛。
- 接口类型：目前存在的所有接口在嵌入式领域都应有广泛，如 RS-232（串口）、IRDA（红外线接口）、SPI（串行外围设备接口）、I2C（现场总线待定）、USB（通用串行接口）、Ethernet（以太接口）和普通并口。
- 显示类型：CRT，LCD 和触摸屏等外围设备。

1.3.1.3 嵌入式系统的特点

嵌入式系统具有以下几个重要特征[5]：

①系统内核小，由于嵌入式系统一般应用于小型电子装置，系统资源相对有限，所以内核较之传统的操作系统要小得多。比如，ENEA 公司的 OSE 分布式系统，内核只有 5KB，而 Windows 的内核则要大得多。

②专用性强，嵌入式系统的个性化很强，其中的软件系统和硬件结合非常紧密，一般要针对硬件进行系统的移植；同时针对不同的任务，往往需要对系统进行较大的更改。另外，程序的编译下载要和系统相结合。

③嵌入式系统一般没有系统软件和应用软件的明显区分，不要求其功能的设计及实现过于复杂，这样既利于控制系统成本，也利于实现系统安全。

④高实时性的操作系统软件是嵌入式软件的基本要求，而且软件要求固化存储，以提高速度软件代码要求高质量和高可靠性。

⑤嵌入式软件开发要想走向标准化，就必须使用多任务操作系统。嵌入式系统的应用程序可以没有操作系统而直接在芯片上运行，但为了更合理的调度多任务，利用系统资源、系统函数，用户必须自行选配 RTOS 开发平台。

⑥执行的实时性、可靠性，并减少开发时间，保障软件质量。嵌入式系统开发

需要专门的开发工具和环境。由于嵌入式系统本身不具备自主开发能力,即使完成设计后,用户通常也不能对其中的程序功能进行修改,因此必须有一套基于通用计算机的开发工具和环境才能进行开发。

1.3.2 嵌入式 Linux 简述

Linux 是一个类似 UNIX 的操作系统，其代码是完全公开的，内核功能强大，实现简洁。它提供了类似 UNIX 的编程接口和系统调用，可以方便的将 UNIX 系统上的应用程序，移植到 Linux 上运行。Linux 内核支持多种体系结构的处理器，包括目前流行的 Intel x86，Motorola/IBM PowerPC，Compaq Alpha，Sun SRARC 等处理器体系结构[6]。

要把 Linux 用于嵌入式环境，就必须修改 Linux 满足嵌入式系统的要求。主要集中在两个方面：一是体积，而是实时性。与目前上的众多商业的 RTOS（实时操作系统）相比，嵌入式 Linux 拥有以下的特点[6]：

① 完全开放源代码

嵌入式 Linux 开放源代码，这使得学习，修改，剪裁 Linux 成为可能，嵌入式系统的设计者可以对嵌入式 Linux 进行二次开发，去掉操作系统的附加功能，只保留必须的操作系统功能,并且可以根据实际应用的需要优化操作系统的源代码，从而降低整个操作系统的开销与消耗。

② 成本低

GPL 协议保证了源自 Linux 的嵌入式 Linux 也是开放源代码的自由软件，也就是说，只要遵守 GPL 协议，嵌入式 Linux 操作系统的源代码可以自由获得。另外，大多数嵌入式 Linux 使用的开发工具也是遵守 GPL 协议的，同样可以免费获得。

③ 丰富的实用软件支持

Linux 操作系统是一个完整的，功能强大的操作系统，提供了大量的实用程序和各种各样的应用软件。这些软件的正确性和有效性都经过了实际应用检验，可以根据需要，利用 Linux 提供的丰富的软件支持，迅速构建嵌入式应用的软件环境。这样可以极大地减小嵌入式系统软件开发的时间和费用，提高系统的可靠性。

④ 嵌入式 Linux 的可移植性

将 Linux 移植到新的微处理器体系非常快捷，一般是将其移植到一种新型的目标板，其中包含有独特的外设。大部分的内核代码都是相同的，因为它们与微处理器无关，所以，移植的工作多集中在一些存储器管理及中断处理程序上。一旦完成，它们将非常稳定。

⑤ 嵌入式 Linux 的应用

嵌入式系统的涵盖面是非常广泛的，其中，家电市场包括机顶盒、数字电视、可视电话、家庭网络等信息家电；工业市场包括工业控制设备、仪器；商用市场包括掌上电脑、瘦客户机、POS 终端等；通信市场包括 WAP 手机、无线 PDA 等。目前被广泛看好的是信息家电市场，国内有很多开发厂商正加大投入、开发和研制新的产品，嵌入式 Linux 将是他们首选的操作系统。嵌入式 Linux 系统的基本构架如图 1-2 所示：

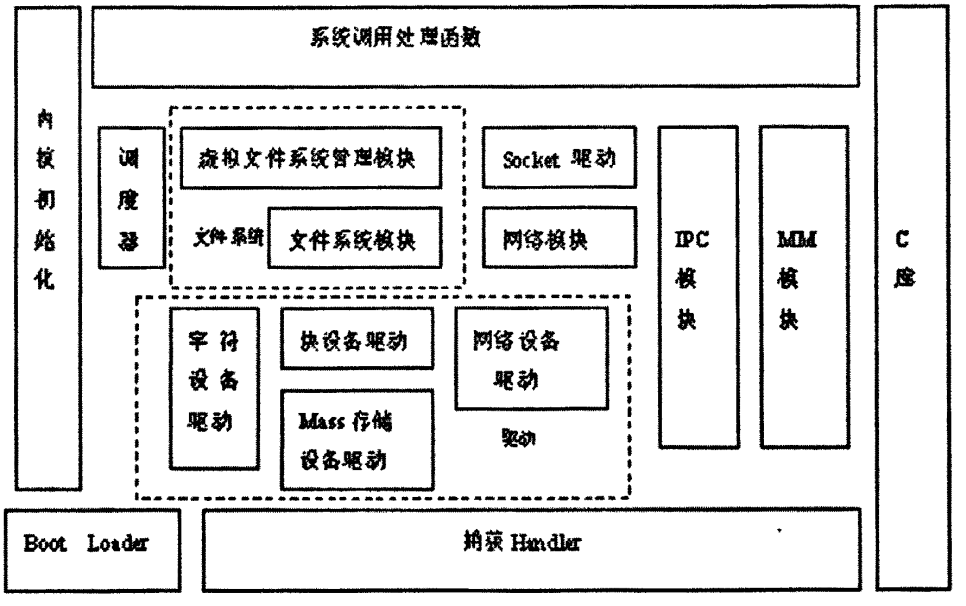

图 1-2 嵌入式系统示意图

Bootloader：负责 Linux 内核的启动，它用于初始化系统资源，包括 SDRAM。这部分代码用于建立 Linux 内核运行环境和从 Flash 中装载初始化 ramdisk。

内核初始化：Linux 内核的入口点是 start_kernel()函数。它初始化内核的其他部分，包括捕获，IRQ 通道，设备驱动，标定延迟循环，最重要的是能够 fork

“init”进程，以启动整个多任务环境。系统调用函数/捕获函数：在执行完“init”程序后，内核对程序流就不再有直接的控制权，此后，它的作用仅仅是处理异步事件（例如硬件中断）和为系统调用提供进程。

设备驱动：设备驱动占据了 Linux 内核很大部分。同其他操作系统一样，设备驱动为它们所控制的硬件设备和操作系统提供接口。

文件系统：Linux 最重要的特性之一就是对多种文件系统的支持。这种特性使得 Linux 很容易地同其他操作系统共存。文件系统的概念使得用户能够查看存储设备上的文件和路径而无须考虑实际物理设备的文件系统类型。Linux 透明的支持许多不同的文件系统，将各种安装的文件和文件系统以一个完整的虚拟文件系统的形式呈现给用户。

1.4 论文的主要内容

随着医疗卫生事业的发展，家庭医疗监护成为一个重要的内容。便携式的医疗仪器由于其体积小、使用方便、价格低等特点容易为大部分家庭所接受，因此，家庭医疗仪器有着巨大的市场空间。

本论文立足于家庭医疗监护，使用压力脉搏传感器和 S3C2410 为核心的嵌入式系统研制了基于 ARM 的心血管参数检测系统。论文主要阐述了基于脉搏波的心血管参数检测的原理和方法；嵌入式系统的开发过程和实现，主要包括开发环境的建立、硬件平台的搭建、软件系统的实现；系统的实验结果和分析等等。

在论文的组织结构如下：

① 第一章是绪论部分，通过查阅大量文献主要介绍心血管参数检测系统的研究意义和实现方法；嵌入式系统的概述；嵌入式 Linux 系统的概述。

② 第二章主要介绍嵌入式平台的搭建。主要阐述嵌入式系统的开发过程；嵌入式 Linux 开发环境的建立；为了调试的方便，系统实现了网络文件系统(NFS)的配置。

③ 第三章主要阐述硬件平台的搭建。主要介绍了硬件设备 S3C2410、压力脉搏传感器、网卡、LCD 和触摸屏。同时阐述了压力脉搏传感器、LCD、触摸屏和网卡与核心处理器 S3C2410 的硬件连接方式。

④ 第四章主要分析了嵌入式系统软件开发流程。主要包括 bootloader 的分

析与移植；嵌入式 Linux 内核的裁减与移植；嵌入式 Linux 设备驱动的开发与移植，主要讲述了网卡驱动、LCD 驱动、A/D 和触摸屏驱动；应用程序的开发，主要包括对采集的脉搏信号进行的处理、脉搏波的检测和心血管参数的计算；嵌入式文件系统的分析、根文件系统的创建与移植。

⑤ 第五章主要介绍了基于 QT 的图形界面开发，包括 QT 图形界面开发概述、触摸屏为 QT 库所支持、QT 在 ARM 板上的移植和基于 QT 的应用程序开发。

⑥ 第六章主要分析了系统的运行结果。

⑦ 第七章主要对所作的工作进行总结并提出几个研究方向。

第二章　嵌入式平台搭建

2.1 嵌入式系统的开发过程

在嵌入式在嵌入式系统开发中，根据用户的应用需求，首先选择相应的嵌入式处理器及外围接口电路来搭建硬件平台，然后选择合适的嵌入式操作系统，在此基础上进行相应的用户应用程序开发，最后是整个系统的调试运行[7]。系统开发流程如图 2-1 所示：

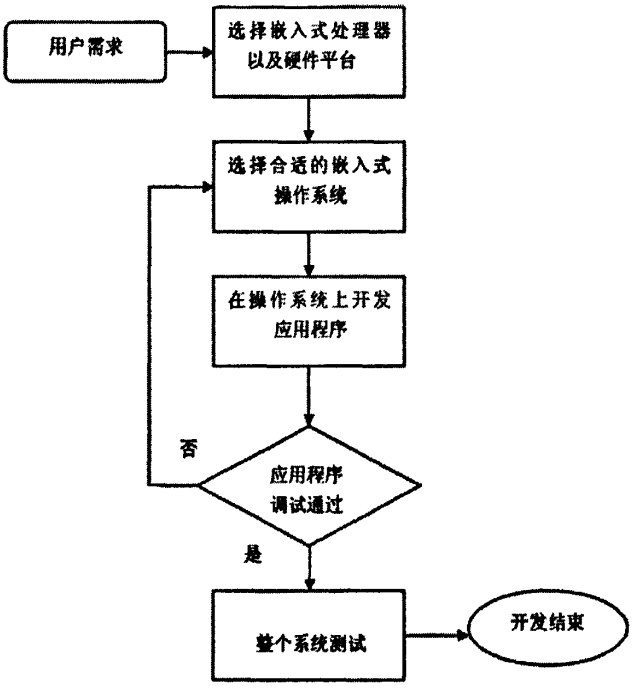

图 2-1 嵌入式系统开发流程图

2.2 嵌入式 Linux 开发环境的建立

建立和安装交叉开发环境通常是在开发嵌入式软件之前要做的第一件事情。因为针对嵌入式程序的编译、连接与通常的 Linux 程序开发不同，在嵌入式系统中，由于其硬件能力所限，不可能再嵌入式系统上安装编译器和连接器，然后再嵌入式系统上编写代码、编译并连接程序。于是就出现了交叉编译器

（cross-compiler）。交叉编译器是指运行在某台功能足够强大的宿主机（host）上，可用来编译某个源程序，然后生成针对特定目标平台（target）的代码的编译器。除了交叉编译器，在开发过程还需要以下几种工具：

①相关二进制工具（连接器、归档工具、符号剥离器等）。

②针对目标平台的C头文件。不同的目标平局具有自身特有的函数库及对应的头文件，这样，在宿主机上，我们就需要保留一份针对该目标平台的头文件，以便用来交叉编译C程序。

③针对目标平台的C函数库。和头文件一样，在宿主机上，我们也要保留一份针对特定目标平台的函数库，以便连接生产最终的目标程序。

以上这些开发嵌入式系统软件所使用的工具通常就被称为交叉开发链或交叉开发环境。目前，开发嵌入式Linux系统首选的开发工具是自由软件基金组织FSF（Free Software Foundation）提供的GNU开发工具。GNU开发工具已经集成到各Linux发行版中，它与Linux内核一脉相承，作为开发Linux内核及应用软件的标准开发工具，因为其效率高、功能强大而被移植到多种平台之上。

跨平台的GNU开发工具链组件包括：二进制工具binutils，编译器gcc，C函数库glibc以及Linux内核头文件。

交叉编译环境的建立步骤如下[8][9][10]：

①下载源代码：下载包括binutils、gcc、glibc以Linux内核的源代码（glibc和内核源代码的版本必须与目标机上实际使用的版本保持一致），并设定shell变量PREFIX指定可执行程序的安装路径。

②编译binutils

运行configure文件，并使用--prefix=$PREFIX参数指定安装路径，使用--target=arm-linux参数指定目标机类型，然后执行make install。

③配置Linux内核文件

首先执行make mrproper进行清理工作，然后执行make config ARCH=arm（或make menuconfig/xconfig ARCH=arm）进行配置，这一步需要根据目标机的实际情况进行详细的配置。

配置完成之后，需要将内核文件拷贝到安装目录：

```
#cp -dR include/asm-arm  $PREFIX/arm-linux/include/asm
```

#cp -dR include/Linux $PREFIX/arm-linux/include/Linux

④第一次编译 gcc

首先运行 configure 文件，使用--prefix=$PREFIX 参数指定安装路径，使用--target=arm-linux 参数指定目标机类型，并使用--disable-threads、--disable-shared、--enable-languages=c 参数，然后执行 make install。这样将生产一个最简单的 gcc。由于编译整个 gcc 是需要目标机的 glibc 库的，它现在还不存在，因此需要首先生产一个最简单的 gcc，它只需要具备编译目标机 glibc 库的能力即可。

⑤交叉编译 glibc

这个步骤生成的代码是针对目标机 CPU 的，因此它属于一个交叉编译过程。该过程要用到 Linux 内核头文件，默认路径为$PREFIX/arm-linux/sys-Linux，因而需要在$PREFIX/arm-linux 中建立一个名为 sys-Linux 的软连接，使其指向内核头文件所在的 include 目录；或者，也可以在接下来要执行的 configure 命令中使用--with-headers 参数指定 Linux 内核头文件的实际路径。

configure 的运行参数设置如下：

CC=arm-linux-gcc

./configure --prefix=$PREFIX/arm-linux --host=arm-linux

--enable-add-ones

最后，按以上配置执行 configure 和 make install，glibc 的交叉编译过程就算完成了，glibc 的安装路径设置为$PREFIXARCH=arm/arm-linux，如果此处设置不当，第二次编译 gcc 时可能找不到 glibc 的头文件和库。

⑥第二次编译 gcc

运行 configure，参数设置为

--prefix=$PREFIX --target=arm-linux --enable-language=c,c++

运行 make install。

到此整个交叉编译环境就完全生成了。

2.3 网络文件系统及其配置

在测试应用程序的时候，如果每次都压缩烧写进 flash，比较麻烦，因此，采用 NFS 文件系统来调试。Linux 内核可以从一个远程 NFS（网络文件系统）服

务器上挂载根文件系统，这样就大大加快了开发速度，避免我们不断地更新目标板上的文件系统内容和不断地重新启动目标板。

NFS 服务器共享的文件系统目录可以被网络中的其他机器以 NFS 客户端的身份挂载到本地目录，然后当作本地文件系统一样来访问和使用。

我们把编译好的程序通过 NFS 服务共享出去，而目标板把该共享目录通过 NFS 挂载到本地目录。当开发时，若重新编译了服务器中的内容，则在目标板上也同时更新了，这样就使得开发和调试变得方便。

NFS 服务的建立过程如下[5][7]：

①服务器配置

以 root 身份登录，编译共享目录的配置文件 exports，指定共享目录及其权限。

#vi /etc/exports

在该文件中添加：

/home/work（共享目录） 192.168.1.*（rw,sync,no_root_squash）

添加的内容表示允许 IP 范围在 192.168.1.*的计算机以读写的权限来访问共享目录/home/work。

rw：读/写权限。

sync：数据同步写入内存和硬盘。

no_root_squash：此参数用来要求服务器允许远程系统以它自己的 root 特权存取该目录。

②服务器启动服务

首先要启动 portmapper（端口映射）服务，这是 NFS 本身要求的。

#/etc/init.d/portmap start

然后启动 NFS Server。此时 NFS 会激活守护进程，然后开始监听客户端的请求。

#/etc/init.d/nfs start

最后还要关闭 iptables 服务。

#/etc/init.d/iptables stop

③目标板的配置

嵌入式 Linux 内核应该支持 NFS 客户端。

在内核配置时，选择如下：

File system→Network File System→

选中 NFS System support 和 Provide NFS vs client support，然后保存退出，重新编译内核，将生成的 zImage 重新下载到目标板。

通过设置启动命令行来实现 NFS 启动。服务器 ip 设置为：192.168.0.120，开发板 ip 设置为：192.168.0.110。则设置启动命令如下：

root=/dev/nfs nfsroot=192.168.0.120:/home/work/root ip=192.168.0.110 console=ttySAC0,115200 devfs=mount init=/linuxrc

第三章 硬件平台设计

3.1 总体设计

检测系统采用以 ARM920T 为内核的 S3C2410 的嵌入式微处理器。脉搏传感器输出的是模拟信号，要经过 A/D 转换，因此把传感器接到 S3C2410 自带的 AD 转换器。由于 S3C2410 自带 LCD 控制器，因此直接把 LCD 连接到 LCD 控制器。因此系统主要有如下模块构成： A/D 转换模块、LCD 模块和触摸屏模块。模块间联系如图 3-1 所示：

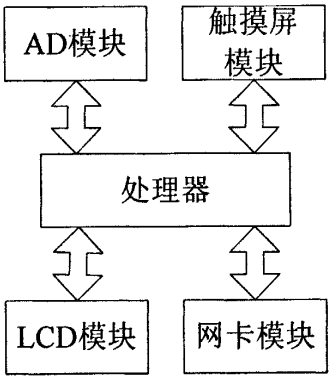

图 3-1 系统模块介绍图

3.2 核心器件选择

系统核心器件包括 S3C2410 处理器、NAND FLASH（64M）、SDRAM（64M）、压力脉搏传感器、LCD 和触摸屏。下面主要介绍 S3C2410 处理器、压力脉搏传感器、LCD 和触摸屏。

3.2.1 S3C2410 处理器介绍

S3C2410 是韩国三星公司的一款基于 ARM920T 内核的 16/32 位的 RISC 嵌入式微处理器，主要优势是性价比高、功耗低、集成度高、运算速度高等。主频可高达 203MHz，被广泛应用在手持设备中；其片内资源丰富，能很好地节省设计者的成本；同时支持睡眠模式，能更好地降低设备功耗[6]。

基于 S3C2410 的以上特点，我们在系统中采用它为核心处理器芯片。

S3C2410 是三星公司推出的支持 32 位 ARM 指令集和 16 位 Thumb 指令集的 32 位嵌入式微处理器，采用 ARM 公司 ARM920T 内核，0.18um 的 CMOS 制作工艺。ARM920T 采用五级整数流水线结构，独立的 16KB 数据 Cache 和 16KB 指令 Cache 实现了对 MMU，AMBA BUS，Harvard 高速缓冲体系结构。S3C2410 还提供了丰富的系统外围设备，为手持设备和一般类型的应用提供了低价格、低功耗、高性能小型微控制器的解决方案[6]。

S3C2410 内部带有全性能的 MMU，支持嵌入式 Linux、WinCE 等操作系统；支持不同类型的引导启动 ROM（NOR/NAND Flash、EEPROM 等），并支持从 NAND Flash 引导系统，芯片内部提供 4 的启动缓存。

S3C2410 提供了丰富的外设接口，片上的资源包括：

- 一个 LCD 控制器（支持 STN 和 TFT 液晶屏）
- SDRAM 控制器
- 三个通道的 UART
- 四个通道的 DMA
- 四个具有 PWM 功能的计时器和一个内部时钟
- 8 通道 10 位 ADC
- 触摸屏接口
- IIS 总线接口
- 2 个 USB 主机接口，1 个 USB 设备接口
- 2 个 SPI 接口
- SD 卡接口和 MMC 卡接口
- 看门狗计数器
- 117 位通用 I/O 口和位外部中断源
- 具有日历功能的实时时钟（RTC）

S3C2410 将系统的存储空间分为 8 组（Bank），每组的大小是 128MB，共 1G。Bank0 到 Bank5 的开始地址是固定的，用于 ROM 或 SRAM。Bank6 和 Bank7 用于 ROM，SRAM 或 SDRAM，这两组可编程且大小相同。Bank7 的开始地址是 Bank6 的结束地址，灵活可变。所有的内存块的访问周期都可编程。S3C2410 采用 Ngcs[7：0]8

个通用片选信号选择这些组。

S3C2410 支持从 NAND FLASH 启动，NAND FLASH 具有容量大，比 NOR FLASH 价格低等特点。系统采用 NAND FLASH 与 SDRAM 组合，可以获得非常高的性价比。S3C2410 具有三种启动方式，可以通过 OM[1:0]管脚进行选择：

