**《嵌入式系统结构与操作系统》课程教学大纲**

**一、课程的基本情况**

课程中文名称：嵌入式系统结构与操作系统

课程英文名称：Embedded System Architecture and Operating System

课程类别：专业课（模块）

课程性质：限选课

总 学 时：54 讲课学时：54 实验学时：单独开设实践课程

学 分：3.5

授课对象：计算机科学与技术

前导课程：操作系统；计算机组成原理

**二、教学目的**

该课程针对目前最流行的32位嵌入式处理器ARM体系结构、指令系统及其应用开发方法进行讲授。通过理论教学和实践教学，使学生掌握嵌入式系统体系结构，嵌入式处理器结构（ARM架构为主），异常处理，存储处理，系统控制过程，流水线作业及各种I/O接口；嵌入式系统开发应用方法；嵌入式操作系统，实时多任务操作系统，基于Android的系统级开发方法。

**三、教学内容及基本要求**

**第1章 ARM微处理器概述**

本章简要介绍ARM微处理器的一些基本概念、特点及应用领域，使读者对ARM微处理器及相关技术有一个大致的了解，为后续章节的学习打下基础。

本章主要内容包括：

● ARM微处理器概述

● ARM微处理器体系结构

● ARM微处理器系列

● ARM微处理器的应用与选型

1.1 ARM微处理器概述

1.1.1 ARM公司简介

ARM于1990年11月在英国伦敦成立，前身为Acorn计算机公司，后改名Advance RISC Machines Limited公司（ARM公司）。ARM既可以认为是一个公司的名字，也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。

1.1.2 ARM微处理器的特点

本章介绍了采用RISC架构的ARM微处理器的一般特点。

1.2 ARM微处理器系统结构

1.2.1 ARM体系结构的版本

ARM指令集体系结构，从最初开发至今已有了重大改进，而且将会不断完善和发展。为了精确表达每个ARM实现中所使用的指令集，到目前ARM体系结构共定义了7个版本，以版本号v1～v7表示。

1.2.2 ARM体系结构的变种及命名格式

（1）Thumb指令集（T变种）

（2）长乘法指令（M变种）

（3）增强型DSP指令（E变种）

（4）Java加速器Jazelle（J变种）

（5）SIMD变种（ARM媒体功能扩展）

1.2.3 ARM系统结构的命名格式

ARM体系架构版本号一般命名格式：ARM vN（x）B（x）B …

1.3 ARM微处理器系列

1.3.1 ARM7微处理器

ARM7采用采用ARMV4T冯·诺依曼（Von-Neumann）结构，数据存储器和程序存储器重合在一起。同时，此结构也被大多数计算机所采用。

1.3.2 ARM9微处理器

ARM9采用ARMV4T哈佛（Harvard）结构，程序存储器与数据存储器分开，提供了较大的存储器带宽。同时，大多数DSP都采用此结构。

1.3.3 ARM9E微处理器

ARM9E系列微处理器为可综合处理器，使用单一的处理器内核提供了微控制器、DSP、Java应用系统的解决方案，极大的减少了芯片的面积和系统的复杂程度。

1.3.4 ARM10E微处理器

ARM10E系列微处理器具有高性能、 低功耗的特点， 由于采用了新的体系结构（6 级整数流水线，指令执行效率更高）， 与同等的ARM9器件相比较，在同样的时钟频率下，性能提高了近50％，同时，ARM10E系列微处理器采用了两种先进的节能方式，使其功耗极低。

1.3.5 SecurCore微处理器

SecurCore系列微处理器专为安全需要而设计，提供了完善的 32 位 RISC 技术的安全解决方案，因此，SecurCore系列微处理器除了具有ARM体系结构的低功耗、高性能的特点外，还具有其独特的优势，即提供了对安全解决方案的支持。

1.3.6 StrongARM微处理器

StrongARM系列处理器是英特尔旗下的嵌入式处理器，基于ARMV4T结构，旨在支持WinCE3.0-PocketPC系统。

1.3.7 Xscale微处理器

Xscale处理器是基于ARMv5TE体系结构的解决方案，是一款全性能、高性价比、低功耗的处理器。它支持16位的Thumb指令和DSP指令集，已使用在数字移动电话、个人数字助理和网络产品等场合。

1.3.8 ARM Cortex系列微处理器

基于ARMv7版本的ARM Cortex系列产品由A、R、M三个系列组成，具体分类延续了一直以来ARM面向具体应用设计CPU的思路。

1.4 ARM微处理器的应用与选型

1.4.1 ARM微处理器的应用领域

到目前为止，ARM微处理器及技术的应用几乎已经深入到各个领域。

1.4.2 ARM微处理器的应用选型

鉴于ARM微处理器的众多优点，随着国内外嵌入式应用领域的逐步发展，ARM微处理器必然会获得广泛的重视和应用。

1.5 本章小结

本章对ARM微处理器、ARM微处理器体系结构等基本概念做了一些简单的介绍，希望读者通过对本章的阅读，能对ARM微处理器、ARM微处理器的相关技术及应用有一个总体上的认识。

**第2章 ARM微处理器的指令系统**

本章介绍ARM指令集、Thumb指令集，以及各类指令对应的寻址方式，通过对本章的阅读，希望读者能了解ARM微处理器所支持的指令集及具体的使用方法。

本章的主要内容包括：

● ARM指令集、Thumb指令集概述。

● ARM指令集的分类与具体应用。

● Thumb指令集简介及应用场合。

2.1 ARM微处理器的指令集概述

2.1.1 ARM微处理器的指令的分类与格式

ARM微处理器的指令集是加载/存储型的，也即指令集仅能处理寄存器中的数据，而且处理结果都要放回寄存器中，而对系统存储器的访问则需要通过专门的加载/存储指令来完成。

