

## 摘 要

本文分析了当前车辆交通管理中的实际问题,介绍了一种车载终端的设计方法。设计采用 ARM9 微处理器构造的嵌入式系统,是集 GPS 全球卫星定位系统和 GPRS 无线通信技术于一体的新型车载电子产品。它为现代交通运输提供了新颖,可靠,有效的控制和管理途径。

车载终端通过将 GPS 模块的定位信息提取出来,一方面将定位信息在车载终端上显示,一方面又结合车辆的状态等信息发送给 GPRS 模块,发送出去的信息通过无线网络传输给车辆管理部门。车辆管理部门根据车辆的位置和状态等,采取一定的措施,从而实现车辆的有效管理。

本设计从硬件和软件两大部分出发,硬件上设计了 ARM 处理器、存储器、内存及其外围电路,另外还有 GPS 模块电路和 GPRS 模块电路;软件上采用 Qt 的人机界面完成数据显示与更新,采用 PPP 拨号脚本完成 GPRS 模块的拨号,通过 Qt 多线程编程的方法完成 GPS 数据的提取和 GPRS 的信息发送。在硬件和软件之间采用了嵌入式 Linux 系统,包括启动代码、内核和文件系统等。

论文的最后总结了所完成的工作,给出了设计的不足之处和有待完善的地方。

**关键词:** ARM, GPS, GPRS, 嵌入式 Linux, Qt

## ABSTRACT

This thesis analyzes the problems of vehicle-management at present. It gives a valid design method which is based ARM9 microprocessor and combines the GPS(Global Positioning System) and GPRS(General Packet Radio Service) in the newly product .It provides a new and efficient controlling and managing way for the transport in the modern cities.

The vehicle terminal obtains the positioning data from GPS module, extracts some useful messages that we need, displays them on the screen and sends out by GPRS module. The vehicle management-department receives the messages by networks, and then adopts some tactics to maintain the traffic order.

The design includes two parts: hardware and software. The hardware design is comprised of ARM, memory, SDRAM,GPS module circuit and GPRS module circuit while The software design adopts Qt for graphic user interface(GUI),utilizes multi-threads for extracting GPS Info and sending GPRS messages. Between the hardware and software is embedded Linux system, it includes Bootloader, kernel, file-system and so on.

At the end, the thesis summaries the completed works, and gives the methods for further improvement.

**Keyword:** ARM, GPS, GPRS, embedded Linux, Qt

## 声 明

本学位论文是我在导师的指导下取得的研究成果，尽我所知，在本学位论文中，除了加以标注和致谢的部分外，不包含其他人已经发表或公布过的研究成果，也不包含我为获得任何教育机构的学位或学历而使用过的材料。与我一同工作的同事对本学位论文做出的贡献均已在论文中作了明确的说明。

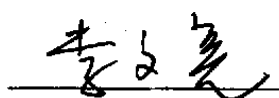
研究生签名：\_\_\_\_\_

年 月 日

## 学位论文使用授权声明

南京理工大学有权保存本学位论文的电子和纸质文档，可以借阅或上网公布本学位论文的部分或全部内容，可以向有关部门或机构送交并授权其保存、借阅或上网公布本学位论文的部分或全部内容。对于保密论文，按保密的有关规定和程序处理。

研究生签名：



年 月 日

## 1 绪论

### 1.1 车载定位系统的研究背景

随着我国经济的高速发展以及人民生活水平的提高,我国的汽车保有量在未来很长一段时间内将持续上涨。交通运输行业作为国家重点扶持的基础产业得到了快速高效的发展,遍布全国的高等级公路网络的建设,方便快捷的城市公共交通体系的建成,都显现出一番蓬勃的繁荣景象。与此同时,城市建设规模日益扩大,高速公路网络不断延伸,城市的车辆,驾驶员及交通流量大幅度增长,这对公安交通管理部门和交通运输部门的管理提出了更新,更高的要求。另外,对于各种车辆的行驶安全以及公路交通的畅通等问题,也逐渐成为当前所面临的急需解决的重要课题。对于上述等问题,必须找到一个行之有效的解决方案。由此,我们发现,只要知道汽车在某一时刻的有效位置,就可以找到解决相关问题的方法,车载 GPS/GPRS 系统集成全球卫星定位技术,现代移动通信技术于一体,对车辆位置和相关状态进行实时监控,有效地解决以上所提到的问题。

### 1.2 国内外的研究发展现状

车载卫星定位系统,属于智能交通系统(即通过运用先进的信息、通信和控制等高新技术对传统运输系统进行改造而形成的一种信息化、智能化和社会化的新型交通运输方式)的分支,在智能交通系统这一庞大的体系中,占有极其重要的地位。作为整个智能交通系统的基础,车载卫星定位系统的主要目的就是找出特定车辆在特定时间的位置。国外对于智能交通系统的研究,已经开展了三、四十年,美国,日本和欧洲等国都为此投入了大量的资金、人力和物力,取得了很多有价值的成果,并成为世界智能交通系统研究的三大基地。另外一些国家和地区对智能交通系统的研究也有相当大的规模,如澳大利亚、韩国、新加坡和香港等。目前可以说,全球正在形式一个新的智能交通系统产业,发展规模和速度惊人。

我国智能交通系统的发展起步较晚,二十世纪九十年代之前,主要是在一些大城市引进和消化城市交通信号控制系统。九十年代以后开始逐步应用 GPS 车辆监控调度系统。目前,我国 GPS 车载系统尚处于市场培育阶段,整体规模较小。相关的基础配套设施与资源正在完善之中,对于实时交通信息的发布和完善的电子导航地图的提供与更新机制,缺乏实际的解决方案和途径,因而市场推进困难重重。但面对庞大的汽车市场, GPS 车载系统的潜力不可估量,其发展前景比较乐观。

### 1.3 本课题的主要研究内容及意义

本论文主要是根据车载定位系统的要求,根据现在市场上出现的电子整机产品,参考相关的功能、性能、价格和实现难易程度等因素。使设计能够按照全面性、可操作性、可比性、系统性和代表性的原则,比较科学合理地选取硬件资源,辅以相应的软件设计,使设计具有一定的前瞻性、实用性和创新性。其成果对于进一步满足运输效率和安全保障的需要,对我国的国民经济建设有重要的现实意义。

根据要求,本设计主要研究了以下几部分内容:

- (1) 车载定位系统的整体方案。
- (2) ARM9 S3C2410 系列处理器、GPS 模块、GPRS 模块以及各部分模块的关系与作用。
- (3) 嵌入式 Linux 系统的移植。
- (4) GPS 输出的 NEMA0183 码及 GPRMC 语句。
- (5) GPRS 模块的拨号与应用。
- (6) Qt/E 的界面设计方法,以及如何来实现 GPS 数据的提取和 GPRS 信息的发送。
- (7) 将应用程序移植到 Qtopia 中去。

## 2 车载定位系统原理概述

### 2.1 车载定位系统概述

车载定位系统由车载定位终端、无线通信链路和车载监控管理系统三部分组成。其主要功能是将移动目标的动态位置（经度和纬度）、时间和状态等信息，通过无线通信链路传送至监控中心，而后在具有强大的地理信息查询功能的电子地图上进行移动目标运动轨迹的显示，并对目标的准确位置、速度、运动方向和车辆状态等用户感兴趣的参数进行监控和查询，为调度管理提供可视化依据，提高车辆的运营效率，并确保车辆的安全。

当前，可以使用的定位系统有美国的全球定位系统（GPS），俄罗斯的 GLONASS，中国的“北斗”计划定位系统等，其中应用最为广泛的，性能最稳定的，定位精度最高的是 GPS 定位系统，本设计采用 OEM 板 GPS 接收机模块，它能够接收卫星的相关数据，然后直接输出定位信息。

车载终端完成的功能就是利用 GPS 模块所接收到的定位数据，提取其中有用的信息，然后通过无线模块将信息发送到通信网络上。就目前来看，无线通信方式主要有 GSM、GPRS 和 CDMA 等方式。其中 GPRS 通信系统具有传输速率高，实时性好的优点，但成本较高。因为车载终端只要在启动之后，就必须和监控管理系统一直相连，一直进行数据传输，这样才能实时地监控车辆，所以车载终端和监控管理系统之间就需要良好的通信。目前车载终端主要采用短消息通信方式。因为 GPRS 是按照流量收费的，它可以一直和监控系统保持连接状态，而费用是按照信息量收费的。和 GSM 相比，它的实时性、突发性和性价比要高很多。本终端采用 GPRS 数据通信方式来完成定位信息的发送。

### 2.2 全球定位系统（GPS）

#### 2.2.1 GPS 全球定位系统的组成

GPS 是 Navigation Satellite Timing and Ranging Global Positioning System 的缩写词 NAVSTAR/GPS 的简称，GPS 是以卫星为基础的无线电导航定位系统，具有全能性、全球性、全天候、连续性和实时性的导航、定位和定时功能。GPS 是 1973 年 12 月美国国防部批准的海陆空三军联合研制的新的卫星导航系统，从 1973 年以来，GPS 经历了方案论证（1974~1978）、系统论证（1979~1987）和生产实验（1988~1993）三个阶段。

GPS 系统包括三大部分：空间部分（GPS 卫星星座），地面控制部分（地面监控系统），用户设备部分（GPS 信号接收机）。

### (1) 空间卫星部分

1) GPS 由 21 颗工作卫星和 3 颗备用卫星组成，工作卫星分布在 6 个轨道平面内，每个轨道面分布有 3~4 颗卫星，卫星轨道面相对地赤道面的倾角为  $55^\circ$ ，各个轨道平面升交点的赤经相隔  $60^\circ$ ，相邻轨道之间的卫星要彼此叉开  $30^\circ$ 。

2) GPS 卫星用 L 波段两种频率的无线电波（1575.42MHz 和 1227.6MHz）向用户发射导航定位信号，同时接收地面发送的导航电文以及调试命令。每个电波用导航信息  $D(t)$  和伪随机码（PRN），测距信号进行双相调制。用于捕获信号及粗略定位的伪随机码称 C/A 码，精密测距码称为 P 码。由导航电文可以知道该卫星当前的位置和卫星的工作情况。

3) 位于地平线以上的卫星颗数随着时间和地点的不同而不同，最少可见到 4 颗，最多可见到 11 颗。在用 GPS 信号导航定位时，为了计算观测点的三级坐标，必须观测 4 颗 GPS 卫星，称为定位星座。这 4 颗卫星在观测过程中的几何位置分布对定位精度有一定的影响。对于某地某时，甚至不能测得精确的点位坐标，这种时间段叫做“间隙段”。但这种时间间隙段是很短暂的，并不影响全球绝大多数地方的全天候、高精度、连续实时的导航定位测量。

### (2) 地面控制系统

对于导航定位来说，GPS 卫星是一动态已知点。星的位置是依据卫星发射的星历——描述卫星运动及其轨道的参数算得的。每颗 GPS 卫星所播发的星历，是由地面监控系统提供的。卫星上的各种设备是否正常工作，以及卫星是否一直沿着预定轨道运行，都要由地面设备进行监测和控制。地面监控系统另一重要作用是保持各颗卫星的时间，求出钟差。然后由地面注入站发给卫星，卫星再由导航电文发给用户设备。

地面控制部分包括一个主控站、三个注入站和五个监测站，实现对 GPS 卫星运行的监控。主控站位于美国的科罗拉多，三个注入站分别设在太平洋、印度洋和大西洋的三个美国军事基地内。监测站设在主控站、三个注入站和夏威夷岛。

### (3) 用户接收机

GPS 的空间卫星部分和地面监控部分是用户应用该系统进行导航定位的基础，而用户只有使用 GPS 接收机才能实现其定位、导航的目的。

GPS 信号接收机的任务是：能够捕获到按一定卫星高度截止角所选择的待测卫星的信号，并跟踪这些卫星的运行，对所接收到的 GPS 信号进行变换、放大和处理，以便测量出 GPS 信号从卫星到接收机天线的传播时间，解译出 GPS 卫星所

发送的导航电文，实时计算出测点的三维位置，甚至三维速度和时间。

接收机硬件和机内软件以及 GPS 数据的后处理软件包，构成完整的 GPS 用户设备，GPS 接收机的结构分为天线单元和接收单元两大部分。

### 2.2.2 GPS 定位的基本原理

GPS 地面观测点定位的方法有伪距定位法、多普勒定位法、载波相位定位法、伪距测量加多普勒定位法和干涉定位法等。常用的是前三种，多普勒定位法和载波相位法定位精度比伪距测量定位法高，但其成本造价也要高出许多。现在导航型 GPS 接收机主要采用伪距测量定位法，本设计采用的 OEM 板 GPS 接收机也是采用这种定位法。

伪距 (Pseudo-Range) 是 GPS 接收机对码的量测而得到卫星到接收机的距离，由于含有接收机卫星时钟的误差及大气传播误差，故称为 P 码伪距，精度约为 2m 左右。

在 GPS 观测中，我们可以得到卫星到接收机的距离，利用三维坐标中的距离公式，利用 3 颗卫星，就可以组成 3 个方程式，解出观测点的位置  $(x, y, z)$ 。考虑到卫星的时钟与接收机之间的误差，实际上有 4 个未知数， $x, y, z$  和钟差，因而需要引入第 4 颗卫星，构成 4 个方程式进行求解，从而得到观测点的经度、纬度和高度，如图 2.2.2 所示。

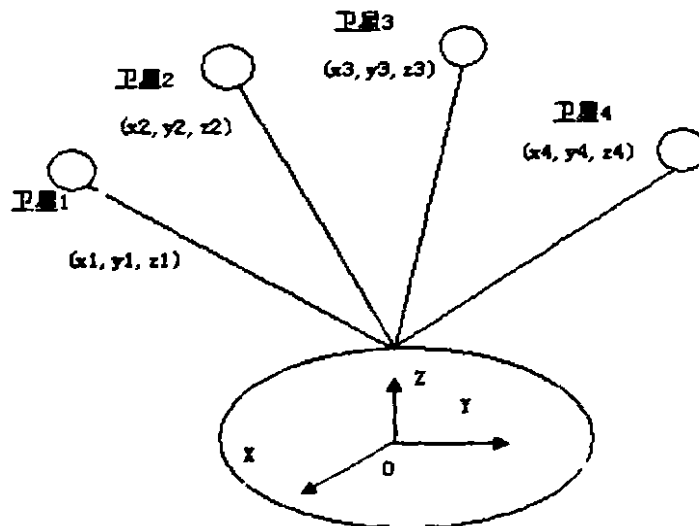


图 2.2.2 GPS 卫星定位原理

根据图 2.2.2 可以确定下面的方程式(2.2.2)：



$$\begin{cases} (x_1-x)^2+(y_1-y)^2+(z_1-z)^2+c^2(t-t_{s1})^2=d_1^2 \\ (x_2-x)^2+(y_2-y)^2+(z_2-z)^2+c^2(t-t_{s2})^2=d_2^2 \\ (x_3-x)^2+(y_3-y)^2+(z_3-z)^2+c^2(t-t_{s3})^2=d_3^2 \\ (x_4-x)^2+(y_4-y)^2+(z_4-z)^2+c^2(t-t_{s4})^2=d_4^2 \end{cases} \quad (2.2.2)$$

求解未知数  $(x, y, z, t)$ ，其中  $(x, y, z)$  是待测点坐标，即定位未知参数； $t$  为定时未知参数，是接收机的时钟差。 $c$  为 GPS 信号的传播速度（即光速）。 $d_1 \sim d_4$  是卫星 1~4 到定位点的距离， $(x_i, y_i, z_i)$  是每个卫星的星历参数，即卫星的轨道坐标  $(i=1, 2, 3, 4; j=1, 2, 3, 4; k=1, 2, 3, 4)$ ； $t_{s1} \sim t_{s4}$  是各个卫星的时钟差。接收机可以锁住 4 颗以上的卫星，接收机可按卫星的星座分布成若干组，每组 4 颗，然后通过算法挑选出误差最小的一组用作定位，从而提高精度。由于卫星运行轨道卫星时钟存在误差，大气对流层和电离层对信号的影响，以及人为的 SA 保护政策，使得民用 GPS 的定位精度只有 100m。为提高定位精度，普遍采用差分 GPS (DGPS) 技术，建立基准站（差分台）进行 GPS 观测，利用已知的基准站精确坐标，与观测值进行比较，从而得出一修正数，并对外发布。接收机收到该修正数后，与自身的观测值进行比较，消去大部分误差，得到一个较准确的位置。

## 2.3 通用分组无线业务 (GPRS)

GPS 接收机提供定位信息，而不能把定位信息发送给监控中心。因此车载终端和监控中心之间必须建立数据通信，使得定位信息能够及时、准确的发送出去。

### 2.3.1 GPRS 系统概述

GPRS (General Packet Radio Service) 是通用分组无线业务的简称，是 GSM 提供的分组交换和分组传输方式的新的承载业务。采用与 GSM 相同的频段、频带宽度、突发结构、无线调制标准、跳频规则以及相同的 TDMA 帧结构。它的基本原理是：当有数据传送需要时，将利用分组在网络中传送数据，而不是利用当前承载服务所采用的固定电路连接。这促进了多用户间对网络资源的共享，并允许运营商最优地使用现有设备，同时利用已安装的设备创造新的收入来源。使用 GPRS 时，数据封装进每个分组并在网上发送。网络容量仅在需要时分配，提供了即时连接和高通过率。GPRS 提供了非常灵活的波特率，从小于 100bit/s 到大于 171.2kbit/s，因此能处理从低速短消息到浏览复杂网站所需的高速传输。GPRS 允许用户在收发数据资料的同时接收电话。不过 GPRS 将要求空中接口和基站分系统两方面作改动，以便能进行此

分组模式的传输。另外, GPRS 需要新的网络组成部分, SGSN 和 GGSN, GPRS IP 骨干网以及新型终端。GPRS 对电子信箱和数据库接入业务较理想, 用户不必为短暂的传输付出高费用。

GPRS 被认为是 2G 向 3G 演进的重要一步, 它不仅能提供 PTP (点对点) 和 PTM (点对多点) 数据业务, 还能支持补充业务和短消息业务。

### 2.3.2 GPRS 特点

GPRS 采用分组交换技术, 在现在 GSM 网络上实现高速数据传输, 最高数据速率可达 171.2kbit/s。这样的数据速率, 可以极大地满足日益增长的移动数据业务需求, 并可与 Internet 有效融合, 实现移动 Internet, 因而有很强的生命力。其优势主要有以下几个方面:

(1) 资源利用率高: GPRS 引入了分组交换的传输模式, 使得原来采用电路交换模式的 GSM 传输数据方式发生了根本性的变化, 这在无线资源稀缺的情况下显得尤为重要。按电路交换模式来说, 在整个连接期内, 用户无论是否传送数据都将独自占有无线信道。而对于分组交换模式, 用户只有在发送或接收数据期间才占用资源, 这意味着多个用户可高效率地共享同一无线信道, 从而提高了资源的利用率。GPRS 用户的计费以通信的数据量为主要依据, 而不是连接时间, 体现了“得到多少, 支付多少”的原则。实际上, GPRS 用户的连接时间可能长达数小时, 却只需支付相对低廉的连接费用。

(2) 传输速率高: GPRS 可提供高达 115kbit/s 的传输速率 (最高值为 171.2kbit/s, 不包括 FEC)。这意味着通过便携式电脑, GPRS 用户能和 ISDN 用户一样快速地上网浏览, 同时也使一些对传输速率敏感的移动多媒体应用成为可能。

(3) 接入时间短: 分组交换接入时间缩短为少于 1 秒, 能提供快速即时的连接, 可大幅度提高一些事务(如信用卡核对、远程监控等)的效率, 并可使已有的 Internet 应用(如 E-mail、网页浏览等)操作更加便捷、流畅。