①OM[1:0]=00 时处理器从 NAND FLASH 启动

②OM[1:0]=01 时处理器从 16 位宽的 ROM 启动

③OM[1:0]=10 时处理器从 32 位宽的 ROM 启动

图 3-2 是 S3C2410 不同启动模式下的地址空间分配。

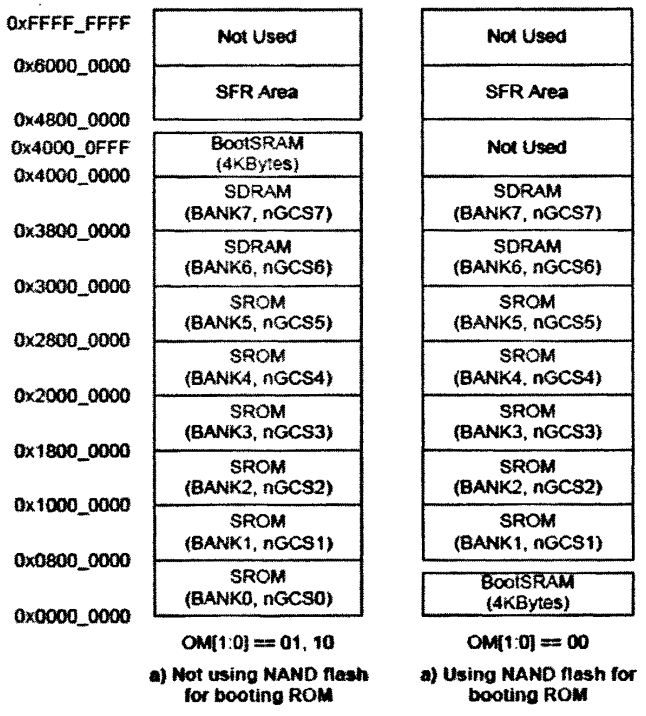

图 3-2 不同启动模式下存储分配图

用户可以将引导代码和操作系统镜像存放在外部的NAND FLASH中，并从NAND FLASH 启动。当处理器在这种启动模式下复位时，内置的 NAND FLASH 将访问控制接口，并经代码自动加载到内部 SRAM（此时该 SRAM 定位于起始地址空间 0x00000000，容量为 4KB）并且运行。之后，SRAM 中的引导程序将操作系统镜像加载到 SDRAM 中，操作系统就能够在 SDRAM 中运行。启动完毕后，4KB 的启动 SRAM 就可以用于其他用途。如果从其它方式启动，启动 ROM 就要定位于内存的启动地

址空间 0x00000000,处理器直接在 ROM 上运行启动程序，而 4KB 启动 SRAM 被定位于内存地址的 0x40000000 处。

本系统采用从 NAND FLASH 启动，设置 OM[1:0]=00。

3.2.2 脉搏传感器简介

脉搏传感器采用的是 HK-2000B 集成脉搏传感器。该脉搏传感器采用高度集成化工艺将立敏元件、灵敏度温度补偿元件、感温元件、信号调理电路集成在传感器内。其主要特点是：灵敏度高、抗干扰性能强、过载能力大、一致性好，性能稳定可靠，使用寿命长[11]。

HK-2000B 集成脉搏传感器的技术指标如下：

①电源电压为：5～6VDC

②压力量程：-50～+300mmHg

③灵敏度：2000μV/mmHg

④精度：0.5%

以上技术指标基本符合系统对脉搏传感器的要求。

3.2.3 LCD 简介

LCD 显示器，即液晶显示器，具有耗电省，体积小等特点，被广泛应用于嵌入式系统中。

在 LCD 显示器中，显示面板薄膜被分为很多小栅格，每个小栅格有一个电极控制，通过改变栅格上的电极就能控制格内液晶分子的排列，从而控制光路的导通。彩色液晶显示通过利用三种原色混合的原理显示不同的色彩：彩色面板中，每个象素都是有三个液晶单元格构成的，其中每个单元格前面都分别有红色，绿色或蓝色的过滤片；光线经过过滤片的处理变成红色，蓝色或者绿色，利用三原色的原理组合出不同的色彩。

常见的 LCD 包括 TN（Twist Nematic）型显示器（扭转向列型显示器，如 TN_LCD,STN_LCD 和 DSTN_LCD）和 TFT（Tin Film Transistor 薄膜晶体管）型显示器。这两种显示器的基本原理比较接近，不同点在于：TN 型显示器通过电极控制液晶分子，而 TFT 型显示器则通过 FET 电子管控制液晶分子。FET 有电容效

应，所以液晶分子能在下一次电极变化前保持原有的排列，因此 TFT 型显示器的颜色数量和刷新速度都优于 TN 型显示器[12]。

系统采用的 LTS350Q1-PE1 作为显示设备。LTS350Q1-PE1 是三星电子公司生产的一款非晶硅有源矩阵 TFT-LCD，它具有功耗低、亮度高和体积小等特点，目前在嵌入式设备中应用非常广泛。

3.2.4 触摸屏简介

触摸屏按其工作原理的不同分为：表面声波屏，电容屏，电阻屏和红外屏。系统采用的是四线电阻式触摸屏。

电阻触摸屏是与显示器表面非常配合的电阻薄膜屏，这是一种多层符合薄膜，它以一层玻璃或硬塑料平板作为基层，表面涂有一层透明氧化金属（ITO 氧化铟，透明的导电电阻）导电层，上面再盖有一层外表面硬化处理、光滑防擦的塑料层、内表面也涂有一层 ITO 涂层、在他们之间有许多细小的（小于 1/1000 英寸）透明隔离点把两层导电层隔开绝缘。触摸屏结构示意图如图 3-3 所示。

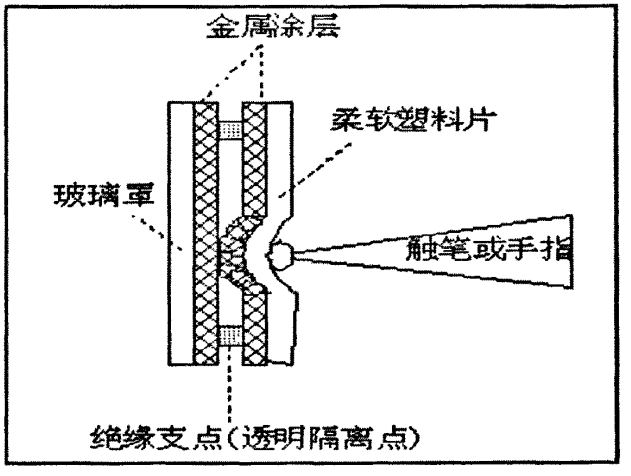

图 3-3 电阻触摸屏结构图

当手指触摸屏幕时，两层导电层在触摸点位置就有了接触，控制器侦测到这一接触并计算出（X，Y）的位置，再根据模拟鼠标的方式运作。

主要特点是：高分辨率，高速传输反应；表面硬度处理，减少擦伤、刮伤及防化学处理；具有光面和雾面处理；一次校正，稳定性高，永不漂移。

四线触摸屏包含两个阻性层。其中一层在屏幕的左右边缘各有一条垂直总

线，另一层在屏幕的底部和顶部各有一条水平总线，如图 3-4 所示。为了在 X 轴方向进行测量，将左侧总线偏置为 0V，右侧总线偏置为 V_{REF}。将顶部或底部总线连接到 ADC，当顶部和底层相接触时即可可作一次测量。

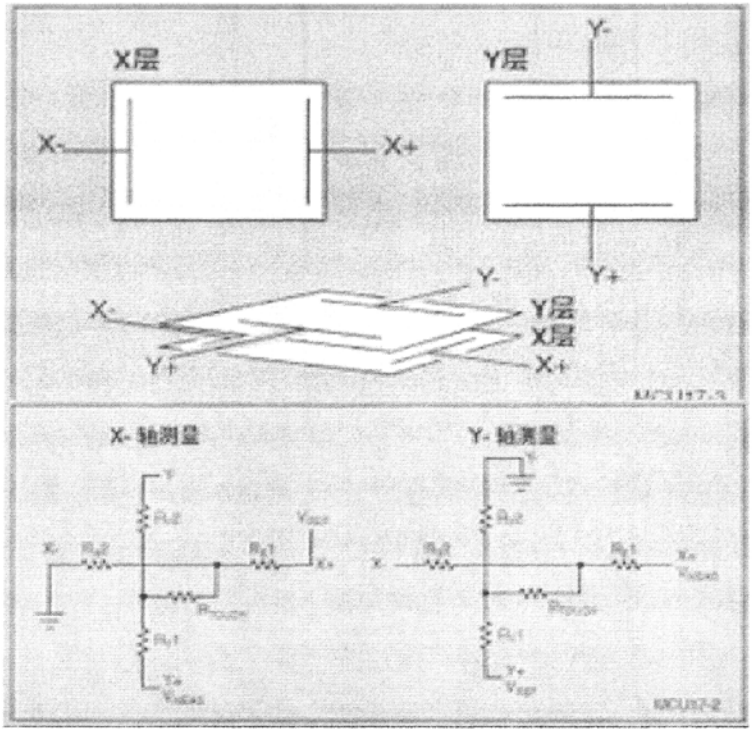

图 3-4 四线电阻触摸屏简化模型图

为了在 Y 轴方向进行测量，将顶部总线偏置为 V_{REF}，底部总线偏置为 0V。将 ADC 输入端接左测总线或右侧总线，当顶层与底层相接触时即可对电压进行测量。图 3-4 显示了四线触摸屏在两层相接触时的简化模型。对于四线触摸屏，最理想的连接方法是将偏置为 V_{REF} 的总线接 ADC 的正参考输入端，并将设置为 0V 的总线接 ADC 的负参考输入端。

3.2.5 网卡简介

系统采用 CS8900A-CQ3 芯片扩展了网络通讯模块，其传输速率工作为 10M。

CS8900A—CQ3 是 100 脚 TQFP 封装，除了具备其他以太网控制芯片所具有的基本功能外，还具有以下优点[13]：

适应工业级温度范围（—40～80℃）；

① 工作电压为 3.3V，功耗低；

② 高度集成，通过使用 CS8900A-CQ3，系统工程可以将一个完整的以太网电路设计在面积不到 $10cm^2$ 的电路板上，适合智能嵌入设备网络接口；

③ 独特的 PacketPage 结构，可自动适应网络通信模式的改变，占用系统资源少，从而增加系统效率；

④ 具有冲突自动重传、自动填充和 CRC 生成可编程发送控制的特点。

CS8900A 芯片主要为嵌入式应用系统、便携式产品和某些适配卡等提供切实可行的以太网解决方案。实践证明，该芯片可靠易用，是实现以太网的良好选择。

CS8900A 是 Cirrus 公司生产的一种高集成度的全面支持 IEEE802.3 标准的以太网控制器。CS8900A 支持 8 位、16 位的微处理器，可以工作在 I/O 方式或 Memory 方式。片内集成了 ISA 总线接口，可以直接和有 ISA 总线的微处理器系统无缝连接。片内集成了 4KB 容量的 PacketPage 结构的 RAM，这 4KB 存储器映像结构的 RAM 包括片内各种控制、状态、命令寄存器，以及片内发送、接收缓存。用户可以以 I/O 方式、Memory 方式或 DMA 方式访问它们[14]。之所以选择 CS8900A，是因为 Cirrus 提供了 CS8900A 的基于各种操作系统的驱动程序源代码，这就为开发带来了方便。

3.3 系统各模块介绍

3.3.1 系统总体模块介绍

系统在以 S3C2410 为核心处理器的基础上，主要分为四个模块：A/D 模块，触摸屏模块、LCD 模块和网络模块。

系统各个模块如图 3-5 所示：

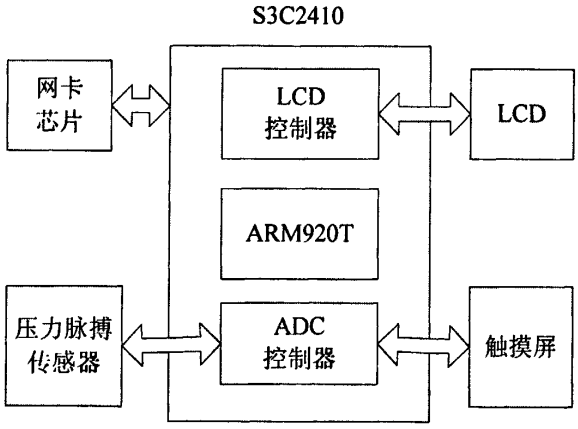

图 3-5 系统模块图

3.3.2 A/D 模块介绍

A/D 模块主要由 S3C2410 的 ADC 控制器和压力脉搏传感器组成。

系统采用的是 HK-2000B 集成脉搏传感器，其压力量程为-50～+300mmHg，灵敏度为 2000μV/mmHg，正常的脉压范围一般小于 200mmHg，因此输出的电压一般小于 0.4V。由于脉搏波的频率较低，可以不加前置低通滤波器，而直接采用软件滤波。

S3C2410 集成的 10 位 A/D 转换器输入电压范围为 0～3.3V，转换速率可以高达 500KSPS，符合要求。由于时间关系，为简单起见，系统采用直接将压力传感器接到系统的 A/D 第六个输入通道（AIN[6]），但是最好添加一个前置放大器来提高输入电压。

3.3.3 触摸屏模块介绍

触摸屏模块主要包括 S3C2410 的 ADC 控制器、4 线电阻式触摸屏和触摸屏滤波电路。S3C2410 接 4 线电阻式触摸屏的电路原理如图 3-6 所示。整个触摸屏由横向电阻线和纵向电阻线组成，由 nYPON、YMON、nXPON、XMON 四个控制信号控制 4 个 MOS 管的通断。其中，通道 7（AIN[7]）作为触摸屏接口的 X 坐标输入，通道 5（AIN[5]）作为触摸屏接口的 Y 坐标输入。在接入 S3C2410 触摸屏接口前，它们都通过一个阻容式低通滤波器滤除坐标信号噪声。因为如果传递给 S3C2410 模拟输入接口的信号中干扰过大，不利于后续的软件处理。

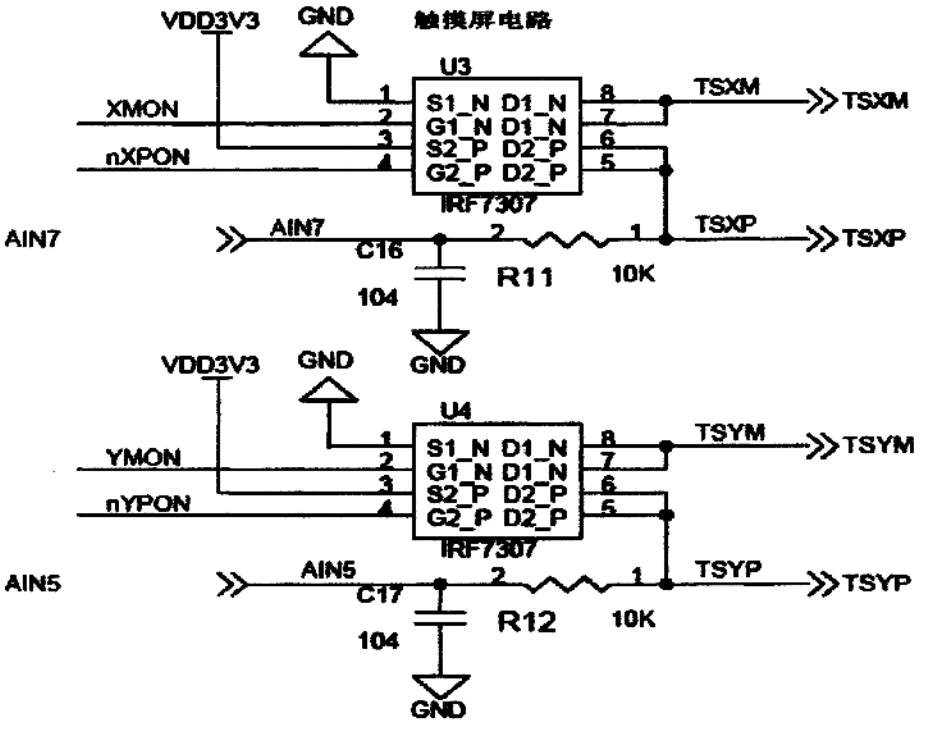

图 3-6 触摸屏接口电路图

3.3.4 LCD 模块介绍

LCD 模块主要包括 S3C2410 的 LCD 控制器、LCD（LTS350Q1-PE1）和硬件驱动电路。驱动电路将 S3C2410 的 LCD 控制器和 TFT-LCD 结合起来构成嵌入式液晶显示系统[15]。LCD 系统结构框图如图 3-7 所示：

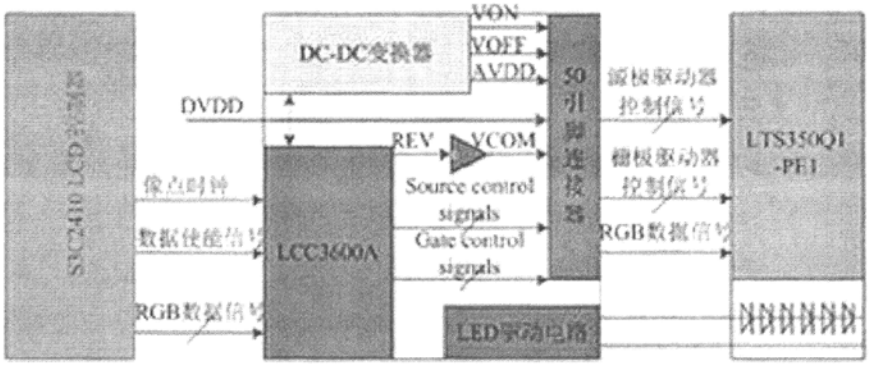

图 3-7 LCD 硬件系统结构框图

TFT-LCD 驱动电路的主体部分由多路电压源、能够给出正确数字逻辑信号的电路以及为了看到显示画面而设计的背光驱动电路构成，另外，不同的液晶显示器因为内部电路的差别还需要一些不同的外围附属电路。

3.3.5 网卡模块介绍

网卡本模块主要包括 CS8900A-CQ3 芯片、S3C2410 和 CS8900 接口电路。电路原理图如图 3-8 所示：

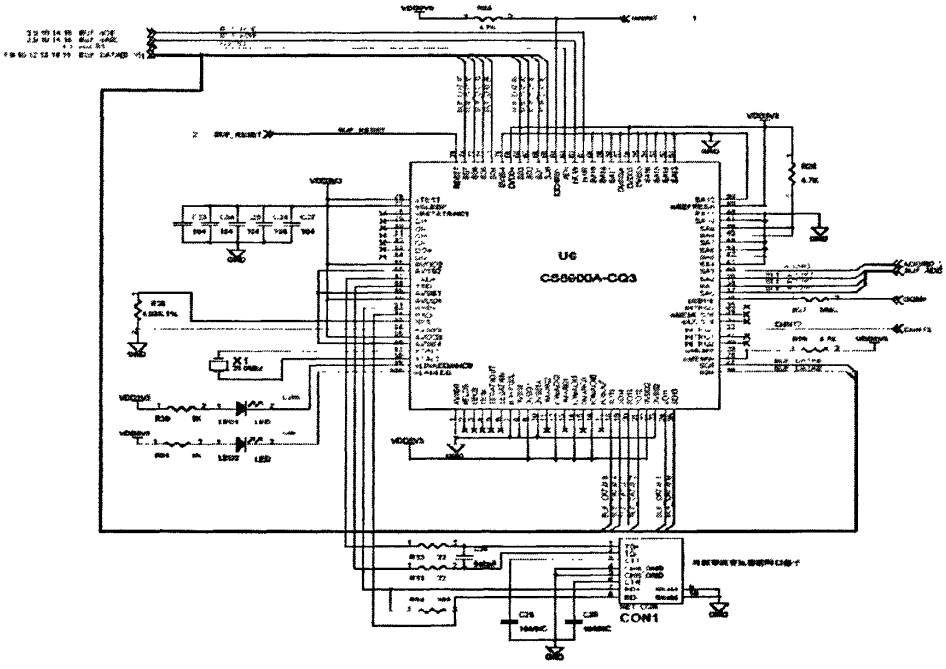

图 3-8 CS8900 电路原理图

CS8900 占用 CPU 的片选线 nGCS3。在电路接口中，CS8900 工作在 16 位模式下，网卡芯片复位默认工作方式为 I/O 连接，寄存器的默认基址是 300H。

第四章 软件设计

4.1 软件设计思路

一个嵌入式 Linux 系统从软件角度看通常可以分为四个层次[16]：

① 引导加载程序：包括固化在固件中的 boot 代码和 bootloader 两大部分；

② Linux 内核：特定于嵌入式板子的内核，以及控制内核引导的系统参数；

③ 用户应用程序。特定于用户的应用程序，有时在用户应用程序和内核层之间可能还会包括一个嵌入式图形界面用户；

④ 文件系统：包括根文件系统和建立于 Flash 内存设备之上文件系统。通常用 RAMDISK 做根文件系统，它是提供管理系统的各种配置文件以及系统执行用户应用程序运行环境和载体。

图 4-1 是一个同时装有 Bootloader、内核启动参数、内核映像、根文件系统映像和应用程序典型结构示意图。

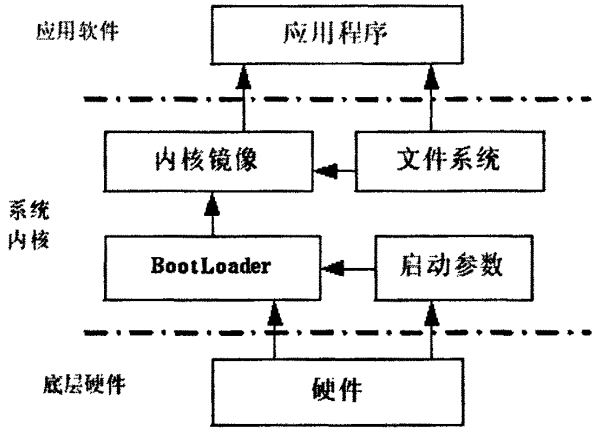

图 4-1 Linux 系统软件逻辑图

下面从以上四个部分进行详细分析。

4.2 bootloader 的移植

4.2.1 bootloader 简介

bootloader 是在操作系统内核运行之前运行的一段小程序。通过这段小程序来初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境。

由于 bootloader 的实现依赖于 CPU 的体系结构，bootloader 的启动大多数都分为两个阶段。

第一阶段（stage1）主要包含依赖于 CPU 的体系结构硬件的初始化的代码，通常都用汇编语言来实现。这个阶段的主要任务有[17]：

①基本的硬件初始化

这是 bootloader 一开始就执行的操作，其目的是为 stage2 的执行以及随后的 kernel 的执行准备好一些基本的硬件环境。

②为第二阶段（stage2）准备 RAM 空间

为了获得更快的执行速度，通常把 stage2 加载到 RAM 空间中来执行，因此必须为加载 bootloader 的 stage2 准备好一段可用的 RAM 空间范围。由于 stage2 通常是 C 语言执行代码，因此在考虑空间大小时，除了 stage2 可执行镜像的大小外，还必须把堆栈空间也考虑进来。此外，空间大小最好是 memory page 大小(通常是 4KB)的倍数。一般而言，1M 的 RAM 空间已经足够了，具体的地址范围可以任意安排。假设我们的内存空间分布如图 4-2 所示：