2.1.2 指令的条件域

当处理器工作在ARM状态时，几乎所有的指令均根据CPSR中条件码的状态和指令的条件域有条件的执行。当指令的执行条件满足时，指令被执行，否则指令被忽略。

2. 2 ARM指令的寻址方式

所谓寻址方式就是处理器根据指令中给出的地址信息来寻找物理地址的方式。

2.2.1 立即寻址

立即寻址也叫立即数寻址，这是一种特殊的寻址方式，操作数本身就在指令中给出，只要取出指令也就取到了操作数。

2.2.2 寄存器寻址

寄存器寻址就是利用寄存器中的数值作为操作数，这种寻址方式是各类微处理器经常采用的一种方式，也是一种执行效率较高的寻址方式。

2.2.3 寄存器间接寻址

寄存器间接寻址就是以寄存器中的值作为操作数的地址，而操作数本身存放在存储器中。

2.2.4 基址变址寻址

基址变址寻址就是将寄存器（该寄存器一般称作基址寄存器）的内容与指令中给出的地址偏移量相加，从而得到一个操作数的有效地址。

2.2.5 多寄存器寻址

采用多寄存器寻址方式，一条指令可以完成多个寄存器值的传送。这种寻址方式可以用一条指令完成传送最多16个通用寄存器的值。

2.2.6 相对寻址

与基址变址寻址方式相类似，相对寻址以程序计数器PC的当前值为基地址，指令中的地址标号作为偏移量，将两者相加之后得到操作数的有效地址。

2.2.7 堆栈寻址

堆栈是一种数据结构，按先进后出（First In Last Out，FILO）的方式工作，使用一个称作堆栈指针的专用寄存器指示当前的操作位置，堆栈指针总是指向栈顶。

2.3 ARM指令集

本节对ARM指令集的六大类指令进行详细的描述。

2.3.1 跳转指令

跳转指令用于实现程序流程的跳转，在ARM程序中有两种方法可以实现程序流程的跳转：

（1）使用专门的跳转指令。

（2）直接向程序计数器PC写入跳转地址值。

2.3.2 数据处理指令

数据处理指令可分为数据传送指令、算术逻辑运算指令和比较指令等。

2.3.3 乘法指令与乘加指令

ARM微处理器支持的乘法指令与乘加指令共有6条，可分为运算结果为32位和运算结果为64位两类，

2.3.4 程序状态寄存器访问指令

ARM微处理器支持程序状态寄存器访问指令，用于在程序状态寄存器和通用寄存器之间传送数据。

2.3.5 加载/存储指令

ARM微处理器支持加载/存储指令用于在寄存器和存储器之间传送数据，加载指令用于将存储器中的数据传送到寄存器，存储指令则完成相反的操作。

2.3.6 批量数据加载/存储指令

ARM微处理器所支持批量数据加载/存储指令可以一次在一片连续的存储器单元和多个寄存器之间传送数据，批量加载指令用于将一片连续的存储器中的数据传送到多个寄存器，批量数据存储指令则完成相反的操作。

2.3.7 数据交换指令

ARM微处理器所支持数据交换指令能在存储器和寄存器之间交换数据。

2.3.8 移位指令（操作）

ARM微处理器内嵌的桶型移位器（Barrel Shifter），支持数据的各种移位操作，移位操作在ARM指令集中不作为单独的指令使用，它只能作为指令格式中是一个字段，在汇编语言中表示为指令中的选项。

2.3.9 协处理器指令

ARM微处理器可支持多达16个协处理器，用于各种协处理操作，在程序执行的过程中，每个协处理器只执行针对自身的协处理指令，忽略ARM处理器和其他协处理器的指令。

2.3.10 异常产生指令

ARM微处理器所支持的异常指令有如下两条：

● SWI 软件中断指令

● BKPT 断点中断指令

2.4 Thumb指令及应用

2.5 本章小节

本章系统的介绍了ARM指令集中的基本指令，以及各指令的应用场合及方法，由基本指令还可以派生出一些新的指令，但使用方法与基本指令类似。与常见的如X86体系结构的汇编指令相比较，ARM指令系统无论是从指令集本身，还是从寻址方式上，都相对复杂一些。

**第3章 ARM11综述**

ARM 11系列微处理器是ARM公司近年推出的基于ARMv6架构的新一代RISC处理器，主要应用在消费类电子、无线设备、网络应用和汽车电子领域。ARM11处理器软件可以与以前所有ARM处理器兼容，井引人了用于媒体处理的32位SIMD、用于提高操作系统上下文切换性能的物理标记高速缓存、强制实施硬件安全性的TrustZone以及针对实时应用的紧密藕合内存。

3.1 ARM11简介

ARM11系列微处理器是ARM公司近年推出的新一代RISC处理器，它是ARM新指令架构——ARMv6的第一代设计实现。该系列主要有ARM1136J，ARM1156T2和ARM1176JZ三个内核型号，分别针对不同应用领域。

3.1.1 目标应用

ARMv6架构是根据下一代的消费类电子、无线设备、网络应用和汽车电子产品等需求而制定的。

3.1.2 ARM11处理器特点

对于各种无线移动应用，毫无节制的提供高性能处理器是无用的。同成本控制类似，功耗的控制也是一个重要因素。

3.1.3 ARM11处理器性能

ARM11处理器的超强性能是由一系列的架构特点所决定的。

3.1.4 ARM11处理器规格

将ARM1136, ARM1156, ARM1176、ARMI1MPCore分别从体系结构、Dhrystone性能、是否多核、ISA支持、内存管理和调试跟踪几方面参数进行总结。