(4) 支持 IP 协议和 X.25 协议: GPRS 支持因特网应用最广泛的 IP 协议和 X.25 协议。而且由于 GSM 网络覆盖面广, 使得 GPRS 能提供 Internet 和其它分组网络系统的全球性无线接入。

(5) 基于以上的特点, GPRS 系统的采用能克服了电路交换型数据传输速率低, 资源利用率差等缺点, 特别适用于像 GPS 系统那样突发性数据的应用。其 GPRS/IP/TCP/UDP 协议栈可以满足数据的实时交换, 在现有的公用无线通信系统中拥有最大的带宽。与目前流行的短消息相比, 在相同数据长度, 相同时间间隔下通讯费用是短消息的 1/6, 甚至更少。

(6) 车载终端以 GPRS 为传输网络, 有效地解决了以往短消息传输方式延时大、系统实时性差的问题。GPRS 技术提供的高速传输速率和“永远在线, 按流量计费”的优点, 使该系统具有良好的适用性、可靠性和可扩展性, 而且易于管理与维护。

## 2.4 ARM 与嵌入式系统

### 2.4.1 ARM 体系结构

ARM (Advanced RISC Machines), 是一个公司名字, 也是一种处理器的通称, 还可以认为是一种技术名字。

ARM 处理器目前包括下面几个系列的处理器产品以及其他厂商实现的基于 ARM 体系结构的处理器。ARM7 系列、ARM9 系列、ARM9E 系列、ARM10E 系列、SecurCore 系列、Intel 的 Xscale 系列和 StrongARM 等系列。

这些处理器最高主频达到了 800MIPS, 功耗数量级为 Mw/MHz。对于支持同样 ARM 体系版本的处理器, 其软件是兼容的。

本设计采用是三星公司的 S3C2410 处理器, 属于 ARM9 系列, 有以下特点:

- 提供 1.1MIPS/MHz 的 5 级流水线结构
- 支持 32 为 ARM 指令和 16 位 Thumb 指令集
- 支持 32 位高速 AMBA 总线和接口
- 全性能 MMU, 支持 Linux、WinCE 等各种嵌入式操作系统
- 支持 Cache 和指令 Cache

### 2.4.2 嵌入式系统

嵌入式系统是嵌入到对象体系中的专用计算机系统, 以嵌入式计算机为核心的嵌入式系统是继 IT 网络技术之后, 又一个新的技术发展方向。嵌入式系统是以应用为中心, 以计算机技术为基础, 软硬件可裁剪, 适用于应用系统对功能、可靠性、成本上升、体积、功耗有严格要求的专用计算机系统。它一般由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户的应用程序等四个部分组成, 用于实现对其他设备的控制、监视或管理等功能。

嵌入式系统是专用计算机应用系统, 它具有一般计算机组成的共性, 也是由硬件和软件组成的。图 2.4.2<sup>[29]</sup>完整的描述了嵌入式系统的软硬件各部分的组成结构。

嵌入式系统的硬件是嵌入式系统软件环境运行的基础, 它提供了嵌入式系统软件运行的物理平台和通信接口; 嵌入式操作系统和嵌入式应用软件则是整个系统的

控制核心，控制整个系统的运行，提供人机交互的信息等。嵌入式系统的软件分为嵌入式操作系统和嵌入式应用软件两大部分。

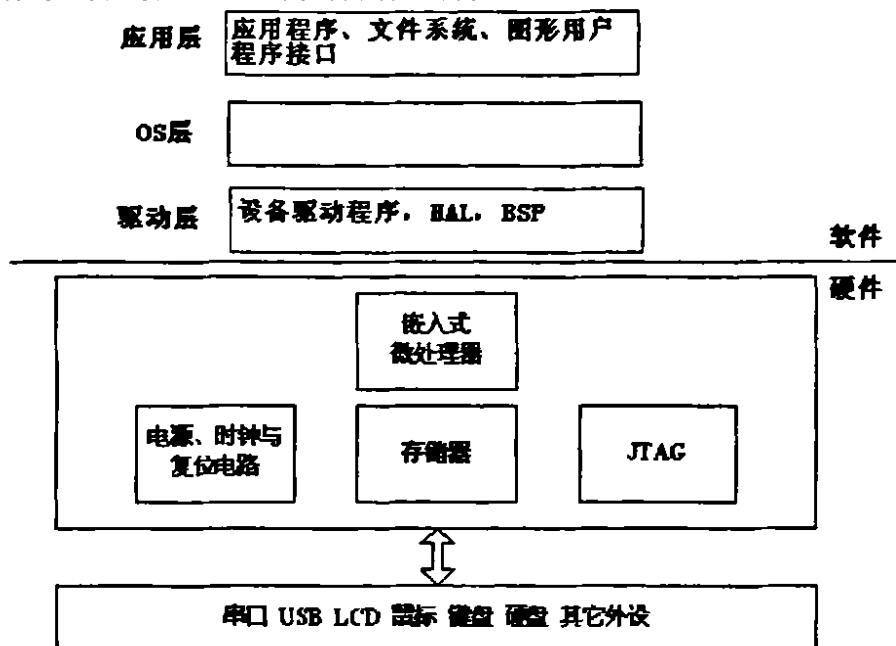


图 2.4.2 嵌入式系统框图

## 2.5 车载系统的组成与实现

车载定位系统的组成如图 2.5 所示。



图 2.5 车载定位系统总体框图

下面一部分是车载终端，上面的是 GPRS 通信网络、监控中心和 GPS 定位卫星。

GPS 接收模块接收 GPS 定位卫星的信号，然后解算出定位数据，车载终端对数据进行提取和格式转换，同时能根据输入（报警、参数设置等）做出相应的响应。车载终端一方面把定位信息用于显示，一方面发送给 GPRS 通信模块。GPRS 模块将定位信息通过通信网络传送给监控中心后，监控计算机通过地图匹配技术对数据进行存储和处理最终在电子地图上显示运行轨迹。监控人员或交通管理部门根据车辆的

位置, 采取一定的措施。

本设计完成车载定位系统的车载终端部分。硬件上, 完成 ARM 处理器、SDRAM、FLASH、GPS 模块、GPRS 模块及其外围电路的搭建; 软件上, 移植启动代码 Bootloader、文件系统、嵌入式 Linux 系统和 Qtopia, 然后在 Qt/E 平台上开发用户界面和应用程序, 建立交叉开发环境, 将程序编译成目标平台上的可执行文件, 完成 GPS 定位数据的提取并发送给 GPRS 模块。

### 3 车载终端的硬件设计

#### 3.1 总体硬件设计方案<sup>[34]</sup>

整体电路框图如图 3.1 所示。

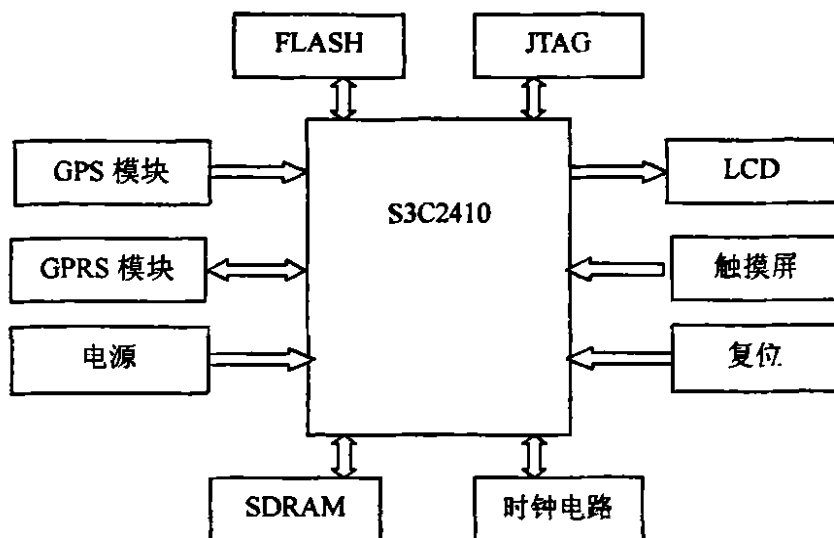


图 3.1 车载终端总体硬件设计框图

车载终端总体上可以分为三大部分：S3C2410 系统电路、GPS 模块电路和 GPRS 模块电路。

电路的核心为 S3C2410 处理器；FLASH 存储系统，应用程序等；SDRAM 作为系统的运行内存，用作程序的运行空间、数据及堆栈区；JTAG 用来开发调试系统。GPS 模块将接收到的定位数据通过串口传送给处理器。GPRS 模块用来发送定位等信息到网络上。

除以上的几个电路部分外，本设计还预留了 USB 电路和扩展槽，用于后续开发和功能扩展。

本设计共分 S3C2410 系统电路（见附录 A）、GPS 接收机电路（见图 3.6）和 GPRS 发送电路（见图 3.7.1，图 3.7.2，图 3.7.3）三个独立的电路部分。三个电路分别用 RS-232 串口相连。下面给出几个主要部分的电路连接图。

#### 3.2 电源

由于本设计的芯片较多，所需的芯片电压也有所不同。根据车载终端的特点，一般而言，汽车上的直流电压源有两种：12V 和 24V。需要对这两种电压进行转换，

得到所需的电压。主要芯片的电压需求如下:

- S3C2410: 3.3V, 1.8V
- SDRAM, NAND, JTAG, MAX3232: 3.3V
- GPS, USB, MAX232: 5.0V
- GPRS: 3.6V

对于汽车电压, 本设计采用 National Semiconductor 公司的 LM2576HV 系列稳压模块, 它的输入电压范围宽(8V~40V), 稳压效果好, 能把 12V、24V 电压转换成 5.0V 电压, 电路如图 3.2.1 所示。

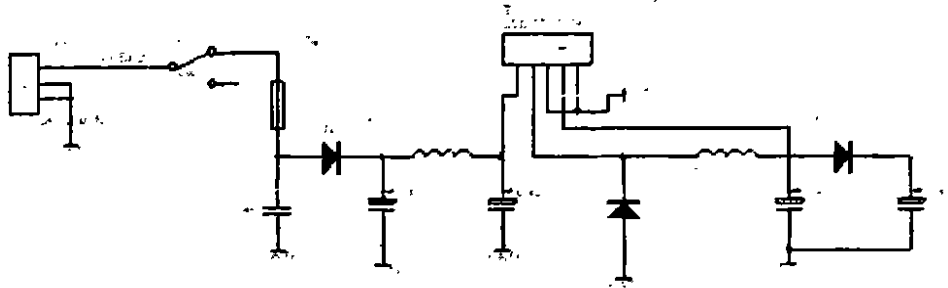


图 3.2.1 12V-24V 电压转 5.0V 电压

采用 LM1117DT-3.3 和 LM1117DT-1.8 芯片电路分别得到 3.3V 和 1.8V, 如图 3.2.2 所示。

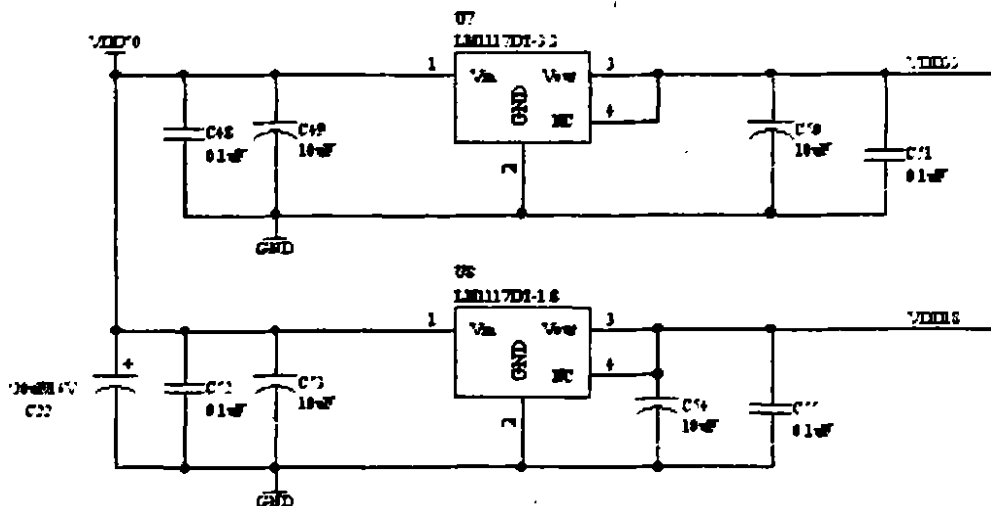


图 3.2.2 3.3V 和 1.8V 转换电路

对于 GPRS 模块, 为保证模块的正常工作, 需要提供两路电源, 其中 VBATT 提供给 RF (射频) 部分, 电压范围为 3.3V ~4.5V, 最大功率为 2W; VDD 提供给基带部分, 电压范围为 3.1V~4.5V, 最大功率为 0.5W。模块内部具有电池充电管理电路, 利用外部电源提供的电流源对内部电池的安全充电进行控制与管理。本设计采用 LM317T 芯片完成 5V 转 3.6V, 电路图略。图 3.7.1 中, 将 VBATT 和 VDD 连在一起接 VDD36。

### 3.3 S3C2410 处理器<sup>[38]</sup>

从产品的定位出发,考虑到功能、性能和性价比,兼顾二次开发和功能拓展,选择一款各适的处理器。S3C2410 微处理器是一款由三星公司为手持设备设计的低功耗,高集成度的基于 ARM920T 的微处理器。采用 16 位/32 位 RISC 结构和 ARM 精简指令集,支持 WinCE、Linux 等操作系统;支持指令 CACHE、数据 CACHE 和写缓冲、支持 ARM 调试结构、片上 ICE 支持 JTAG 调试方式;支持大端(Big Endian)小端(Little Endian)模式。8 个内存块,6 个用于 ROM、SRAM 及其他,2 个用于 ROM/SRAM/SDRAM;每个内存块 128MB (共 1GB),每个内存块支持 8/16/32 位数据总线编程;1 个起始地址和大小可编程的内存块 (Bank7),7 个起始地址固定的内存块 (Bank0~Bank6);所有的内存块可编程寻址周期,支持 SDRAM 自动刷新模式,支持多种类型 ROM 启动,包括 NOR/NAND FLASH,EEPROM 等。

(1) S3C2410X 处理器时钟和电源管理。

(2) 片上 MPLL 和 UPLL: UPLL 产生操作 USB 主/从的时钟;MPLL 产生操作 MCU 的时钟,1.8V 电压供电最高可达 203MHz;每个模块的时钟可由软件控制,电源模式有四种,分别为正常模式,处于正常运行的状态下;休眠模式下只使 CPU 的时钟停止;低能模式,不带 PLL 的低频时钟;停止模式下,所有时钟都停止。可以用 EINT[15:0] 或 RTC 告警中断从停止模式唤醒。

(3) S3C2410X 处理器中断控制器。

(4) 它带有 55 个中断源 (看门狗定时器,5 个定时器,9 个 UART,24 个外部中断,4 个 DMA,2 个 RTC,2 个 ADC,1 个 I2C,2 个 SPI,2 个 USB,1 个 LCD 和一个电池管理);外部中断源的触发模式可为电平触发也可为边沿触发;对紧急中断请求支持 FIQ(快速中断请求)。

(5) S3C2410X 处理器定时器。

(6) 它带有四个 16 位带 PWM 的定时器,1 个 16 位基于 DMA 或基于中断的定时器;具有可编程的占空比,频率和极性;支持外部时钟源,具有看门狗定时器:16 位看门狗定时器,定时中断请求和系统复位。

(7) S3C2410X 处理器实时时钟。

(8) 全时钟特点:毫秒,秒,分,时,日,星期,月,年;以 32.768KHz 运行,可以产生告警中断,时钟滴答中断。

(9) S3C2410X 处理器通用输入/输出端口及接口。

(10) 具有 24 个外部中断端口,多路输入/输出口;具有 3 个带有 DMA 和中断的 UART,支持 5 位,6 位,7 位,8 位串行数据传送/接收,并且在传送/接收时支持双向握手,具有可编程的波特率,支持 IrDA1.0(115.2KBPS),支持回环测试模式,每个通道有 16 字节 TX FIFO 和 16 字节 RX FIFO。



S3C2410X 的存储空间映射如图 3.3<sup>[29]</sup>。

为了使处理器对各个设备的访问互不干扰，所以要将不同类的设备映射到不同的 bank 内。

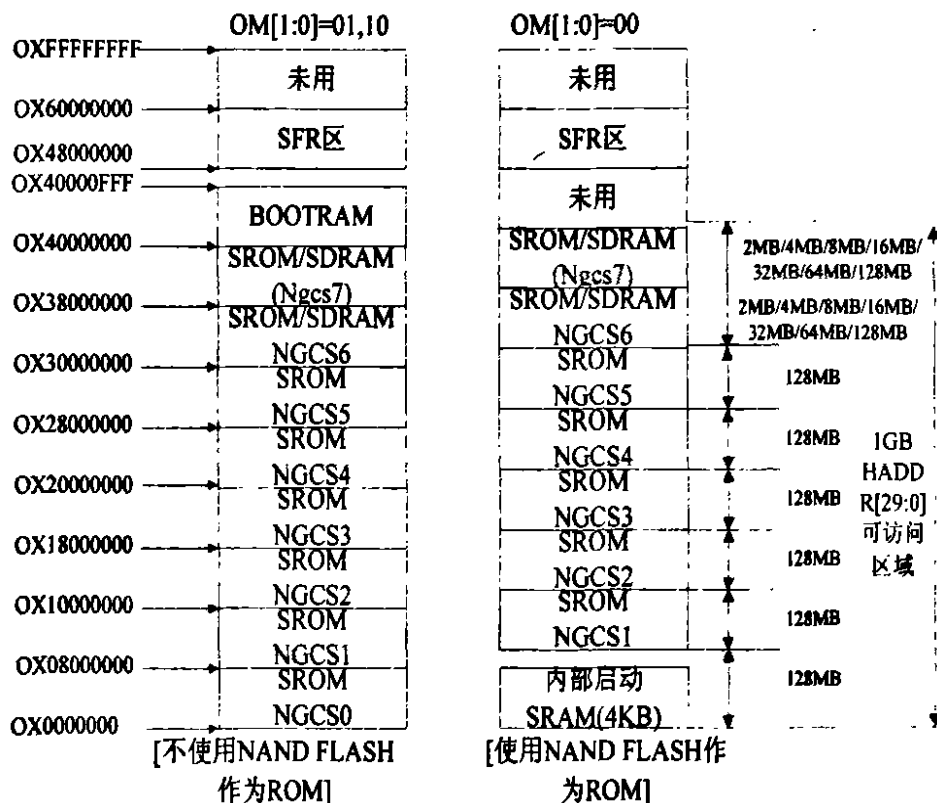


图 3.3 S3C2410X 存储空间映射框图

### 3.4 存储电路设计

#### 3.4.1 SDRAM<sup>[40]</sup>

SDRAM 是嵌入式微处理器的内存，在掉电以后数据即消失，不能够长久保存。SDRAM 在系统中主要用作程序的运行空间、数据及堆栈区。当系统启动时，CPU 首先从复位地址 0x0 处读取启动代码，在完成系统的初始化后，程序代码一般应调入 SDRAM 中运行，以提高系统的运行速度。同时，系统及用户堆栈、运行数据也都放在 SDRAM 中。

SDRAM 具有单位空间存储容量大和价格便宜的优点，已广泛应用在各种嵌入式系统中。SDRAM 的存储单元可以理解为一个电容，总是倾向于放电，为避免数据丢失，必须定时刷新（充电）。因此，要在系统中使用 SDRAM，就要求微处理器具有刷新控制逻辑，或在系统中另外加入刷新控制逻辑电路。S3C2410 芯片在片内具有独立的 SDRAM 刷新控制逻辑，可方便地与 SDRAM 接口。