在上述一切都就绪后，就可以跳转到 bootloader 的 stage2 去执行了。比如，在 ARM 系统中，这可以通过修改 PC 寄存器为合适的地址来实现。

第二阶段通常由 C 语言完成，以便实现更复杂的功能，也使程序有更好的可读性和可移植性。这个阶段的主要任务有[15]：

①初始化本阶段所要用到的硬件设备

这通常包括：第一，初始化至少一个串口，以便和终端用户进行 I/O 输出信息；第二，初始化计时器等。在初始化这些设备之前，也可以重新把 LED 灯点亮，以表明我们已经进入 main()函数执行。设备初始化完成后，可以输出一些打印信息，程序名字字符串、版本号等。

②检测系统内存映射

所谓内存映射就是指在整个 4GB 物理地址空间中有哪些地址范围被分配用来寻址系统的 RAM 单元。比如，在 SA-1100 CPU 中，从 0XC000,0000 开始的 512M 地址空间被用作系统的 RAM 地址空间，而在 Samsung S3C44BOX CPU 中，从 0X0c00,0000 到 0x1000,0000 之间的 64M 地址空间被用作系统的 RAM 地址空间。

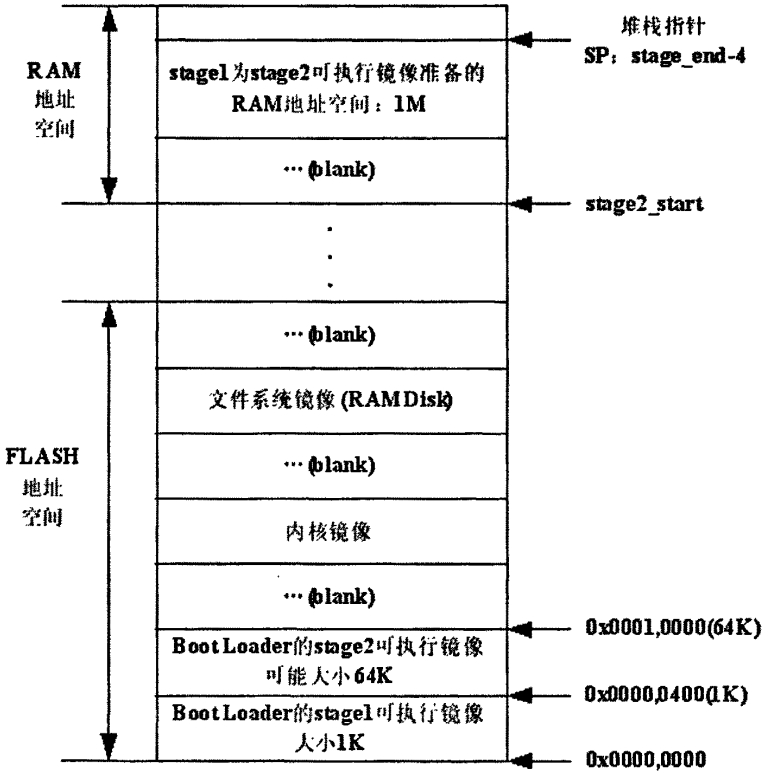

图 4-2 内存空间分配图

虽然 CPU 通常预留出一大段足够的地址空间给系统 RAM，但是在搭建具体的嵌入式系统时却不一定会实现 CPU 预留的全部 RAM 地址空间。也就是说，具体的嵌入式系统往往只把 CPU 预留的全部 RAM 地址空间中的一部分映射到 RAM 单元上，而让剩下的那部分预留 RAM 地址空间处于未使用状态。由于上述这个事实，因此 Boot Loader 的 stage2 必须在它想任何事(比如，将存储在 FLASH 上的内核镜像读到 RAM 空间中)之前检测整个系统的内存映射情况，也即它必须知道 CPU 预留的全部 RAM 地址空间中的哪些被真正映射到 RAM 地址单元，哪些是处于“unused”

状态的。

③将内核镜像和根文件系统从 Flash 读到 RAM 中

④为内核设置启动参数

应该说，在将内核镜像和根文件系统镜像拷贝到 RAM 空间中后，就可以准备启动 Linux 内核了。但是在调用内核之前，应该作一步准备工作，即：设置 Linux 内核的启动参数。在嵌入式 Linux 系统中，通常需要由 bootlader 设置的常见启动参数有：

ATAG_ CORE, ATAG_MEM, ATAG_CMDLINE, ATAG_RAMDISK, ATAG_INITRD 等。

⑤调用内核

bootloader 调用 Linux 内核的方法是直接跳转到内核的第一条指令处，也即直接跳转到 MEM_START+0X8000 地址处。

4.2.2 bootloader 的烧写

系统根据开发板商提供的 sjf2410 工具使用 JTAG 来将 BIOS 烧写到 NAND FLASH。烧写步骤如下：

①用 20 针的排线将 20 芯 JTAG 接口与 JTAG 小板的 JP3 接口相连。安装 giveio 驱动。然后点击 SJF2410_BIOS.BAT 批处理文件。烧写程序所支持的 FLASH 都会列出来了，有 K9S1208（NAND，64M）、28F128J3A、AM29LV800、SST39VF160 等。

②由于系统采用的是 K9S1208（NAND，64M）内存。因此选择 K9S1208。然后选择开始烧写程序。烧写结束后程序会自动退出。

③关闭电源，拔掉 JTAG 插头，将核心板的 JP1 的短路帽接上，将 PC 的串口和开发板的串口 0 通过串口线连接好，在 PC 上启动 DNW 程序，并通过 configure 选项设置好 PC 的串口和波特率。如图 4-3 所示。

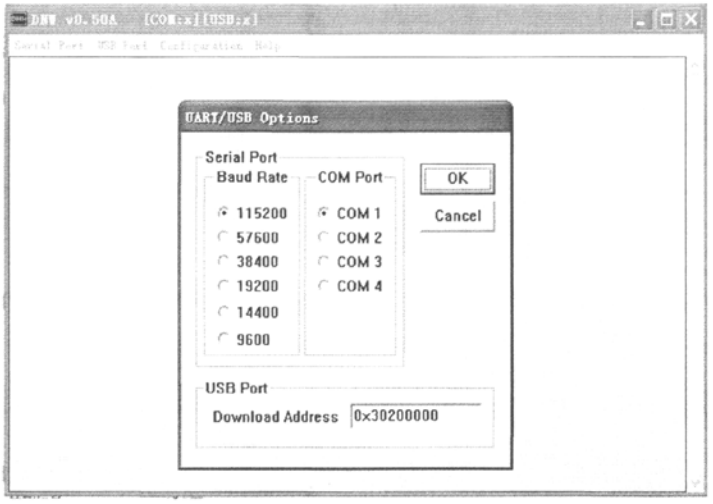

图 4-3 DNW 设置图

④打开开发板电源，烧进 NAND FLASH 的程序 BIOS 会启动运行。这样就将一个 BIOS 烧写到 NAND FLASH 中。

4.2.3 bootloader 的使用

系统采用的是深圳优龙科技有限公司提供的 bootloader。通过烧写 bootloader 可以实现图 4-4 所示的功能。

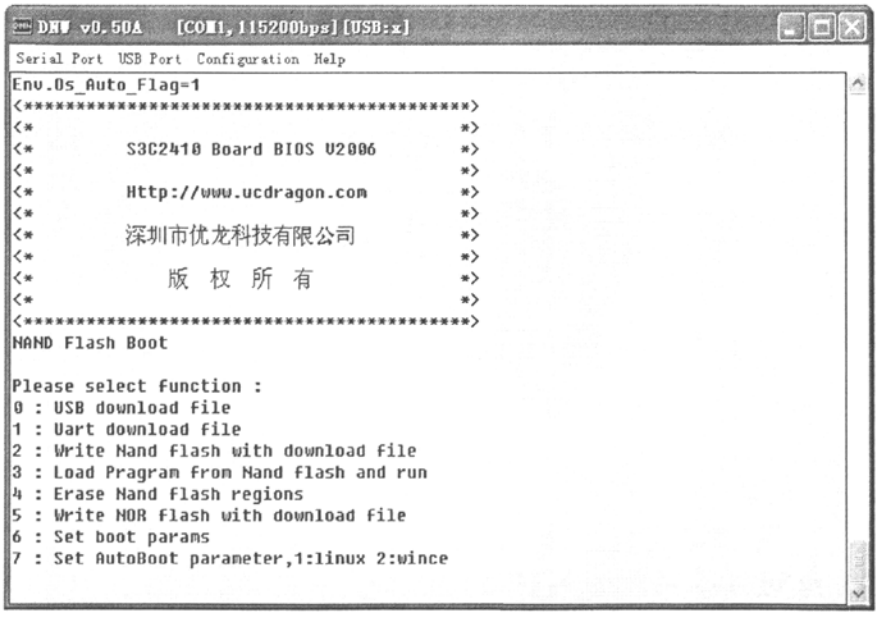

图 4-4 bootload 启动显示图

在烧写了 bootloader 之后，使用 bootloader 的功能 0 来下载内核映象。如图 4-5 所示。

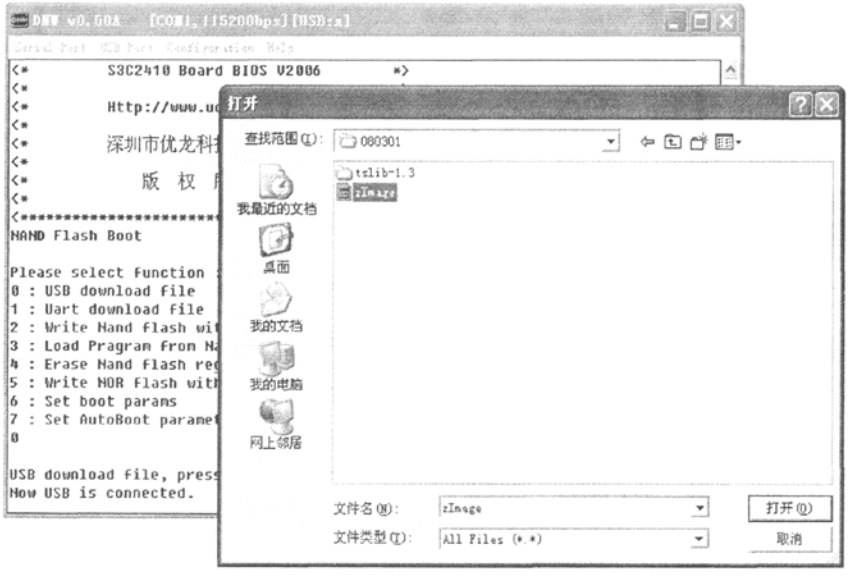

图 4-5 bootloader 下载内核映像图

可以使用功能 2 来烧写 bootloader，内核映象和文件系统。FLASH 烧写如图 4-6 所示。

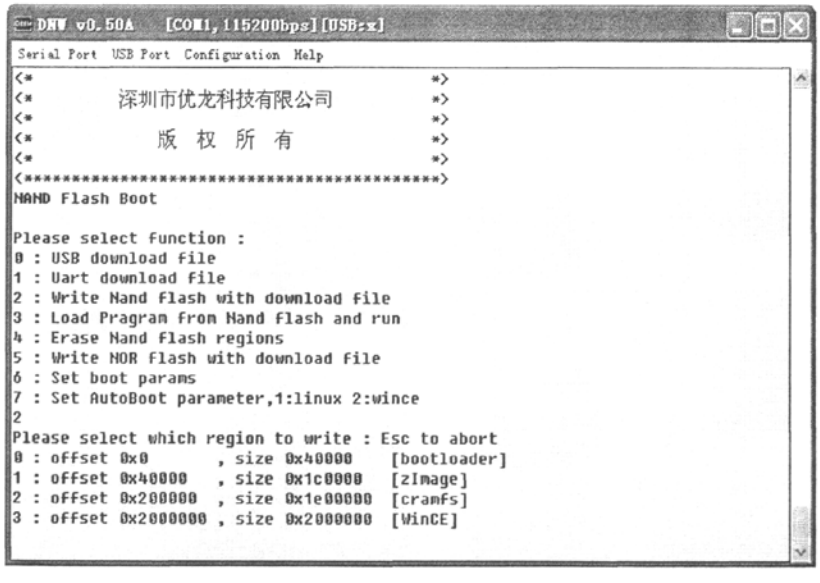

图 4-6 烧写 FLASH 示意图

使用功能 6 可以设置启动参数。本系统设置的启动参数（使用 NFS 情况时）如下：

root=/dev/nfs rw nfsroot=192.168.0.120:/home/root

ip=192.168.0.110 console=ttySAC0,115200 devfs=mount init=/linuxrc

4.3 Linux 操作系统的移植

由于 2.6 版本的 Linux 内核支持 S3C2410，提供了很多新的扩展和功能，系统选择采用 2.6.14 版本的内核。2.6 内核针对嵌入式系统做了如下改进。

①速度更快

改进的同步多线程技术提高了任务调度器对虚拟处理器和实际处理器之间的负载均衡优化能力。同时新的任务调度器优化了诸如 I/O 设备、多 CPU 和内存等相关系统资源，可以在 16 个以上的 CPU 间进行无缝的进程切换。

②系统更大

2.6 内核支持更多的设备。2.4 内核支持个 255 主设备和 255 个子设备，而新的 2.6 内核支持 4095 个主设备和上千万个子设备。

③性能更优

2.6 内核的 NFS（网络文件系统）在速度和安全性上有了很大改善。可支持 NFSv4 协议，使用新的密码方式提供更安全的授权，令 NFS 服务器同时最多可支持用户的数目是 2.4 内核的 64 倍。2.6 内核中的 EXT2 文件系统提供更安全的数据存储和更快捷的数据恢复。

④多媒体功能

2.6 内核包含了 ALSA（高级 Linux 声音结构）。ALSA 完全基于线程,并且是 SMP 可靠的。使得 Linux 具有更好的多声卡，硬件混音能力。同时也增强了游戏手柄支持。

⑤无线设备

2.6 内核将无线设备的各个子系统合并成一个集中式 API 无线接口，这样使得无线工具可以轻松的应用于所有已支持的无线设备。同时，2.6 内核包含蓝牙技术所需协议，可以对蓝牙设备进行更好的支持。

⑥电池保护

2.6 内核支持新处理器所具有的根据电源状况调节运行频率的能力。

⑦设备支持

2.6 内核采用的统一的设备模型集中控制了系统资源，对热插拔、PC 卡、USB 和火线设备有了更好的支持，是真正的即插即用操作系统，并且可以在系统的 BIOS 中进行配置。2.6 内核支持 USB2.0，并且基于 2.6 内核的系统可以同时成为 USB 主机和 USB 设备。这样可以方便的连接和同步诸如 PDA 的手持设备。

⑧安全性

2.6 内核支持 Ipsec 网络协议（例如 IPV6），这样就提供了网络协议层的密码保护。

⑨嵌入式应用

支持 μCLinux。支持更多没有 MMU 的处理器，这些处理器广泛应用于 PDA 等设备上。允许新内核能方便的运用于嵌入式设备和消费电子类设备上。

4.3.1 设置参数

修改内核的 Makefile 文件。包括指定交叉编译器和设置环境变量。

#vi Makefile

修改如下：ARCH =arm

CROSS_COMPILE = arm-linux-

#export PATH=交叉编译器所在目录:$PATH

4.3.2 修改内核

内核启动部分的修改主要包括加上对 NAND FLASH 的支持和 MTD 分区表的填写。目的是在内核启动时能初始化 NAND FLASH。具体修改如下[17][18][19]：

①将分区信息加入内核

根据 bootloader 的 NAND Flash 的配置状况来修改内核分区信息

#vi $KERNEL/arch/arm/mach-s3c2410/devs.c

添加头文件：

```
#include <linux/mtd/partitions.h>
#include <linux/mtd/nand.h>
#include <asm/arch/nand.h>
```

添加分区结构：

```
static struct mtd_partition partition_info[]={
{
    name: "bootloader",        //分区名字，任意
    size: 0x40000,             //分区大小
    offset: 0x0,               //分区的起始地址，相对于 0x0 的偏移
},{
    name: "kernel",
    size: 0x1c0000,
    offset: 0x40000,
},{
    name: "root",
    size: 0x1e00000,
    offset: 0x200000,
},{
    name: "user",
    size: 0x2000000,
    offset: 0x2000000,
}
};
struct s3c2410_nand_set nandset={
    nr_partitions: 4,              //分区数量
    partitions: partition_info,    //分区表
};
struct s3c2410_platform_nand superlpplatform={
     //NAND Flash 芯片支持
```

```
        //参数意义可参考 S3C2410 手册
        tacls: 0,
        twrph0: 30,
        twrph1: 0,
        sets: &nandset,
        nr_sets: 1,
};
```

修改 s3c_device_nand，以加入 NAND Flash 驱动：

```
struct platform_device s3c_device_nand = {
  .name = "s3c2410-nand", //设备名称
  .id = -1,                    //有效设备编号，-1 表示唯一设备
  .num_resources = ARRAY_SIZE(s3c_nand_resource),
                               //寄存器区的个数
  .resource = s3c_nand_resource,
                               //寄存器区的首地址
                               //添加如下信息，表示 NAND Flash 设备
 .dev={
        .platform_data=&superlpplatform
       }
};
```

②指定启动时的设备初始化

#vi $KERNEL/arch/arm/mach-s3c2410/mach-smdk2410.c

修改如下：

```
static struct platform_device *smdk2410_devices[] __initdata = {
  s3c_device_usb,
  s3c_device_lcd,
  s3c_device_wdt,
  s3c_device_i2c,
  s3c_device_iis,
```

```
    //添加如下信息
    s3c_device_nand,
};
```

③禁止 Flash Ecc 效验

内核通过 bootloader 把数据写入 NAND Flash，而 bootloader 的 ECC 效验算法和内核不同，内核的校验码是由 NAND Flash 控制器产生的，所以在此必须禁用 NAND Flash ECC。

#vi $KERNEL/drivers/mtd/nand/s3c2410.c

找到 s3c2410_nand_init_chip 函数，将 chip->eccmode 的值修改为：NAND_ECC_NONE

4.3.3 内核配置与编译

- 内核配置

  #make menuconfig

  选中与 S3C2410 相关的配置。

- 内核编译

  #make zImage

  在$KERNEL/arch/arm/boot 目录下就可以找到内核映象 zImage。

4.4 驱动程序的实现

4.4.1 Linux 设备驱动概述

Linux 操作系统将所有的设备（而不仅是存储器里的文件）全部都看成文件，都纳入文件系统的范畴，都通过文件的操作界面进行操作。

每一个设备都至少由文件系统的一个文件代表，因而都有一个“文件名”。每个这样的“设备文件”都唯一地确定了系统中的一项设备。应用程序通过设备的文件寻找访问具体的设备，而设备则象普通文件一样受到文件系统访问权限控制机制的保护。

应用程序通常可以通过系统调用 open()“打开”这个设备文件，建立起与目标设备的连接[20]。代表着该设备的文件节点中记载着建立这种连接所需的信息。对于执行该应用程序的进程而言，建立起的连接就表现为一个已经打开的文件。打开了代表着目标设备的文件，即建立起与设备的连接后，就可以通过 read()、write()、ioctl()等常规的文件操作对目标设备进行操作。

驱动程序需要完成的工作有：初始化设备、管理设备、提供文件读写操作的接口、处理设备出现的错误等[21]。

系统加载驱动有两种方式：一是把驱动程序直接编译进内核；二是使用模块加载的方式。第一种是写好驱动程序之后，放到内核代码的目录里，修改 Makefile 文件，与内核一同编译。第二种方式，可以给调试带来很大的方便。

Linux 把设备分为 3 种基本设备类型：字符设备、块设备和网络设备[22][23]。

①字符设备

一个字符设备是一种可以当作一个字节流来存取的设备（ 如同一个文件 ）；一个字符驱动负责实现这种行为。这样的驱动常常至少实现 open，close，read 和 write 系统调用。字符设备通过文件系统结点来存取，例如 /dev/tty1 和 /dev/lp0。在一个字符设备和一个普通文件之间唯一有关的不同就是，可以在普通文件中移来移去，但是大部分字符设备仅仅是数据通道，只能顺序存取。然而，存在看起来象数据区的字符设备，可以在里面移来移去。例如，frame grabber 经常这样，应用程序可以使用 mmap 或者 lseek 存取整个要求的图像。

②块设备

如同字符设备，块设备通过位于 /dev 目录的文件系统结点来存取。一个块设备(例如一个磁盘)应该是可以驻有一个文件系统的。在大部分的 Unix 系统，一个块设备只能处理这样的 I/O 操作，传送一个或多个长度经常是 512 字节(或一个更大的 2 的幂的数)的整块。Linux，相反，允许应用程序读写一个块设备象一个字符设备一样，即允许一次传送任意数目的字节。结果就是，块和字符设备的区别仅仅在内核在内部管理数据的方式上，并且因此在内核/驱动的软件接口上不同。如同一个字符设备，每个块设备都通过一个文件系统结点被存取的，它们之间的区别对用户是透明的。块驱动和字符驱动相比，与内核的接口完全不同。

③网络接口

任何网络事务都通过一个接口来进行，就是说，一个能够与其他主机交换数据的设备。通常，一个接口是一个硬件设备，但是它也可能是一个纯粹的软件设备，比如环回接口。一个网络接口负责发送和接收数据报文，在内核网络子系统的驱动下，不必知道单个事务是如何映射到实际的被发送的报文上的。很多网络连接(特别那些使用 TCP 的)是面向流的，但是网络设备却常常设计成处理报文的发送和接收。一个网络驱动对单个连接一无所知， 它只处理报文。

本系统主要涉及字符设备驱动。

4.4.2 字符设备驱动

4.4.2.1 驱动程序的加载

Linux 驱动程序分为两种形式：一种是直接编译进内核，另一种是编译成 module 形式，然后在需要该驱动 module 时手动加载。对于 Module 形式的驱动，Linux 提供了两个命令用来加载：modprobe 和 insmod[24]。