3.2 ARM11系列处理器架构

ARM11系列包括了ARM11MPCore处理器、ARM1176处理器、ARM1156处理器、ARM1136处理器，它们是基于ARMv6架构，分别针对不同应用领域。

3.3 ARM11流水线

流水线是RISC执行指令时采用的一种重要机制。在流水线既要能达到更高的性能，还要让客户更加方便的实现流程。

3.3.1 流水线结构的性能

系统处理数据时，1个指令周期含有4~6个时钟脉冲，每个脉冲周期有不同的部件完成不同的操作。

3.3.2 流水线级数的影响

ARM7采用的是三级流水线，ARM9采用的是5级流水线，ARM10采用的是6级流水线，ARM11处理器的流水线和由8级组成，比以前ARM内核减少了40%的吞吐量。

3.3.3 ARM11处理器中流水线管理

为了解决这种延迟，避免流水线中的这种数据冲突，让前面的指令执行的结果能够快速进入到后面指令的流水线中，ARM11处理器采用了预测技术，存储管理，并行机制等技术来保持最佳的流水线效率。

3.4 ARM工作模式及寄存器组

3.4.1 ARM核工作模式

常规CPU工作核心模块都是由：寄存器组和ALU两大模块组成，ALU完成数据的加工处理，寄存器用来保存数据，这些数据是直接参与ALU运算。

3.4.2 ARM寄存器分组：

ARM芯片将寄存器分成很多个组，来运行不同模式下的程序，让CPU程序运行得更加稳健。

3.4.3 工作模式分析

3.5 各种模式工作机制

ARM工作模式切换的2种方式：

（1）程序手动切换，在某种特权模式下，可以切换到另外一个特权模式下，不过User用户模式没有这个能力，因为它不是特权模式。

（2）异常切换，异常事件发生，可从User模式下切换到异常模式下，或从另外一种异常模式下切换到该异常事件模式下。

3.5.1 CPSR、PC和SPSR\_xxx、LR\_xxx寄存器工作关系

3.5.2 R13\_xxx寄存器用途（sp\_xxx堆栈指针）

3.5.3 FIQ和IRQ特权模式（异常或叫中断模式）

3.5.4 Supervisor特权模式（异常）

3.5.5 Abort特权模式（异常）

3.5.6 Undefined特权模式（异常）

3.5.7 Secure Monitor特权（异常）模式

3.5.8 System特权模式

3.5.9 ARM中各个异常处理响应优先级

3.6 进入和退出异常中断的过程

3.6.1 ARM处理器对异常中断的相应过程

3.6.2 从异常中断处理程序种返回

3.7本章小结

本章对ARM微处理器的体系结构、寄存器的组织、处理器的工作状态、运行模式以及处理器异常等内容进行了描述，这此内容也是ARM休系结构的基本内容，是系统软、硬件设计的基础。

**第4章 GPIO接口**

GPIO即为通用输入/输出，每个GPIO接口可以根据需要设置为输人或输出I/O口。GPIO接口功耗低、封装小、成本低、应用广泛，可用来驱动LED、键盘、LCD等。

本章主要内容如下：

● 介绍GPIO接口的结构。

● 介绍GPIO接口的主要寄存器。

● 给出GPIO接口的操作实例，包括硬件设计和软件设计。

4.1 GPIO接口概述

GPIO接口是S3C6410最常用的接口，一共有17个端口。这17个端口为复用端口，在使用时可根据小同的功能，通过软件来控制相应端口的控制寄存器，以实现其特定的功能。