目前常用的 SDRAM 为 8 位/16 位的数据宽度, 工作电压一般为 3.3V。主要的生产厂商为 HYUNDAI、Winbond 和三星等。它们生产的同型器件一般都具有相同的电气特性和封装形式, 可通用。

在本设计中使用 HYUNDAI 公司的 HY57V561620C, 这是 32M CMOS 同步 DRAM, 是一种高密度、高带宽的存储器。它的容量为  $4 \times 4 \times 16\text{Mbit}$  (32M 字节), 主要特点有: 时钟频率 133MHz, 数据宽度 16 位, 工作电压为 3.3V, 所有的管脚都与 LVTTTL 接口兼容, 支持自动刷新和自刷新。图 3.4.1 为 SDRAM 内存电路连接图。

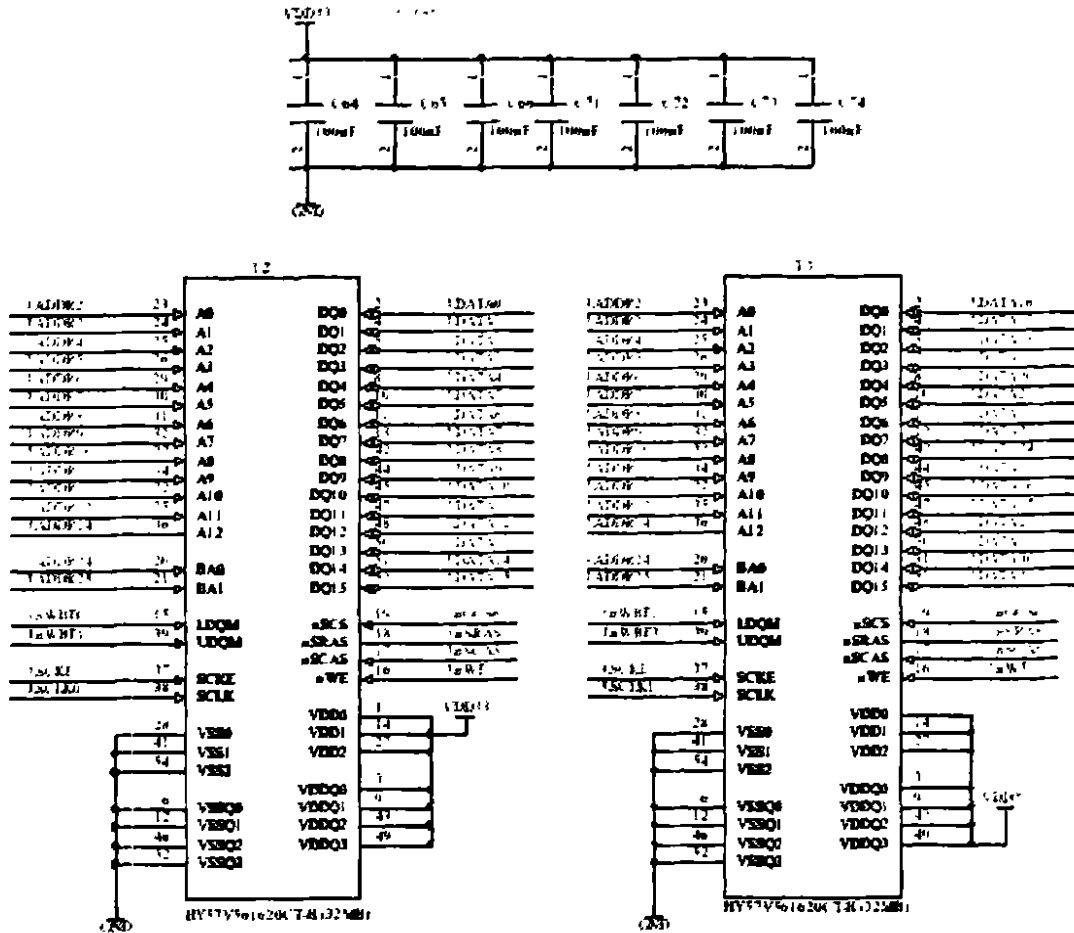


图 3.4.1 SDRAM 内存电路连接图

S3C2410 数据宽度为 32 位, 因此使用了 2 片 K4S561632C, 一片连 S3C2410 数据线的低 16 位, 一片连数据线的高 16 位。这两片并联为 32 位数据宽度的 SDRAM 存储系统, 并映射到 S3C2410 BANK6, 地址范围为 0x3000 0000~0x33FF FFFF。

### 3.4.2 FLASH<sup>[30]</sup>

FLASH 是一种非易失, 可在系统 (In-System) 电擦写, 掉电后信息不丢失的存储器。它具有低功耗、大容量、擦写速度快、可整片或分扇区在系统编程 (烧写)、

擦除等特点，并且可由内部嵌入的算法完成对芯片的操作，在各种嵌入式系统中得到了广泛的应用。FLASH 在系统中通常用于存放程序代码、常量表以及一些在系统掉电后需要保存的用户数据等。

FLASH 主要可分为 NOR FLASH 和 NAND FLASH。NOR FLASH 的特点是芯片内执行，这样应用可以直接在 FLASH 内运行，不必再把代码读到系统 RAM 中，NOR 的传输效率很高，在 1~4MB 的小容量时具有很高的成本效益，但是很低的写入和擦除速度大大影响了它的性能；NAND FLASH 结构能提供极高的单元密度，可以达到高存储密度，并且写入和擦除的速度也很快，应用 NAND 的困难在于 FLASH 的管理和需要特殊的系统接口。

FLASH 闪存是非易失存储器，可以对成块的存储器单元进行擦写和再编程。任何 FLASH 器件的写入操作只能在空或已擦除的单元内进行，所以大多数情况下，在进行写入操作之前必须先执行擦除。NAND 器件执行擦除操作是十分简单的，而 NOR 则要求在进行擦除前先要将目标块内所有的位都写为 0。一般 NAND FLASH 的存储容量较大，虽然 NAND 的管理比较困难，但是 S3C2410 提供了能够从 NAND FLASH 启动系统的 Steppingstone 机制，因此 S3C2410 系统可以采用 1 片 NAND FLASH 同时作为启动 ROM 和系统程序保存 ROM，减少了硬件成本。

在本设计中，使用三星公司的 NAND FLASH 芯片 K9F1208U0M，用来存放启动代码 (Bootloader)、Linux 内核映像、文件系统，另外还有应用程序。它的单片存储容量为 64MB，8 位数据宽度，快速的写周期时间 (编程时间为 200us，块擦除时间为 2ms)，命令、地址、数据 I/O 口复用。芯片上包含有写控制器，能够自动编程和擦写。

图 3.4.2 为 NAND FLASH 部分的电路图。

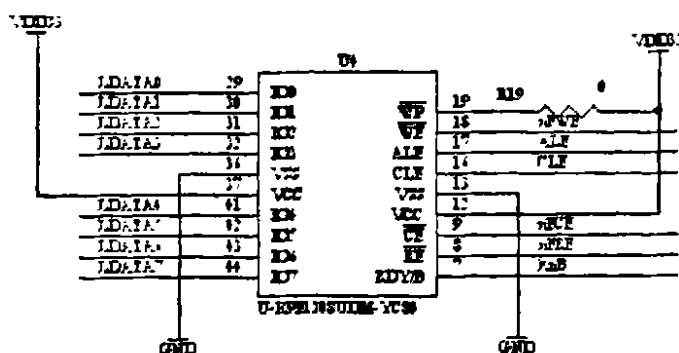


图 3.4.2 FLASH 电路连接图

S3C2410 选择从 NAND FLASH 启动, 因此 S3C2410 的 OM[1:0]为 00。在图附录 A S3C2410 系统电路中, 将这两端直接接地。

### 3.5 串口电路

串行口是计算机一种常用的接口,具有连接线少,通讯简单,得到广泛的使用。常用的串口是 RS-232-C 接口(又称 EIA RS-232-C)它是在 1970 年由美国电子工业协会(EIA)联合贝尔系统、调制解调器厂家及计算机终端生产厂家共同制定的用于串行通讯的标准。它的全名是“数据终端设备(DTE)和数据通讯设备(DCE)之间串行二进制数据交换接口技术标准”。它所具有的特性有:

- 机械特性: RS-232C 接口是单端发送,单端接收,传输线上允许一个驱动器和一个发送器。RS-232C 标准接口有 25 针(DB25)和 9 针(DB9)两种。它的最大传输距离可达 30m,最大速率 20kb/s,适于相距较近设备的通信
- 电气特性: RS-232C 标准定义-15V~-3V 表示逻辑“1”,+3V~+15V 表示逻辑“0”。它选择-15V~-3V 和+3V~+15V 这个范围而不采用 TTL 逻辑(0V~5V)的原因是为了提高抗干扰能力和增加传输距离,因此与 TTL 设备连接时需加电平转换接口

由于 S3C2410 的异步串行通讯接口不是 RS232 逻辑,因此,需加电平转换接口,采用德州仪器公司的 MAX3232<sup>[4]</sup>低功耗芯片,将 TTL 电平转换为 RS232 电平。

在本设计中采用两个 9 针 DB9 串口,分别用来连接 GPS 模块和 GPRS 模块。另外还用来和 PC 机进行通信,观察系统启动情况和下载代码等。COM1(串口一)包含了四个信号(TXD、RXD、CTS、RTS),可用来连接 GPRS 模块;COM2(串口二)只有两个信号(TXD, RXD),可用来连接 GPS 模块。两个串口的 TXD, RXD 信号,分别与两个模块的 TX, RX 交叉相接,完成数据的发送和接收。

其电路如图 3.5 所示。

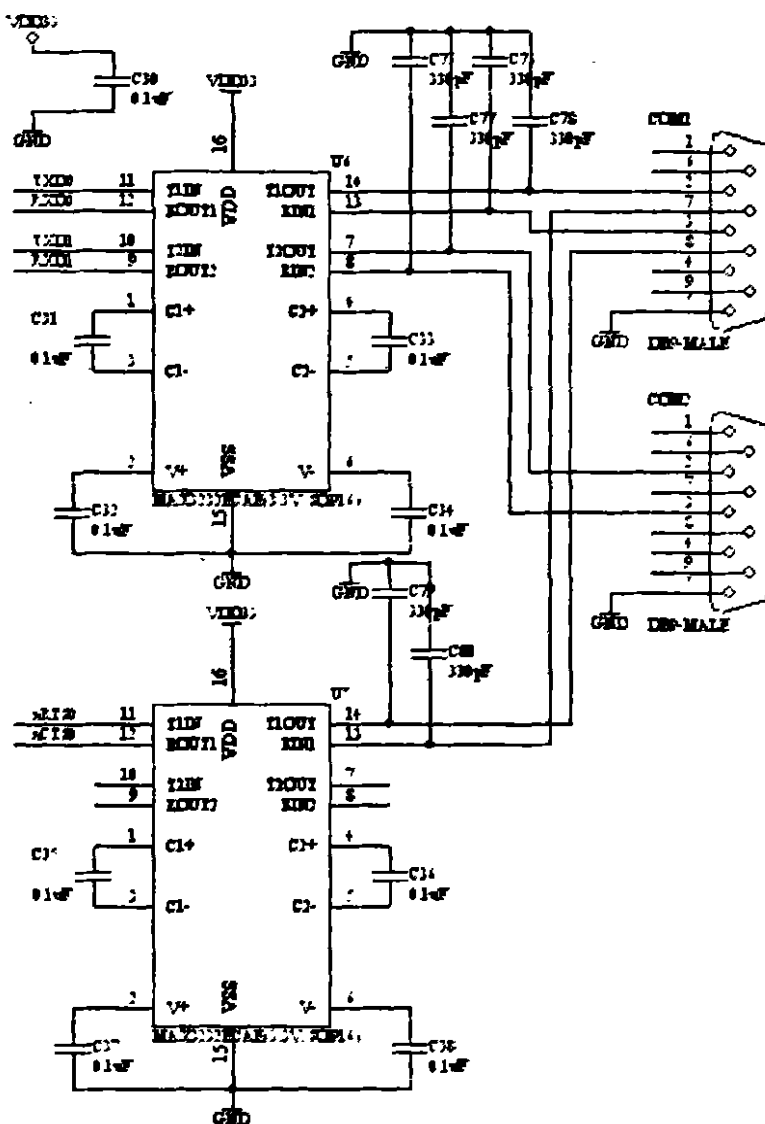


图 3.5 串口电平转换电路连接图

### 3.6 GPS 模块<sup>[35]</sup>电路

GPS 是 GPS 定位信息的接收机, 本设计直接采用 OEM 板 GPS 模块, 从 GPS 模块可以直接得到 GPS 卫星的定位信息, 而不需要了解太多 GPS 原理上的知识、算法等, 因此加快了开发速度。并且, GPS 模块大多采用专用的处理芯片和相关算法, 提高了 GPS 定位信息的精度。

本设计采用合众思壮公司的 E531 接收机模块。它是 12 通道的 GPS 接收机模块，同时可以跟踪 12 颗 GPS 卫星，跟踪性能优越，能够快速定位。E531 接收机功耗低，数据更新率为每秒一次，能满足车载系统的需求，性价比较高。

其主要指标如下:

- 电气指标：输入电压：DC3.3V~5.5V
- GPS 指标：接收机频率：L1, C/A code (SPS)，接收机通道：并行 12 通道，定位时间：快速启动：约 8 秒，热启动：约 15 秒，冷启动：约 50 秒（典型值）
- 自由定位：2 分钟，更新率：1Hz 精度：定位精度：3m(CEP)，小于 6m 2DRMS
- 速度精度：0.2m/s RMS (50%) 接收机灵敏度：-152dBm（跟踪）；-139dBm（捕获）
- 接口：接口特性：CMOS 电平输出
- 串口 0：默认波特率为 4800 输出：NMEA0183 版本 3.01 的 ASCII 码语句，默认输出语句包括 GGA, GSA, GSV, RMC；可选输出：GLL, VTG, ZDA, DTM。输入：选择坐标系、波特率设置、输出语句选择等
- 串口 1：默认波特率为 115200，二进制协议

在本设计中,使用串口 0,通过串口提取 ASCII 码。串口数据通过 MAX232 电平转换为 RS232 电平。另外,将 GPS 模块的发送端和接收端经电平转换后与串口 DB9 交叉相连。

注：由于 GPS 模块输出 CMOS 电平，可以直接驱动 S3C2410 芯片，因此在系统集成时，可直接将 GPS 模块的串口与 S3C2410 的串口相连，而不需要电平转换。

一个完整的 GPS 接收机包括电源、接收机和天线等。图 3.6 为 GPS 接收机连接图。

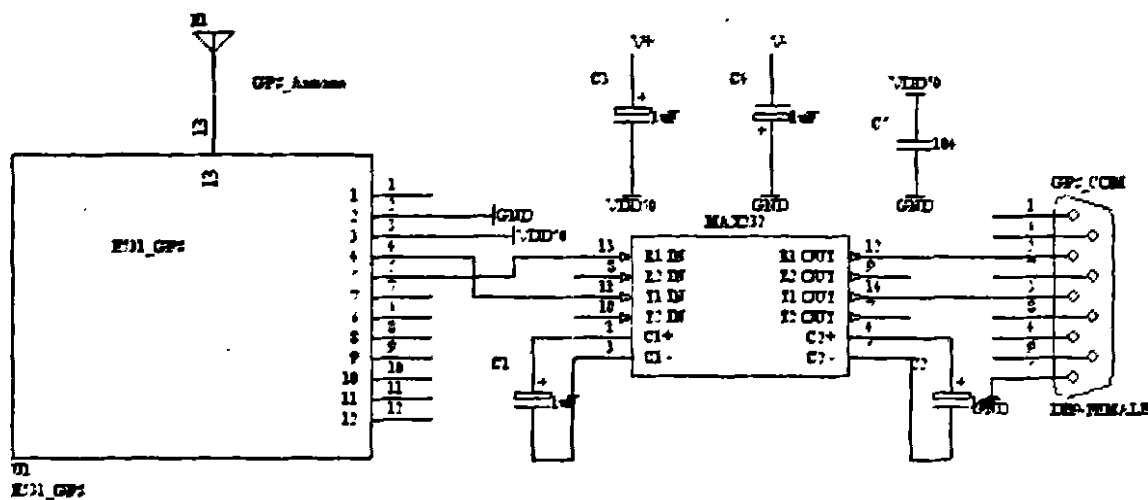


图 3.6 GPS 接收机连接图

### 3.7 GPRS 模块<sup>[42]</sup>电路

在车载终端中，短消息的发送、接收和语音通话是由无线通信模块来负责完成的。GPRS 模块是指带有 GPRS 功能的 GSM 模块，可以通过 GPRS 网络进行数据传输。现在使用较多的有法国 WAVECOM 公司的 WISMO 系列、Siemens 公司的 MC35、MC45 系

列、Motorola 公司的 G18, G20 系列, BENQ 公司的 M22 系列等。这些模块接口简单、使用方便并且功能非常强大,为 GPRS 应用提供理想的解决方案,在工业与民用等诸多领域已得到了广泛的应用。在结构和功能上都有相似之处。一般具有以下特点:

- 接口简单,使用方便:一般都提供电源接口、SIM 卡接口、RS232 数据口,利用 AT 指令进行控制
- 功能齐全,有两种工作模式:GSM Phase 2 模式,支持语音服务;GPRS 分组交换模式

模块本身支持的数据业务包括 SMS、CSD、HSCSD 和 GPRS。同时也支持语音,传真等服务。有的模块内部集成了 TCP/IP 协议栈,方便了网络应用程序的开发。

本设计采用的 GPRS 模块是 WAVECOM 公司的 WISMO Q2406A。WISMO Q2406A 模块是一个 GSM/GPRS 900/1800 MHz 的双频模块,执行 ETSI GSM Phase 2+的标准,在 900MHz 时,功率为 2W,1800/1900MHz 时为 1W;带有 16Mb 的 Flash 存储器和 2Mb 的 SRAM;支持 class 10,数据线路异步传输和同步可达 14400bits/s;通过 AT 指令控制,波特率从 300 到 115200bits/s,自动速率从 2400 到 19200bits/s;单一天线接口 (900/1800),SIM 3V/5V 和 SIM 检测;3.6V DC 供电。

WISMO Q2406A 模块包括以下几个部分:电源、异步串行接口、SIM 卡接口、天线等。其主串行口 (UART1) 包含六路信号:TX、RX、RTS、CTS、DTR、DSR,另外还有两个附加信号 DCD 和 RI。Q2406 如全性能工作,则须使用所有的接口信号,但是在本设计中,只采用了 TX、RX 和 CTS、RTS 两个握手信号 (实际上用两根线 TX, RX 也能实现通常的功能)。