使用 modprobe 会自动加载驱动 module，不过，使用 modprobe 时，必须把要加载的驱动 module 放在当前模块搜索路径中。而 insmod 命令不会考虑驱动 module 的依赖性，可以加载任意目录下的驱动 module。一般来说，在驱动开发阶段，使用/sbin/insmod 比较方便，因为不必将 module 放入当前 module 的搜索路径中。

一旦 insmod 加载模块， Linux 内核就会调用 module_init(devicename_ init_module)特殊宏，其中 devicename_init_module 是驱动初始化函数，devicename 一般是设备名称。

驱动程序 module 被加载后，若对设备进行操作(如 open，read，write 等)，驱动 module 就会调用相应的函数响应该操作。Linux 内核有两种方式访问设备：其一是通过某些设备的 ID（比如 PCI 设备和 USB 设备的 Device ID 和 Product ID），Linux 内核根据这些 ID 调用驱动 module；其二是在/dev 目录下根据设备的主次设备号创建对应的设备节点（即设备文件），这样当操作 /dev 目录下的设备文件时，就会调用相应的驱动 module。

4.4.2.2 驱动初始化

驱动模块的初始化包括以下几个方面[25]：

①分配并注册主设备号和次设备号

设备号是在驱动 module 中分配并注册的，/dev 目录下的设备文件是根据这个设备号创建的，因此，当访问/dev 目录下的设备文件时，驱动 module 就知道自己被调用了。

在 Linux 内核看来，主设备号标识设备对应的驱动程序，告诉 Linux 内核使用哪一个驱动程序来为该设备(也就是/dev 下的设备文件)服务；而次设备号则用来标识具体且唯一的某个设备。

在内核中，用 dev_t 类型(其实就是一个 32 位的无符号整数)的变量来保存设备的主次设备号，其中高 12 位表示主设备号，低 20 位表示次设备号。

设备获得主次设备号有两种方式：一种是手动给定一个 32 位数，并将它与设备联系起来(即用某个函数注册)；另一种是调用系统函数给设备动态分配一个主次设备号。

对于手动给定一个主次设备号，使用下面函数：

```
int register_chrdev_region(dev_t first, unsigned int count, char *name)
```

其中 first 是我们手动给定的设备号，count 是所请求的连续设备号的个数，而 name 是和该设备号范围关联的设备名称，它将出现在/proc/devices 和 sysfs 中。

对于动态分配设备号，使用下面函数：

```
int alloc_chrdev_region(dev_t *dev, unsigned int firstminor, unsigned int count, char *name)
```

该函数需要传递给它指定的第一个次设备号 firstminor（一般为 0）和要分配的设备数 count，以及设备名，调用该函数后自动分配得到的设备号保存在 dev 中。

与主次设备号相关的 3 个宏：

MAJOR(dev_t dev)：根据设备号 dev 获得主设备号；

MINOR(dev_t dev)：根据设备号 dev 获得次设备号；

MKDEV(int major, int minor)：根据主设备号 major 和次设备号 minor 构建设备号。

②初始化代表设备的 struct 结构体

如果设备设置了自己的数据结构体，那么在 module 初始化函数中，可以初始化一些设备相关的变量。

③初始化在内核中代表设备的 cdev 结构体

在 Linux 内核中，cdev 结构体才是真正代表了某个设备。在内核调用设备的 open，read 等操作之前，必须先分配并注册一个或者多个 cdev 结构。

4.4.2.3 设备操作

在内核内部，I/O 设备的存/取通过一组固定的入口点来进行，这组入口点是由每个设备的设备驱动程序提供的。Linux 系统，设备驱动程序所提供的这组入口点由一个文件操作结构来向系统进行说明，即 file_operations 结构[25]。

file_operations 结构中的成员全部是函数指针，所以实质上就是函数跳转表。每个进程对设备的操作，都会根据 major、minor 设备号，转换成对 file_operations 结构的访问。

常用的操作包括以下几种：

open：打开设备文件，进行必要的初始化和填写相关的数据结构。

lseek：移动文件指针的位置，只能用于可以随机存取的设备。

read：进行读操作，并通过 copy_to_user 函数把要传送的数据 copy 到用户空间。

write：进行写操作，与 read 类似。

select：进行选择操作。如果驱动程序没有提供 select 入口，select 操作将会认为已经准备好进行任何的 I/O 操作。

ioctl：进行读、写以外的其他操作，参数 cmd 为自定义的命令。

4.4.2.4 卸载驱动

每个重要的模块都需要一个清除函数，该函数在模块被移除前注销接口并向系统返回所有资源。如果一个模块未定义清除函数，则内核不允许卸载该模块[25]。

Linux 驱动 module 的卸载可以用/sbin/rmmod devicename.ko 命令，这时 Linux 内核就会调用驱动程序中用 module_exit(devicename_cleanup_module) 特殊宏定义的清除函数，也就是说，module_exit 声明用来帮助 Linux 内核找到模块的清除函数。

模块的清除函数需要撤销初始化函数注册的所有资源，并且习惯上(但不是必须的)以与初始化函数注册相反的顺序进行撤销。初始化函数是指用 module_init 宏声明的初始化函数，而不是指 open 函数，与 open 函数对应的应当是 release 函数。

清除函数主要做两件事：一是 free 了所有为设备分配的内存；二是收回了初始化函数所注册的设备号。

4.4.3 网卡驱动

系统采用 NFS 进行应用程序的调试，所以要实现网卡的驱动。网卡驱动的体系结构如下图 4-7 所示：

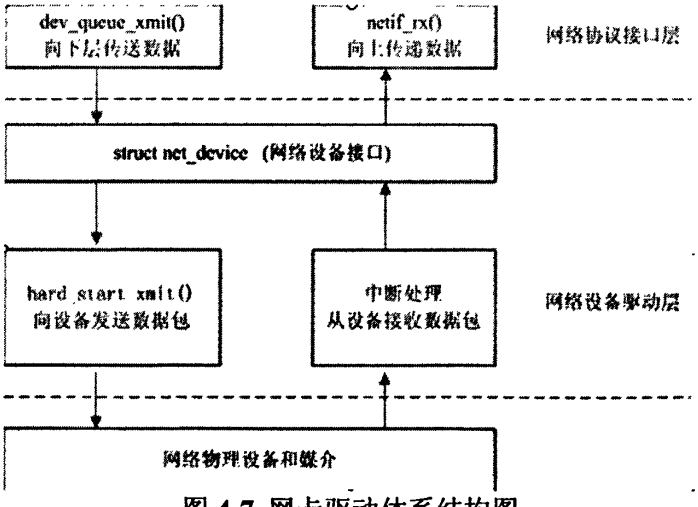

图 4-7 网卡驱动体系结构图

4.4.3.1 网卡芯片简介

系统使用 CS8900A-CQ3 芯片扩展网络通讯模块。CS8900 占用 CPU 的片选线 nGCS3，CS8900 寄存器的基址空间为 0X6000000+300H。

CS8900 相关的几个寄存器如下[6]：

- LINECTL（0112H）

  LINECTL 决定 CS8900 的基本配置和物理接口。在本系统中，设置初始值为 00d3H，选择物理接口为 10BASE-T，并使能设备的发送和接收控制位。

- RXCTL（0104H）

  RXCTL 控制 CS8900 接收特定数据报。设置 RXCTL 的初始值为 0d05H，接收网络上的广播或者目标地址同本地物理地址相同的正确数据报。

- RXCFG（0102H）

  RXCFG 控制 CS8900 接收到特定数据报后会引发接收中断。RXCFG 可设置为 0103H，这样当收到一个正确的数据报后，CS8900 会产生一个接收中断。

- BUSCT（0116H）

  BUSCT 可控制芯片的 I/O 接口的一些操作。设置初始值为 8017H，打开 CS8900 的中断总控制位。

- ISQ（0120H）

  ISQ 是网卡芯片的中断状态寄存器，内部映射接收中断状态寄存器和发送中断状态寄存器的内容。

- PORT0（0000H）

  发送和接收数据时，CPU 通过 PORT0 传递数据。

- TXCMD（0004H）

  发送控制寄存器，如果写入数据 00C0H，那么网卡芯片在全部写入后开始发送数据。

- TXLENG（0006H）

  发送数据长度寄存器，发送数据时，首先写入发送数据长度，然后将数据通过 PORT0 写入芯片。

4.4.3.2 网卡驱动分析

驱动首先对网卡芯片进行初始化，即写寄存器 LINECTL、RXCTL、RCCFG、BUSCT。发数据时，写控制寄存器 TXCMD，并将发送数据长度写入 TXLENG，然后将数据依次写入 PORT0 口。网卡芯片将数据组织为链路层类型并添加填充

位和 CRC 校验到网络。同样，CPU 查询 ISO 的数据，当有数据来到后，读取接收到的数据帧。读数据时，CPU 依次读地址 300H，301H，300H，301H。

网卡驱动通过modprobe来自动加载驱动module。因此需要实现cs8900_probe初始化函数。包括初始化特定网卡设备结构体 cs8900_t，通用网卡设备结构体net_device，检测分配给设备的内存空间，检测芯片版本和初始化中断等。

网卡设备的操作主要包括对设备驱动程序所提供的入口点的实现。即file_operations 结构所包含的函数。CS8900A 网卡设备的操作主要包括如下：

```
static struct file_operations cs8900_eeprom_fops = {
    owner:      THIS_MODULE,
    open:       cs8900_eeprom_fopen,
    release:    cs8900_eeprom_frelease,
    llseek:     cs8900_eeprom_fllseek,
    read:       cs8900_eeprom_fread,
    write:      cs8900_eeprom_fwrite,
};
```

CS8900A 驱动程序主要函数有 CS_Init()、CS_Identification()、CS_Close()[13]。

① CS_Init()，CS8900A 初始化。初始化步骤为：

- 检测 CS8900A 芯片是否存在，然后软件复位 CS8900A；
- 如果使用 Memory Map 方式访问 CS8900A 芯片内部寄存器，就设置 CS8900A 内部寄存器基地址（默认为 I/O 方式访问）；
- 设置 CS8900A 的 MAC 地址；
- 关闭事件中断（使用查询方式，如果使用中断方式，则添加中断服务程序，再打开 CS8900A 中断允许位）；
- 配置 CS8900A，然后允许 CS8900A 接收和发送，接收发送 64－1518B 的网络帧和网络广播帧。

② CS_Reset()，复位 CS8900A 芯片。

③ CS_Identification()获得 CS8900A 芯片 ID 和修订版本号。

④ CS_Close()关闭 CS8900A 芯片数据收发功能，并关闭中断请求。

4.4.3.3 网卡驱动移植

网卡驱动移植过程如下所示[25]：

① 把驱动程序 cs8900.c，cs8900.h 拷贝到$KERNEL/drivers/net 目录下

② 修改$KERNEL/drivers/net/Kconfig，增加以下内容

```
Config ARM_CS8900
    tristate "CS8900 support"
    depends on NET_ETHERNET && ARM
    help
```

③ 修改$KERNEL/drivers/net/Makefile，

增加如下内容：

```
obj-$(CONFIG_ARM_CS8900) += cs8900.o
```

④ 在$KERNEL/include/asm-arm/arch-s3c2410 目录下创建 smdk2410.h 文件

增加如下内容：

```
#include <linux/config.h>
#define pSMDK2410_ETH_IO        0x19000000
#define vSMDK2410_ETH_IO        0xE0000000
#define SMDK2410_ETH_IRQ        IRQ_EINT9
```

⑤ 修改$KERNEL/arch/arm/mach-s3c2410/mach-smdk2410.c 文件

添加如下内容：

```
#include asm/arch/smdk2410.h
static struct map_desc smdk2410_iodesc[] __initdata = {
[0] = {vSMDK2410_ETN_IO,Psmdk2410_ETN_IO,SZ_1M,MT_DEVICE},
};
```

⑥ 在内核配置中选上

```
Device Drivers..>
   Network device support…>
        Ethernet (10 or 100 Mbit)
                [*]    CS8900 support.
```

4.4.4 LCD 驱动

系统要通过 LCD 来显示脉搏波。因此要实现 LCD 的驱动。

4.4.4.1 LCD 控制器简介

S3C2410 有内置的 LCD 控制器，它支持单色，每象素 2 位、4 位的黑白屏，也可以支持每象素 8 位和 12 位的彩色 STN LCD 屏，也支持每象素 1 位、2 位、

4 位和 8 位的调色 TFT 彩色 LCD，并且支持每象素 16 位和每象素 24 位的真彩显示。LCD 控制器可以通过编程选择支持不同的 LCD 屏的要求，例如行和列象素，数据总线宽度，接口时序和刷新频率。

LCD 控制器的主要作用是，将定位于系统存储器的显示缓存区的 LCD 图像数据传送到外部 LCD 驱动器。

S3C2410 内部的 LCD 控制器的逻辑示意图如图 4-8 所示[12]：

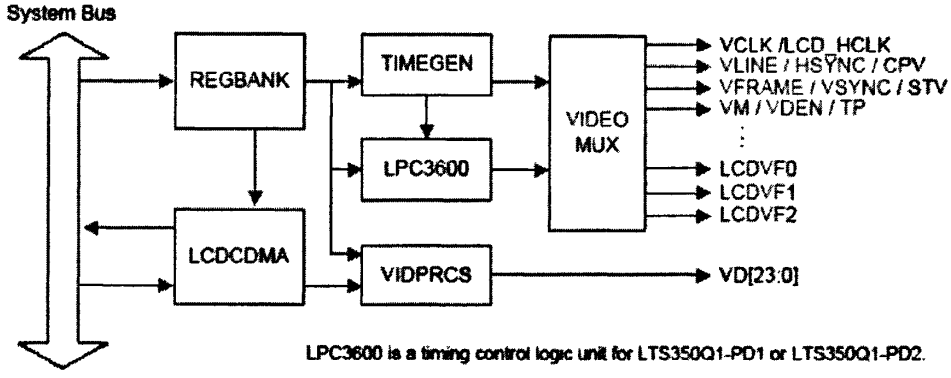

图 4-8　LCD 模块功能图

REGBANK 是 LCD 控制器的寄存器组，用来对 LCD 控制器的各项参数进行设置。而 LCDCDMA 则是 LCD 控制器专用的 DMA 信道，负责将视频资料从系统总线（System Bus）上取来，通过 VIDPRCS 从 VD[23：0]发送给 LCD 屏。同时 LPC3600 负责产生 LCD 屏所需要的控制时序，例如 VSYNC、HSYNC、VCLK、VDEN，然后从 VIDEO MUX 送给 LCD 屏。

与 LCD 相关的控制信号如下：

- VFRAME/VSYNC/STV：LCD 控制器和 LCD 驱动器之间的帧同步信号。该信号告诉 LCD 屏的新的一帧开始了。LCD 控制器在一个完整帧显示完成后立即插入一个 VFRAME 信号，开始新一帧的显示；
- VLINE/HSYNC/CPV：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 驱动器将水平线（行）移位寄存器的内容传送给 LCD 屏显示。LCD 控制器在整个水平线（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 驱动器之间的像素时钟信号，由 LCD 控制器送

出的数据在 VCLK 的上升沿处送出，在 VCLK 的下降沿处被 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，从而控制像素点的显示或熄灭。VM 信号可以与每个帧同步，也可以与可变数量的 VLINE 信号同步。
- VD[0..23]：LCD 象素数据输出端口，即 RGB 信号线，采用的是 5：6：5 模式。

TFT 液晶屏时序分析如图 4-9 所示[12]：

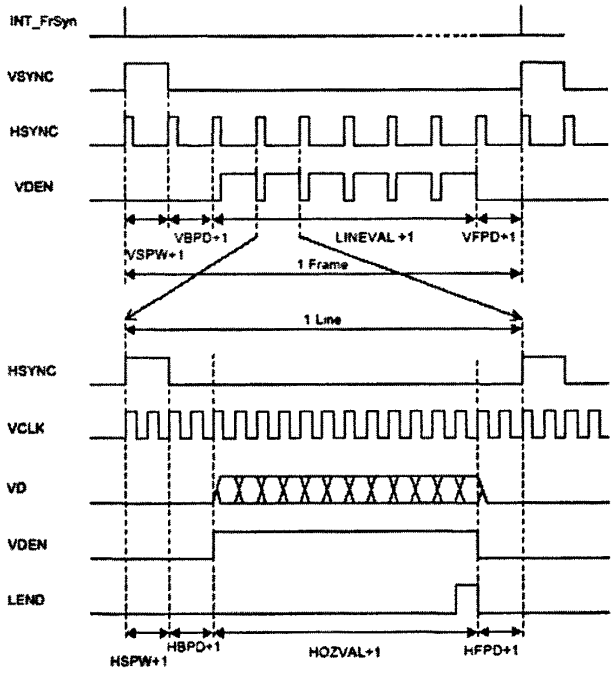

图 4-9 TFT 液晶屏时序图

其中 VSYNC 是帧同步信号，VSYNC 每发出一个脉冲，意味着新的一屏视频资料开始发送。而 HSYNC 为行同步信号，每个 HSYNC 脉冲表明新的一行视频资料开始发送。而 VDEN 用来标明视频资料有效，VCLK 用来锁存视频资料的像素时钟。

在帧同步以及行同步的头尾都必须留有回扫时间，例如对于 VSYNC 来说前回扫时间就是（VSPW+1）+（VBPD+1），后回扫时间就是（VFPD+1）；HSYNC 亦类同。

4.4.4.2 LCD 驱动分析

LCD 控制器的功能是显示驱动信号，进而驱动 LCD。用户需要通过读写一系列的寄存器，完成配置和显示驱动。

在驱动 LCD 设计的过程中首要的是配置 LCD 控制器，而在配置 LCD 控制器中最重要的一步是帧缓冲区（FrameBuffer）的指定。用户所要显示的内容皆是从缓冲区中读出，从而显示到屏幕上的。帧缓冲区的大小由屏幕的分辨率和显示色彩数决定。驱动帧缓冲的实现是整个驱动开发过程的重点。

帧缓冲区是出现在 Linux 2.2.xx 及以后版本内核当中的一种驱动程序接口，这种接口将显示设备抽象为帧缓冲区设备区。帧缓冲区为图像硬件设备提供了一种抽象化处理，它代表了一些视频硬件设备,允许应用软件通过定义明确的界面来访问图像硬件设备。这样软件无须了解任何涉及硬件底层驱动的东西（如硬件寄存器）。它允许上层应用程序在图形模式下直接对显示缓冲区进行读写和 I/O 控制等操作。通过专门的设备节点可对该设备进行访问，如/dev/fb*。用户可以将它看成是显示内存的一个映像，将其映射到进程地址空间之后，就可以进行读写操作，而读写操作可以反映到 LCD[26]。

帧缓冲设备对应的设备文件是/dev/fb*。如果系统有多个显卡，Linux 还支持多个帧缓冲设备，最多可达 32 个，即/dev/fb0～/dev/fb31。而/dev/fb 则指向当前的帧缓冲设备，通常情况下，默认的帧缓冲设备为/dev/fb0[26]。

帧缓冲设备也属于字符设备，采用“文件层-驱动层”的接口方式。在文件层为之定义了以下数据结构。

```
static struct file_operations fb_fops={
    ower: THIS_MODULE,
    read: fb_read,          /*读操作*/
    write: fb_write,        /*写操作*/
    ioctl: fb_ioctl,        /*I/O 操作*/
    mmap: fb_mmap,          /*映射操作*/
    open: fb_open,          /*打开操作*/
    release: fb_release,    /*关闭操作*/
}
```

其成员函数都在 linux/driver/video/fbmem.c 中定义，其中的函数对具体的硬件进行操作，对寄存器进行设置，对显示缓冲进行映射。主要结构体还有以下几个：

- struct fb_fix_screeninfo：记录了帧缓冲设备和指定显示模式的不可修改信息。它包含了屏幕缓冲区的物理地址和长度。
- struct fb_var_screeninfo：记录了帧缓冲设备和指定显示模式的可修改信息。它包括显示屏幕的分辨率、每个像素的比特数和一些时序变量。其中变量 xres 定义了屏幕一行所占的像素数，yres 定义了屏幕一列所占的像素数，bits_per_pixel 定义了每个像素用多少个位来表示。
- struct fb_info：Linux 是帧缓冲设备定义的驱动层接口。它不仅包含了底层函数，而且还有记录设备状态的数据。每个帧缓冲设备都与一个 fb_info 结构相对应。其中成员变量 modename 为设备名称，fontname 为显示字体，fbops 为指向底层操作的函数的指针。

4.4.4.3 LCD 驱动移植

Linux 2.6.14 内核已经包含了该驱动，所以只要在系统初始化文件中对 LCD 进行设置、初始化即可[27][28][29]。

①#vi $KERNEL/arch/arm/mach-s3c2410/mach-smdk2410.c

添加头文件：

```
#include <asm/arch/regs-lcd.h>
#include <asm/arch-s3c2410/fb.h>
```

②添加 LCD 初始化代码：

```
static struct s3c2410fb_mach_info s3c2410_fb_cfg __initdata = {
    .fixed_syncs = 0,
    .width = 240,
    .height = 320,
    .xres = {240,240,240},
    .yres = {320,320,320},
    .bpp = {16,16,16},
    .regs = {
        .lcdcon1 = S3C2410_LCDCON1_TFT16BPP|S3C2410_LCDCON1_
TFT|S3C2410_LCDCON1_CLKVAL(5),
        .lcdcon2 = S3C2410_LCDCON2_VBPD(2)|S3C2410_LCDCON2_
LINEVAL(319)|S3C2410_LCDCON2_VFPD(2)|S3C2410_LCDCON2_VSPW(4),
        .lcdcon3 = S3C2410_LCDCON3_HBPD(8)|S3C2410_LCDCON3_
HOZVAL(239)|S3C2410_LCDCON3_HFPD(8),
        .lcdcon4 = S3C2410_LCDCON4_MVAL(13)|S3C2410_LCDCON4_
```

```
HSPW(6),
            .lcdcon5 = S3C2410_LCDCON5_FRM565|S3C2410_LCDCON5_
HWSWP|S3C2410_LCDCON5_PWREN,
    },
        .gpcup = 0xffffffff,
        .gpcup_mask = 0xffffff,
        .gpccon = 0xaaaaaaaa,
        .gpccon_mask = 0xffffffff,
        .gpdup = 0xffffffff,
        .gpdup_mask = 0xffffffff,
        .gpdcon = 0xaaaaaaaa,
        .gpdcon_mask = 0x0,
        .lpcsel =0x0,
};
```

③在 static void __init smdk2410_map_io(void)函数中添加：

```
set_s3c2410fb_info(&s3c2410_fb_cfg);
```