4.2 GPIO结构

GPIO有以下特性：

（1）可以控制127个外部中断

（2）有187个多功能输入/输出端口

（3）控制管脚的睡眠模式状态，除了GPK, GPL, GPM、和GPN管脚以外。

4.3 GPIO端口

S3C6410包含了187个多功能输入/输出端口管脚。表7-1列出了S3C6410的17个端口。

4.4 GPIO寄存器

4.4.1端口A控制寄存器

端口A控制寄存器包括五个控制寄存器，分别是GPACON、GPADAT、GPAPUD、GPACONSLP，GPAPUDSLP。

4.4.2端口B控制寄存器

端口B控制寄存器包括五个控制寄存器，分别是GPBCON、GPBDAT、GPBPUD、GPBCONSLP、GPBPUDSLP。

4.4.3端口C控制寄存器

端口C控制寄存器包括五个控制寄存器，分别是GPCCON、GPCDAT、GPCPUD、GPCCONSLP、GPCPUDSLP。

4.4.4端口D控制寄存器

端口D控制寄存器包括五个控制寄存器，分别是GPDCON、GPDDAT、GPDPUD、GPDCONSLP、GPDPUDSLP。

4.4.5端口E控制寄存器

端口E控制寄存器包括五个控制寄存器，分别是GPECON、GPEDAT、GPEPUD、GPECONSLP、GPEPUDSLP。

4.4.6端口F控制寄存器

端口F控制寄存器包括五个控制寄存器，分别是GPFCON、GPFDAT、GPFPUD、GPFCONSLP、GPFPUDSLP。

4.4.7端口G控制寄存器

端口G控制寄存器包括五个控制寄存器，分别是GPGCON、GPGDAT、GPGPUD、GPGCONSLP、GPGPUDSLP。

4.4.8端口H控制寄存器

端口H控制寄存器包括六个控制寄存器，分别是GPHCONO、GPHCONI、GPHDAT、GPHPUD、GPHCONSLP、GPHPUDSLP。

4.4.9端口I控制寄存器

端口I控制寄存器包括五个控制寄存器，分别是GPICON、GPIDAT、GPIPUD、GPICONSLP、GPIPUDSLP。

4.4.10端口J控制寄存器

端口J控制寄存器包括五个控制寄存器，分别是GPJCON、GPJDAT、GPJPUD、GPJCONSLP、GPJPUDSLP。

4.4.11端口K控制寄存器

端口K控制寄存器包括四个控制寄存器，分别是GPKCONO、GPKCONI、GPKDAT、GPKPUD。

4.4.12端口L控制寄存器

端口L控制寄存器包括四个控制寄存器，分别是GPLCONO、GPLCON1、GPLDAT、GPLPUD。

4.4.13端口M控制寄存器

端口M控制寄存器包括三个控制寄存器，分别是GPMCON、GPMDAT、GPMPUD。

4.4.14端口N控制寄存器

端口N控制寄存器包括三个控制寄存器，分别是GPNCON、GPNDAT、GPNPUD。GPNCON、GPNDAT、GPNPUD都是alive部分。

4.4.15端口O控制寄存器

端口O控制寄存器包括五个控制寄存器，分别是GPOCON、GPODAT、GPOPUD、GPOCONSLP、GPOPUDSLP。

4.4.16端口P控制寄存器

端口P控制寄存器包括五个控制寄存器，分别是GPPCON、GPPDAT、GPPPUD、GPPCONSLP、GPPPUDSLP。

4.4.17端口Q控制寄存器

端口Q控制寄存器包括五个控制寄存器，分别是GPQCON、GPQDAT、GPQPUD、GPQCONSLP、GPQPUDSLP。

4.5 GPIO应用举例

**第5章 IIC总线接口**

IIC总线是由PHILIPS公司开发的两线式串行总线，用于连接微处理器及其外围设备，实现全双工同步数据处理。S3C6410处理器支持一个多主控器IIC串行接口，在多主控制IIC总线模式下，多个S3C6410处理器能发送（或接收）串行数据到从属设备，主控器S3C6410控制IIC总线上的数据传输。

5.1 IIC总线接口概述

S3C6410处理器IIC接口为一个多主控器IIC串行接口。通过一个专用的串行数据线（SDA）和一个串行时钟线（SCL）实现在总线主控器和连接到IIC总线的外部设备之间传输数据。

5.2 IIC总线接口操作模式

S3C6410 IIC总线接口有四种操作模式：主控器发送模式、主控器接收模式、从属器发送模式、从属器接收模式。

5.2.1开始和停止条件

IIC总线接口无效时，它通常是在从属器模式下。换句话说就是，在SDA线检测一个开始条件前(当时钟信号SCL在高位时，SDA线发生高位到低位的跃变，开始条件启动)，接口必须在从属器模式下。

5.2.2 数据传输格式

在SDA线上的每一个字节长度必须是8位。起始条件后的第一个字节有一个地址域。当IIC主线在主控器模式下操作时，地址域能通过主控器被传输。每一个字节后面跟随一个ACK（acknow ledgement）位。

5.2.3 ACK信号传输

为了完成一个字节的发送操作，接收器必须将一个ACK位发送到发送器。ACK脉冲在SCL线的第九个时钟产生。

5.2.4 读写操作

在发送模式下，当发送数据时，IIC总线接口将一直等待，直到数据移位（IICDS）寄存器接收到一个新的数据。

5.2.5 异常中断条件

如果一个从属接收器不承认该从属地址，它将保持SDA线为高位。在这种情况下，主控器产生一个中断条件中断传输。

5.2.6 IIC总线配置

为了控制串行时钟的频率（SCL），在IICCON寄存器中，4位的预分频值被执行。IIC总线接口地址被存储在IIC总线地址（IICADD）寄存器。

5.2.7 每个模块的操作流程图

5.3 IIC总线接口特殊的寄存器

5.3.1 多主控器IIC总线控制（IICCON）寄存器

5.3.2 多主控器工IC总线控制/状态（IICSTAT）寄存器

5.3.3 多主控器IIC总线地址（IICADD）寄存器

5.3.4 多主控制器IIC总线发送/接收数据移位（IICDS）寄存器

5.3.5 多主控器IIC总线控制寄存器

5.4 IIC总线编程举例

**第6章 UART接口**

UART即通用异步收发器，它用于数据的串行传输。应用于嵌入式系统时，UART主要实现与PC的通信，或与外接EEPROM的数据交互。

本章主要内容如下：

● UART接口的结构。

● UART接口的主要操作。

● UART接口的主要寄存器。

● UART的具体操作实例，包括硬件设计和软件设计

6.1 UART接口概述

S3C6410的UART（通用异步接收和发送器）提供了四个独立的异步串行I/O（SIO）端口。

6.2 UART接口特性

UART的特性包括：

（1）基于rxd0，txd0，rxdl，txdl，rxd2，txd2，rxd3和txd3的DMA或中断来操作。

（2）UART通道0，1，2符合IrDA 1.0要求，且具有16字节的FIFO。

（3）UART通道0，1具有nRTS0，nCTS0，nRTS1和nCTSl。

（4）支持收发时握手模式。

6.3 UART的操作

本章介绍UART的操作，包括数据传输，数据接收，中断产生，波特率产生，环回模式，红外线模式和自动流量控制。

6.3.1 数据发送

数据帧发送是可编程的。

6.3.2 数据接收

和数据发送一样，数据帧接收也是可编程的。

6.3.3 自动流量控制（AFC）

该S3C6410的UART0和UART1通过nRTS和nCTS信号支持自动流量控制。

6.3.4 接收FIFO的操作

6.3.5 发送FIFO的操作

6.3.6 RS-232C接口

要将UART连接到调制解调器接口（而不是零调制解调器）需要nRTS，nCTS，nDSR，nDTR，DCD和nRI信号。在这种情况下，可以通过软件来控制这些信号与一般的I/O端口。