Q2406 模块的连接如图 3.7.1 所示。

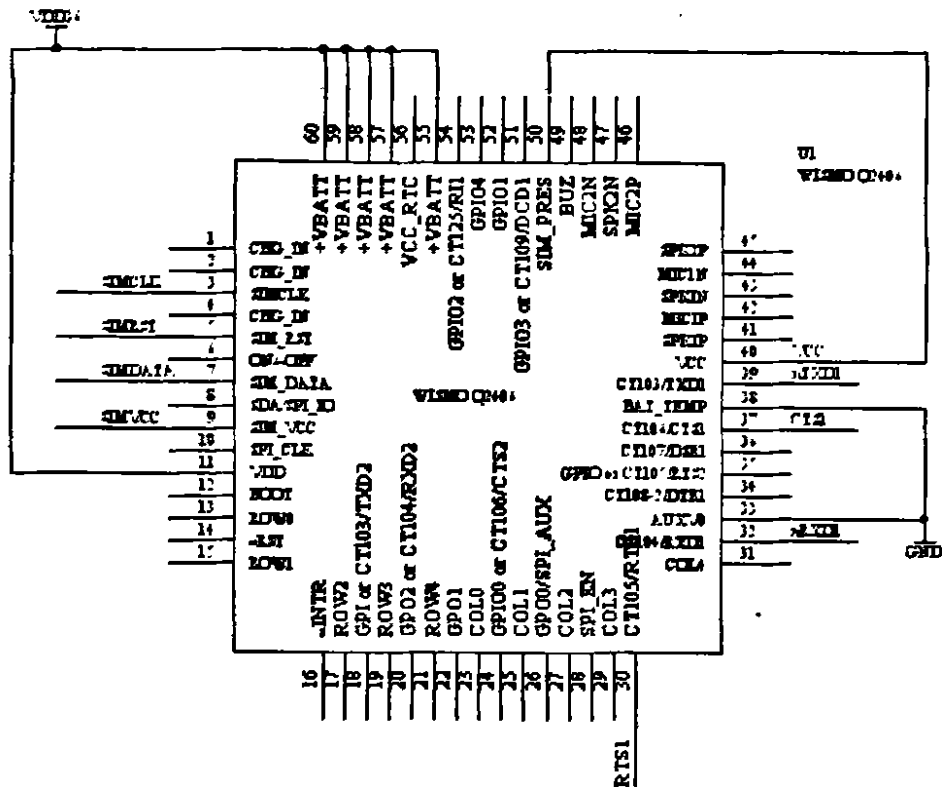


图 3.7.1 WISMO Q2406 模块硬件连接图

SIM 卡可以根据电压不同，分为 3V 和 5V 等类型（5V 的 SIM 卡需要外部的驱动器 external SIM driver, 也就是进行电平转换）。现在国内通用 SIM 卡以 3V 为主，可以直接和模块连接。模块上与 SIM 卡相关的信号线有 5 条，如表 3.7 所示。

表 3.7 Q2406 模块的 SIM 卡引脚描述

信号线名称	引脚号	输入/输出	功能描述
SIM_CLK	3	输出	信号传输时的同步时钟
SIM_RST	5	输出	RESET SIM 卡
SIM_DATA	7	输入/输出	与 SIM 卡进行数据传输
SIM_VCC	9	输出	SIM 卡的电源线
SIM_PRES	50	输入	检测是否有 SIM 卡插入

SIM\_PRES 可以直接接电源，如图 3.7.1 所示，表示 SIM 一直处于插入状态。其他 4 根线可以和 SIM 卡的相应线直接连接。



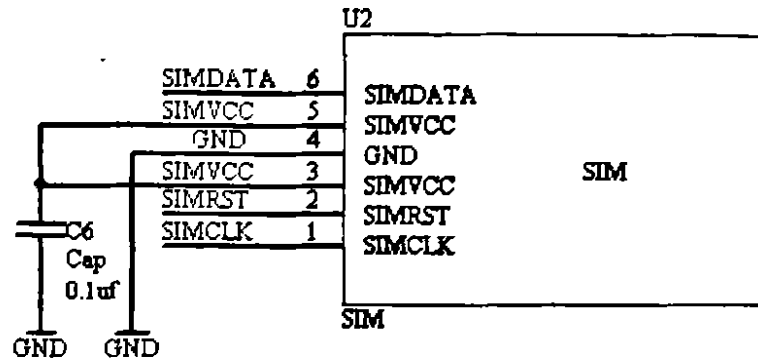


图 3.7.2 SIM 卡连接图

注：同 GPS 模块一样，GPRS 模块也要进行电平转换，但在系统集成时，就不需要了。GPRS 模块的接口信号 RXD、TXD 与 ARM 的 TXD、RXD 对应相接，RTS 与 CTS 对应相接。电平转换如图 3.7.3 所示。

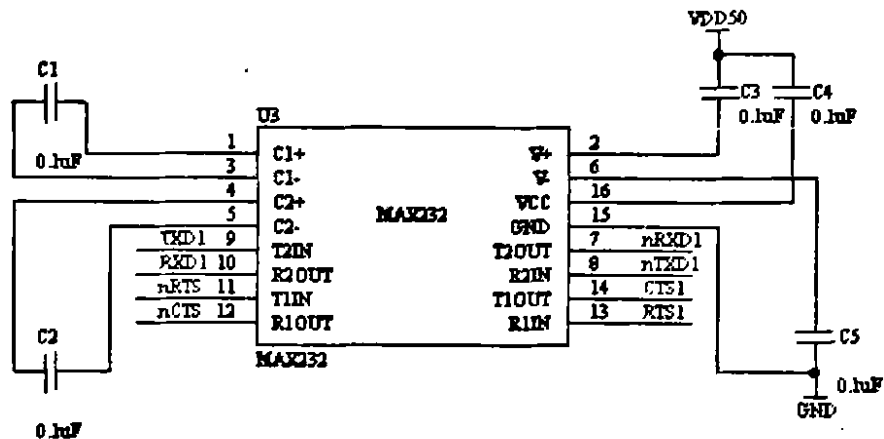


图 3.7.3 GPRS 电平转换电路

## 4 嵌入式操作系统及其开发环境介绍

以嵌入式处理器为中心, 搭建好硬件电路, 仅仅提供了裸机运行平台, 要使整个系统的各部分资源充分利用起来, 还需要(嵌入式)操作系统的软件支持。

### 4.1 嵌入式操作系统

嵌入式操作系统 EOS(Embedded Operation System)定义为一种实时的, 支持嵌入式系统应用的操作系统软件。它是嵌入式系统(包括软、硬件)极为重要的组成部分, 通常包括与硬件相关的底层驱动软件、系统内核、通信协议、图形用户界面和标准化浏览器。嵌入式与一般的商用多任务 OS(如 Unix、Windows 以及 Linux 等)有共同的一面, 也有不同的一面。商用多任务 OS 主要目的是方便用户管理计算机资源和追求系统资源的最大利用率; 而嵌入式操作系统重点追求的是可确定性, 可靠性, 当然也包括有限资源的管理。一般嵌入式操作系统内核都很小, 在几 KB~十几 KB 之间。基本上都支持基于优先级的抢占式调度策略和时间片轮转, 具有微内核结构, 有标准组件可代选用, 支持虚拟存储技术和存储保护机制。一个典型的嵌入式操作系统一般要包括操作系统所具备的基本功能, 如进程调度、内存管理以及中断处理等, 同时要具有小巧、速度快和响应可预测性等特点, 必须保证实时任务在要求的时间内完成。

### 4.2 嵌入式系统选择与嵌入式 Linux

由于嵌入式应用的功能需求差异以及不同嵌入式操作系统间不同的性能指标, 其选择有三种方案: 一是根据应用需要和公司技术实力, 考虑自主开发研究; 一是充分考虑系统需求和嵌入式操作系统性能指标及性价比选择商用嵌入式操作系统, 如 WinCE、VxWorks 等; 三是考虑嵌入式 Linux, 它是开放源代码的免费自由软件, 互联网技术论坛较多, 具备较好的裁减性以支持不同的应用范畴。当前全球范围内商用嵌入式操作系统很多, 如 VxWorks、pSOS、Nucleus、QNX、WinCE 等。这些不同名称的嵌入式操作系统, 选择的重点是考虑它们的性能评价指标, 主要包括高度算法、本身内存开销、内存管理模式、最大中断禁止时间和最大任务切换时间。当然, 也包括购买成本和提供的技术支持等相关因素。通常还要考虑系统功能方面支持何种处理器硬件平台, 何种 API, 是否支持内存管理单元 MMU, 可移植性, 调试支持和标准支持等。如果开发网络应用, 还需要考虑该嵌入式操作系统是否 TCP/IP 网络组

件和 I/O 服务等。

结合以上原因,本设计从成本、开发难易度、资料方面、开发周期以及 S3C2410 硬件开发平台的特点和本设计所需完成的功能出发,选择嵌入式 Linux 系统。Linux 是一款免费的操作系统,三星公司已经为 S3C2410 这款嵌入式 CPU 开发出已经移植好的嵌入式 Linux 系统,开发者只需要在此系统上根据具体差异做一些修改即可使用。

下面介绍其特点<sup>[29]</sup>:

(1) Linux 可以移植到多个不同架构的 CPU 和硬件平台上,具有良好的稳定性、升级能力,而且开发方便。因此,有着得天独厚的优势。

(2) Linux 是开放源代码的,不存在黑箱技术,易于裁减,成本上极具竞争力。

(3) Linux 内核虽小,但功能强大、运行稳定、系统健壮、效率高。

(4) Linux 支持包括 X86、ARM、MIPS、Motorola 68K、PowerPC、SUN SPARC 等几乎目前所有的系统架构。

(5) Linux 有大量的开发工具,这些工具为嵌入式系统的开发提供了良好的开发环境。

(6) Linux 沿用 UNIX 的发展模式,遵循国际标准,可以方便的获得软硬件厂商的支持。

(7) 在图象处理、文件管理、多任务等诸多方面,Linux 的表现都非常出色。

(8) Linux 具有强大的网络功能,对现在通用流行的各种网络协议都有着很好的支持。

(9) Linux 的程序执行遵循按需载入内存的策略,当程序部分真正要执行的时候才会被载入系统,有效的降低了内存需求量。另外它的 Memory Page 管理,也让实际内存的使用更加富于效率。

(10) Linux 内核可根据实际需要做得很小。并且驱动模块具有动态加载的能力,这样不仅减少了所占用的内存,并且对新硬件的支持也很容易。

(11) Linux 不仅适于作为嵌入式系统,其本身也是嵌入式系统应用开发的优秀工具。现在,越来越多的嵌入式系统开发商开始使用 Linux 作为嵌入式操作系统,可以看到 Linux 在嵌入式领域的巨大潜力。

#### 4.3 Linux 交叉编译环境

嵌入式应用系统的开发属于跨平台开发,即开发平台使用的处理器和开发对象的处理器往往不是同一类型,需要交叉的软件集成开发环境,即进行代码编写,编

译, 链接和调试应用程序的集成开发环境。与运行应用程序的环境不同, 它分散在有通信连接的主机和目标机环境之中。在主机上系统开发者利用丰富的软硬件资源, 开发工具, 仿真系统, 通过与目标机的通信, 生成能够在目标机上调试, 运行的代码。

通常嵌入式系统软件的编译和执行是在两个不同平台上进行的。编译是在 Host 端, 就是桌面主机; 执行是在 Target 端, 也就是嵌入式系统的硬件平台。一般是在 Host 端上通过跨平台交叉编译器(toolchain)把源文件编译成目标平台上可执行的文件, 再通过串口, 并口或网络下载至目标平台上的 FLASH 或者其它存储介质, 然后由目标机来运行这些软件。这里所说的跨平台编译器和一般的编译器功能类似, 都是把源代码通过编译器编译成目标文件, 然后通过链接器、可重定位程序和定位器把目标文件重新定位成可执行文件。和通用的编译器之间最大的差别就在于跨平台编译器编译出来的可执行程序通常只能在特定 CPU 所属平台上运行。所以一般来说每种 CPU 都对应有不同的跨平台编译器。对于 ARM920T 的 S3C2410, 可以使用常用的 ARM 交叉编译器。

在本设计中, 首先搭建硬件环境, 把主机和 S3C2410 开发板通过三种连接电缆连接起来, 如图 4.3 所示。

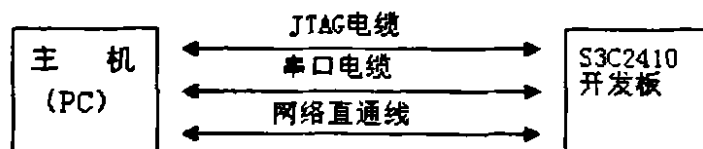


图 4.3 搭建硬件开发平台

串口用来观察目标板在程序运行过程中的输出信息, 同时也用来向目标板发送命令。连接完成后可以用 Linux 系统自带的 minicom 工具来进行控制。JTAG 用来固化 bootloader, 由并口线与 JTAG 小板构成。网络通讯可以采用以太网交叉线来直接连接 PC 机和开发板, 也可以用直通网线来连接 Hub 和开发板。网络通讯在 NFS 方式 mount 主机、root 文件系统更新、以及进行各种网络应用调试时必须用到。

搭建软件环境, 在主机上安装 RedHat9.0 Linux 操作系统; 安装交叉编译 toolchain; 将 Vivi Bootloader 固化到目标板; 将 kernel 映像文件固化到目标板; 把 YAFFS 文件系统固化在 NAND Flash 中; 安装和编译 Qt/E 开发环境。建立好之后, 就可以开发基于 Qt 的应用程序了。

#### 4.4 嵌入式操作系统移植

一个嵌入式 Linux 系统从软件的角度看通常可以分为四个层次：

- (1) 启动引导加载程序：即 Bootloader 程序。
- (2) Linux 内核：特定于嵌入式板子的定制内核以及内核的启动参数。
- (3) 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。
- (4) 用户应用程序：特定于用户的应用程序。

在一般的嵌入式系统中，系统所有的软件和数据都保存在 Flash 存储器中。对于 S3C2410 来说，其上电或复位后 PC 指针的地址是 0x0000 0000。因此 Flash 的起始地址就应该为 0x0000 0000。图 4.4 是一个典型的系统内存结构图。

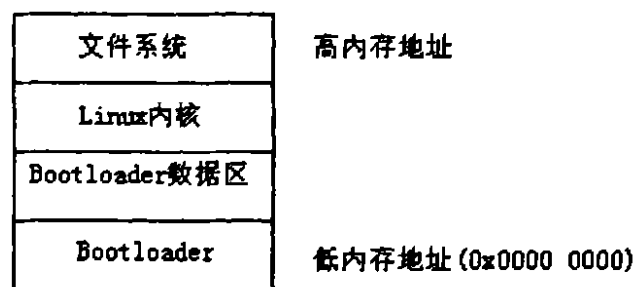


图 4.4 系统层次结构图

在本设计中，建立 3 个 MTD 分区分别用来存放 bootloader、kernel 和 rootfs。地址分别是：

0x0000 0000~0x0003 0000: “boot”

0x0003 0000~0x0020 0000: “kernel”

0x0020 0000~0x0400 0000: “rootfs”

#### 4.4.1 启动代码 Bootloader

Bootloader（引导加载程序）是系统加电后运行的第一段代码，根据 PC 的体系结构可以知道，PC 机中的引导加载程序由 BIOS（其本质就是一段固件程序）和位于硬盘 MBR 中的 OS Bootloader（比如 LILO 和 GRUB 等）一起组成。BIOS 在完成硬件检测和资源分配后，将硬盘 MBR 中的 Bootloader 读到系统的 RAM 中，然后将控制权交给 OS Bootloader。Bootloader 的主要运行任务就是将内核映像从硬盘上读到 RAM 中，然后跳转到内核的入口点去运行，也即开始启动操作系统。在嵌入式系统中，通常并没有像 BIOS 那样的固件程序（有的嵌入式 CPU 会在芯片内部嵌入一段短小的程序，一般用来将 Bootloader 装载进 ARM 中，因此 Bootloader 的作用与 PC 机上的

BIOS 类似), 通过 Bootloader 可以完成对系统板上的主要部件如 CPU、SDRAM、FLASH 和串行口等进行初始化, 也可以下载文件到系统板、对 Flash 进行擦除与编程。在一个基于 ARM 的嵌入式系统中, 系统在上电或复位时通常都从地址 0X00000000 处开始执行, 而在这个地址安排的通常就是系统的 Bootloader。通过这段小程序可以初始化硬件设备, 建立内存空间的映射, 从而将系统的软硬件环境带到一个合适的状态, 以便最终调用操作系统内核准备好正确的环境。

Bootloader 一般包含以下几个必备的功能<sup>[30]</sup>:

(1) 初始化处理器。这个动作都是用汇编语言完成的, 称为重置码(reset code)或者称为 boot code, 而且对于每个 CPU 都不一样的, 当电源接通就会执行这个动作, 通常只有两三个汇编指令, 目的是将 CPU 的控制权转给硬件初始化的程序。

(2) 初始化一些必要的硬件。这个动作也大都由汇编语言来完成, 主要是初始化 CPU、SDRAM 等, 其他的硬件, 例如串口, 可以由 C 语言等比较高级的语言程序来完成后续动作。

(3) 设置处理器的寄存器以及内存, 关掉所有的输入管脚(包括中断管脚), 以防止突然有信号进入妨碍我们接下来的硬件初始动作。然后初始化串口, 以便后续运行的程序能够同 Host 端进行通信, 便于调试。

(4) 从特定的位置把操作系统和文件系统调入内存, 并设置一些操作系统所必要的参数, 然后把 CPU 控制权交给操作系统。有的 Bootloader 会先从串口或者网络等其他途径得到内核的影像文件, 然后把这些文件写入 target 系统的 Flash 或者其它存储介质, 然后把内核载入 RAM 执行, 交出控制权。

大多数 Bootloader 都包括两种不同的操作模式:

- 启动加载模式: 在这种模式下, Bootloader 从目标机上的某个固态存储设备上将操作系统加载到 RAM 中运行, 整个过程没有用户接入。这是 Bootloader 的正常工作模式
- 下载模式: 在这种模式下, 目标机上的 Bootloader 将通过串口或网络等通信手段从开发主机上下载内核映像和根文件系统映像等到 RAM 中。然后可以再被 Bootloader 写到目标机上的固态存储介质中, 或者直接进行系统的引导

在本设计中, 采用的是 Mizi 公司开发的 vivi Bootloader。执行“make menuconfig”命令进行配置, 实际上, 并不需要对其进行手工配置, 而只需装载一个缺省的配置文件(default-configuration-file)即可, 使用这个配置文件生长成的 vivi 正好适合于目标板。然后编译 vivi。如果编译过程顺利, 将会生成 vivi 二进制映像文件。通过 JTAG 口, 用烧写工具 Jflash 烧写到 NAND FLASH 中。然后通过 minicom 工具可以查看相应的启动信息。

#### 4.4.2 内核<sup>[36]</sup>

内核是一整块可执行代码。在 Linux 系统中, 若干并发进程会参加不同的任务。每个进程都要求获得系统资源, 可以是计算、内存、网络连接或别的资源。通常由内核负责处理所有这样的请求。内核按照作用可以划分为如下部分。进程管理: 内核负责创建和终止进程, 并且处理它们和外部世界的联系(输入和输出)。内存管理: 计算机内存是主要资源, 而使用内存和策略是影响个系统性能的关键。

Linux 系统是建立在文件系统这个概念上的。Linux 里几乎所有东西都可以看作文件。这意味着:

- 每一个设备都至少由文件系统的文件代表, 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设备, 而设备则像普通文件一样受到文件系统访问权限控制机制保护
- 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。代表着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

Linux 将设备分成两大类。一类是像磁盘那样以记录块或“扇区”为单位, 成块进行输入/输出设备, 称为“块设备”; 另一类是像键盘那样以字符(字节)为单位, 逐个进行输入/输出的设备, 称为“字符设备”。文件系统通常都建立在块设备上。网络设备是介于块设备和字符设备之间的一种特殊设备。

在 Linux 操作系统中, 驱动程序是操作系统内核与硬件设备的直接接口, 驱动程序屏蔽了硬件的细节, 驱动程序是内核的一部分, 完成以下功能:

- 对设备初始化和释放
- 对设备进行管理, 包括实时参数设置以及提供对设备的操作接口
- 读取应用程序传送给设备文件的数据并回送应用程序请求的数据
- 检测是处理设备出现的错误