④修改内核配置文件。

```
$ make menuconfig
```

在 Device Drivers->Graphics support->下选中：

```
Support for frame buffer devices
S3C2410 LCD framebuffer support
S3C2410 lcd debug messages
Virtual Frame Buffer Support
    Console display driver support
Backlight & LCD device support
```

⑤至此，重新编译内核，就完成 LCD 的驱动。

4.4.5 A/D 和触摸屏驱动

系统要使用 A/D 控制器来转换传感器采集的数据，使用 TouchScreen 来选择实现特定的功能。因此，要实现 A/D 和触摸屏的驱动。

系统使用的是电阻式触摸屏。电阻式触摸屏是使用最广泛的触摸屏，它的屏体部分是一块与显示器表面非常配合的多层复合薄膜，由一层玻璃或有机玻璃作为基层，表明涂有一层透明的导电层，上面再盖有一层外表面硬化处理、光滑防刮的塑料层，它的内表面也涂有一层透明导电层，在两层导电层之间有许多细小（小于千分之一英寸）的透明隔离点把它们隔开绝缘[6]。

当手指触摸屏幕时，平常相互绝缘的两层导电层就在触摸点位置有了一个接

触，因其中一面导电层接通 Y 轴方向的 5V 均匀电压场，使得侦测层的电压由零变为非零，控制器侦测到这个接通后，进行 A/转换，并将得到的电压值与 5V 相比即可得触摸屏 Y 轴坐标，同理得出 X 轴的坐标，这就是电阻触摸屏的基本原理[6]。

S3C2410 内置一个 8 信道的 10 位模数转换器(ADC)，该 ADC 能以 500KSPS 的采样资料将外部的模拟信号转换为 10 位分辨率的数字量。同时 ADC 部分能与 CPU 的触摸屏控制器协同工作，完成对触摸屏绝对地址的测量。转换器的主要特征如下[6]：

分辨率：±1LSB，微分线性度误差：±1.5LSB；

积分线形度误差：±1LSB，最大转化速率：500KSPS；

输入电压范围：0~3.3v，片上采样保持功能；

中断等待模式；

独立/自动 X/Y 位置转换模式。

图 4-10 展示了 S3C2410 A/D 转换器和触摸屏接口的功能块图。

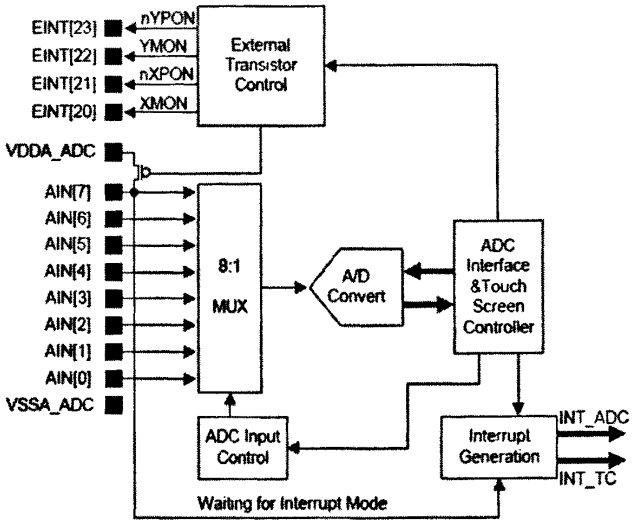

图 4-10 ADC 及触摸屏控制器逻辑示意图

图 4-11 是 S3C2410 的 ADC 以及触摸屏控制器的基础上外接触摸屏的示意图[6]。

S3C2410 有 8 个模拟输入通道。其中，通道 7（AIN[7]）作为触摸屏接口的

X 坐标输入，通道 5（AIN[5]）作为触摸屏接口的 Y 坐标输入。在采样过程中，通过软件对特殊寄存器置位，S3C2410 的触摸屏控制器就会自动控制触摸屏接口打开或关闭控制信号，按顺序完成 X 坐标点采集和 Y 坐标点采集。

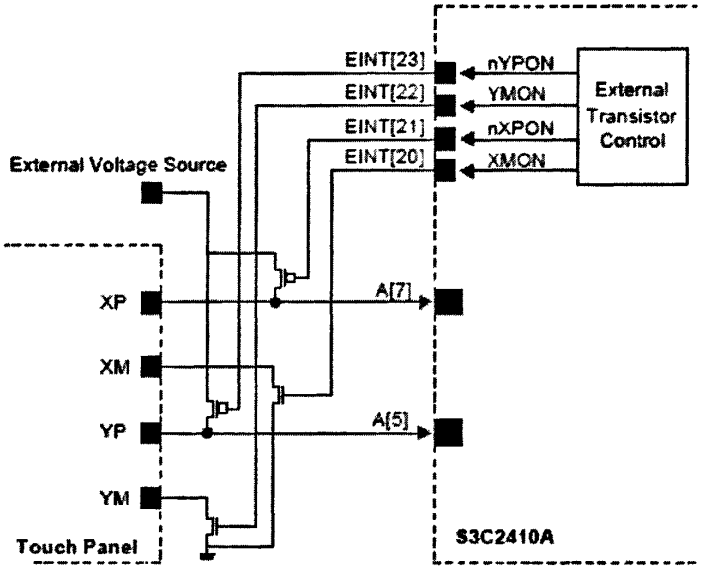

图 4-11 ADC 外接触摸屏示意图

系统把通道 6（AIN[6]）作为脉搏传感器的输入。传感器的采集数据后的 A/D 转换和触摸屏使用的是同一个 ADC，因此要解决两者的资源共享问题。

4.4.5.1 ADC 控制器简介

①ADC 及触摸屏控制器的工作模式如下[6]：

- ADC 普通转换模式

  普通转换模式（AUTO_PST=0，XY_PST=0）是用来进行一般的 ADC 转换之用的，例如通过 ADC 测量电池电压等等。

- 独立 X/Y 轴坐标转换模式

  独立 X/Y 轴坐标转换模式包含了 X 轴模式和 Y 轴模式。首先进行 X 轴的坐标转换（AUTO_PST=0，XY_PST=1），X 轴的转换资料会写到 ADCDAT0 寄存器的 XPDAT 中，等待转换完成后，触摸屏控制器会产生相应的中断。然后进行 Y 轴的坐标转换（AUTO_PST=0，XY_PST=2），Y 轴的转换资料会写到 ADCDAT1 寄存器的 YPDAT 中，等待转换完成后，触摸屏控制器会

产生相应的中断。

- 自动 X/Y 轴坐标转换模式

自动 X/Y 轴坐标转换模式（AUTO_PST=1，XY_PST=0）将会自动地进行 X 轴和 Y 轴的转换操作，随后产生相应的中断。

- 中断等待模式

在系统等待“Pen Down”，即触摸屏按下的时候，其实是处于中断等待模式。一旦被按下，实时产生“INT_TC”中断信号。每次发生此中断后，X 轴和 Y 轴坐标转换资料都可以从相应的资料寄存器中读出。

- 闲置模式

在该模式下，转换资料寄存器中的值都被保留为上次转换时的资料。

②与 ADC 和触摸屏相关的寄存器如下[6]：

- A/D 控制寄存器 ADCCON

ENABLE_START： 置 1 表示启动 ADC 转换

置 0 表示无操作

RESR_START：置 1 表示允许读操作启动 ADC 转换

置 0 表示禁止读操作启动 ADC 转换

STDBM： 置 1 表示将 ADC 置为闲置转换（模式）

置 0 表示将 ADC 置为正常操作状态

SEL_MUX：选择需要进行转换的 ADC 信道

PRSCVL：ADC 转换时钟预分频参数

PRSCEN：ADC 转换时钟使能

ECFLG：ADC 转换完成标志位（只读）

置 1 表示 ADC 转换结束；置 0 表示 ADC 转换进行中。

- A/D 数据寄存器 ADCDAT0 和 ADCDAT1

XY_PST：选择 X/Y 轴自动转换模式

AUTO_PST：X/Y 轴自动转换使能位

UPDOWN：选择中断等待模式的类型

0 表示按下产生中断；1 表示释放产生中断。

- ADC 触摸屏控制寄存器（ADCTSC）

XY_PST：对 X/Y 轴手动测量模式进行选择

AUTO_PST：X/Y 轴的自动转换模式使能位

PULL_UP：XP 端的上拉电阻使能位

XP_SEN：设置 nXPON 输出状态

XM_SEN：设置 XMON 输出状态

YP_SEN：设置 nYPON 输出状态

YM_SEN：设置 YMON 输出状态

- ADC 启动延迟寄存器（ADCDLY）

4.4.5.2 A/D 驱动

A/D 驱动主要是设置 ADCCON 和 ADCTSC 寄存器，然后开启 A/D 转换，等待转换结束后，从资料寄存器中读取转换结果即可。

①寄存器设置

设置寄存器 ADCCON 如下：允许设置预分频，设置预分频值，设置通道数。

S3C2410_ADCCON_PRSCEN|S3C2410_ADCCON_PRSCVL(49)|S3C2410_ADCCON_SELMUX(6)

②开启 AD 转换

设置寄存器 ADCCON 如下：设置 A/D 转换开始。

S3C2410_ADCCON_ENABLE_START

③等待转换结束

当 ADCCON 寄存器的 ECFLG 为 1 时，表示 A/D 转换结束。

④读取转换结果

A/D 转换的结果存放在 ADCDAT0 寄存器中，因此直接读取即可。

4.4.5.3 触摸屏驱动

触摸屏的驱动程序主要处理触摸屏中断处理程序和 A/D 转换中断处理程序。触摸屏驱动程序的流程图如图 4-12 所示。

触摸屏中断处理程序根据数据资料寄存器来判断触摸屏的铁笔（stylus）状态是按下还是释放。如果释放的话，则直接返回。如果是按下的话，则设置 AD

转换模式然后开启 AD 转换，AD 转换结束并填满缓冲区之后，激活下半部分处理程序并处理接收数据，同时等待 stylus 被释放。在下半部分处理程序中，如果触摸屏缓冲区已满，且 stylus 的状态还未释放，则算出平均值后，交给事件处理层处理，主要是填写缓冲区，然后唤醒等待输入数据的进程。如果 stylus 的状态已经释放，则报告 stylus 释放事件，重新等待 stylus 按下[30][31]。

AD 中断处理程序主要判断缓冲区是否已满，如果未满的话，则设置 AD 转换模式并开启 AD 转换；如果已满的话，则激活下半部分处理程序并处理接收数据。

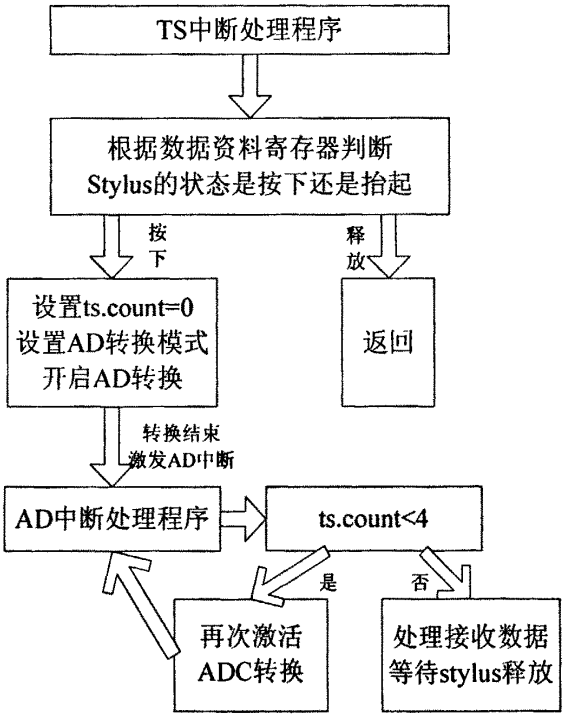

图 4-12 触摸屏驱动流程图

AD 中断处理程序中寄存器设置如下：

- ADCTSC 寄存器设置为：XP pull-up 禁止（PULL_UP=1），自动转换模式（AUTO_PST=1），无操作模式（XY_PST=0）
- ADCCON 寄存器设置为：设置 AD 转换开始

  S3C2410_ADCCON_ENABLE_START

4.4.5.4 互斥解决

由于触摸屏驱动和传感器采集 A/D 转换驱动都要用到 ADC 控制器。因此要解决互斥问题。本系统采用信号量来解决互斥问题，由于触摸屏的使用频率相对传感器采集 A/D 转换而言少很多，而且系统通过触摸来选择要实现的功能，因此给触摸屏驱动更高的优先级。

系统通过在触摸屏驱动中设置全局变量 s3c2410_flag 和信号量 sem 来解决互斥问题。在 2.6 版本的 Linux 内核中，使用 EXPORT_SYMBOL 可以使得一个内核模块的变量被另外一个内核模块所使用。系统中具体使用如下：

在触摸屏驱动中定义如下：

```
int s3c2410_flag=1;
EXPORT_SYMBOL(s3c2410_flag);
struct semaphore sem;
EXPORT_SYMBOL(sem);
```

在 A/D 驱动中引用变量如下：

```
extern int s3c2410_flag;
extern struct semaphore sem;
```

在触摸屏驱动中，当触摸屏按下时，设置变量 s3c2410_flag 为 1，在进行 A/D 设置和开启转换之前要先获取信号量，在转换结束之后释放信号量，在 stylus 释放之后把变量 s3c2410_flag 设置为 0。

在 A/D 驱动中，如果变量 s3c2410_flag 为 1 时，则表示触摸屏在使用 ADC 控制器，因此直接返回。如果变量 s3c2410_flag 为 0 时，则表示 ADC 控制器空闲中，可以使用，先获取信号量然后设置 A/D 转换模式并开启 A/D 转换，在转换结束并读出 A/D 转换结果之后重新设置 A/D 转换模式并释放信号量。

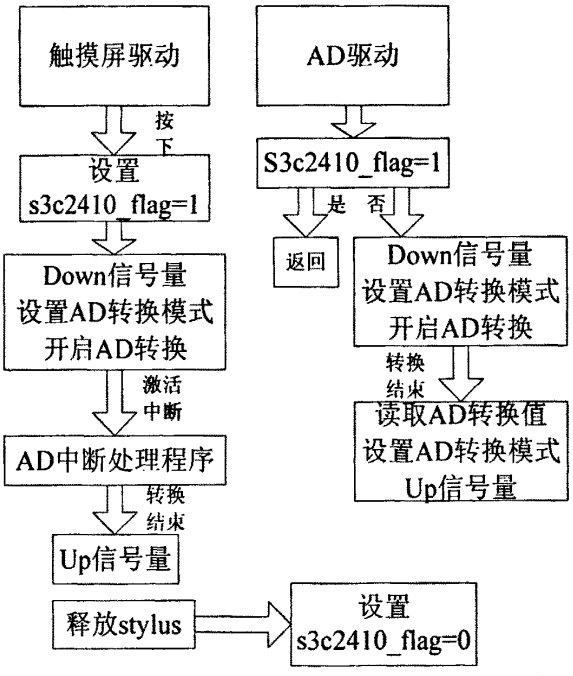

图 4-13 触摸屏和 A/D 驱动互斥解决流程图

系统在互斥的实现中，还要注意对 ADCTSC 寄存器的设置。

在 A/D 驱动中，在获取信号量之后，设置 ADCTSC 为：XP pull-up 禁止（PULL_UP=1），正常 ADC 转换模式（AUTO_PST=0），无操作模式（XY_PST=0）。

在 A/D 驱动中，在释放信号量之前，设置 ADCTSC 为：XP pull-up 允许（PULL_UP=0），正常 ADC 转换模式（AUTO_PST=0），等待中断模式（XY_PST=3）。

4.4.5.5 驱动移植

4.4.5.5.1 触摸屏驱动移植

系统把触摸屏作为系统的输入设备。因此触摸屏驱动的移植过程如下[32]：

①把 s3c2410_ts.c 文件拷贝到 drivers/input/touchscreen/目录下

②在内核文件 arch/arm/mach-s3c2410/devs.c 中添加如下内容：

```
static struct s3c2410_ts_mach_info s3c2410ts_info;
void __init set_s3c2410ts_info
(struct s3c2410_ts_mach_info *hard_s3c2410ts_info)
```

```
{
memcpy(&s3c2410ts_info,hard_s3c2410ts_info,sizeof(struct
s3c2410_ts_mach_info));
}
EXPORT_SYMBOL(set_s3c2410ts_info);

struct platform_device s3c_device_ts = {
.name = "s3c2410-ts",
.id = -1,
.dev = {
.platform_data = &s3c2410ts_info,
}
};
EXPORT_SYMBOL(s3c_device_ts);
```

③在内核文件 arch/arm/mach-s3c2410/devs.h 中添加如下内容：

```
extern struct platform_device s3c_device_ts;
```

④在内核文件 arch/arm/mach-s3c2410/mach-smdk2410.c 中添加如下内容：

```
static struct s3c2410_ts_mach_info sbc2410_ts_cfg __initdata = {
.delay = 1000,
.presc = 49,
.oversampling_shift = 2,
};
```

在 smdk2410_devices 结构中，添加：

```
&s3c_device_ts,
```

在 smdk2410_map_io 函数中添加：

```
set_s3c2410ts_info(&sbc2410_ts_cfg);
```

⑤在内核文件 drivers/input/touchscreen/Kconfig 中添加如下内容：

```
config TOUCHSCREEN_S3C2410
tristate "Samsung S3C2410 touchscreen input driver"
depends on ARCH_S3C2410 && INPUT && INPUT_TOUCHSCREEN
```

select SERIO

help

⑥在内核文件 drivers/input/touchscreen/Makefile 中添加如下内容：

obj-$(CONFIG_TOUCHSCREEN_S3C2410) += s3c2410_ts.o

同时选中编译如下内容：

CONFIG_INPUT+=input.o

CONFIG_INPUT_TSDEV+=tsdev.o

CONFIG_INPUT_TOUCHSCREEN+=s3c2410_ts.o

⑦至此，重新编译内核，即可使用触摸屏。

4.4.5.5.2 A/D 驱动移植

A/D 驱动移植方法如下：将 A/D 驱动文件拷贝到系统根文件系统 tmp 文件下。由于 A/D 驱动要自动加载，因此在根文件系统 linuxrc 文件中。添加如下命令：

/sbin/insmod /tmp/adc.ko

/bin/mknod /dev/adc c 253 0

由此，重启系统，即可自动加载 A/D 驱动。

4.5 应用程序的设计与实现

4.5.1 应用程序总体设计

从人体桡动脉测得的脉搏信号，由于手臂和人的整体系统有着不可分割的联系，必然会受到体内环境的随即影响；再加上测量过程中身体的移动或大声讲话等干扰，使脉搏信号带有一定的内外干扰。因此在对波形进行分析之前，需对其进行一系列的软件滤波（数字滤波），提取有用信号，消除和减少各种干扰。在本论文中我们根据采集信号的特点对采集到的信号进行了滑动平均滤波处理和脉搏波自动检测。在脉搏波检波后，计算心血管参数。应用程序总体流程图如图 4-14 所示。

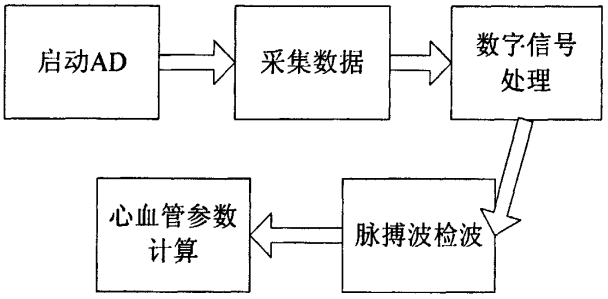

图 4-14 应用程序流程图

4.5.2 脉搏波采集

系统通过离散采样的方法，对连续的脉搏波信号进行采集，因此，应考虑下面两个问题。

4.5.2.1 采样周期的确定

由传感器检测的桡动脉脉搏信号，一般为频率范围为 40~180beat/min（0.66~3beat/s）的连续信号。根据采样定理，频率为 0~ω 的连续信号 $f(t)$，可以用一系列离散采样值 $f(t_1)$，$f(t_2)$，…，$f(t_i)$来表示，只要这些采样点的时间间隔 $T \le \frac{1}{2} \bullet \frac{2\pi}{\omega}$，则该信号就可以由这些采样值完全恢复出来。脉搏波信号的频谱分析表明，脉搏波的信号的最高谱线成分，一般不超过 20Hz。由此可以初步确定脉搏波离散采样周期为 $T \le \frac{1}{2 \times 20} = 25ms$[3]。

系统采用的是 S3C2410 集成的 A/D 转换器。当 PCLK 频率为 50MHZ，预分频值为 49，10 位数字量的转换时间如下：

A/D 转换频率＝50MHz/（49＋1）＝1MHz

转换时间＝1/（1MHz/5 周期）＝1/200KHz＝5μs

系统在触摸屏驱动中，设置 ADCDLY 寄存器为 1000，因此，A/D 转换的延迟为 5μs×1000＝5ms。因此设置采样周期为 10ms 即可。

4.5.2.2 采样持续时间的确定

本系统采用多波叠加平均处理法来计算各个心血管参数。这时，即使在采集最长周期的脉搏波时要求脉搏波的采样持续时间应该在多个脉搏波周期数以上。

一般正常的脉搏波周期范围为 0.9~0.7s（65~85beat/min）。为了使系统能够适应更宽的检测范围，可以确定系统能够检测到的脉搏波周期范围为 1.5~0.5s（40~200beat/min）[3]。系统采用每隔 20s 对采样到的多个脉搏波进行一次参数处理。

4.5.3 滑动平均滤波

从桡动脉提取的脉搏信号不可避免带有很大的随即干扰，消除随即干扰的影响可以使用滑动平均方法。滑动平均方法是一种非常灵活的滤波方法，它可以作为低通滤波器用于消除存在于各种不同波形中的噪声，也可以作为高通滤波器消除较高频率信号的基线漂移。该算法在使用过程中需要用到平滑系数 s，来指定用于滑动平均的波形数据个数或者采样点数。滑动平均技术最初是用于消除一段数据中的异常点，展示其真实趋势的方法。它在本质上其实相当于一种有限冲击响应（FIR）滤波器[1]。