6.3.7 中断/DMA请求的产生

每个S3C6410的UART有七个状态（发射/接收/错误）信号：溢出错误，奇偶错误，帧错误，中断，接收缓冲区数据就绪，传输缓冲区为空，发送移位寄存器为空。其状态信号靠相应的UART的状态寄存器（UTRSTATn/UERSTATn）来指示。

6.3.8 UART错误状态FIFO

除了Rx FIFO寄存器之外，UART还具有一个错误状态FIFO。错误状态FIFO中表示了在FIFO寄存器中，哪一个数据在接收时出错。错误中断发生在有错误的数据被读取时。为清除错误状态FIFO，寄存器URXHn和UERSTATn会被读取。

6.3.9 红外线（IR）模式

S3C6410的UART模块支持红外线（IR）发送和接收，可以通过设置UART控制寄存器（ULCONn）中的红外模式位来选择这一模式。

6.4外部接口

6.5寄存器描述

6.5.1 UART行控制寄存器

6.5.2 UART控制寄存器

6.5.3 UART的FIFO控制寄存器

6.5.4 UART Modem控制寄存器

6.5.5 UART接收（Rx）/（Tx）发送状态寄存器

6.5.6 UART错误状态寄存器

6.5.7 UART的FIFO状态寄存器

6.5.8 UART Modem状态寄存器

6.5.9 UART发送缓冲寄存器（保存寄存器和FIFO寄存器）

6.5.10 UART接收缓冲寄存器(保存寄存器和FIFO寄存器)

6.5.11 UART波特率分频寄存器

6.5.12 波特率错误容限

6.5.13 UART中断处理寄存器

6.5.14 UART中断源处理寄存器

6.5.15 UART中断屏蔽寄存器

6.6 UART接口应用举例

**第7章ADC接口**

ADC的作用是将模拟信号转变成数字信号。A/D转换通常要经过采样、保持、量化及编码4个过程。触摸屏的本质是传感器，基本原理是用手指或其他物体触摸安装在显示器前端的触摸屏，所触摸的位置（以坐标形式）由触摸屏控制器检测，并通过接口送到CPU，从而确定输入的信息。触摸屏的主要结构由触摸检测组件和触摸控制器组成。

本章主要内容如下：

● ADC与触摸屏接口的基本原理。

● ADC与触摸屏接口的主要功能。

● ADC与触摸屏接口的主要寄存器。

● ADC的具体操作实例，包括硬件设计和软件设计。

7.1 ADC及触摸屏概述

触摸屏接口控制触摸屏的位置和方位（XP，XM，YP，YM），为X坐标转换和Y坐标转换选择触摸屏的位置和方位。触摸屏界面包含了位置和方位控制逻辑、ADC界面逻辑和中断发生逻辑。

7.2 ADC及触摸屏的特性

7.3 ADC及触摸屏界面操作

7.4功能描述

7.4.1 A/D转换时间

7.4.2 触摸屏接口方式

7.4.3 待机模式

当ADCCON [2」被设定为‘1’时，待机模式被激活。在此模式下，A/D转换操作停止，并且ADCDAT0，ADCDAT1寄存器包含先前转换的数据。

7.4.4 编程记录

7.5 ADC和触摸屏界面的特殊寄存器

7.5.1 ADC的控制寄存器（ADCCON）

7.5.2 ADC的触摸屏控制寄存器（ADCTSC）

7.5.3 在X/Y坐标转换下触摸屏的PIN的条件

7.5.4 ADC开始延迟寄存器（ADCDLY）

7.5.5 ADC的数据转换寄存器（ADCDAT0）

7.5.6 ADC的数据转换寄存器（ADCDAT1）

7.5.7 ADC的触摸屏UP-DOWN寄存器（ADCUPDN）

7.5.8 ADC触摸屏中断清除寄存器

7.6 ADC应用举例

**第8章 Android系统开发概述**

随着智能手机的发展，越来越多的手机操作系统进入了人们的视野。由Google及其手机联盟推出的基于Linux平台的开源手机操作系统Android凭借着华丽的图形界面及友好的开发环境，吸引了全球范围内的电信行业、手机制造商加入到了Android阵营中，逐渐形成了与iOS平分天下的趋势。本章将详细介绍Android的系统架构、内核以及Android系统移植的概念和驱动开发的方法。

8.1 Android系统架构

Android将系统架构分为了4层，从上到下依次为：应用程序层（Application）、应用程序框架层（Application Framework）、系统运行库层（Libraries）以及Linux内核层（Linux Kernel）。

8.1.1 应用程序层（Application）

应用程序层包含一个核心应用集合，如SMS短消息程序、联系人管理、E-mail客户端、浏览器、地图、同时，开发人员利用Java语言设计和编写的程序（如音乐播放器、游戏等）也都运行在这一层上。

8.1.2 应用程序框架层（Application Framework）

应用程序框架层是Android应用开发的基础，开发人员可以使用Google发布的API框架方便地进行程序开发，简化了程序开发的架构设计，但必须遵守框架的开发原则。