如图 4.4.2 所示, 应用程序通过 Linux 的系统调用与内核通信。由于 Linux 中将设备当作文件, 文件所以对设备进行操作的调用和对文件操作的操作类似, 应用程序发出系统调用命令后, 会从用户态转到内核态, 通过内核将 `open()` 这样的系统调用转换成对物理设备的操作。这一点将在下一节, 对串口操作时, 其应用就比较

明显。在进行内核配置时，可以把应用系统不需要的驱动程序去掉，以减少系统内核的大小。

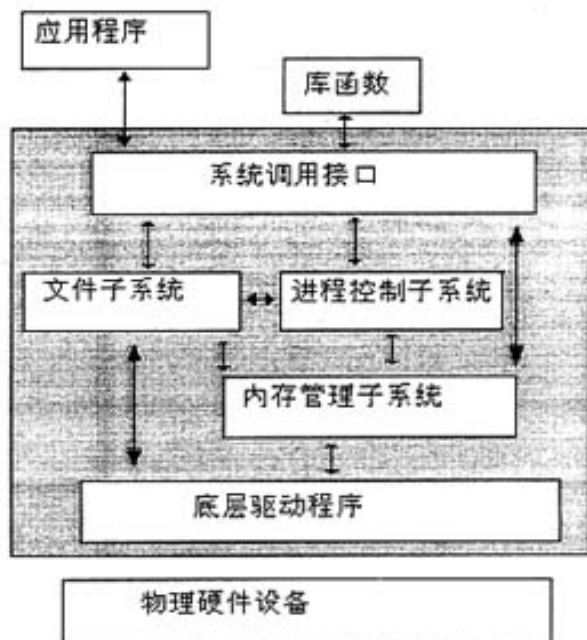


图 4.4.2 Linux 内核结构联系图

Linux 支持多个不同的文件系统：ext3、Yaffs、minix、iso9660、msdos、xia、fat、vfat、proc、nfs 以及 ufs。每一种文件系统都有自己的组织结构和文件操作函数，相互之间差别很大。Linux 文件管理的最上层模块是文件系统。系统启动时，必首先装入“根”文件系统，然后根据/etc/fstab 中指定，逐个建立文件系统。此外，用户也可以通过 mount, umount 操作，随时安装或卸载文件系统。Linux 的一个最大的优点是内核可裁减。根据产品的实际需要对 Linux 内核进行配置。

在本设计中，我们从网上下载相应的内核，由于目标板采用的是三星的 S3C2410 芯片，故可以从三星网站上下载与 S3C2410 相应的 Linux 核，对其解压后，进行配置 (make menuconfig)，使内核与我们所求的相合。然后编译 (make clean, make dep, make zImage)。编译结束后将在 kernel/arch/arm/boot 目录下得到 linux 内核压缩映像文件：zImage。最后通过 minicom 把内核固化在 Flash 中。

#### 4.4.3 文件系统

嵌入式系统对文件的操作是通过层次结构实现的。文件系统是操作系统的一部分，它负责管理逻辑文件，包括提供对逻辑文件的操作接口。文件系统不能直接控制物理设备，通常通过 Flash 驱动实现对文件和目录的操作。目前支持闪存 Flash



的文件系统技术有: Yaffs、JFFS2、RAMFS、ROMFS 等。

本设计采用的文件系统是 Yaffs。因为它存取速度快, 占用内存更少。此外, Yaffs 还自带 NAND Flash 驱动, 并为嵌入式系统提供了直接访问文件系统的 API。

一个完整的 Linux 系统包括一个 Kernel 和一个 root 文件系统。如果只有一个 Linux 内核, 系统最后是无法正确引导的。Linux 内核引导完毕, 就试图 mount 一个 root 文件系统 (根文件系统), 如果找不到, 内核将报如下错误:

Kernel Panic: VFS: Unable to mount root fs on 02:00

这个文件系统的位置由环境变量 root 指定, S3C2410 目前支持 3 种启动方式:

- root=/dev/bon/2
- root=/dev/mtdblock/0 本地启动 (缺省启动方式)
- root=/dev/nfs

其中第一种 root=/dev/bon/2 是 mizi 公司提供的启动方法, 它使用只读文件系统 cramfs 作为根系统。基于该文件系统的所有文件操作均是只读的, 用户只能在内存中创建文件, 系统掉电后, 文件也就不存在了。

Yaffs 是一种专门为嵌入式系统中常用的 flash 设备设计的一种可读写的文件系统, 它比 jffs2 文件系统具有更快的启动速度, 对 Flash 使用寿命有更好的保护机制。还可以使用 NFS 通过以太网挂接根文件系统, 这是一种经常用作为调试使用的文件系统启动方式。通过网络挂接的根文件系统, 可以在主机上生成 ARM 交叉编译版本的目标文件或二进制可执行文件, 然后就可以直接装载或执行它, 而不用频繁地写入 FLASH。

#### 4.5 嵌入式图形用户界面

GUI (graphical user interface) 图形用户接口, 用户通过 GUI 与系统进行交互, 实现相应的功能。在本设计中, 主要是用来显示 GPS 的信息, 用户可以直观, 方便的使用软件。另外, 在后续开发中, 还可以加入 GIS (地理信息系统) 等, 使用户更能直观的查看当前的位置。

当前应用较广的嵌入式图形用户软件主要有 Microwindows、MiniGUI、Qt/E 等。下面对这几种软件作一个简要的阐述:

(1) Microwindows 是嵌入式系统中广为使用的一种图形用户接口, 它的早期目标是在嵌入式 Linux 平台上提供和普通个人电脑上类似的图形用户界面。作为 PC 上 X-Windows 的替代品, Microwindows 提供了和 X-Windows 类似的功能, 但是占用的内存要少得多, 根据用户的配置, Microwindows 占用的内存资源在 100KB-60KB。

Microwindows 支持多种外部设备的输入, 包括 LCD、Mouse、Keyboard 等。在嵌入式 Linux 平台上, 从 Linux 2.2.X 的内核开始, 为了方便图形的显示, 使用了 framebuffer 的技术。Microwindows 的核心基于显示设备接口, 因此可移植性很好, 在 eCos、FreeBSD、RTEMS 等操作系统上都能很好地运行。另外, 它是完全免费的一个用户图形系统。

(2) MiniGUI 是由北京飞漫软件技术有限公司主持一个自由软件项目(遵循 GPL 条款), 其目标是为基于 Linux 的实时嵌入式系统提供一个轻量级的图形用户界面支持系统。MiniGUI 为应用程序定义了一组轻量级的窗口和图形设备接口。利用这些接口, 每个应用程序可以建立多个窗口, 而且可以在这些窗口中绘制图形。用户也可以利用 MiniGUI 建立菜单、按钮、列表框等常见的 GUI 元素。用户可以将 MiniGUI 配置成“MiniGUI-Threads”或者“MiniGUI-Lite”。运行在 MiniGUI-Threads 上的程序可以在不同的线程中建立多个窗口, 但所有的窗口在一个进程中运行。相反, 运行在 MiniGUI-Lite 上的每个程序是单独的进程, 每个进程也可以建立多个窗口。MiniGUI-Threads 适合于具有单一功能的实时系统, 而 MiniGUI-Lite 刚适合于类似于 PDA 和瘦客户机等嵌入式系统。如果用户的专用商业产品使用了 MiniGUI, 需要向飞漫公司支持商业授权费用。

(3) Qt/Embedded<sup>[32]</sup> (简称 Qt/E) 是一个专门为嵌入式系统设计图形用户界面的工具包。Qt 是挪威 Trolltech 软件公司的产品, 它为各种系统提供图形用户界面的工具包, 是一个跨平台的 C++ 图形界面库。它的目的是提供开发应用程序用户界面所需要的一切, 主要通过汇集 C++ 类的形式来实现的。Qt/E 就是 Qt 的嵌入式版本。

Qt 是一个支持多操作系统平台的应用程序开发框架, 它的开发语言是 C++。Qt 最初主要是为跨平台的软件开发者提供统一的, 精美的图形用户编程接口, 但是现在它也提供了统一的网络和数据库操作的编程接口。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, 字体国际化工具, Makefile 制作工具。目前 Qt 系列的软件主要包括 Qt, 基于 Framebuffer 的 Qt Embedded, 快速开发工具 Qt Designer, 国际化工具 Qt Linguist 等部分。Qt 具有下列优点:

- 优良的跨平台特性: 支持多种操作系统
- 面向对象: Qt 的良好封装机制使得 Qt 的模块化程度非常高, 可重用性较好
- 丰富的 API: 多达 250 个以上的 C++ 类
- 支持 2D/3D 图形渲染, 支持 OpenGL
- 大量的开发文档
- XML 支持

Qt/Embedded 是一个为嵌入式设备上的图形用户接口和应用开发而订做的 C++ 工

具开发包,它通常可以运行在多种不同的处理器上部署的嵌入式 Linux 操作系统上。如果不考虑 X 窗口系统的需要,基于 Qt/Embedded 的应用程序可以直接对缓冲帧进行写操作。除类库以外,Qt/Embedded 还包括了几个提高开发速度的工具,使用标准的 Qt API,可以非常方便的在 Windows 和 Unix 编程环境里开发应用程序。

Qt/Embedded 是一组用于访问嵌入式设备的 Qt C++ API: Qt/Embedded 的 Qt/X11, Qt/Windows 和 Qt/Mac 版本提供的都是相同 API 和工具。Qt/Embedded 还包括类库以及支持嵌入式开发的工具。

Qt/Embedded 提供了一种类型安全的被称之为信号与槽的真正的组件化编程机制,这种机制和以前的回调函数有所不同。Qt/Embedded 还提供了一个通用的 widgets 类,这个类可以很容易的被子类为客户自己的组件或是对话框。针对一些通用的任务,Qt 还预先为客户定制了象消息框和向导这样的对话框。

运行 Qt/Embedded 所需的系统资源可以很小,相对 X 窗口下的嵌入解决方案而言,Qt/Embedded 只要求一个较小的存储空间(Flash)和内存。Qt/Embedded 可以运行在不同的处理器上部署的 Linux 系统,只要这个系统有一个线性地址的缓冲帧并支持 C++ 的编译器。可以选择不编译 Qt/Embedded 的某些不需要的功能,从而大大减小它的内存占有量。

Qt 的图形设计器(designer)用来可视化地设计用户接口,使用 UNIX/Linux 操作系统,可以在工作站上通过一个虚拟缓冲帧的应用程序嵌入式系统的显示终端。另外 Qt/Embedded 也提供了许多特定用途的非图形组件,例如国际化、网络和数据库交互组件。Designer 和网络在本设计中都会用到的。

Qtopia 是 Trolltech 为采用嵌入式 Linux 操作系统的消费电子设备而开发的综合应用平台,Qtopia 包含完整的应用层、灵活的用户界面、窗口操作系统、应用程序、启动程序以及开发框架。Trolltech 提供三个 Qtopia 版本:手机版、PDA 版和消费类电子产品平台。在这里采用的是 PDA 版本的。

本设计采用的就是 Qt/E,先建立和安装 Qt/E 的开发平台,使用 Qt Designer 设计图形用户界面,然后编写应用程序,编译后,再把应用程序加入到 Qtopia 中去。

## 5 车载终端的软件设计

### 5.1 Qt/E 开发平台的建立

为了建立嵌入式 Qt 开发环境,可从网上([www.trolltech.com](http://www.trolltech.com))下载 Qt 的相关资源包,所需要的工具<sup>[33]</sup>如下:

- tmake 工具安装包: `tmake-1.11.tar.gz`——用于生成应用工程的 Makefile 文件
- Qt/Embedded 安装包: `qt-embedded-2.3.7.tar.gz`——用于 Qt/Embedded 的安装
- Qt 的 X11 版安装包: `qt-x11-2.3.2.tar.gz`——用于产生一些必要的工具
- Qtopia 安装包: `qtopia-free-1.7.0.tar.gz` 提供手持机的图形界面平台

下载的时候要注意,由于软件安装包有许多不同的版本,不同的版本在使用时可能造成冲突,所以当下载了 Qt/Embedded 的某个版本的安装包之后,选择安装的 Qt for X11 安装包的版本必须比最先下载的 Qt/Embedded 的版本要旧,这是因为 Qt for X11 的安装包的两个工具 `uic` 和 `designer` 产生源文件会和 Qt/Embedded 库一起被编译链接,本着“向前兼容”的原则,Qt for X11 的版本应比 Qt/Embedded 的版本旧。

可以在系统上安装两个开发环境:一个是基于 PC 的,一个是基于 ARM 的。前者可以不需要 ARM 硬件平台,而在主机上通过运行虚拟 Framebuffer,实现嵌入式的 Qt 开发;后者把应用程序移植到 S3C2410 的开发板上。两者的应用程序源代码都是相同的,不需要做任何修改。

下面以建立 PC 的开发环境为例,下面的 `tar` 命令指的是解压缩, `export` 命令是将路径加到环境变量, `mv` 是更名, `make` 执行编译:

#### (1) 安装 tmake

在 Linux 命令模式下运行以下命令:

```
tar xzf tmake-1.11.tar.gz
export TMKEDIR=$PWD/tmake-1.11
export TMAKEPATH=$TMKEDIR/lib/qws/linux-x86-g++
export PATH=$TMKEDIR/bin:$PATH
```

#### (2) 安装 Qt/Embedded 2.3.7

```
tar xzf qt-embedded-2.3.7.tar.gz
cd qt-2.3.7
export QTDIR=$PWD6
export QTEDIR=$QTDIR
```

```
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$QtDIR/lib:$LD_LIBRARY_PATH
./configure -xplatform linux-x86-g++ -thread -qvfb -depths
4, 8, 16, 32
make sub-src
```

这里要注意的是，因为本设计的程序需要多线程支持的，所以在配置时一定要加上`-thread`选项；`make sub-src`指定按精简方式编译开发包，也就是说有些 Qt 类未被编译，这样可以控制 Qt 生成的库文件的大小。

### (3) 安装 Qt/X11 2.3.3

```
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 -platform linux-x86-g++ -thread -no-xft -no-opengl
make
make -c tools/qvfb
mv tools/qvfb/qvfb bin
cp bin/uic $QTEDIR/bin
```

### (4) 安装 qtopia 1.7.0

```
tar xfvz qtopia-free-1.7.0.tar.gz
cd qtopia-free-1.7.0
export QTDIR=$QTEDIR
export QPEDIR=$PWD
export PATH=$QPEDIR/bin:$PATH
./configure -platform linux-x86-g++
make
```

## 5.2 总体软件设计

### 5.2.1 Qt/Embedded 的架构

Qt/Embedded 为带有轻量级窗口系统的嵌入式设备提供了标准的 API。Qt/Embedded 面向对象的设计思想，使得它能一直向前支持键盘、鼠标和图形加速卡

这样的附加设备。Qt/Embedded 的实现结构图如图 5.2.1.1 所示。

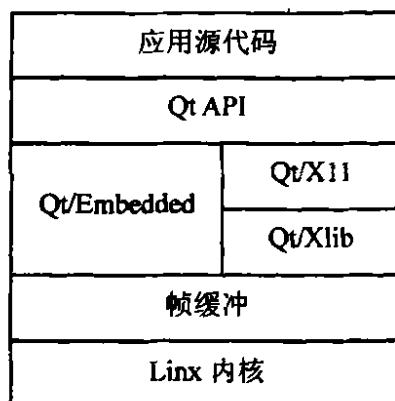


图 5.2.1.1 Qt/Embedded 的实现层次结构图

可以看到，Qt/Embedded 和 Qt/X11 一样都处在 Linux 内核和缓冲帧之上。它延续了 Qt 在 X 上的强大功能，在底层摒弃了 Xlib，由于 Qt/Embedded 的底层图引擎采用缓冲帧(Frame Buffer)，是针对高端嵌入式图形领域的应用而设计的。Qt/Embedded 还可以编译生成在主机虚拟缓冲帧 (Virtual Frame Buffer) 中显示输出的版本，可以仿真一个和嵌入式显示终端大小、像素相同的模拟仿真显示环境，这样界面开发人员可以在没有开发板的情况下做图形界面系统的开发。

图形开发模型最终是为了快速进行界面的原型设计，再辅助以 Qt/Embedded 提供强大的设计工具，使得开发者能迅速建立定制化的图形界面。

从全局的角度讲，开发流程[32]如下：

(1) 用 progen 命令生成一个工程文件 (.pro) 文件(本设计直接在 Qt Designer 中生成 C++ Project)。

(2) 新建一个窗体。

(3) 用 uic 命令窗体类的头文件和实现文件。

(4) 编写主函数 main() 和应用程序。

(5) 生成 makefile 文件。

(6) 编译链接整个工程。

图形开发模型如图 5.2.1.2 所示。

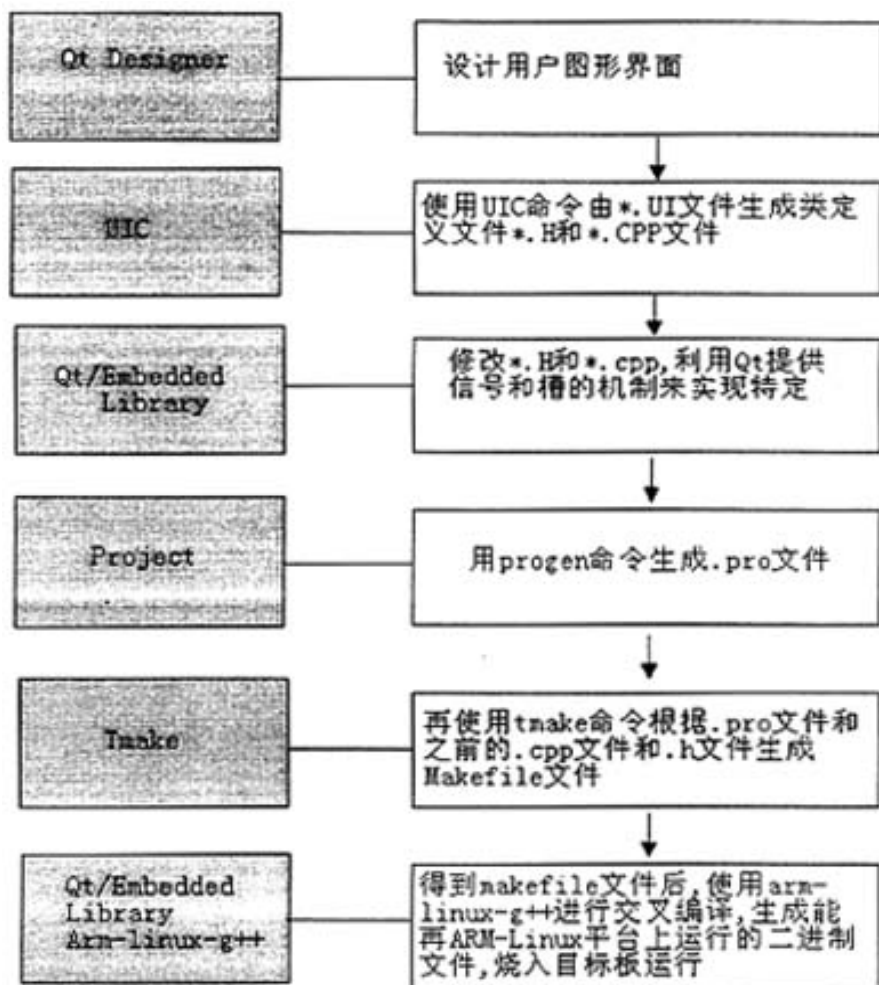


图 5.2.1.2 图形开发模型

### 5.2.2 Qt Designer 设计方法<sup>[43]</sup>

图形界面设计可以采用纯 C++ 程序调用 API 函数类库的方式实现, 也可以采用 Qt 提供的界面设计工具 Qt Designer 来设计, 后者可以通过拖放控件完成。这种方法快速, 简单, 修改方便, 很容易就能完成界面的设计。本设计就是采用 Qt Designer 来设计用户界面的。