图 4-15 为滑动平均的过程，以三点平均为例。

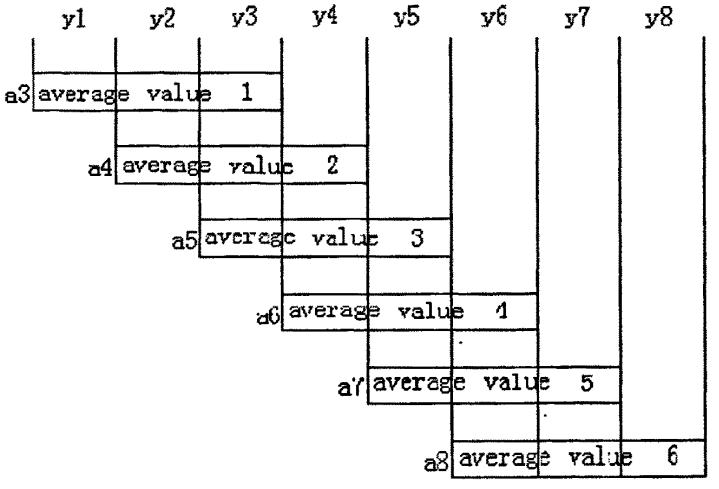

图 4-15 滑动平均滤波示意图

假设平均点数为 m，则由下式可得平均点 a(n)：

$$a(n)=\frac{1}{s}\sum_{n}^{n-(m-1)}y(n) \qquad \begin{cases} s=m.....低通 \\ s=-m.....高通 \end{cases}$$

式中 a 为平均值，n 为数据点的位置，y 为 n 处的实际数值。

对于 m 的选择主要是根据滑动平均滤波后的信号波形是否具备脉搏波的波形特征来确定。图 4-16 是对于同一采样获得的信号，对几个不同的 m 值进行滑动平均滤波后获得的信号波形。

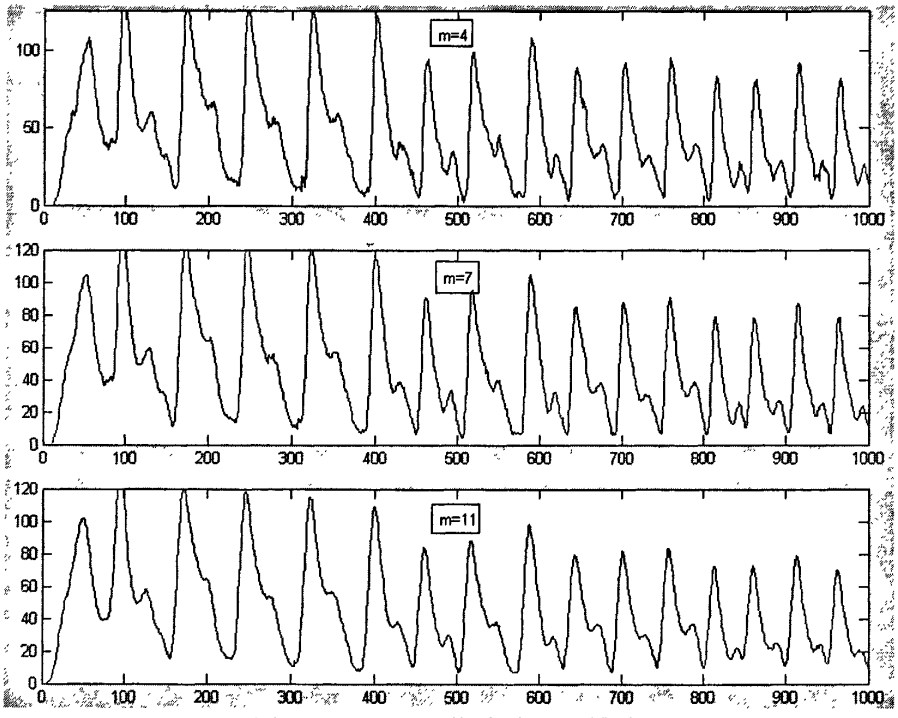

图 4-16 不同 m 值滤波后脉搏波形图

由于 m 值越大，需要的运算越多，越不利于嵌入式系统。根据图 4-16，取 m 为 7 时即可获得较好的脉搏波形。因此本系统取 m 值为 7。

图 4-17 是采样到的信号和经过滑动平均滤波后（m 取为 7）的信号比较。

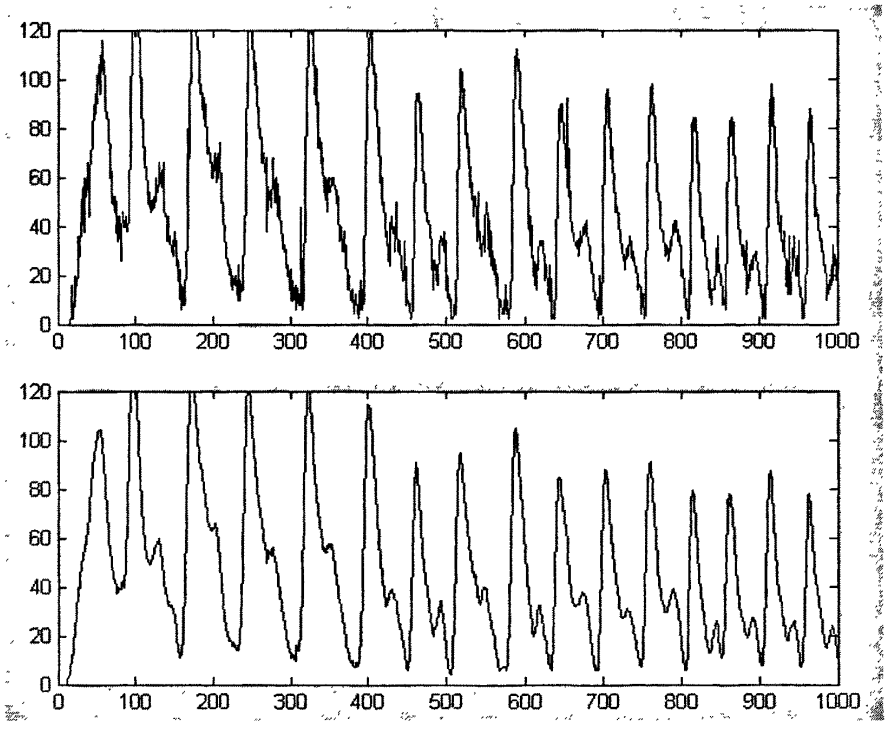

图 4-17 滤波前后采样信号的波形图

4.5.4 脉搏波检测

由于心血管参数的计算需要完整的脉搏波形，因此需要从一组脉图中检测出单波（完整的一个脉搏波）来。检测脉搏的方法如下：选择从波形稳定后开始采集数据，然后从数据中分离出一个个脉搏波。检测脉搏波的关键是找到脉搏波的起点和终点，即找到极小值点。根据脉搏波的波形特性，从起点开始很快上升至最高值点。因此首先找到极大值点，从而找到最高值点，然后从最高值点反方向找到最靠近的极小值点，这个点即是我们找到的起点。找到起点后，从最高点之后找到和起始点值相近的极小值点，从而找到终点。找到起点和终点，就检测到了一个脉搏波。

用脉搏传感器检测到的脉搏信号实际上是桡动脉处血管半径随时间的变化曲线。理论分析表明，它可用来近似代表桡动脉处血管压力随时间变化的脉搏信号，但这信号是以电压量变化来表示的，因而必须要以毫米汞柱对曲线定标后才能作分析处理。

脉搏波定标前后的信号有以下关系：

$$P(i)=\frac{P_s-P_d}{M_s-M_d}[M(i)-M_d]+P_d \quad (\text{mmHg})$$

其中，M(i)表示脉搏波定标前的实际采样信号波形上任一点值，M_s 为采样波峰值，M_d 为波谷值；P(i)为定标后的脉搏波压力波形上的任一点压力值，其中 P_s 为收缩压，P_d 为舒张压。

检测到的脉搏波可能是奇异波，因此要对脉搏波的检测进行重新设置限制。脉搏波的主波波峰（舒张压），即是我们所说的最大值要进行限制，限制为 60mmHg~180mmHg。起点值和终点值相差也要小于 10mmHg。主波波峰和起点值的差值限制为 20mmHg~85mmHg。从起点到主波波峰的间隔要小于 0.25s，起点到终点的间隔即周期也设置为在 0.5s~1.5s 之内。

脉搏波检测的流程如图 4-18 所示。

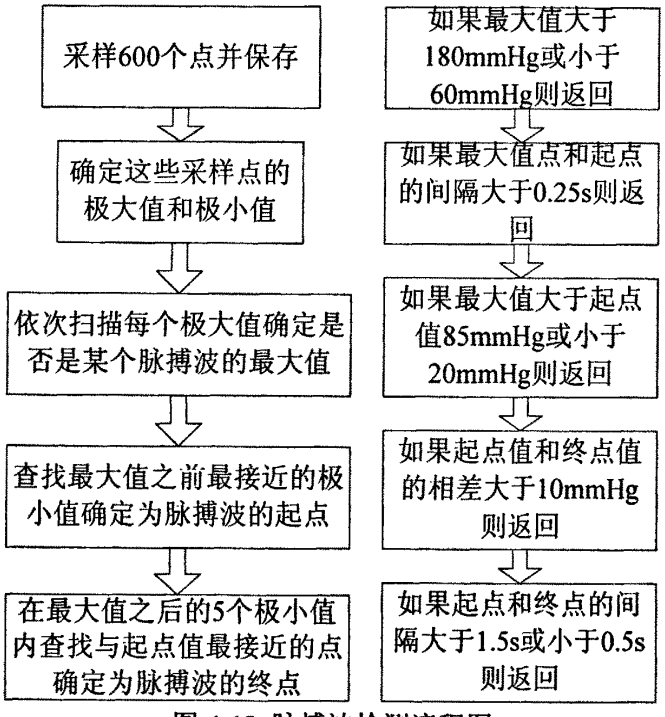

图 4-18 脉搏波检测流程图

4.5.5 心血管参数计算

任一心动周期中，血压是随时间脉动变化的，不同的血压波形和数值将代表人体不同的生理病理状态。要完整描述出血压随时间的变化，除表示为收缩压

P_s和舒张压 P_d外，还应包含有描述血压波形随时间变化的参数 K 值。K 值的大小决定于脉搏波的波形，并和人体的外周阻力、血管壁弹性以及血液黏性等生理因素有关，所以不同的 K 值将从量的方面代表不同的心血管生理状态。因此，只有得到 P_s，P_d和 K 三个参数，才能完整地描述出人体血压地生理特征。临床上 Ps、Pd 和 K 值都可用无创方法方便地得到，它们三者将完全代表血压的数值和波形，是无创检测心血管血流参数的基础。

心血管血流参数主要是指收缩压 P_s，舒张压 P_d、脉压差（P_s-P_d）、平均动脉压 P_m、心率 HR、心搏出量 SV、心输出量 CO、心搏指数 SI、心脏指数 CI、外周阻力 TPR、血管顺应性 AC、血流半更新率 ALK、血流半更新时间 ALT 以及血流平均滞留时间 T_m等十余项指标。系统主要计算如下心血管参数。

①K 值和平均动脉压 P_m

由脉搏波的产生与传播的机理可知，随着血管阻力和动脉弹性等生理变化，脉搏波波形特征变化首先反映在脉搏波波形面积的变化上，为了能用一个简单的特征量来描述上述变化，北京工业大学罗志昌教授提出一个以脉图面积变化为基础的脉搏波波形特征量 K 值，其定义为[3]

$$K=\frac{P_m-P_d}{P_s-P_d}$$

或

$$P_m=P_d+K*(P_s-P_d)$$

其中，$P_m=\frac{1}{T}\int_0^T P(t)dt$ 为平均动脉压，它等于一个心动周期 T 中，脉搏压力 P(t)的平均值；P_s、P_d分别为收缩压和舒张压。

②心率 HR

心率 HR 是心脏每分钟博动的次数。根据定义：

$$HR=\frac{60}{T}$$

T 为心动周期，即一个脉搏波的周期。

③心博出量 SV 和心输出量 CO

血流量是估计心血管参数的重要参数。血流量的主要参数是心博出量和心输出量。由于血流量在人体内部血管中流动，要在人体外部无创地检测到其流量十分困难。20 世纪 60 年代以来许多研究人员通过脉搏波压力波形导出脉搏流量的

方法计算心博出量，其中比较重要的是 Wesseling 于 1983 年提出的用弹性管模型计算心输出量的方法。1987 年北京工业大学罗志昌等在此基础上继续研究，并将 Wesseling 提出的公式简化为能用于临床的使用形式。得出的公式如下[3]：

心博出量 $SV = \frac{0.283}{K \times K} T(P_s - P_d)$ （mL）

心输出量 $CO = SV \times \frac{60}{T} = \frac{17}{K \times K}(P_s - P_d)$ （mL/min）

④血管顺应性 AC 和外周阻力 TPR

血管信息主要指血管的外周阻力 TPR 和动脉顺应性 AC 两个血流参数。

根据外周阻力的定义[3]

$$TPR = \frac{P_m}{CO} = \frac{P_d(1 + K\frac{P_s - P_d}{P_d})}{\frac{17}{K \times K}(P_s - P_d)} = \frac{K \times K}{17}\left[\frac{1}{\frac{1}{P_d}(P_s - P_d) + K}\right] \times 60 \quad (\text{PRU})$$

外周阻力 TPR 的大小和 P_d、P_s、K 等三个参数都有关系。当脉压差（P_s—P_d）增加或舒张压 P_d 减少时，TPR 将减少；反之，当（P_s—P_d）减少或 Pd 增加时，TPR 将增加。同时，外周阻力 TPR 还受到 K 值的强烈影响，当 K 值增加时，TPR 将升高。

根据动脉顺应性的定义，当动脉内血液压力改变一个单位时，所对应动脉体积的变化量，它近似地可表示为[3]：

$$AC = \frac{SV}{P_s - P_d} = \frac{0.28}{K \times K} T \quad (\text{mL/mmHg})$$

动脉顺应性 AC 是动脉管段本身的固有特性，它与血压大小无关。它仅决定于代表血管阻力、血管壁弹性和血液黏性等因素的脉搏波波形特征量 K 值的大小和心动周期 T 的大小。K 值越大，心动周期越小（心跳越快），则动脉顺应性也越小，即动脉弹性性越差；反之，K 值越小，心动周期越长（心跳越慢），则动脉顺应性越大，即动脉弹性较好。

4.6 构建根文件系统

4.6.1 嵌入式 Linux 文件系统的介绍

文件系统向用户或程序提供一个使用文件的统一界面，从而使得对文件的各

种操作能够在更加抽象、简便的层次上进行，文件系统的引入通常有以下几种目的[33]：

① 满足用户管理数据的需求，包括数据存储和对数据的操作。

② 尽可能保证文件中数据的有限性。

③ 性能优化，以提高系统的吞吐量和响应速度。

④ 提供不同类型的存储设备的 I/O 支持。

⑤ 消除或降低数据丢失或遭破坏的可能性。

⑥ 提供一个标准的 I/O 界面。

⑦ 在多用户系统中，向多个用户提供 I/O 支持等。

Linux 系统中每个分区都是一个文件系统，都有自己的目录层次结构。Linux 会将这些分属不同分区的、单独的文件系统按一定的方式形成一个系统的总的目录层次结构。一个操作系统的运行离不开对文件的操作，因此必然要拥有并维护自己的文件系统。

Linux 文件系统将文件索引节点号和文件名同时保存在目录中。所以，目录只是将文件的名称和它的索引节点号结合在一起的一张表，目录中每一对文件名称和索引节点号称为一个连接。对于一个文件系统来说，有唯一的索引节点号与之对应，对于一个索引节点号，却可以有多个文件名与之对应。

4.6.2 嵌入式 Linux 文件系统的组成

嵌入式 Linux 的文件系统和标准 Linux 文件系统的目录结构相似，包含如下目录（或更多目录）[34]：

- /bin (binary)：包含着所有的标准命令和应用程序；
- /dev (device)：包含外设的文件接口，在 Linux 下，文件和设备采用同种方法访问的，系统上的每个设备都在/dev 里有一个对应的设备文件；
- /etc (etcetera)：这个目录包含着系统设置文件和其他的系统文件，例如 /etc/fstab(file system table) 记录了启动时要 mount 的 filesystem；
- /home：存放用户主目录；
- /lib(library)：存放系统最基本的库文件；

- /mnt：用户临时挂载文件系统的地方；
- /proc：linux 提供的一个虚拟系统，系统启动时在内存中产生，用户可以直接通过访问这些文件来获得系统信息；
- /sbin：这个目录存放着系统管理程序，如 fsck、mount 等；
- /tmp(temporary)：存放不同的程序执行时产生的临时文件；
- /usr(user)：存放用户应用程序和文件；
- /root：超级用户主目录。

系统采用的是 cramfs 文件系统。

4.6.3 CRAMFS 文件系统

Cramfs（Compressed ROM File System）是 Linux 的创始人 Linus Torvalds 参与开发的一种只读的压缩文件系统。它是基于 MTD 驱动程序。

在 cramfs 文件系统中，每一页(4KB)被单独压缩，可以随机页访问，其压缩比高达 2:1,为嵌入式系统节省大量的 Flash 存储空间，使系统可通过更低容量的 FLASH 存储相同的文件，从而降低系统成本。

Cramfs 文件系统以压缩方式存储，在运行时解压缩，所以不支持应用程序以 XIP 方式运行，所有的应用程序要求被拷到 RAM 里去运行，但这并不代表比 Ramfs 需求的 RAM 空间要大一点，因为 Cramfs 是采用分页压缩的方式存放档案，在读取档案时，不会一下子就耗用过多的内存空间，只针对目前实际读取的部分分配内存，尚没有读取的部分不分配内存空间，当读取的档案不在内存时，Cramfs 文件系统自动计算压缩后的资料所存的位置，再即时解压缩到 RAM 中[35]。

另外，它的速度快，效率高，其只读的特点有利于保护文件系统免受破坏，提高了系统的可靠性。由于以上特性，Cramfs 在嵌入式系统中应用广泛。但是它的只读属性同时又是它的一大缺陷，使得用户无法对其内容对进扩充。

Cramfs 映像通常是放在 Flash 中，但是也能放在别的文件系统里，使用 loopback 设备可以把它安装别的文件系统里。

4.6.4 构建根文件系统

构建文件系统步骤如下[36][37][38]：创建一个目标板的空根目录，然后构建根

文件系统，创建基础目录结构，存放交叉编译后生成的目标应用程序（BUSYBOX等），存放库文件等。

①创建一个空目录

```
#mkdir rootfs
#cd rootfs
#mkdir dev proc etc sbin bin lib mnt usr tmp
#mkdir /mnt/etc
#mkdir usr/bin usr/sbin
#touch linuxrc
```

②指定交叉编译器的路径，交叉编译 busybox，获得 Linux 系统所需的命令[39]

```
#make menuconfig
#make install
```

③配置 linuxrc 文件

```
#vi linuxrc
!/bin/sh
/sbin/insmod   /tmp/adc.ko
/bin/mknod /dev/adc c 253 0
/bin/mount –f –t cramfs –o remount,ro /dev/mtdblock/2 /
exec /sbin/init
```

④将需要的应用程序拷贝到 rootfs 目录下，例如目录名为 rootfs，运行如下命令就可以生成一个 cramfs 文件系统的 image 文件。

```
#mkcramfs rootfs rootfs.cramfs
```

⑤将生成的 cramfs 镜像文件通过 bootloader 加载到 NAND FLASH 上。设置 linux 的启动模式为：

```
noinitrd root=/dev/mtdblock2 ro init=/linuxrc console=ttySAC0,115200
```

重启系统后，内核就加载在 NAND FLASH 第三个分区的 cramfs 文件系统。

第五章 基于QT库的图形界面设计与实现

5.1 基于QT的图形界面开发概述

5.1.1 QT简介

Qt是Trolltech公司开发的一个多平台的C++图形用户界面应用程序框架。Qt采用了完全面向对象组件编程技术，Qt也是流行的Linux桌面环境KDE的基础，并且支持多平台。同时Qt提供图形用户界面在嵌入式系统上开发的一系列开发工具包。该图形系统最大的优点是采用面向对象设计，移植性好，基于X Window的Qt桌面应用程序可以非常方便的移植到嵌入式系统上[37]。

Qt具有以下优点[40]：

① 优良的跨平台特性：Qt支持多种操作系统；

② 面向对象：Qt的良好封装机制使得的Qt模块化的程度非常高，可重用性较好，对于用户开发来说非常方便。Qt提供了一种称为信号/槽（Signals/Slots）的安全类型来替代回调函数，这使得各个组件之间的协同工作变得十分简单。

③ 丰富的API：Qt包括多达250个以上的C++类。

④ 支持2D/3D图形渲染，支持OpenGL和XML。

⑤ 拥有大量的开发文档。

Qt是以工具开发包的形式提供给开发者的，这些工具开发包包括了图形设计器，Makefile制作工具，字体国际化工具，Qt的C++类库等等；同时Qt的类库是支持跨平台的类库，也就是说Qt类库封装了适应不同操作系统的访问细节，这正是Qt的魅力所在[12]。

目前，Qt可以支持的操作系统平台如下[12]：

- MS/Windows 95、Windows 98、WindowsNT 4.0、Windows 2000、Windows XP；
- Unix/X11 Linux、Sun Solaris、HP-UX、Compaq True64Unix、IBM AIX、SGI IRIX和很多其它X11平台；
- Macintoshi Mac OSX；
- 嵌入式的，包含有FramBuffer的Linux平台。

5.1.2 QT/Embedded 简介

Qt/Embedded 是在嵌入式环境下所使用的 Qt。目前市面上所有上市的 Linux PDA 都是采用 Qt/Embedded 作为图形接口的函数库。Qt/Embedded 的特性是可以直接在 Framebuffer 上显示图形接口，反应的速度更快，这对硬件和容量都有限制的嵌入式环境非常重要[41]。

图 5-1 是 Qt/Embedded 的实现结构。

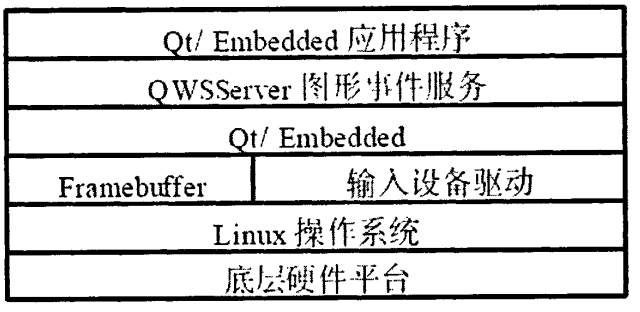

图5-1 Qt/Embedded实现结构图

许多基于 Qt 的 X Window 程序可以非常方便地移植到 Qt/Embedded 上，与 X11 版本的 Qt 在最大程序上接口兼容，延续了在 X 上地强大功能，在底层彻底摒弃了 X lib，仅采用 framebuffer 作为底层图形接口。Qt/Embedded 类库完全采用 C++封装。丰富的空间资源和较好的可移植性是 Qt/Embedded 最为优秀的一方面，使用 X 下的开发工具 Qt Designer 可以直接开发基于 Qt/Embedded 的 UI（用户操作接口）界面[42]。