8.1.3 系统运行库层（Libraries）

当使用Android应用框架时，Android系统会通过一些C/C++库来支持我们使用的各个组件，使其能更好地位我们服务。

8.1.4 Linux内核层

Android以Linux内核为基础，借助Linux内核提高核心系统服务，例如安全性、内存管理、进程管理、网络协议栈和驱动模型等等方面。

8.2 Android系统内核

Android采用Linux内核，与Linux内核的一些特性有着重要的关系。

8.2.1 Linux内核结构

Linux内核主要由五个子系统组成：进程调度，内存管理，虚拟文件系统，网络接口，进程间通信。

8.2.2 Android内核和驱动

Android在Linux内核的基础上进行了增强，增加了一些面向移动计算的特有功能。例如，低内存管理器LMK（Low Memory Killer）、匿名共享内存（Ashmem），以及轻量级的进程间通信Binder进制等。

8.3 系统移植的概念和驱动开发的方法

嵌入式操作系统与桌面操作系统以及服务器操作系统最显著的区别之一就是它的可移植性。对嵌入式操作系统而言，它往往需要运行在不同体系结构的处理器和开发板上。而移植的目的就是为了改动较小的内容，支撑较为庞大上层的系统。

**第9章 开发环境**

9.1开发环境和工具

9.1.1 安装虚拟机和Ubuntu12.04

9.1.2 JDK的安装

9.1.3 安装Android编译需要软件包

2.2 系统源码获取和编译

9.2.1 安装Git和Repo·

Git是Linus Torvalds 为了帮助管理Linux 内核开发而开发的一个开源的分布式版本控制系统，用以敏捷高效的处理任何或大或小的项目。

9.2.2 获取Android源码

9.2.3 获取Android Linux Kernel源码

9.3 辅助开发工具介绍

9.3.1 Minicom的安装和使用

minicom 是个串口通信程序，有点象共享软件 TELIX，但其源码可以自由获得，并能够运行于多数Unix系统。

9.3.2 Partitionmanager安装和使用

9.3.3 Vim使用简介

9.3.4 其他工具介绍

**第10章 Linux设备驱动开发简介**

设备驱动程序在Linux中扮演着比较重要的作用。用户可以通过一组标准化的程序接口调用这些不同功能的驱动程序。而将这些调用映射到实际设备的操作上的任务则是设备驱动程序来实现。在编写驱动程序时，应该要注意我们编写的硬件的内核代码不要给用户强加任何特定策略，因为不同的用户有不同的需求。驱动程序应该处理如何使硬件可用，而怎样使用的问题应该留给上层应用程序，这样的驱动程序就比较灵活。

10.1 Linux系统简介

10.1.1 Linux系统特点

Linux是一类Unix操作系统的统称。最初是由Linus Torvalds带头开发，并由全世界众多爱好者共同维护的操作系统。

10.1.2 文件系统介绍

文件系统是磁盘上存储文件的方法和数据结构，是操作系统组织、存取和保存信息的重要手段，每种操作系统都有自己的文件系统。

10.1.3 Linux 2.6内核的特点

Linux 2.6内核是目前应用得最广泛的内核，相较于Linux 2.4，Linux 2.6有了相当大的改进。

10.2 交叉编译工具选项说明

源文件需要经过编译才能生成可执行文件。在Windows下进行开发时，只需要单击几个按钮即可编译，集成开发环境（比如Visual studio）已经将各种编译工具的使用封装好了。

10.2.1 arm-linux-gcc选项

一个C/C++文件要经过预处理（preprocessing）、编译（compilation）、汇编（assembly）和连接（linking）等4步才能变成可执行文件。

10.2.2 arm-linux-objcopy选项

arm-linux-objcopy被用来复制一个目标文件的内容到另一个目标文件中，可以使用不同于源文件的格式来自输出目的文件，即可以进行格式转换。

10.2.3 arm-linux-objdump选项

arm-linux-objdump用于显示二进制文件信息，常用来查看反汇编代码。

10.3 Makefile介绍

make命令执行时，需要一个 makefile 文件，以告诉make命令如何去编译和链接程序。

10.3.1 Makefile的规则

10.3.2 Makefile示例

10.3.3 make是如何工作的

10.3.4 Makefile中使用变量

10.3.5 让make自动推导

10.3.6 清空目标文件的规则

10.3.7 Makefile里的内容

Makefile里主要包含了五个东西：显式规则、隐晦规则、变量定义、文件指示和注释。

10.3.8 Makefile的文件名

10.3.9 引用其它的Makefile

10.3.10 环境变量MAKEFILES

10.3.11 make的工作方式

**第11章 HAL层介绍**

HAL（Hardware Abstraction Layer）层又被称为硬件抽象层，是建立在Linux驱动上的一套程序库，主要作用是对Linux内核驱动程序的封装，向上提供接口，屏蔽底层的实现细节。HAL层运行在用户空间，属于Linux内核层之上的应用层。Google这样设计的原因，主要是为了保护各硬件厂商的利益。也正是因为如此，Android被踢出了Linux内核主线代码树。

11.1 初识HAL层

Linux内核采用了GPL协议，这要求Linux内核在发布时必须公布所有源代码。如果把对硬件的支持逻辑放在Linux驱动层，则在公布源代码的时候把硬件的相关参数和实现也同时公开了，这设计到了技术专利和商业秘密，会对硬件厂商的利益造成很大的损害。正是为了保护这些厂商的利益，Google才决定将对硬件的支持分为了硬件抽象层和内核驱动层。