Qt Designer 是一个可视化的图形界面设计器, 界面文件是 .ui 的文件格式。Designer 能够读写 .ui 文件, .ui 文件是以 XML 语言描述的。需要使用用户界面编译器 (uic) 把 XML 语言转换为 C++ 代码。

Qt Designer 共有三种设计方式: 第一种是最简单的通过信号和槽的方式; 第二种是继承子类的方式; 第三种是 ui.h 拓展的方式。本设计采用的是第二种, 其拓扑结构如图 5.2.2 所示。

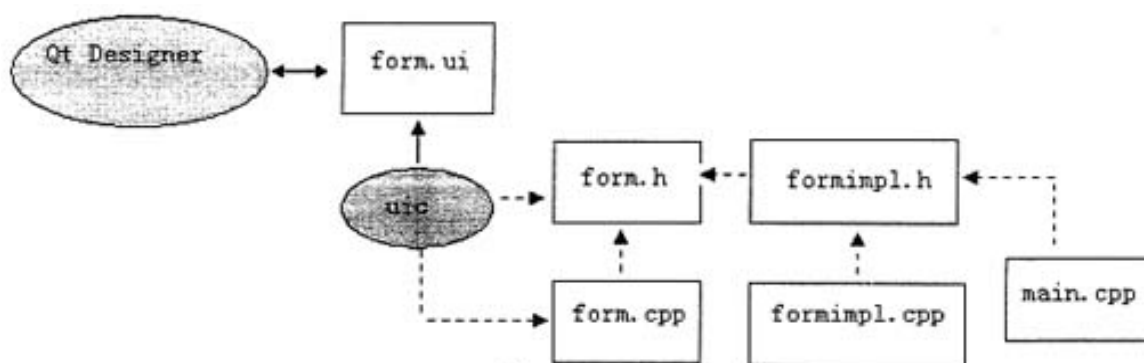


图 5.2.2 继承子类的 Qt Designer 设计方法

Qt Designer 设计好图形界面.ui 文件后, 使用 uic 将其转换为 form.h 和 form.cpp 文件。然后在界面实现的头文件 formimpl.h 中, 通过继承(#include)form.h 头文件, 来完成图形界面的功能。

### 5.2.3 车载终端的总体软件框图

在本设计中<sup>[37]</sup>, 先用 Qt Designer 设计好用户界面, 然后编写界面实现文件。通过继承 Qt 中的线程类 QThread 完成 GPS 和 GPRS 的功能。GPS 模块接收到的数据一方面发送到图形界面, 一方面送给 GPRS 模块。总体软件设计(源文件)框图如图 5.2.3 所示。

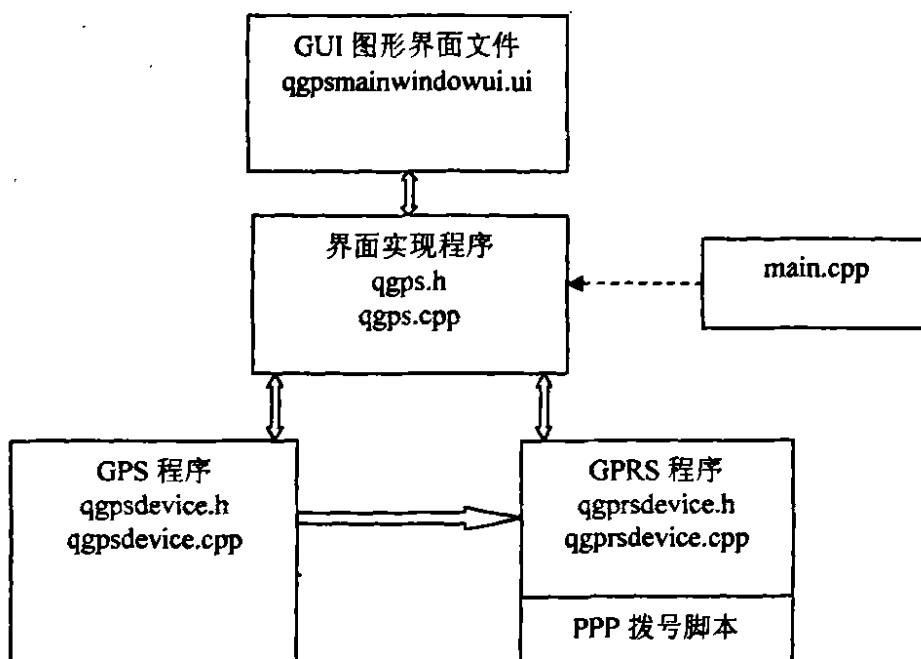


图 5.2.3 车载终端总体软件框图

另外还要用工具产生一些文件: makefile, qgps.pro, qgpsmainwindowui.h,



qgpsmainwindowui.cpp 等。

在编译链接的过程中,还会自动产生一些过程文件,不再详述。

本设计的每一个程序都是通过封装一个类来实现的(Qt 本身也是通过提供类库来实现的),每一个类包括.h 头文件和.cpp 实现文件。

## 5.3 界面和界面实现程序设计

### 5.3.1 界面设计

打开 Qt Designer,创建一个 Project——qgps.pro。然后新建一个 Main Window 框。用六个 LineEdit 来显示数据;六个 TextLabel 来标注;再用两个 GroupBox 来包容两个组;在 MenuBar 中添加 Device,用来控制 GPS、GPRS 的开启和停止;在最下面的状态栏中显示时间信息。

采用 Qt Designer 直接进行 GUI 设计如图 5.3.1 所示。

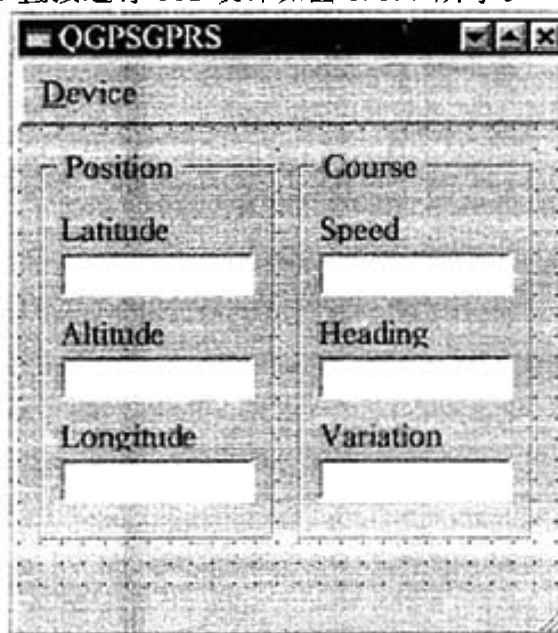


图 5.3.1 用户界面

用户界面是.ui 文件,使用 uic 将其转化为.h 和.cpp 文件。

### 5.3.2 界面实现程序设计

上面用 Qt Designer 设计好界面后,还需要实现文件来实现诸如数据的显示、控制等功能。本设计封装了一个类 qgps.h、qgps.cpp。在其头文件 qgps.h 中加入.ui 文件经 uic 转换后的.h 文件。

```
#include "qgpsmainwindowui.h"
```

部分实现文件 qgps.cpp 如下:

● 继承界面类头文件的构造函数:

```
QGPS::QGPS(QWidget *parent, const char *name)
: QGPSMainWindowUI(parent, name)           //构造函数继承界面类
{
    createStatusBars();                    //创建状态栏

    loadSettings();

    gpsDevice = new QGPSDevice(serialPort.latin1());

    gpsDevice->setTargetWidget(this);       //设为 GPS 接收界面

    connect(startGps, SIGNAL(clicked()), this, SLOT(startGps()));
    connect(stopGps, SIGNAL(clicked()), this, SLOT(stopGps()));
    connect(startGprs, SIGNAL(clicked()), this, SLOT(startGprs()));
    connect(stopGprs, SIGNAL(clicked()), this, SLOT(stopGprs()));
    //Qt 特有的信号和槽机制用来控制 GPS、GPRS 设备的打开和关闭
}
```

注: Qt 提供的信号和槽机制只能在同一线程中使用。

● 状态栏:

```
void QGPS::createStatusBars()              //状态栏用来显示定位状态和时间
{
    lblFixStatus = new QLabel(tr("No Position Fix"), this);
    lblFixStatus->setAlignment(AlignLeft);
    lblFixStatus->setMinimumSize(lblFixStatus->sizeHint());

    lblFixTime = new QLabel(tr("No UTC Time"), this);
    lblFixTime->setAlignment(AlignRight);
    lblFixTime->setMinimumSize(lblFixTime->sizeHint());

    statusBar()->addWidget(lblFixStatus);  //定位状态
    statusBar()->addWidget(lblFixTime);    //定位时间
}
```

● 在界面控件中显示定位信息:

```
void QGPS::customEvent(QCustomEvent *event) //定制用户事件, 来接收 GPS
{                                           //通知的事件
    if(event->type() == 10000){           //是否为所定义的事件
        QString latCardinal, longCardinal, varCardinal;

        if(gpsDevice->latCardinal() == QGPSDevice::CardinalNorth)
            //确定指向
            latCardinal = "N";
        else
            latCardinal = "S";
        if(gpsDevice->longCardinal() == QGPSDevice::CardinalEast)
            longCardinal = "E";
        else
            longCardinal = "W";
        if(gpsDevice->varCardinal() == QGPSDevice::CardinalEast)
```

```

        varCardinal = "E";
    else
        varCardinal = "W";
    txtLatitude->setText(
        QString("%1 %2\' %3\'" %4")
        .arg(gpsDevice->latDegrees())
        .arg(gpsDevice->latMinutes())
        .arg(gpsDevice->latSeconds())
        .arg(latCardinal));
    txtLongitude->setText(
        QString("%1 %2\' %3\'" %4")
        .arg(gpsDevice->longDegrees())
        .arg(gpsDevice->longMinutes())
        .arg(gpsDevice->longSeconds())
        .arg(longCardinal));

    ...
}
else{
    QMainWindow::customEvent(event);
}
}

```

//六个 LineEdit 控件显示数据

//其它数据显示方式相同

在 Qt 中, 线程安全的类有: QThread、QMutex、QMutexLocker、Semaphore、QTreadStorgge<T>和 QWaitCondition。另外还有 QApplication::postEvent()、QApplication::removePostedEvents()、QApplication::removePostedEvent() 和 QEventLoop::wakeup()。在本设计中, 界面实现程序相当于一个界面线程 GUI-Thread。而 GPS 和 GPRS 程序都是通过继承线程类 QThread 来实现的, 是非界面线程 Non-GUI Thread。GUI-Thread 可以通过继承 QThread 子类来创建一个新线程 Non-GUI Thread。在界面类中通过调用该线程类的启动函数 start() 开启线程, 这些新线程相互之间可以通过 mutexes、semaphores 和 wait conditions 相互通信, 但 GUI-Thread 和 Non-GUI Thread 数据间的传递只能通过事件 QApplication::postEvent() 来实现, 因为 postEvent() 线程安全的, Non-GUI Thread 通过它通知事件 (post Events) 给 GUI Thread, 否则会造成界面的死锁。在上面的程序中, 采用了定制事件 customEvent() 来接收事件, 之后在 GPS 程序中, 可以看到 postEvent() 发送事件。

每当数据有所更新的时候, GPS 线程将通知界面线程。因为采用的 GPS 模块是一秒更新一次数据, 所以界面的显示也是每秒刷新一次。

## 5.4 GPS 程序设计

### 5.4.1 NMEA0183 协议

如前 3.4 节所述, GPS 模块两个串口分别输出 NMEA0183 版本 3.01 的 ASCII 码

和二进制码。本设计只使用它的串口一，即提取的是 NMEA0183 码。它是美国国家海洋电子协会制定和一种基于 ASCII 字符的串行通信的数据协议，所有输入输出信息均为 ASCII 字符。它的一条消息称为语句 (Sentence)，每条语句以 “\$” (0x24) 作为开始，以回车符 (<CR><LF>或 0x0D0A) 作为结束，中间包含若干个域，每个域以逗号 “,” (0x2C) 分隔。一条 NMEA0183 语句包括以下几个部分：

- 起始符\$：表示一条语句的开始
- 标识符域：用来表示一条语句的全部数据域的特定格式，长度可变
- 数据域：包含各种类型的数据，一条语句内数据域之间用逗号分隔，数据域可以为空，但数据域间分隔用的逗号不可省略
- 校验和：紧跟在数据域后面，长度为两个字节，用星号 “\*” (0x2A) 与数据域分开
- 结束符<CR><LF>：表示一条语句的结束

一般的语句格式为：\$<Address Field>, D1, D2, D3, ……., Dn\*hh<CR><LF>。

NMEA0183 最常见的几种格式为：

- GPGGA：GPS 定位数据
- GPRMC：导航卫星特定精简资料
- GPGSV：导航卫星资料
- GPGSA：当前卫星信息

GPS 模块每秒发送的数据包括以上的几条，另外还有 GPVTG、GPGLL、GPUTC 等信息。本设计所提取的就是 GPRMC 语句，它是 Recommended Minimum Specific GNSS Data. Time, date, position, course and speed data. (RMC) 推荐定位信息。它包含的数据较全，满足车载系统的需求，主要包括的信息[33]如表 5.4.1 所示。

表 5.4.1 GPRMC 信息格式

\$GPRMC, <1>, <2>, <3>, <4>, <5>, <6>, <7>, <8>, <9>, <10>, <11>, <12>*hh<CR><LF>	
<1>	UTC 时间, hhmmss.ddd(时分秒)格式
<2>	定位状态, A=有效定位, V=无效定位
<3>	纬度 ddmm.mmmmm(度分)格式(前面的 0 也将被传输)
<4>	纬度半球 N(北半球)或 S(南半球)
<5>	经度 dddmm.mmmmm(度分)格式(前面的 0 也将被传输)
<6>	经度半球 E(东经)或 W(西经)
<7>	地面速率(000.0~999.9 节)
<8>	地面航向(000.0~359.9 度, 以真北为参考基准)

- <9> UTC 日期, ddmmyy (日月年) 格式
  - <10> 磁偏角 (000.0~180.0 度)
  - <11> 磁偏角方向, E (东) 或 W (西)
  - <12> 模式指示 A=自主定位, N=数据无效
- 

#### 5.4.2 GPS 数据提取程序

对 GPS 数据的提取, 本设计主要包括三个部分:

- (1) 对串口的操作: 打开和读串口。
- (2) 提取 RMC 出语句: 从 GPS 信息流语句中提取出想要的这个语句。
- (3) NEMA0183 定位信息的提取: 把这个语句中定位信息 (包括时间、经度和纬度等) 提取出来。

程序框图 5.4.2 如下所示。

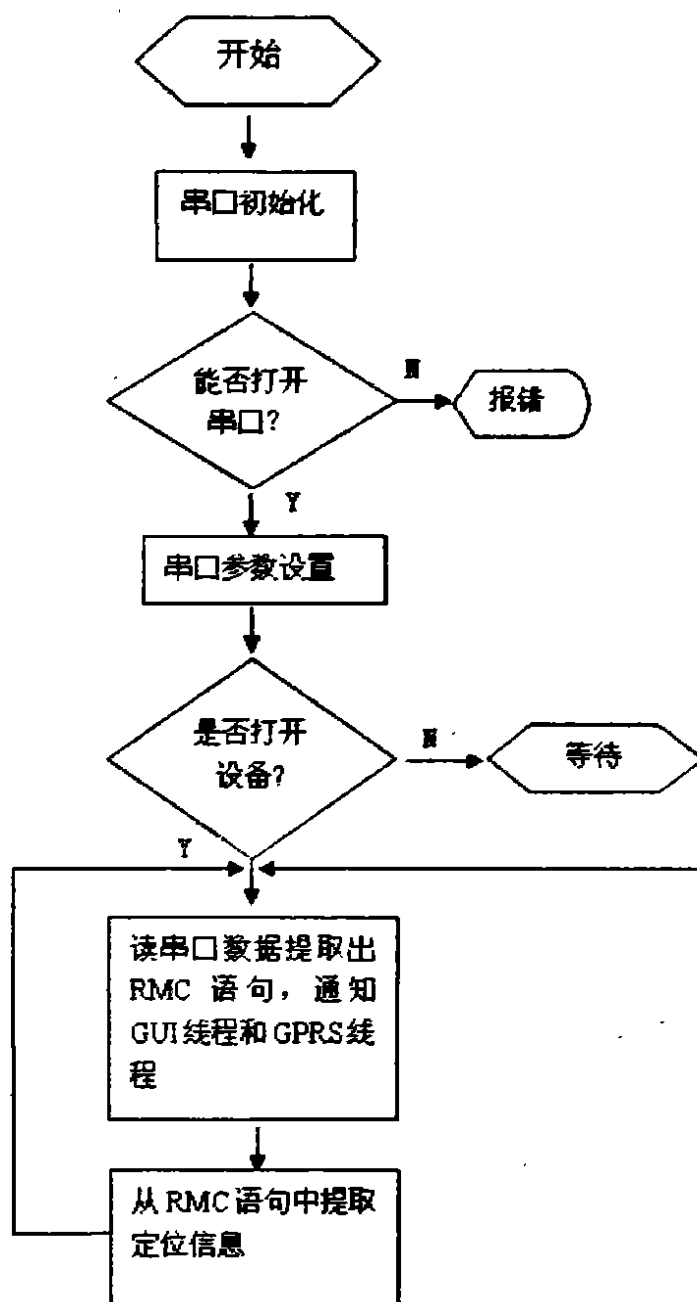


图 5.4.2 GPS 信息提取流程图

相关函数如下:

● 封装的类:

```

class QGPSDevice : public QThread
{
    ...
};
    
```

//头文件中封装了从  
//QThread 继承的类  
//创建了一个线程

● 对串口的操作:

```
void QGPSDevice::openDevice()           //打开串口, 成功后进行参数设置
{
    termios options;                     //termios 的变量声明
    fd = open(serialPort().latin1(), O_RDONLY | O_NOCTTY | O_NDELAY);
                                           //打开串口
    if(fd == -1)
    {
        perror("Could not open serial port!"); // 打开失败报错
    }
    fcntl(fd, F_SETFL, 0);                //设置为阻塞模式
    tcgetattr(fd, &options);              //获取当前的端口配置
    cfsetispeed(&options, B4800);         //波特率为 4800bps
    options.c_cflag |= (CLOCAL | CREAD);  //挂起时不占有端口
    options.c_cflag &= ~PARENB;           //取消校验位
    options.c_cflag &= ~CSTOPB;           //使用一个停止位
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;                //采用 8 个数据位
    tcsetattr(fd, TCSANOW, &options);     //新的设置生效
}
```