嵌入式 Qt 的 API 大致可以分为三个部分：控件、框架和工具。控件部分大都是一些可见的 GUI 类，主要包括：主窗口、标准对话框、基本控件、扩展控件、GUI 组织以及环境类等。框架部分包括一些抽象的类，通常是不可视的，主要包括：对象模型、抽象空间、绘图类、拖放类、控件外观类等。工具部分包括日期、时间以及链表和树等数据结构，它们和 GUI 无关，主要用来方便编程。

5.1.3 QT/Embedded 开发环境简介

Qt/Embedded应用程序的开发可以在安装了一个跨平台开发工具链的不同的平台上编译。系统采用的是在 Linux 平台下开发，在 Linux 平台下以虚拟缓冲帧

的方式来运行，其实是有一个 X11 的应用程序虚拟了一个缓冲帧。通过指定显示设备的宽度，高度和颜色深度，虚拟出来的缓冲帧将和物理的显示设备在每个像素上保持一致。这样每次调试应用时开发人员就不用总是刷新嵌入式设备的 FLASH 存储空间，从而加速了应用的编译、链接和运行周期[43]。

运行 Qt 的虚拟缓冲帧工具的方法是：在 Linux 的图形模式下运行命令：

qvfb (回车)

当 Qt 嵌入式的应用程序要把显示结果输出到虚拟缓冲帧时，我们在命令行运行这个程序时，在程序名后加上-qws 的选项。例如： $> hello -qws

Qt 包含了许多支持嵌入式系统开发的工具，其中一些工具我们会在别的地方介绍。最实用的工具（除了上面我们提到的虚拟缓冲帧）有 tmake、Qt designer(图形设计器)、uic 和 moc[44][45]。

① tmake

tmake 是一个为编译 Qt/Embedded 库和应用而提供的 Makefile 生成器。它能够根据一个工程文件（.pro）产生不同平台下的 Makefile 文件。

开发者可以使用 Qt 图形设计器可视化地设计对话框而不需编写一行代码。使用 Qt 图形设计器的布局管理可以生成具有平滑改变尺寸的对话框，tmake 和 Qt 图形设计器是完全集成在一起的。

② Qt designer

Qt Designer 是设计窗口组件（Widget）的应用程序，在安装 Qt 的 bin 目录下输入./designer 命令，就启动一个包含很多 Qt 组件的可视化界面。在此组织应用程序的各组建分布很方便，最后生成一个 file.ui 和 main.cpp 文件；file.ui 是用 XML 语言编写的一个文本。

③ uic（user interface compiler）

uic 是从 XML 文件生成代码的用户界面编译器，用来将 file.ui 文件生成 file.h 和 file.cpp 文件（命令如：uic -o file.h file.ui　uic -o file.cpp file.ui），但生成的这两个文件不是标准的纯 C++代码，通常称为 Qt 的 C++扩展，因为 Qt 的对象间运用了信号和插槽的通信机制，在文件中用 Q_OBJECT 宏来标识。

④ moc（元对象编译器）

moc 用来解析一个 C++文件中的类声明并且生成初始化对象的 C++代码，moc 在读取 C++源文件，如果发现其中一个或多个类的声明中含有 Q_OBJECT 宏，就给出这个使用 Q_OBJECT 宏的类生成另外一个包含元对象代码的 C++元文件；元对象代码对信号插槽机制、运行时的类型信息和动态属性系统是需要的。

使用 Qt/Embedded 开发一个嵌入式应用的过程大体可用下面的流程如图 5-2 表示：

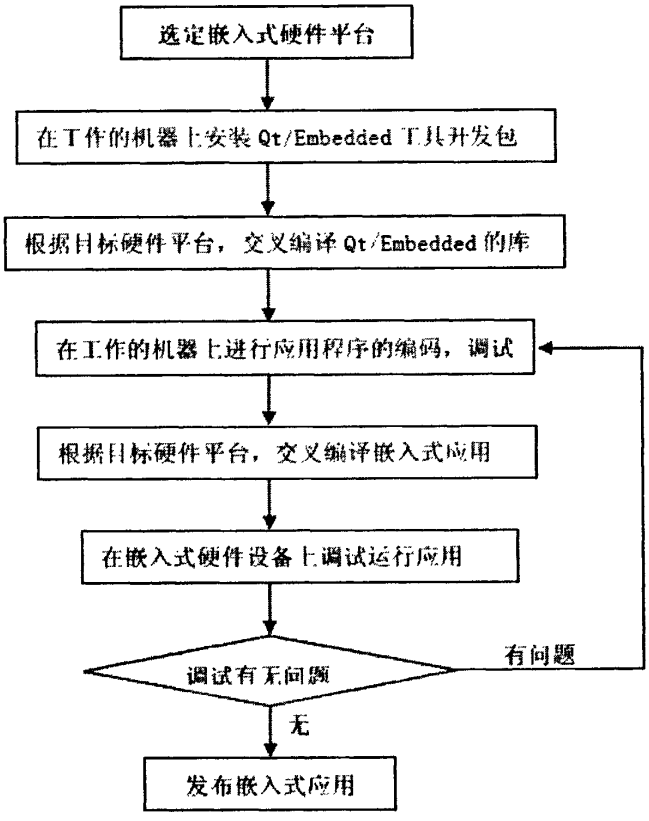

图5-2 QT/E应用程序开发流程图

5.1.4 信号与插槽

Qt/Embedded使系统获得高效的工作性能是它拥有一个重要的机制－信号与插槽机制。这个机制提供了对象间的通信机制，它易于理解和使用，并完全被 Qt 图形设计器所支持。

图形用户接口的应用需要对用户的动作做出响应。例如，当用户点击了一个菜单项或是工具栏的按钮时，应用程序会执行某些代码。大部分情况下，我们希

望不同类型的对象之间能够进行通信。程序员必须把事件和相关代码联系起来，这样才能对事件做出响应。以前的工具开发包使用的事件响应机制是易崩溃的，不够健壮的，同时也不是面向对象的。Trolltech 已经创立了一种新的机制，叫做“信号与插槽”。信号与插槽是一种强有力的对象间通信机制，它完全可以取代原始的回调和消息映射机制；信号与插槽是迅速的，类型安全的，健壮的，完全面向对象并用 C++来实现的一种机制[46][47]。

在以前，当我们使用回调函数机制来把某段响应代码和一个按钮的动作相关联时，我们通常把那段响应代码写成一个函数，然后把这个函数的地址指针传给按钮，当那个按钮被按下时，这个函数就会被执行。对于这种方式，以前的开发包不能够确保回调函数被执行时所传递进来的函数参数就是正确的类型，因此容易造成进程崩溃，另外一个问题是，回调这种方式紧紧的绑定了图形用户接口的功能元素，因而很难把开发进行独立的分类。

Qt 的信号与插槽机制是不同的。Qt 的窗口在事件发生后会激发信号。例如一个按钮被点击时会激发一个“clicked”信号。程序员通过建立一个函数（称作一个插槽），

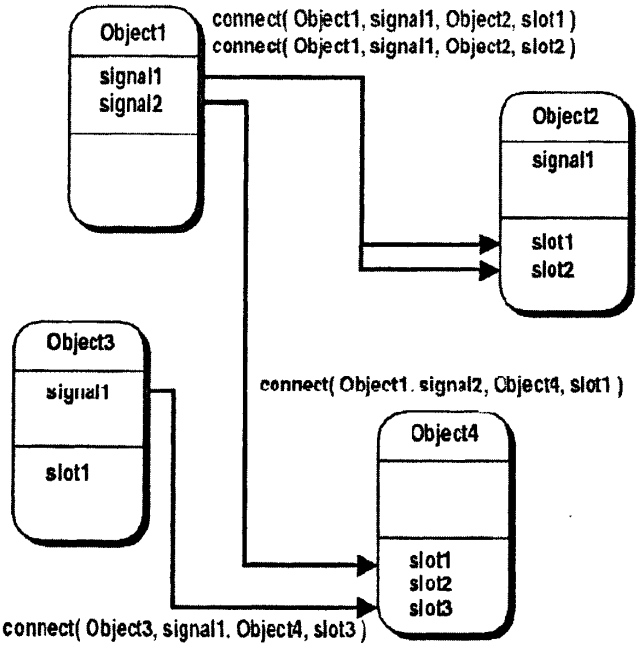

图5-3 信号与插槽链接的抽象图

然后调用connect()函数把这个插槽和一个信号连接起来，这样就完成了一个事件和响应代码的连接。信号和插槽的链接方式如图5-3所示。信号与插槽机制并不要求类之间互相知道细节，这样就可以相对容易的开发出代码可高重用的类。信号与插槽机制是类型安全的，它以警告的方式报告类型错误，而不会使系统产生崩溃[46][47]。

例如，如果一个退出按钮的 clicked() 信号被连接到了一个应用的退出函数－quit() 插槽。那么一个用户点击退出键将使应用程序终止运行。上述的连接过程用代码写出来就是这样

connect(button, SIGNAL(clicked()), qApp, SLOT(quit()));

我们可以在Qt应用程序的执行过程中增加或是减少信号与插槽的连接。信号与插槽的实现扩展了C++的语法，同时也完全利用了C++面向对象的特征。信号与插槽可以被重载或者重新实现，它们可以定义为类的公有，私有或是保护成员。

信号：当对象的内部状态发生改变，信号就被发射。只有定义了一个信号的类和它的子类才能发射这个信号。

例如，一个列表框同时发射highlighted()和activated()这两个信号。绝大多数对象也许只对activated()这个信号感兴趣，但是有时想知道列表框中的哪个条目在当前是高亮的。如果两个不同的类对同一个信号感兴趣，可以把这个信号和这两个对象连接起来。当一个信号被发射，它所连接的槽会被立即执行，就像一个普通函数调用一样。信号/槽机制完全不依赖于任何一种图形用户界面地事件回路。当所有的槽都返回后emit也将返回。

如果几个槽被连接到一个信号，当信号被发射时，这些槽就会被按任意顺序一个接一个地执行。

槽：当一个和槽连接的信号被发射的时候，这个槽被调用。槽也是普通的C++函数并且可以像它们一样被调用；唯一的特点是它们可以被信号连接。槽的参数不能含有默认值，并且和信号一样，为了槽的参数而使用自己特定的类型是不明智的。槽和普通成员函数一样有访问权限。一个槽的访问权限决定了谁可以和它相连。

一个public slots：区包含了任何信号都可以相连的槽。

一个 protected slots：区包含了之后这个类和它的子类的信号才能连接的槽。这些槽只是类的实现的一部分，而不是它和外界的接口。

一个 private slots：区包含了之后这个类本身的信号可以连接的槽。这就是说它和这个类是非常紧密的，甚至它的子类都没有获得连接权利这样的信任。

还可以把槽定义为虚的，这在实践中被发现是非常有用的。

信号与插槽机制是以纯 C++代码来实现的，实现的过程使用到了 Qt 开发工具包提供的预处理器和元对象编译器（moc）。

5.2 QT 库对触摸屏的支持

5.2.1 QT/E 加载触摸屏

Qt/Embedded 2.x 系列对于输入设备的底层接口与 3.x 系列不同，触摸屏设备和键盘设备需要根据具体的驱动程序接口在 Qt/Embedded 中设备实现对应的设备操作类。其中对应于鼠标类设备的实现位于 src/kernel/qwsmouse_qws.cpp 中。

在文件 qwsmouse_qws.cpp 中添加对触摸屏的支持。具体修改如下[46][49][50]：

①定义和 Linux 内核文件 driver/input/tsdev.c 中数据结构 ts_event 相一致的 TS_EVENT 数据结构，定义如下：

```
#if defined(QT_QWS_IPAQ)
typedef struct {
        short pressure;
        short x;
        short y;
        short millisecs;
} TS_EVENT;
```

②修改校准文件的位置

在函数 void QCalibratedMouseHandler::writeCalibration()和

void QCalibratedMouseHandler::readCalibration()中修改如下：

```
Qstring calFile = "/tmp/pointercal";
```

③对打开的设备文件进行修改

在函数 QTPanelHandlerPrivate::QTPanelHandlerPrivate 中，修改如下：

mouseFD = open("/dev/input/ts0" ,O_RDONLY | O_NDELAY) ;

④由于内核 TS_EVENT 结构中，当触摸屏按下时对 pressure 的设置为 1，因此在 void QTPanelHandlerPrivate::readMouseData()函数中把

if(data->pressure>=QT_QWS_TP_PRESSURE_THRESHOLD)

修改为：

if(data->pressure)

5.2.2 触摸屏的校准

由于触摸屏在实现原理上存在着 A/D 量化误差的问题，因此所有的触摸屏接口实现类需要从特殊的 QcalibratedMouseHandler 继承，并获得校正功能。

触摸屏上点击范围内的数据，称为触摸屏的显示数据，其范围内的点可以用显示坐标表示；触摸屏点击后产生模拟数据通过 A/D 转换器后生成的数据量数据，称为触摸屏的物理数据，对应的点可以用物理坐标来表示。触摸屏校准的工作过程就是将物理数据转换为显示数据并加以显示的过程。由于本系统采用 10 位 A/D 转换器，其物理数据的 x，y 坐标都是在 0~1023 范围内，又由于临界点误差较大，所以一般采用 100~1000 之间的数据为宜；系统中采用 320×240 的触摸屏，所以显示数据的 x，y 坐标分别在 0~320 和 0~240 范围内[51]。

常用的校准方法就当屏幕上依次出现 topleft、bottomleft、bottomright、topright 和 center 这 5 个点时，用户必须在这 5 个点上点击，这样就可以得到物理点的数据，而显示点是已知的，这样就可以确定它们之间的转换系数。计算公式如下[51][52][53][54]：

```
div = 1<<16;
a = div*(d_tl.x-d_br.x)/(p_tl.x-p_br.x);
b = div*d_tl.x-a*p_tl.x;
c = div*(d_tl.y-d_br.y)/(p_tl.y-p_br.y);
d = div*d_tl.y-c*p_tl.y;
```

式中，d_tl、d_br 分别表示显示坐标的左上角和显示坐标的右下角；p_tl、p_br 分别表示物理坐标的左上角和物理坐标的右下角。a、b、c、d 分别为待定

系数，div 是公因子。下面是确定系数之后由物理坐标得到显示坐标的公式：

d.x = (a*p.x+b)/div;

d.y = (c*p.y+d)/div;

系统通过校准后，获得校准参数为：-17792、17284992、-24066、23288556、65536。把这些参数写入校准文件/tmp/pointercal。由于就实现了触摸屏的校验。

5.3 QT 在 ARM 板上的移植

5.3.1 安装 QT/Embedded 工具开发包

在一台装有 Linux 操作系统的机器上建立 Qt/Embedded 开发环境。开发环境建立过程如下[12][46][48]：

①安装 tmake

在 Linux 命令模式下运行以下命令：

#tar xfz tmake-1.11.tar.gz

#export TMAKEDIR=$PWD/tmake-1.11

#export TMAKEPATH=$TMAKEDIR/lib/qws/linux-x86-g++

#export PATH=$TMAKEDIR/lib:$PATH

②安装 Qt/Embedded 2.3.7

在 Linux 命令模式下允许以下命令：

#tar xfz qt-embedded-2.3.7.tar.gz

#cd qt-2.3.7

#export QTDIR=$PWD

#export QTEDIR=$QTDIR

#export PATH=$QTDIR/bin:$PATH

#export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

#./configure –qconfig –qvfb –depths 4,8,16,32

#make sub-src

③安装 Qt/X11 2.3.2

在 Linux 命令模式下允许以下命令：

#tar xfz qt-x11-2.3.2.tar.gz

```
#cd qt-2.3.2
#export QTDIR=$PWD
#export PATH=$QTDIR/bin:$PATH
#export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
#./configure –no-opengl –no-xfs
#make
#make –C tools/qvfb
#mv tools/qvfb/qvfb    bin
#cp bin/uic $QTEDIR/bin
```

5.3.2 交叉编译 QT/Embedded 库

开发居于 Qt/Embedded 的应用程序要使用到 Qt/Embedded 的库，编写的 Qt 嵌入式应用程序最终是要在 ARM 开发板上运行的，因此在把 Qt 嵌入式应用程序编译成支持 ARM 的目标代码之前，需要经过 ARM9 的 Linux 编译器编译过的 Qt/Embedded 库。

Qt/Embedded 的安装选项允许我们自己定制一个配置文件，来有选择的编译 Qt/Embedded 库，这个安装选项是“-qconfig local”。当我们指定这个选项时，Qt/Embedded 库在安装过程中会寻找 qt-2.3.7/src/tools/qconfig-local.h 这个文件，如果找到这个文件，就会以该文件里面定义的宏来编译链接 Qt/Embedded 库。

系统要使用到触摸屏，为了支持对触摸屏的显示，我们需要定义一些宏，我们把这些宏定义到一个名为 qconfig-local.h 的安装配置文件中，这个配置文件定义了宏 QT_QWS_IPAQ 和 QT_NO_QWS_CURSOR。

在 Linux 命令模式下允许以下命令：

```
#tar xfz qt-embedded-2.3.7.tar.gz
#cd qt-2.3.7
#export QTDIR=$PWD
#export QTEDIR=$QTDIR
#cp qconfig-local.h    ./src/tools
#make clean
```

#./configure –xplatform linux-arm-g++ -shared –debug –qconfig local –qvfb –depth 4,8,16,32

#make

5.3.3 移植 QT/Embedded 库

在 ARM 板的根文件系统的 tmp 目录下创建新目录 qt，在 qt 目录下创建新目录 lib。把交叉编译后生成的 Qt/Embedded 库拷贝到 ARM 板上的根文件系统下的/tmp/qt/lib 目录下。在使用 QT 库之前要设置环境变量。命令如下：

#export QTDIR=/tmp/qt

#export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

这样就可以在应用程序中使用 QT 库了。

5.4 基于 QT 的应用程序开发

5.4.1 系统 QT 应用程序

系统使用 QT 作为图形界面开发库，主要实现显示脉搏波、通过按钮选择显示功能和定时器的使用。应用程序启动后，会生成一个带有三个按钮的界面。点击“start”按钮后，会画出坐标，并启动定时器。第一次点击“suspend”按钮，则停止定时器，由此可以查看到特定时刻的波形图，第二次点击按钮时重新启动定时器。点击“test”按钮，通过 messagebox 来显示通过脉搏波计算得到的心血管参数。系统使用按钮使用情况如图 5-4 所示：

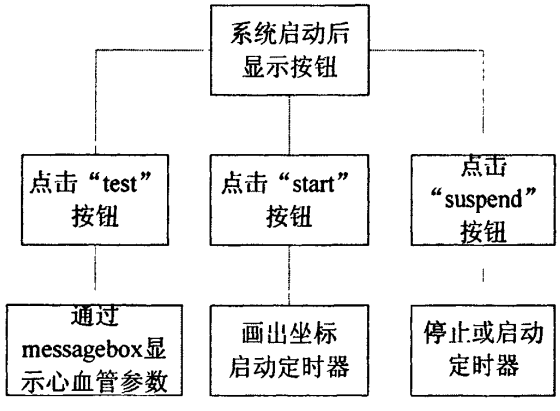

图 5-4 按钮使用状况图

系统使用了三个定时器，一个定时器用于采集从传感器采集获得的数据；另一个定时器用于显示经过滤波后信号的波形；由于经过滤波后得到的波形不一定就是脉搏波，应该还要经过脉搏波的检测，最后一个定时器就是用来显示确认后的脉搏波。定时器使用状态如图5-5所示。

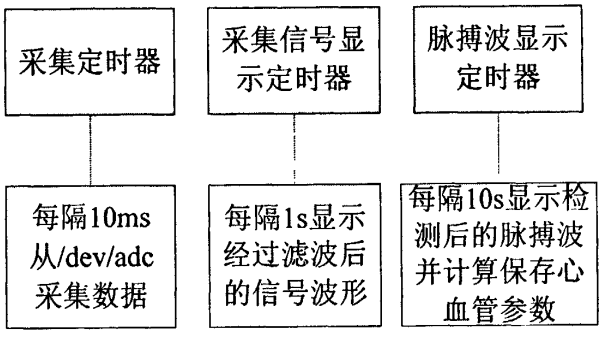

图5-5 定时器使用状况图

5.4.2 QT应用程序开发流程

具体开发一个Qt应用程序的过程如下[12][55]：

①生成一个工程文件（.pro文件）

一个应用通常对应一个工程文件，生成一个工程文件，并对它做一些简单的编辑，然后使用一个专门的工具（例如tmake）处理这个工程文件，就可以生成一个Makefile文件。

产生一个工程文件的其中一个方法是使用progen命令（progen程序可以在tmake的安装路径下找到）。下面是使用progen产生一个名为hello的工程文件的命令：

```
progen -t app.t -o project.pro
```

产生的project.pro工程文件并不完整，开发者还需手动往里添加工程所包含的头文件，源文件等信息。

②新建一个窗体

在qt-2.3.2的安装路径的bin目录下运行“./designer”命令，就启动了一个Qt图形编辑器。点击编辑器的“new”菜单，弹出了一个“new Form”对话框，在这个对话框里我们选择“Widget”，然后点击“OK”按钮，这样

我们就新建了一个窗体。接着，我们可以对这个窗体进行设置。

设置完成后，将其保存为 project.ui 文件，这个文件就是 project 窗体的界面存储文件。

③生成窗体类的头文件和实现文件

界面文件使用 uic 工具产生出窗体类的头文件和实现文件，例如 project.ui 节目文件产生 project 窗体类的头文件和实现文件，具体方法如下：

```
#cd qt-2.3.7/bin
#uic -o project.h project.ui
#uic -o project.cpp -impl project.h project.ui
```

这样我们就得到 hello 窗体类的头文件 project.h 和实现文件 project.cpp。接下来根据我们要实现的具体功能，在 project.cpp 文件里添加相应的代码。

④编写主函数 main()