11.2 HAL层的实现原理

11.2.1 HAL层的架构

本节主要介绍早期的Android和新版本的HAL层的架构及对比。

11.2.2 HAL module分析

本节主要介绍在HAL module中的三个重要数据结构。

11.3 HAL层实例

本小结将通过一个实例来帮助大家更好地理解支持HAL的Linux驱动程序。

11.3.1 编写硬件驱动程序

本节将为一个虚拟的字符设备编写内核驱动程序。

11.3.2 增加硬件抽象层（HAL）模块

本节实现virtualcdev设备驱动的HAL模块。

**第12章 Input输入子系统**

Linux系统提供了input子系统，按键、触摸屏、键盘、鼠标等输入都可以通过input接口函数来实现设备驱动。本章将详细讲解Android输入系统驱动的基本知识，介绍这些源代码的实现原理。

12.1 用户输入系统介绍

12.1.1 Android输入系统的结构

Android输入系统的结构比较简单，从下到上包含了驱动程序、本地库处理部分、Java类对输入事件的处理、对Java的接口等

12.1.2 移植工作

本节介绍了在移植Android输入系统时，需要完成的两个工作。

12.2 Input输入驱动

Input输入驱入程序是Linux输入设备的驱动程序，可以进一步分成游戏杆（joystick）、鼠标（mouse和mice）和事件设备三种驱动程序。

12.2.1 文件input.h

本节介绍了在手机系统中使用的键盘（keyboard）和小键盘（kaypad），它们属于按键设备EV\_KEY，轨迹球属于相对设备EV\_REL，触摸屏属于绝对设备EV\_ABS。

12.2.2 文件KeycodeLabels.h

本节介绍了文件KeycodeLabels.h，它的功能是存储从字符串到整数的映射关系。

12.2.3 文件KeyCharacterMap.h

本节介绍文件frameworks/base/include/ui/KeyCharacterMap.h也是本地框架层libui的头文件，在其中定义了按键的字符映射关系

12.2.4 kl格式文件

本节介绍Android默认提供的按键布局文件，主要包括qwerty.kl和AVRCP.kl。

12.2.5 kcm格式文件

本文介绍kcm格式文件，它是按键字符映射文件，用于表示按键字符的映射关系，功能是将整数类型按键码（keycode）转化成可以显示的字符

12.2.6 文件EventHub.cpp

本节介绍文件EventHub.cpp，它位于libhui库中的frameworks/base/libs/ui目录下，此文件是输入系统的核心控制文件，整个输入系统的主要功能都是在此文件中实现的。

12.3 模拟器中的实现

本节介绍GoldFish虚拟处理器键盘输入部分的驱动程序，即event驱动程序的核心。

**第13章LCD驱动开发**

本章介绍LCD驱动开发。主要内容包括了对LCD基础介绍、LCD驱动介绍、FrameBuffer驱动的HAL层分析等内容

13.1 LCD基础介绍

13.1.1 LCD的类型

13.1.2 LCD的技术参数

13.1.3 工作原理

本小节介绍LCD的类型、技术参数以及工作原理等。

13.2 LCD驱动介绍

13.1.2 LCD的技术参数

13.2.2 FrameBuffer驱动的移植

13.3 FrameBuffer驱动的HAL层分析

13.3.1 Gralloc库

3.3.2 Gralloc与上层的关系

13.3.3 调用Gralloc HAL库

本节主要介绍FrameBuffer驱动的HAL层

**第14章 Audio子系统**

本章介绍LCD驱动开发。主要内容包括了音频驱动基础、音频系统的层次、音频系统的硬件抽象层等内容。

14.1音频驱动基础

14.1.1 数字音频简介

14.1.2 ALSA架构简介

14.1.3 音频驱动的目录结构

14.1.4 音频设备硬件接口

本小节介绍Linux音频驱动的基本架构，以及Linux内核与音频驱动相关的目录结构。

14.2 音频系统的层次

14.2.1 层次说明

14.2.2 Media库中的Audio框架

本节介绍从上到下分别是由Java的Audio类、Audio本地框架类、AudioFlinger和Audio的硬件抽象层几个部分组成的Audio系统

14.3 音频系统的硬件抽象层

14.3.1 Audio硬件抽象层的接口

14.3.2 实现Audio硬件抽象层

14.3.3 Audio策略管理的内容

14.3.4 上层的情况和注意事项

**第15章 WiFi系统**

WiFi（Wireless Fidelity）是一种可以将个人电脑、手持设备（如PDA、手机）等终端以无线方式互相连接的技术。WiFi是一个无线网络通信技术的品牌，为WiFi联盟所持有，目的是改善基于IEEE 802.11标准的无线网络产品之间的互通性。

15.1 WiFi系统的结构

WiFi使用IEEE的802.11协议的技术，目前在智能手机中使用WiFi已经成为智能手机的核心功能之一。无线局域网的底层硬件通常是WiFi芯片，这个芯片通常提供了集成化的WiFi功能。

15.2 移植和调试

15.2.1 协议和驱动程序

在Linux中，WiFi的部分包括协议和驱动程序两个部分的内容，对用户空间提供的是网络驱动程序。

15.2.2 本地实现

本地实现部分主要包括wpa\_supplicant及wpa\_supplicant适配层。WPA是WiFi Protected Access的缩写，中文含义为“WiFi网络安全存取”。WPA是一种基于标准的可互操作的WLAN安全性增强解决方案，可大大增强现有及未来WiFi系统的数据保护和访问控制水平。