● 从数据流中提取 RMC 语句:

```
void QGPSDevice::run()                  //QTread 线程类的执行从 run()开始
{
    int index = 0;                       //初始化
    char bufferChar;
    char bufferString[100];
    bool safeStopLoop = false;
    QString parseRMCStr;
    int seconds=0;
    do                                    //只要 GPS 开机, 就一直循环
    {
        mutex->lock();                   //加锁
        safeStopLoop = stopLoop;
        mutex->unlock();                 //解锁
        read(fd, &bufferChar, 1);       //读串口“一个数据”
        if(bufferChar == '$')           //检测是否为 GPS 信息头
        {
            index = 0;
            bufferString[index] = bufferChar;
            do                            //把整个数据串提取出来
            {
                read(fd, &bufferChar, 1);
                if(bufferChar != '\0' && (isalnum(bufferChar) || isspace(bufferChar) ||
                ispunct(bufferChar)))
                {
                    index ++;
                    bufferString[index] = bufferChar;
                }
            } while(bufferChar != 0x0a && bufferChar != 0x0d);
            bufferString[index + 1] = '\0';
        }
    }
```

```

        mutex->lock();
        if(bufferString[3] == 'R' && bufferString[4] == 'M' && bufferString[5] == 'C')
            //检测是否为 RMC 语句
        {
            ++seconds;
            if(seconds==setSeconds)                //每 5 秒种唤醒 GPRS 一次
            {
                for(int i=0;i<strlen(bufferString);i++)
                    parseRMCStr+=bufferString;
                setParseRMCStr(parseRMCStr);
                waitForSend.wakeAll();
                seconds=0;
            }
            parseRMC(bufferString);                //转到 RMC 提取函数
            QApplication::postEvent(targetWidget,new QCustomEvent(10000));
            //通知界面线程更新数据
        }
        mutex->unlock();
    } while(safeStopLoop == false);
}

```

● 在 RMC 语句中，提取定位信息

void QGPSDevice::parseRMC(const char \*rmcString)

```

{
    mutex->lock();

    int j, k, commaPos;
    char tempString[100];
    int tempInt;
    float tempFloat;
    long tempLong;

    j = 7;                                //提取定位时间
    k = 0;
    while(*(rmcString + j) != 0x2c)
    {
        tempString[k] = *(rmcString + j);
        j ++;
        k ++;
    }

    tempString[k] = '\0';
    commaPos = j;

    sscanf(tempString, "%d", &tempLong);

    setUtcTime(tempLong);
    setUtcHour((utcTime() / 10000));
    setUtcMinute((utcTime() - (utcHour() * 10000)) / 100);
    setUtcSecond(utcTime() - (utcHour() * 10000) - (utcMinute() * 100));

    //其它定位信息提取方法相同
}

```

注：以上使用了 Linux 下关于串口的操作函数，所以要在程序的实现中，要将这些 Linux 头文件包括(#include<>)进去。



## 5.5 GPRS 程序设计

如前 3.7 节所述, GPRS 模块与处理器是通过串口相连接的, GPRS 移动台设备 ME (GPRS 模块) 与对端的 ME 或 ISP 服务商之间通信是利用了 TCP/IP 协议。从 ME 发出或者结束的数据包都是基于 IP 协议的 IP 数据包。GPRS 模块必须经过两个过程才能完成通信<sup>[31]</sup>:

(1) GPRS 移动台 ME 在 GPRS 网络中的附着。

(2) GPRS 移动台 ME 能够与 Linux 的应用进行 IP 数据交互——PDP 上下文激活过程 (Packet Data Protocol 分组数据协议)。

附着过程用于接入 GPRS 网, 通过附着过程将自己的信息登记在 SGSN (Serving GPRS Support Node, 业务支持节点) 中, 只有这样, SGSN 才能对用户进行移动性管理。PDP 激活过程用于激活 IP 协议, 保证数据能以 IP 包的形式传送, 只有通过激活过程, 才能使移动台与 GGSN (Gate Way GPRS Support Node, 网关支持节点) 建立一条逻辑通路进行数据传输。PDP 激活过程较为复杂, 涉及到网络的多个协议, 如 PPP 协议、LCP (链路控制协议)、NCP (网络控制协议)、PAP (密码认证协议) 和 IPCP (Internet) 协议控制协议等。

GPRS 模块在 Linux 下要完成与挂接在 Internet 上的通信服务器通信, 首先需要经过 GPRS 模块初始化和 GPRS 网络连接, 然后才能使用 TCP/IP 协议进行数据传输。前者是通过 PPP 协议实现的, 后者是通过 Qt 应用程序实现的。

### 5.5.1 PPP 拨号程序

#### 5.5.1.1 PPP 协议

PPP 协议 (Point to Point) 在 OS (开放系统) 互连网络参考模型处于数据链路层。这种链路提供全双工操作, 并按照顺序传递数据包, 它提供了在串行点对点链路上传输数据包的方法, 支持异步 8 位数据同步连接。并提供了一种管理两点间会话的有效方法, 正在取代 SLIP (Serial Line IP) 协议成为点对点网络的标准。一个 PPP 会话分为四个步骤: 连接建立、连接质量控制、网络层协议配置、连接终止; PPP 会话提供了密码认证协议 (PAP) 或者邀请握手认证协议 (CHAP) 来保证连接安全。

PPP 协议包括以下三个部分:

- 数据帧封装方法
- 链路控制协议 LCP (Link Control Protocol): 用于对封装格式选项的自动

协商, 建立和终止连接, 探测链路错误和配置错误

- 针对不同网络协议的网络控制协议 NCP (Network Control Protocol): PPP 协议规定了针对每一种网络协议都有相应的网络控制协议, 并用它们来管理各个协议不同的需求

GPRS 通过 PPP 协议对 GPRS 模块进行控制可以用两种方式进行:

- (1) 用工具软件, 如 pppd 等。
- (2) 编写软件, 使用 AT 指令通过对串口操作, 来控制 GPRS 模块。

本设计就是采用 pppd 进行控制的, 车载终端连接到 Internet 需要作如下的准备工作:

- (1) 支持 PPP 协议的 Linux 内核。
- (2) 交叉编译生成两个实用程序 pppd (PPP 守护进程) 和 chat (会话程序), 这个程序用于实现 PPP 连接。
- (3) 编写拨号脚本实现自动拨号。

#### 5.5.1.2 Linux 下拨号环境的建立

Linux 系统下的拨号脚本程序需要利用的程序有 pppd 和 chat。pppd 程序是一个 PPP 守护进程, 提供对 PPP 协议的支持, 它的用途是建立并维持与服务器的 PPP 连接, 传输数据; chat 程序的用途是拨号并等待提示, 根据提示输入用户名和密码等登录信息。Linux 内核必须支持 PPP 协议, 在 kernel 目录路径下用 make menuconfig 语句查看 linux 内核是否支持 PPP 协议。如果不支持, 则需重新编译内核, 并做以下的工作:

- 配置 kernel 使它支持如下 PPP 选项:
  - PPP(point-to-point) support
  - PPP multilink support (EXPERIMENTAL)
  - PPP support for async serial prots
  - PPP support for sync tty ports
  - PPP Deflate compression
  - PPP BSD-Compress compression
- 交叉编译 ppp-2.4.1:
  - 将 ./pppd/pppd, ./chat/chat, ./pppdump/pppdump, ./pppstats/pppstats 四个文件拷贝到嵌入式文件系统的 /usr/sbin 目

录下，并将它们的文件属性改为 755，如：chmod 755 /usr/sbin/pppd

- 在嵌入式文件系统中，建立一个 ppp 设备：mknod /dev/ppp c 108 0
- 改变/etc/ppp 文件夹的属性，chmod 600 /etc/ppp
- 编辑/etc/modules.conf 文档，增加一行：  
options ppp\_async flag\_time=0
- 编辑/etc/resolv.conf，加入一行：  
nameserver 211.136.17.107

### 5.5.1.3 PPP 拨号脚本

Linux 下的拨号过程如下：

- (1) pppd 程序调用 chat 会话程序。
- (2) chat 程序负责拨号登录，启动服务器端的 pppd 程序验证身份，然后结束。
- (3) 由 pppd 程序继续 chat 的工作，与服务端的 pppd 程序进行握手，建立 PPP 连接。

脚本示意如图 5.5.1.3 所示。

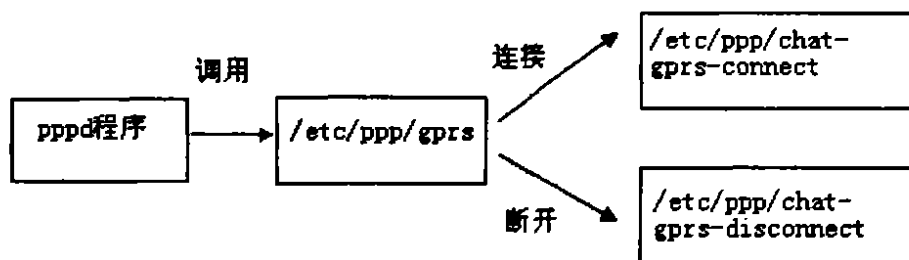


图 5.5.1.3 PPP 拨号脚本示意图

pppd 程序通过一些选项来控制 PPP 链路的所有方面，这些选项放在一个脚本文件中，系统可以让 pppd 程序使用的选项存放在/etc/ppp/options 文件中，这个文件由 pppd 程序直接读取。文件可以为空，但必须存在。在主机上调试时，可以使用自定义的文件，存放目录为/etc/ppp/gprs，在终端下运行 pppd file /etc/ppp/gprs 进行拨号。

chat 是在嵌入式 Linux 系统和 GPRS 模块进行自动交互的程序，主要用于在本地和远程 pppd 守护进程之间建立连接。

文件/etc/ppp/chat-gprs-connect 和/etc/ppp/chat-gprs-disconnect 用于连接和断开拨号。

除了以上脚本文件外，如果拨入的服务器端需要密码认证协议（PAP）或邀请握手认证协议（CHAP）来保证连接安全，还需要在/etc/ppp 目录下配置 pap-secrets

或 chap-secrets。

PPP 拨号脚本见附录 B。

附录 B 说明：

- (1) “TIMEOUT” 设定接收所期待的超时时限。
- (2) “ABORT” 若收到后的字符串则终止执行，“OK” 收到 OK，接着执行。
- (3) “AT+CGATT” 登记接入 GPRS 网络，操作成功将返回 OK。
- (4) “A+CGATT?” 查看附着状态。
- (5) “AT+CGACT=1” 使用所存储的第一个 PDP 信息激活 PDP。
- (6) “AT+CGDCONT=1,“IP”” 用来设置 GPRS 节点服务器的 APN (Access Point Name, 接入点) 名称和属性，操作成功则返回 OK。在调试时，使用 “AT+CGDCONT=1,“IP”,“cmnet”,“0.0.0.0”,0,0” 登入中国移动网。如果车管系统向移动公司申请了专用 APN，则需将 cmnet 替换为移动公司提供的 APN 名称即可。
- (7) “ATDT\*99\*\*\*1#” 用来拨通连接 GPRS 节点服务器，操作成功则返回 CONNECT。“\*99\*\*\*1#” 是 GPRS 业务号码。目前拨号接入 GPRS 网络不需要使用用户名和密码。

通过将 pppd 的调用放入一个 Shell 文件中，使其可以开机自启动。Shell 脚本文件就是一个普通的文本文件，它可以用任何文本编辑器来创建和修改。在 Shell 文件中，可以输入命令调用，并加入所在要在命令行上使用的参数。这里使用 vi 来创建一个 /etc/ppp/startgprs 的 Shell 脚本。

```
#!/bin/sh
```

```
exec /usr/sbin/pppd
```

然后通过命令：chmod a+x startgprs 使其成为可执行脚本。

通过命令 echo “/etc/ppp/startgprs” >>/etc/rc.d/rc.local，然后使用 vi 编辑器，后面加上 start 完成自启动功能。

### 5.5.2 Qt 网络通信

在 GPRS 模块拨号上网之后，下面要做的是把定位信息等发送到网络上。本设计只做了数据发送这一块的工作，每 5 秒种发送一次信息，程序流程图 5.5.2。

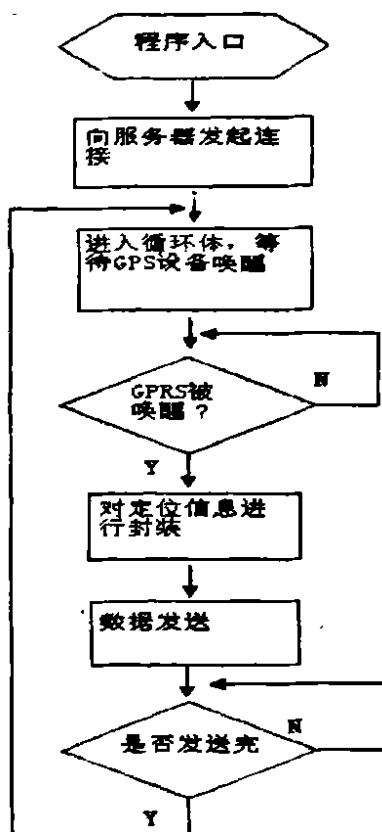


图 5.5.2 GPRS 信息发送流程图

消息封装的格式如下:

标志位+移动终端 ID+经度+纬度+速度+定位时间+车辆状态

说明:

(1) 标志位是用来标示该数据是上行数据或是下行数据, 该位为“0”表示是上行数据; 为“1”时表示下行数据。

(2) 移动终端 ID 指被监控车辆的识别号码(唯一), 这里可用车牌号码。

(3) 经度、纬度、速度、定位时间, 可以用提取后的各定位信息, 也可以直接使用 RMC 语句, 在服务器端提取即可。

(4) 车辆状态包括: 故障等(可根据实际情况增减), 可自行定义, 例如: “0”表示正常, “1”表示报警, “2”表示故障等。本设计中暂不使用, 以“0”表示。

如前所述, Qt 网络通信程序也是通过继承线程类 QThread 的方式实现的, 属于非界面线程 non-GUI 线程。在 Qt 中, 提供的网络类包括 QFtp、QHttp、QSocket 和 QSocketNotifier 都是依赖事件环(event loop)的, 不能在非界面线程中使用的。Qt 提供了一个低等级网络类 QSocketDevice, 它是一个平台独立的 socket API, 可以用在非界面线程中。由于把网络通信全部放在一个线程里执行, 这样就可使用阻塞

式 (blocking) 的网络操作, 可一直发送数据, 而不会发生死锁现象。QSocket 类作为封装级别高的类, 只能以 TCP 方式工作。而 QSocketDevice 能够以 TCP (QSocketDevice::Stream) 和 UDP (QSocketDevice::Datagram) 两种工作方式工作。UDP 无连接且不可靠, 但相对 TCP 而言, 是一种轻量级的传输协议, 在发送数据的时候可以使用其 writeBlock(\*data, len, HostAddress, Port) 函数。本设计采用的是 TCP 协议, QSocketDevice 默认的工作方式是 TCP 方式, 使用 writeBlock(\*data, len) 函数发送数据。

```

Class QGprsDevice : public QThread           //同 GPS 一样, 从 QThread
{
    ...                                     //类继承一个线程
};

    ● 执行线程:

void QGprsDevice ::run()                    //线程开始
{
    socket=new QSocketDevice;               //在头文件中声明过的 socket 指针
    socket->setBlocking(true);               //设置为阻塞模式
    if(!socket->connect(QHostAddress(HostAddresss,Portnumber)))
        return;                            //连接到服务器

    while(1)                                //进入循环体
    {
        mutex.lock
        waitforSend.wait(&mutex);           //等待唤醒
        mutex.unlock;
        QString gpsInfo=gpsDevice->getGpsInfo();
        QString sendInfo="0"+"CAR_ID"+gpsInfo+"0"; //信息封装

        writeToSocket(sendInfo);            //数据发送
    }

    ● 发送函数

void QGprsDevice :: writeToSocket(const QString &str)
{
    QByteArray ba;
    QTextOutputStream out(ba);
    out<<str;                               //把字符串转换为字符数组

    int wrote=socket->writeBlock(ba.data(),ba.size()); //调用发送函数
    while(wrote<(int)ba.size())              //没发送完, 继续发送
        wrote+=socket->writeBlock(ba.data()+wrote,ba.size()-wrote);
}

```

QSocketDevice 作为低等级的网络类, 不能对字符串直接操作, 需要对其转换后才能发送。writeBlock() 函数返回发送的字符数。

## 5.6 main 函数

一个 Qt 程序就是一个 C++ 程序。main() 函数是程序的入口，所有的 C/C++ 程序都必须包含一个 main() 函数，main() 函数总是首先被执行。几乎在使用 Qt 的所有情况下，main() 只需要在把控制权交给 Qt 库之前执行一些初始化，然后 Qt 库通过事件来向程序告知用户的行为。程序如下：

```
#include <qapplication.h>           //每个 Qt 程序都必须使用一个 QApplication 对象
#include "qgps.h"                   //使用了界面类，则包含其头文件

int main( int argc, char ** argv ) {
    QApplication a( argc, argv );    //创建 QApplication 对象
    QGPS * mainWindow = new QGPS();  //创建界面类对象指针
    mainWindow->setCaption( "QGPSGPRS" ); //设置标题
    mainWindow->show();               //调用 show() 显示窗口部件
    a.connect(&a, SIGNAL(lastWindowClosed()), &a, SLOT(quit()));
                                    //窗口关闭时退出
    return a.exec();
}
```

在 exec() 函数中，Qt 接收和处理用户及系统事件，并把这些事件传递给相应的窗口部件。return a.exec() 是 main() 把控制权交给 Qt。当应用程序关闭时，exec() 函数返回。

## 5.7 调试

### 5.7.1 生成可执行文件

以上设计好源文件后，下面对其进行编译：

#### (1) 生成 .h 头文件和 .cpp 实现文件

在命令行中输入下面命令：

```
cd qt-2.3.2/bin
uic -o qgpsmainwindowui.h qgpsmainwindowui.ui
uic -o qgpsmainwindowui.cpp -iml qgpsmainwindowui.h
qgpsmainwindowui.ui
```

这样就自动生成了头文件 qgpsmainwindowui.h 和实现文件 qgpsmainwindowui.cpp。

#### (2) 编译链接工程

先用 tmake 生成 makefile 文件：

```
tmake -o makefile qgps.pro
```

然后会在当前路径下生成一个文件 makefile，打开这个文件，把文件中的

“LINK=arm-linux-gcc”改为“LINK=arm-linux-g++”，然后输入 make 命令，就可以编译链接整个工程，生成可执行文件 qgps。

### 5.7.2 固化到板上<sup>[33]</sup>

上面生成的二进制可执行文件要烧写固化到 Flash 中，在开发板上运行需要 Linux 系统和 Qt/Embedded 库支持。Linux 内核已经烧写到 Flash 中，需要将 Qt/Embedded 库包含在文件系统中烧写到 Flash 中以支持上面二进制可执行文件 qgps 运行。

下面是具体的方法与步骤：

(1) 前 5.1 节建立的是基于 PC 的开发平台，这里需重新安装基于 ARM 的开发环境。

(2) 将 qgps 下载到开发板的/opt/qtopia/bin 目录下，在开发板上设置环境变量。

(3) 制作一个 32×32 大小的 PNG 格式的图标文件，将文件存放在 \$QPEDIR/pics/inline 目录下，然后使用以下命令将 \$QPEDIR/pics/inline 目录下的所有图形文件转换成为一个 c 语言文件，这个头文件包含了该目录下的图形文件的 rgb 信息。

```
Qtembed -images $QPEDIR/pics/inline/*. *  
>$QPEDIR/src/libraries/qtopia/inlinepics_p.h
```

上面 QPEDIR 是上面解压缩路径的环境变量，qmbed 是 qt for X11 上发布的工具。

(4) 建立一个文本文件，在文件中添加以下内容，这些内容指明了应用的名称，图标名等，等然后将文件更名为 qgps.desktop，保存在 \$QPEDIR/apps/applications 目录下。

```
[Destop Entry]  
Comment=qgps  
Exec= qgps  
Icon= qgps  
Type= Application  
Name= qgps
```

(5) 利用原有的 root\_yaffs 的根文件系统映象，把新建的应用的相关文件添加到这个根文件系统中。这些文件包括: qgps.desktop 文件，包含了图标的库文件 libqte.so.\*，和应用程序的可执行文件 qgps。将生成的新的根文件系统烧写到 S3C2410 开发板上，重新启动即可看到 Qt 界面。