一个 Qt/Embedded 应用程序应该包含一个主函数，主函数所在的文件名是 main.cpp。主函数是应用程序执行的入口点。

⑤编辑工程文件

在工程文件中添加窗体类的头文件、实现文件和主函数文件。

⑥生成 Makefile 文件

编译器是根据 Makefile 文件内容来进行编译的，所以需要生成 Makefile 文件。Qt 提供的 tmake 工具可以帮助我们从一个工程文件（.pro 文件）中产生 Makefile 文件。从工程文件生成一个 Makefile 文件的具体做法如下：

把环境变量$TMAKEPATH 所指的目录设置为 arm 编译器的配置目录，把当前 QTDIR 环境变量指向 Qt/Embedded 的安装路径，这样就可以使用 tmake 来生成 Makefile 文件。

```
#export TMAKEPATH=/tmake 安装路径/qws/linux-arm-g++
#export QTDIR=.../qt-2.3.7
#tmake -o Makefile hello.pro
```

当前目录生成的 Makefile 文件，需要进行一些修改，具体修改如下：

将 LINK = arm-linux-gcc 改为 LINK = arm-linux-g++

⑦编译链接整个工程

在命令行下输入 make 命令对整个工程进行编译链接。

#make

这样 make 生成的二进制文件就可以在 ARM 上允许了。

第六章　系统运行结果与分析

6.1 心血管参数分析

利用双弹性腔模型研究脉搏波结果表明，人体脉搏波波形特征随血管外周阻力和血管壁硬化程度等生理因素的变化将出现一系列规律性的变化。当血管弹性较好、外周阻力较低时，波形特征是主波窄而高、重搏前波不明显、重搏波峰和波谷很明显，这时的 K 值约在 0.35 左右，是健康的表现。随着外周阻力和血管壁硬化程度的增加，波形的重搏前波变为明显，它相对于主波的位置也升高，并自前向后与主波接近并呈现一定程度的融合，甚至超过主波。同时重搏波峰与波谷相对主波的位置抬高，与主波混为一体不易区分，使整个脉搏波波形呈馒头形。这时的 K 值在 0.45 左右，是非健康状态的表现。

同时根据临床实验结果也表明：年轻健康人、运动员或怀孕妇女的血管阻力低、动脉弹性较好，没有重搏前波、重搏波谷很低，其 K 值约为 0.33；健康的中青年人的血管阻力和动脉弹性适中，重搏前波与重搏波都不很明显，其 K 值约在 0.34～0.39 之间；中老年人的血管阻力较高、动脉弹性较差，重搏前波高度几乎与主波持平，其 K 值约为 0.4 左右；严重高血压和动脉粥样硬化的患者的血管阻力极高、动脉弹性极差，重搏前波与重搏波与主波融合，脉搏波形呈馒头状，其 K 值约在 0.45～0.5 之间。由此可见，随年龄增加或高血压、动脉粥样硬化的发展，血管阻力增加、脉搏波形由陡直的波形逐渐发展为馒头型波形，波形系数 K 值也相应增加（一般在 0.35～0.5 之间变化）[4]。

由此可见，K 值的大小和变化能够从宏观上描述出脉搏波波形变化的平均特征，并使外周阻力、血管壁硬化程度等生理参数用 K 值进行量化，具有重要的医学应用价值。

心率（HR）是单位时间内心脏博动的次数。正常成年人安静时的心率有显著的个体差异，平均在 75 次/分左右（60～100 次/分之间）。心率也因年龄、性别及其他生理情况而不同[1]。

心博出量（SV）是心脏每博动一次的输出血量[1][4]。正常范围大致为：60～110mL。

心输出量（CO）是心脏每分钟博动的输出血量[1][4]。正常范围大致为：3～

8L/min。

外周阻力（TPR）是人体外周血管的总阻力[1][4]。正常范围大致为：9～12。

血管顺应性（AC）反映血管弹性的量[1][4]。正常范围大致为：1.2～2.0mL/mmHg。

6.2 系统运行结果及分析

系统启动后的运行应用程序，显示界面如图 6-1 所示：

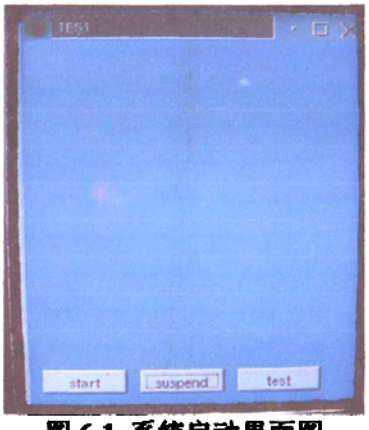

图 6-1 系统启动界面图

点击“start”按钮，则将画出坐标。在系统启动的初始，由于脉搏传感器采集到的信号比较不稳定，因此显示的波形不稳定。系统启动初始时，显示界面和波形图 6-2 所示：

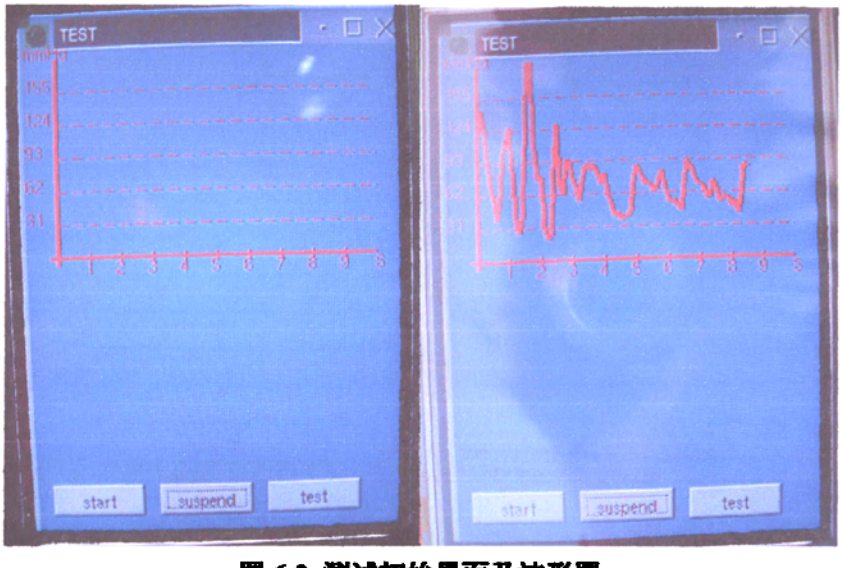

图 6-2 测试初始界面及波形图

系统对不同年纪和健康状况的测试者进行测试，有如下不同的运行结果。测试结果如下所示：

①健康的年轻人测试的结果如图 6-3 所示：

图 6-3 健康年轻人测试结果图

②患有高血压的中年人的测试结果如图 6-4 所示：

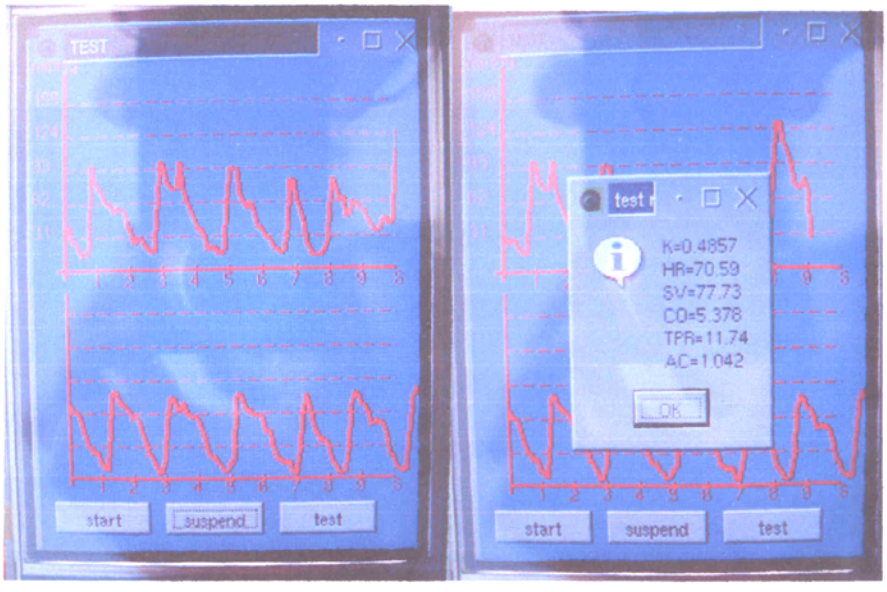

图 6-4 高血压中年人测试结果图

测试结果表明：系统测试结果和目前研究的结构基本一致。

第七章 总结与展望

7.1 总结

随着人口年龄结构的老龄化，心血管疾病特别是高血压、动脉硬化疾病的发病率和死亡率较 30 年前有了明显提高，据统计世界上有三分之一的人死于这类疾病，很多病人由于没有及时发现病变延误了治疗而死于非命。因此，对人体心血管参数的预测，能及时的预防和治疗心血管疾病，具有重要的意义。

便携式检测仪器是医疗检测仪器发展的一个重要方向，具有广阔的发展空间。同时嵌入式系统具有体积小、功耗低、处理能力强等特点决定了它非常适合于便携式检测设备的开发。因此系统实现了嵌入式心血管检测系统。系统通过脉搏传感器采集到脉搏信号，然后通过处理脉搏信号来计算出心血管参数，从而能够有效判断心血管功能状态，对于医疗保健事业具有重要的意义。

本文主要完成以下工作：

①研究了以 ARM9 为内核的微处理器结构和技术特点，阐述了本文采用的以 S3C2410 为核心的 ARM920T 微处理器的硬件平台体系结构和存储空间分配。

②在硬件平台上，建立了交叉编译环境并建立了基于 NFS 的调试环境。分析并移植了 bootloader。

③分析了 Linux 内核体系结构和特点，选中了 Linux2.6 版本的内核。根据硬件平台进行了适当的裁减和修改，然后交叉编译移植到硬件平台上。

④阐述了嵌入式 Linux 操作系统下设备驱动程序开发的步骤，详细叙述了字符型驱动的开发，具体介绍了网卡、LCD、A/D 和触摸屏驱动的开发和移植。

⑤分析了采集到的脉搏信号存在的干扰，并叙述了对信号的处理方法和处理过程。

⑥根据系统的需求和特点，采用了基于 QT 的图形界面的设计与实现。主要阐述了 QT 图形库的特点、QT/E 库在硬件平台上的移植过程以及基于 QT 的应用程序的开发过程。

目前，基本完成了系统的开发和测试，达到了预期的目的。

7.2 展望

由于时间和条件的限制以及个人能力有限，本文所作的工作还有很多不足，拟在今后的工作中对以下几个方面的问题进行进一步的研究：

①压力脉搏传感器从桡动脉采集的脉搏波与传感器放置的位置和施加的外在压力有较大的关系，故不具备较好的重复性和稳定性。因此可以考虑采用光电容积脉搏波描记法。

②系统采用滑动平均滤波法对采集的信号进行处理，虽然可以获得较好的脉搏波形，但是为了获得更为准确的脉搏波形，更好地消除基线漂移，可以考虑尝试其他的滤波方法。

③系统未实现电源模块，因此还要通过外部来供电。

④系统采用 A/D 采集和触摸屏共享 ADC 控制器的情况，在触摸屏使用不频繁的时候可以较好地采集脉搏波，但是为了提供系统的稳定性，可以考虑扩展一个 ADC 控制器用来专门采集脉搏信号。

⑤可以考虑扩展网络模块，通过无线网络来传输测试结果，并且当心血管参数出现异常时，可通过无线网络来通知医院和患者家属。

⑥可以考虑扩展语音模块，当心血管参数异常时，通过语音来提醒检测者。

便携式医疗检测设备的检测精度和系统稳定性必定要从现有的系统不断发展改进，因此，本论文只是构建了一个简单的系统，希望能为后续的开发提供较好的基础。

[参 考 文 献]

[1] 全晓莉．心血管动力学参数检测方法的研究[D]．重庆大学硕士论文．2006.

[2] 刘娜．基于脉搏波的血压和心血管状态检测算法的研究[D]．浙江大学硕士论文．2004.

[3] 罗志昌，张松，杨益明．脉搏波的工程分析与临床应用[R]．科学出版社．2006.

[4] 刁越．脉搏波分析方法及心血管参数检测系统的研究[D]．北京工业大学硕士论文．2005.

[5] 颜庭柏．基于 ARM-Linux 的嵌入式数据采集和发布系统[D]．南京信息工程大学硕士论文．2007.

[6] 深圳市优龙科技有限公司著．sARM9 FS2410P 教学平台实验指导手册[P]．深圳市优龙科技有限公司．2005.

[7] 赵国义．基于 ARM 处理器的嵌入式 Linux 系统的研究及应用[D]．北京邮电大学硕士论文．2007.

[8] 戴丽．基于 ARM 平台嵌入式 Linux 系统的构建与应用程序[D]．合肥工业大学硕士论文．2007.

[9] 张积红，吴强．嵌入式 Linux 研究及其在 ARM 上的移植[J]．电脑知识与技术．2005，(8)：45－46.

[10] 李宇丽．基于 ARM 的嵌入式 Linux 系统的研究及应用[D]．西安电子科技硕士论文．2007.

[11] 合肥华科电子技术研究所著，HK-2000 系列集成化脉搏传感器[P]．合肥华科电子技术研究所．2004.

[12] 深圳市优龙科技有限公司著．ARM9 FS2410P 应用教程[P]．深圳市优龙科技有限公司．2005.

[13] 陈宁，冷建筑，宋文宁，崔丽娜．基于 ARM 的 CS8900A 网络电路设计及实现[J]．重庆科技学院学报(自然科学版)．2008，(2)：69－72.

[14] 殷海兵，钟晓桢．TM1300 DSP 系统以太网接口的设计[J]．单片机与嵌入式系统应用．2002，(11)：26－28.

[15] 2323．基于 S3C2410 的 TFT-LCD 驱动电路设计[EB/01]．http://www.avrw.com/article/art_101_4985.htm．2006-08-30.

[16] 杜冠．基于 ARM 的嵌入式 Linux 系统移植的研究与实现[D]．华中科技大学硕士论文．2006.

[17] 谢清，张建刚．基于 S3C2410 处理器的 ARM Linux 操作系统移植实现[J]．软件导刊．2007，(11)：51－52.

[18] 田家林，陈利学，寇向辉．LINUX 嵌入式操作系统在 ARM 上的移植[J]．微计算机信息．2007，(23)：60－61.

[19] 李绍勋，陈朔鹰，罗国良．Linux 2.6 内核测试及其到 ARM 嵌入式平台的移植[D]．电子质量．2005，(5)：5－7.

[20] 方红萍，钱照华．基于 Linux 的嵌入式平台下设备驱动程序的开发与实现[J]．计算机与现代化．2008，(1)：82－84.

[21] 何世烈，陈健．基于嵌入式 Linux 的设备驱动程序设计[J]．单片机与嵌入式系统应用．2007，(7)：65－67．

[22] 朱园．Linux 设备驱动程序的研究与开发[J]．仪表技术．2008，(2)：32－34．

[23] 邓鹏鹏．基于 Linux 嵌入式操作系统的设备驱动程序的研究[D]．华中科技大学硕士论文．2006．

[24] 胡科路，黄本雄．Linux 内核动态模块开发分析[J]．计算机与数字工程．2007，(2)：156－158．

[25] Jonathan Corbet，Alessandro Rubini，Greg Kroah-Hartman．Linux Device Drivers，3rd Edition[EB/01]．O'Reilly．2005．

[25] 林逸．S3C2410 网卡 CS8900A 驱动程序的移植及问题解析[EB/01]．http://www.gd-emb.org/detail/id-45203.htm．2008-02-26．

[26] 许庆丰．嵌入式 Linux 下彩色 LCD 驱动的设计与实现[EB/01]．http://www.dzkf.cn/html/qianrushixitong/2007/0523/2134.html．2007-05-23．

[27] 张义磊，安吉宁，仲崇亮，任爽，于涛，孙铁铮．ARM 芯片 S3C2410 驱动 TFT 的研究[J]．液晶与显示．2005，(1)：61－64．

[28] 王成儒，朱振涛．基于 ARM9 处理器 S3C2410 的 LCD 显示系统设计[J]．电子元器件应用．2006，(6)：82－83．

[29] 刘利国．S3C2410 LCD 驱动程序移植及 GUI 程序编写[EB/01]．http://www.dzkf.cn/html/qianrushixitong/2006/1228/1298_2.html．2006-12-28．

[30] 陈辰，韩秋实，徐小力．ARM 芯片 S3C2410 触摸屏驱动的研究与开发[J]．科学技术与工程．2006，(3)：327－330．

[31] 畅卫功，丁衷林．嵌入式 Linux 系统中触摸屏驱动的研究[J]．嵌入式操作系统应用．2007，(12)：103－105．

[32] psmax2008．触摸屏驱动移植[EB/01]．http://blog.21ic.com/user1/2712/archives/2006/16659.html．2006-5-18．

[33] 杜丽君．嵌入式 Linux 文件系统的构建[D]．电子科技大学硕士论文．2004．

[34] 李毅．嵌入式 Linux 设备驱动程序的设计与研究[D]．电子科技大学硕士论文．2007．

[35] 中国 IT 实验室收集整理．嵌入式 Linux 文件系统详细介绍[EB/01]．http://linux.chinaitlab.com/kernel/730220.html．2007-9-19．

[36] 张方樱．构建嵌入式 Linux 的根文件系统[J]．实验科学与技术．2008，(1)：50-51．

[37] 胡和智，刘军芳．嵌入式 Linux 文件系统的构造[J]．科技信息．2006，(4)，21．

[38] 吴娴．嵌入式 Linux 文件系统的设计和实现[J．计算机工程与应用．2005，(9)：111－112．

[39] 杨延军．用 busybox 制作嵌入式 Linux 的文件系统[J]．单片机与嵌入式系统应用．2005，(4)：8－10．

[40] 吕亮．基于 ARM 的便携式雷达终端设计[D]．南京理工大学硕士论文．2007．

[41] 王存健，张建正．嵌入式 Linux 下 Qt/Embedded 的应用[J]．计算机技术与发展．2006，(11)：179-181．

[42] 徐广毅，张晓林，崔迎炜，蒋文军．QT/Embedded 在嵌入式 Linux 系统中的应用[J]．单片机与嵌入式系统应用．2004，(12)：14-17．

[43] 白玉霞，刘旭辉，孙肖子．基于 Qt/Embedded 的 GUI 移植及应用程序开发[J]．电子产品世界．2005，(7)：98-100．

[44] 任善全，吕强，钱培德，杨季文．一个基于 Qt/Embedded 的嵌入式 Linux 应用程序的实现[J]．计算机应用于软件．2006，(2)：105-107．

[45] 王丽洁．嵌入式 Linux 的图形界面技术研究与实现[D]．国防科学技术大学硕士论文．2006．

[46] 张方辉，王建群．Qt/Embedded 在嵌入式 Linux 上的移植[J]．计算机技术与发展．2006，(7)：64-66．

[47] 高磊．嵌入式 Linux 系统开发中关键技术的研究与实现[D]．国防科学技术大学硕士论文．2005．

[48] 张协国．嵌入式 Linux 在 ARM9 上的移植研究与实现[D]．哈尔滨工程大学硕士论文．2007．

[49] 申伟杰，彭楚武，胡辉红．嵌入式 Linux 中基于 Qt/Embedded 触摸屏驱动的设计[J]．中国仪器仪表．2006，(4)：48－51．

[50] 唐威．为 Qt/Embedded 添加触摸屏驱动[J]．单片机与嵌入式系统应用．2004，(2)：72－73．

[51] 刘军良，潘刚，李平．嵌入式 Linux 下一种新的触摸屏定标方法的研究[J]．工业控制计算机．2006，(9)：42－43．

[52] 罗勇刚，夏定纯．电阻式触摸屏的校准与应用研究[J]．武汉科技学院学报．2007，(12)：47－49．

[53] 许荣斌，谢莹，朱永红．触摸屏校准常用算法分析[J]．工业控制计算机．2006，(4)：77－78．

[54] 王丁，闫瑶，张延宇．触摸屏校准的一种通用算法[J]．自动化技术与应用．2008，(2)：116－117．

[55] Jasmin Blanchette，Mark Summerfield．C++ CUI Programming with Qt 3[M]．Prentice Hall in association with Trolltech Press．2004．

硕士期间发表的论文和参与的项目

发表论文情况

[1] 杨剑萍，达力，王博亮 基于 ARM 的心血管参数检测系统的研究与设计. 科技创新导报（CN 11-5640/N）. 2007，67（31）. 3－4. 已发表.

科研项目

2007. 09～2008. 05

参与毕业课题“基于 ARM 的心血管参数检测系统的研究与实现”项目的开发.

致 谢

首先我要感谢我的导师达力副教授。在三年的研究生学习期间，导师精深的学术造诣，严谨求实的治学风范，正直诚信的人格魅力和省体力行的工作态度使我受益匪浅，并潜移默化地影响着我，促使我在学习中不断进步。同时还要感谢王博亮教授、黎忠文教授、杨晨辉教授、李绍滋教授和林凡老师一直以来对我的帮助、教导、启迪和关怀，我所取得的成绩和进步和您们是分不开的。在此，向辛勤培育我成长的所有老师们致以最崇高的敬礼和最诚挚的谢意。

感谢张晓军、林章渊、洪必海、杨一麟、蒋业逢、陈宏等师兄在学术上给我无私的指点和启发、在平常生活中给我及时的建议和帮助，你们都是我学习的榜样；感谢陈初铨、卢敏和许振明同学在我遇到学术问题的时候，总是第一时间给我解答和指导；感谢余宾、林炼、刘峰、陈琼、曾坤同学三年来在科研上共同努力，互相帮助，相互促进，相互学习；感谢陈国武师弟和实验室师弟师妹的关照和支持。

感谢“六人行”、宿舍所有同学三年来对我的关系和照顾，感谢你们一直鼓励和陪伴我。特别感谢李凡姐一直以来，在我难过的时候陪伴我，在我失落的时候给我信心，激励我，开导我，让我看到希望。还有黄荔丽同学总是以开朗乐观的心态给我帮助，影响我，鼓励我。

感谢班级所有同学三年来共同努力，共同奋斗，相互学习，相互帮助；感谢我的朋友风中云、Victor.Woo、阿 Vin 一直以来对我的支持和帮助。

感谢我的奶奶，爸爸，妈妈，哥哥，妹妹，一直以来总是给我最大的支持和鼓励，没有你们就没有我的一切。

感谢所有关心和帮助过我的人！