15.2.3 JNI层

Android中的WiFi系统的JNI部分实现的源代码如下。

frameworks/base/core/jni/android\_net\_wifi\_Wifi.cpp

JNI层的接口注册到Java层的源代码如下。

frameworks/base/wifi/java/android/net/wifi/WifiNative.java

15.2.4 Java FrameWork层

15.3 配置WiFi的流程

下面将简要介绍实现并配置WiFi系统的基本流程。

（1） 配置Android支持WiFi

（2） 使wpa\_supplicant能够调试信息

（3） 配置wpa\_supplicant.conf

（4） 配置路径和权限

（5） 运行wpa\_supplicant和dhcpcd

（6） 编译WiFi驱动为module或kernel built in

（7） WiFi需要的firmware

（8） 修改WiFi驱动适合Android

（9） 设置dhcpcd.conf

**第16章 GPS子系统**

GPS定位系统（Global Positioning System）是一个由覆盖全球的24颗卫星组成的卫星系统。这个系统可以实现导航、定位、授时等功能，通过此技术可以引导飞机、船舶、车辆以及个人，安全、准确地沿着选定的路线，准时到达目的地，在当前市面上的很多智能手机中，都配备了GPS定位系统的功能。本章将简要讲解Android平台中GPS系统驱动的移植知识。

16.1 GPS系统简介

16.1.1 GPS系统的发展

GPS的前身是美国军方研制的一种子午仪卫星定位系统(Transit)，1958年研制，1964年正式投入使用

16.1.2 定位原理

GPS导航系统的基本原理是测量出已知位置的卫星到用户接收机之间的距离，然后综合多颗卫星的数据就可知道接收机的具体位置。

16.1.3 GPS系统的组成部分

（1） 空间部分

（2） 地面控制系统

（3） 用户设备部分

16.2 GPS系统的结构

Android在GPS方面有着比较系统的架构，让各种定位设备可以方便地集成进来，也让基于定位的应用开发变得更加容易。Android平台中GPS系统自下而上包含了驱动程序、硬件抽象层、Java框架类等几个部分。

16.3 移植和调试

16.3.1 移植的内容

在Android平台中，需要移植的GPS内容位于JNI层下面，即驱动层和硬件抽象层。GPS硬件的驱动程序通常是串口驱动程序，比较容易实现，在Linux系统的用户空间中通常用ttf设备来表示。

16.3.2 驱动程序

GPS的硬件接口相对简单，除开供电，reset等控制，一般仅通过串口和应用处理器连接。因此驱动程序通常是标准的串口驱动

16.3.3 硬件抽象层

GPS系统的硬件抽象层在如下头文件中定义。

hardware\libhardware\_legacy\include\hardware\_legacy\gps.h

16.3.4 上层应用

**第17章 传感器系统**

随着物联网概念的推出和发展，传感器这一名词逐渐为开发人员所熟知。其实对于广大读者，传感器也是经常可以见到以及用到的，比如楼宇里的楼梯灯、马路上的路灯等。本章将详细讲解Android系统传感器系统的基本知识和移植方法，为学习本书后面的知识打下基础。

17.1 传感器系统的结构

Android的传感器系统用于获取外部的信息，在传感器系统下层的硬件是各种传感器设备。

17.2 移植Sensor驱动

Android传感器系统自传感器硬件抽象层接口以下的部分是非标准的，因此传感器系统移植包括传感器的驱动程序和硬件抽象层。

17.2.1 移植驱动程序

从Linux操作系统的角度，Sensor的驱动程序没有公认的标准定义。

17.2.2 移植硬件抽象层

Sensor传感器系统HAL层的实现文件目录是“hardware/libhardware/include/hardware/”。我们先看看其中的Android.mk文件，其代码如下所示。

17.2.3 实现上层部分

传感器部分的上层包括了以下内容.

17.3 实现传感器

Android为模拟器提供了一个Sensor硬件抽象层的示例实现，它本身具有实际的功能，可以作为实际系统的传感器硬件抽象层的示例。

**四、课程内容与学时分配**

课程内容与学时分配表

|  |  |
| --- | --- |
| 内容 | 学时 |
| 1. 第1章 ARM微处理器概述 | 2 |
| 2. 第2章 ARM微处理器的指令系统 | 6 |
| 3. 第3章 ARM11综述 | 4 |
| 4. 第4章 GPIO接口 | 4 |
| 5. 第5章 IIC总线接口 | 2 |
| 6. 第6章 UART接口 | 2 |
| 7. 第7章 ADC接口 | 2 |
| 8. 第8章 Android系统开发概述 | 4 |
| 9. 第9章 开发环境 | 4 |
| 10. 第10章 Linux设备驱动开发简介 | 4 |
| 11. 第11章 HAL层介绍 | 4 |
| 12. 第12章 Input输入子系统 | 4 |
| 13. 第13章LCD驱动开发 | 2 |
| 14. 第14章 Audio子系统 | 2 |
| 15. 第15章 WiFi系统 | 2 |
| 16. 第16章 GPS子系统 | 2 |
| 17. 第17章 传感器系统 | 2 |
| 复习与机动 | 2 |
| 总学时 | 54 |

**五、教材与参考书**

参考书：

1. (美)刘易斯著，陈文智，胡威译，嵌入式软件设计基础(基于ARM Cortex-M3原书第2版)，北京：机械工业出版社，2015.
2. 陈文智，王总辉，胡威，嵌入式软件开发，北京：高等教育出版社，2015.
3. 吴志祥，柯鹏，张智，胡威，Android应用开发案例教程，武汉：华中科技大学出版社，2014

**六、教学方式和考核方式**

1、教学方式：以课堂讲授为主，辅以实验教学、课后作业；

2、考核方式：闭卷笔试

大纲制订人：

大纲审定人：

制订 日期：2016-3-2