## 6 总结与展望

### 6.1 总结

本文从当今车载定位终端设备的特点和要求出发，设计了基于 ARM 的车载 GPS/GPRS 系统，主要完成以下几项工作：

- (1) 根据现在市场上主流的方案，选择合适的芯片、模块、系统、开发平台等，完成总本设计方案。
- (2) 采用 S3C2410 作为系统微处理器，设计了 ARM 嵌入式系统的外围电路，结合 GPS 模块和 GPRS 模块电路，完成车载定位系统的硬件设计。
- (3) 采用嵌入式 Linux 操作系统，完成 Qt 界面设计，GPS 和 GPRS 应用程序设计。
- (4) 建立交叉开发平台，对系统的部分功能进行测试。

### 6.2 展望

由于时间限制，本车载终端只是具备了基本的功能，还有很多方面需要完善，还有很多的工作需要去做：

- (1) 功能上还比较单一，应根据市场的需求，添加更多实用的功能。比如：摄像功能；语音通信；建立数据库，能够储存和查询行驶路线和数据；建立电子地图，使其更具实用价值。
- (2) 由于本设计只是在实验室里进行调试，还没有进行实际产品的生产，因此对终端的抗干扰性和通信性能还需做进一步的测试。

## 致 谢

时间飞逝，两年的研究生生活即将结束。从课题的选题、论证、设计和调试到现在，已经有一年的时间了。在这段时间中，我的导师张永清副教授在课题的选定、课题的指导、论文的撰写、经费的保证和实验条件的提供等方面给予了全面的支持。在我攻读硕士期间，导师在学习和工作方面对我严格要求，并且在实验方法的确立和一些技术细节、经验等方面都给了我指导性的意见和建议，使我在完成学业的过程中受益匪浅。我更是从导师那儿学到了一丝不苟的科研态度，踏踏实实的工作作风，在此，向张老师表达我最深的敬意和谢意。导师严谨、求实的治学精神，将成为我今后工作和学习的榜样。

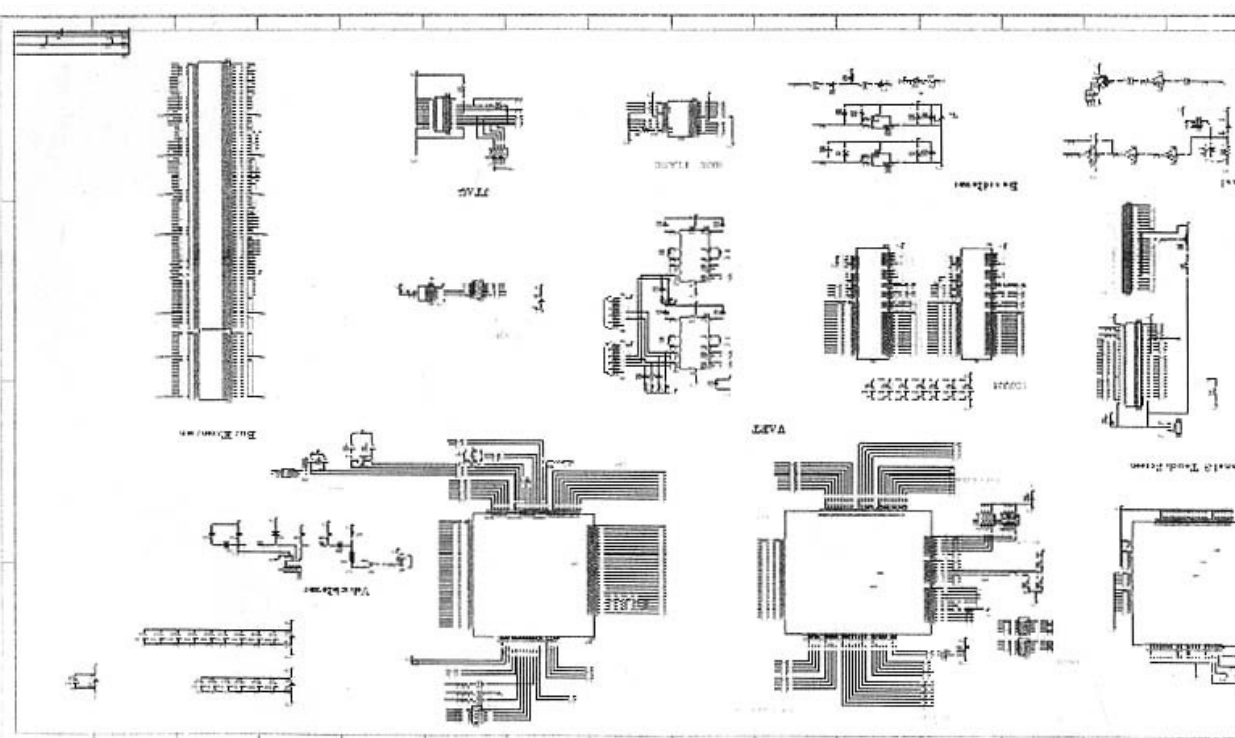
另外，我也要衷心感谢本教研室的王锦涛，刘忠治，林景源等同学，他们和我一起度过了这段时光。

最后，要向远方默默支持、鼓励和关怀我的父母和姐姐道声谢谢，他们无私的爱是我永久的动力。

## 参考文献

- 1 杜春雷. ARM 体系系统结构与编程. 第一版. 北京: 清华大学出版社, 2003
- 2 李天文. GPS 原理及应用. 北京: 科学出版社, 2003
- 3 李洪涛等. GPS 应用程序设计. 北京: 科学出版社, 1999
- 4 林宇, 郭凌云. Linux 网络编程. 北京: 人民邮电出版社, 2000
- 5 范永开, 杨爱林. Linux 应用开发技术详解. 北京: 人民邮电出版社, 2006
- 6 郑灵翔. 嵌入式系统设计与应用开发. 北京: 北京航空航天大学出版社, 2006
- 7 张晓林, 崔迎炜等. 嵌入式系统设计与实践. 北京: 北京航空航天大学出版社, 2006
- 8 郑莉等. C++语言程序设计案例教程. 北京: 清华大学出版社, 2005
- 9 李现勇. Visual C++串口通信技术与工程实践. 北京: 人民邮电出版社, 2004
- 10 Arthur Griffith. KDE 2/Qt 编程宝典. 北京: 电子工业出版社, 2002
- 11 Daniel Solin. 24 小时学通 Qt 编程. 北京: 人民邮电出版社, 2000
- 12 Xteam (中国). Qt 程序设计. 北京: 清华大学出版社, 2002
- 13 李强. C++语言程序设计. 北京: 北方交通大学出版社, 2003
- 14 钟章队, 蒋文怡, 李红君等. GPRS 通用分组无线业务. 北京: 人民邮电出版社, 2002
- 15 Jasmin Blanchette 等. C++ GUI Programming with Qt 3. New Jersey: Prentice Hall PTR, 2004
- 16 田泽. 嵌入式系统开发与应用教程. 北京: 北京航空航天大学出版社, 2005
- 17 孟庆昌. 操作系统教程. 第一版. 西安: 西安电子科技大学出版社, 2004
- 18 张艳红. 基于嵌入式 Linux 系统 PPP 协议的实现. 信息与通信. 2006 (3): 43-46
- 19 赵立权. 在 Linux 下实现安全 PPP 连接. 云南师范大学学报. 2002 (3): 7-10
- 20 张建军, 陈志雄, 韩江洪等. Linux 平台下串行口全双工通讯的实现. 合肥工业大学学报. 2003 (4): 165-170
- 21 王作羽. Qt/Embedded 下 GUI 应用程序开发. 开放系统世界. 2006 (6): 93-100
- 22 李外云, 胡文静, 刘锦高. 基于 Sinstang 的嵌入式 Linux 手持设备的 GSM 和 GPS 应用研究. 华东师范大学学报. 2006 (3): 120-125
- 23 成洁, 路欣. 嵌入式 Linux 平台的 GPS 数据采集研究. 江西理工大学学报. 2006 (3): 24-26
- 24 周云辉, 范立志. 中远程终端拨号接入技术的实现. 湖南理工学院学报. 2004 (3): 66-69
- 25 周明, 李军, 赵辉. 基于嵌入式 Linux 的移动互联网的实现. 天津理工学院学报

- 2003(3): 39-41
- 26 王新云. 玩转 Linux 下的 PPP 连接. 开放系统世界. 2003(6): 54-56
- 27 罗亮, 彭容修. GPRS 在嵌入式手持终端上的实现. 现代电子技术. 2004(4): 76-81
- 28 赵庆丽, 魏东兴, 郭永山. 基于嵌入式 Linux 的 GPRS 数据传输系统. 信息技术. 2004(3): 1-19
- 29 万国建. 基于 ARM 的智能线径测控装置的设计与研究
- 30 林燕琼. 基于嵌入式系统的广播覆盖效果监测记录装置的设计
- 31 杨小东. 基于 ARM 的 GPRS 无线数传温室测控系统
- 32 祁小钰. 基于 ARM 的数据处理终端设计
- 33 ARMSYS2410 Linux 用户手册. 1.3 版. 杭州: 杭州立宇泰电子有限公司编著
- 34 ARMSYS2410-B 开发板硬件用户手册. 1.4 版. 杭州: 杭州立宇泰电子有限公司编著
- 35 罗志勇. E531 用户使用维护说明书. 1.0 版. 上海: 上海易罗信息科技有限公司, E5312.026.005SS
- 36 SBC2410-2410X 使用手册. 广州友善之臂科技有限公司
- 37 neozenkai. qgps. [www.sourceforge.net](http://www.sourceforge.net)
- 38 S3C2410X 32-BIT RISC MICROPROCESSOR USER'S MANUAL. Samsung Electronics.
- 39 K9F1208U0M-YCB0 64M×8Bit NAND Flash Memory. Samsung Electronics
- 40 HY57V561620C(L)T(P) 4 Banks×4M×16Bit Synchronous DRAM. Hynix
- 41 MAX3232 MULTICHANNEL RS232 LINE DRIVER/RECEIVER. Texas INSTRUMENTS
- 42 Wismo Quik Q2406 and Q2426 Product Specification. Wavecom Corporation
- 43 Qt Desiger Manual



图附录 A: S3C2410 系统电路原理图

## 附录 B: PPP 拨号脚本

```
#-----#
# File:
# /etc/ppp/gprs
receive-all
# Give some debug info
debug
kdebug 7
# Print out all the option values which have been set.
dump
/dev/ttyS0 # serial cable
# Serial port line speed
115200
# Turn off waiting of carrier detect or flow control signal
# With IrDA it should be disabled with nocrtscts option.
-crtscts # serial cable
#nocrtscts # IrDA
# Ignore carrier detect signal from the modem
local
# To keep pppd on the terminal
nodetach
# Accept the peer's idea of our local IP address
ipcp-accept-local
# Accept the peer's idea of its (remote) IP address
ipcp-accept-remote
# IP addresses:
# - accept peers idea of our local address and set address peer as 10.0.0.1
# (any address would do, since IPCP gives 0.0.0.0 to it)
# - if you use the 10. network at home or something and pppd rejects it,
# change the address to something else
0.0.0.0:0.0.0.0
-chap
#-pap
# pppd must not propose any IP address to the peer!
#noipdefault
# No ppp compression
novj
novjccomp
papcrypt
nodeflate
#No ppp magic number
nomagic
# no asyn cmap
asynctest 0
# Add default route
defaultroute
# Connect script
connect /etc/ppp/gprs-connect-chat
```

```
# Disconnect script
disconnect /etc/ppp/gprs-disconnect-chat
#-----#
# File:
# /etc/ppp/options
receive-all
nopcomp
noaccomp
nomagic
debug
/dev/ttyS0
115200
connect '/usr/sbin/chat -e -f /etc/ppp/gprs-connect-chat -v'
disconnect '/usr/sbin/chat -e -f /etc/ppp/chat-disconnect -v'
modem
noauth
noccp
novj
novjccomp
defaultroute
noipdefault
user njust
lock
#-----#
#!/bin/sh
# File:
# /etc/ppp/chat-gprs-connect
exec chat \
TIMEOUT 5 \
ECHO ON \
ABORT '\nBUSY\r' \
ABORT '\nERROR\r' \
ABORT '\nNO ANSWER\r' \
ABORT '\nNO CARRIER\r' \
ABORT '\nNO DIALTONE\r' \
ABORT '\nRINGING\r\n\r\nRINGING\r' \
" AT \
TIMEOUT 12 \
SAY "Press CTRL-C to close the connection at any stage!" \
SAY "\ndefining PDP context...\n" \
OK AT+CGATT? \
OK AT+CGATT=1 \
OK AT+CGATT? \
OK 'AT+CGDCONT=1,"IP" \
OK AT+CGQREQ=1,0,0,3,0,0 \
OK AT+CGACT=1,1 \
OK ATDT*99***1# \
TIMEOUT 120 \
SAY "\nwaiting up to 2 mintues for connect...\n"
```

```
CONNECT "" \
SAY "\nConnected. now logging in...\n" \
SAY "\nIf the following ppp negotiations fail,\n" \
SAY "try restarting the phone.\n"
#-----#
#!/bin/sh
# File:
# /etc/ppp/chat-gprs-disconnect
# send break
exec /usr/sbin/chat -V -s -S \
ABORT "BUSY" \
ABORT "ERROR" \
ABORT "NO DIALTONE" \
SAY "\nSending break to the modem\n" \
"" "\K" \
"" "+++ATH" \
SAY "\nPDP context detached\n"
#-----#
```



作者: [李文亮](#)  
学位授予单位: [南京理工大学](#)

## 相似文献(5条)

### 1. 期刊论文 [李彩红, Li Cai-hong 基于ARM的车载GPS定位终端的设计 -微计算机信息2008, 24\(19\)](#)

论文设计了一套基于ARM处理器的车载GPS系统. 采用AT91RM9200处理器为硬件平台, 在该处理器上移植Linux操作系统, 利用操作系统的资源编写程序实现GPS和GPRS的功能在车载GPS系统中的应用. 本文提供了一套切实可行的具有实时监控能力的车载GPS卫星定位系统设计方案.

### 2. 学位论文 [罗致 基于ARM处理器的车载GPS定位终端 2006](#)

车载GPS定位终端在过去十年内已经成为汽车工业发展的焦点. 在欧美国家和日本, 车载GPS定位终端在最近几年内得以广泛的应用. 车载GPS定位终端是融全球卫星定位技术(GPS)和现代无线通信技术于一体的高科技系统. 该终端的主要功能是通过GPS模块从卫星获取GPS数据, 将移动车辆的动态位置(经度、纬度、时间、速度)等信息实时地通过无线通信链路上传至监控中心, 同时接收监控中心发送的控制命令. 目前的车辆监控系统中大多采用GSM通信网以短信息的方式进行通信, 不能充分满足实际应用的需要. 而GPRS通用分组无线业务是一种以分组交换技术为基础, 采用IP数据网络协议的高效数据传输网络, 可以弥补GSM网络的不足. 车载GPS定位终端不仅在智能交通系统中担负主要作用, 同时还可以提供防盗防抢报警, 公交车报站, 物流车辆调度等多种服务.

本文通过对GPS卫星定位理论的详细介绍, 对GPRS移动通信技术规范的细致分析以及对ARM嵌入式硬件系统和ARMLinux嵌入式操作系统等技术的不断实践提出了一套基于ARM处理器的车载GPS系统的设计方案. 在本文的设计方案中采用AT 91RM9200ARM9处理器为硬件平台, 在该处理器上移植ARMLinux操作系统, 利用操作系统的资源编写程序实现GPS和GPRS的功能在车载GPS系统中的应用. 本文提供了一套切实可行的具有实时监控能力的车载GPS卫星定位系统设计方案. 实际的工程设计已经进入试验阶段, 运行测试效果良好.

### 3. 期刊论文 [张勤, 赵珊, 张立芬 车载GPS导航系统的设计 -今日电子2009, ""\(6\)](#)

本设计在系统终端采用了ARM处理器和嵌入式操作系统 $\mu C/OS-II$ 作为开发平台, 通过采用ARM处理器可达到最大为60MHz的CPU操作频率, 使得数据处理能力大大加强, 同时, 基于嵌入式操作系统 $\mu C/OS-II$ 开发设计的软件具备了很强的扩展性和稳定性.

### 4. 期刊论文 [黄勋, 唐慧强, Huang Xun, Tang Huiqian 嵌入式平台ARM-uClinux的构建与应用开发 -武汉理工大学学报\(交通科学与工程版\) 2006, 30\(1\)](#)

介绍了ARM处理器以及嵌入式操作系统uClinux的特点. 将uClinux操作系统移植到ARM7上, 构建了ARM-uClinux嵌入式平台. 以此为基础完成了一种车载GPS接收机的软硬件设计. GP2015作为接收机的射频前端, 内嵌ARM7核的GP4020作为接收机的数字基带处理器. 此接收机消除以往处理器数据处理的瓶颈效应, 解决了体积小, 功耗低的问题.

### 5. 学位论文 [杨永志 基于ARM和嵌入式操作系统的GPS车载终端系统的研究与开发 2005](#)

GPS(全球定位系统)是一种全方位的实时定位技术. GPS车载定位与监控系统结合了GPS定位技术、计算机科学技术、数字无线通信技术和电子技术, 在汽车上实现GPS数据的接收和记录, 并结合GSM/GPRS无线传输网络技术, 利用其数据传输功能, 实现移动车辆与监控中心的双向数据传输, 对车辆进行跟踪和远程监控. GPS车载定位与监控终端是集各种高科技于一身的开放式车辆监控平台, 它能够跨地域对机动性强、数量众多的移动目标实行监控、紧急救援和提供各种信息服务.

本课题从工程实际出发, 对ARM处理器和嵌入式操作系统进行了深入研究, 设计了新一代的GPS车载终端. 这有别于以8051单片机为中央处理器的GPS车载终端系统, 处理器的位数由16位提高到32位, 数据吞吐能力大为加强. 同时软件采用嵌入式操作系统, 增加了系统的稳定性和安全性, 并增加了代码的可移植性. 在这一平台上进行相应的硬件和软件开发, 完成了无线数据通信系统和射频无线拨号器的设计和开发.

首先, 介绍了GPS车载终端的现状和发展前景, 这是开发新一代的GPS车载终端的出发点. 接着阐述了车载GPS全球定位系统的工作原理及特点, 并详细叙述了基于ARM处理器和嵌入式操作系统的GPS车载终端的总体结构, 给出了新一代GPS车载终端的实际设计方案.

接着, 在GPS车载终端的设计中, 分别叙述了新一代基于ARM处理器和 $\mu C/OS-II$ 嵌入式操作系统的GPS车载终端的硬件设计方案和软件设计流程. 描述了ARM微处理器的性能特点及其外围电路的组成, 同时也介绍了在软件方面采用 $\mu C/OS-II$ 这种嵌入式操作系统的优势和具体应用.

最后, 详细论述了GSM/GPRS无线通信模块及其外围电路的设计和射频无线拨号器的设计. 其中GSM/GPRS无线通信模块在车载终端中担负着网络信息传输的重要功能; 射频无线拨号器集成了拨打电话、信息求助等功能, 把有线通信转变为无线通信, 提高了车载终端通话应用方面的灵活性及安全性. 同时, 还探讨了在车载终端相关领域进行深入研究的可能性.

本文链接: [http://d.g.wanfangdata.com.cn/Thesis\\_Y1154667.aspx](http://d.g.wanfangdata.com.cn/Thesis_Y1154667.aspx)

授权使用: 湖南大学(hunandx), 授权号: 8e60540e-bb79-4c06-a060-9dad0102e821

下载时间: 2010年7月8日