<<专业嵌入式软件开发>>

图书基本信息

书名：<<专业嵌入式软件开发>>

13位ISBN编号：9787121147838

10位ISBN编号：7121147831

出版时间：2012-1

出版时间：电子工业出版社

作者：李云

页数：640

<<专业嵌入式软件开发>>

内容概要

本书分为6篇。
硬件篇就嵌入式软件开发所需掌握的处理器概念进行了介绍。
工具篇对make、gcc编译器、bintuils工具集、ld链接器和gdb调试器进行了讲解，其中对make这一嵌入式开发环境的全能管家进行了精辟的介绍，致力于帮助读者成为Makefile方面的专家。
编程语言篇致力于让读者更深入地理解C编程语言。
操作系统篇通过循序渐进的方式介绍ClearRTOS的设计与实现，使得读者能透彻地理解操作系统的关键概念和实现原理。
设计篇和质量保证篇通过实践的方式逐步展开讲解，以帮助读者获得一些实用的设计原则、最佳实践和一套有效的质量保证方法论。

本书适合嵌入式软件开发领域的新手和在工作中碰到瓶颈的老手阅读。
阅读本书要求读者已掌握C编程语言和基本的UML知识。

作者简介

李云，现任诺基亚西门子网络技术有限公司软件架构师。

2年电气和电子开发工作经验，自1999年至今从事软件开发工作曾在浙江大立科技有限公司、UT斯达康杭州研发中心、摩托罗拉杭州研发中心担任软件工程师、项目负责人和软件架构师等职早在2000年开始从事嵌入式软件开发工作，内容主要包括：操作系统内核、文件系统和IP协议栈的移植，IDE硬盘、以太网芯片和PCI总线等外设的驱动开发熟悉x86、PowerPC、XScale处理器，以及TIDSP源码级熟悉uC/OS、RTEMS和VxWorks实时操作系统的实现，并在这些操作系统和Linux操作系统上有着丰富的软件开发经验。

<<专业嵌入式软件开发>>

书籍目录

硬件篇
第1章 处理器的基本概念
1.1 区分微处理器与微控制器
1.2 寄存器
1.3 处理器是如何启动的
1.4 输入与输出
1.5 指令与数据
1.6 中断
1.7 字节序
1.8 边界对齐
1.9 程序断点和数据断点
1.10 内存管理单元
1.11 缓存
1.12 小结
第2章 开发活动中的硬件问题
2.1 两个案例
2.2 案例的背后——信号完整性
2.3 应对方法
2.4 小结
工具篇
第3章 make，开发环境全能管家
3.1 从最简单的Makefile中了解规则
3.2 创建基本的编译环境
3.2.1 将规则运用于程序编译
3.2.2 让Makefile更专业
3.3 提高编译环境的实用性
3.3.1 让编译环境更加有序
3.3.2 提升依赖关系管理
3.4 打造更专业的编译环境
3.4.1 规划项目目录结构
3.4.2 增进复用性
3.4.3 支持头文件目录的指定
3.4.4 实现库链接
3.4.5 增强可使用性
3.4.6 管理对库的依赖关系
3.4.7 改善编译效率
3.4.8 恰当地书写注释
3.5 理解make的解析行为
3.6 Makefile的调试
3.7 make的常用选项
3.8 活用make
3.9 小结
第4章 gcc，C语言编译器
4.1 什么是交叉编译器
4.2 gcc幕后工作揭示

4.3 实用的gcc选项
4.3.1 解决宏错误的好帮手
4.3.2 辅助编写汇编程序的好方法
4.3.3 获取系统头文件路径
4.3.4 产生映射文件
4.3.5 通过选项定义宏
4.3.6 生成依赖关系
4.3.7 指定链接库
第5章 binutils工具集，软件开发利器
5.1 addr2line，指令地址翻译器
5.2 ar，静态库生成器
5.3 nm，符号显示器
5.4 objdump，信息查看器
5.5 objcopy，段剪辑器
5.6 ranlib，库索引生成器
5.7 size，段大小观察器
5.8 strings，字符串窥视器
5.9 strip，程序文件瘦身器
第6章 ld，链接器
6.1 重定位的概念
6.2 链接脚本
6.2.1 段
6.2.2 符号
6.2.3 存储区域
6.2.4 常用命令
6.3 常用选项
6.3.1 指定程序的入口点
6.3.2 生成可重定位的中间文件
6.3.3 指定链接脚本
练习与思考
第7章 gdb，程序调试助手
7.1 启动和退出gdb
7.2 获取帮助
7.3 调试程序
7.3.1 断点设置
7.3.2 控制程序运行
7.3.3 检查程序
7.3.4 提高调试效率
7.4 查看符号表
7.5 控制gdb的行为
编程语言篇
第8章 掌握必要的汇编知识
8.1 as的语法
8.1.1 宏
8.1.2 汇编命令
8.1.3 符号和标签
8.1.4 汇编指令

8.2 嵌入汇编的语法
第9章 深入理解程序的结构
9.1 段
9.1.1 指令段
9.1.2 数据段
9.2 栈
9.3 堆
9.4 小结
第10章 ABI/EABI规范，缔造程序兼容合约
10.1 定义基本数据类型
10.2 规范字节对齐处理
10.3 分配寄存器的功能
10.4 规定栈帧结构
10.4.1 栈帧的含义和作用
10.4.2 函数参数的传递方法
10.4.3 函数返回值的返回方法
10.5 小结
练习与思考
第11章 混淆指针与数组所导致的问题
11.1 问题示例
11.2 问题分析
11.2.1 数组的内存模型
11.2.2 指针的内存模型
11.3 问题成因
11.4 预防措施
11.5 小结
第12章 volatile，让我保持原样
设计篇
第13章 设计，软件质量之本
13.1 软件设计是什么
13.2 软件质量的概念
13.3 阻碍改善设计的常见观念
13.3.1 测试是替罪羊或救命稻草
13.3.2 资源永远不足
13.3.3 不改变就可以规避风险
13.4 如何提高设计能力
13.5 设计模式、设计原则和设计思想
13.6 放之四海皆适用的设计原则
13.6.1 以人为本
13.6.2 追求简单性
13.6.3 让模块善始善终
13.6.4 重视收集统计信息
13.6.5 借助命名传达设计意图
13.6.6 消除“审美告警”
13.6.7 通过机制解决问题
13.6.8 防止他人犯错
13.6.9 考虑可查错性

13.7 小结
第14章 模块管理，保障系统有序运行
14.1 管理参照系
14.2 设计思路
14.3 程序实现
14.3.1 引入模块标识
14.3.2 实现层与级的表达
14.3.3 系统状态和回调函数原型定义
14.3.4 模块注册
14.3.5 系统启动
14.3.6 系统关闭
14.4 module示例程序
14.5 模块管理的一些思考
14.6 小结
练习与思考
第15章 错误管理，不可或缺的用户需求
15.1 表达错误的通用方法
15.1.1 错误码格式
15.1.2 定义方法
15.1.3 使用示例
15.1.4 提高可使用性
15.1.5 定义和使用错误码的准则
15.2 优化错误日志的输出
15.2.1 传统方法
15.2.2 更有效的方法
15.3 平台和框架层的错误处理
15.4 小结
第16章 目录结构管理，使项目进展更顺利
16.1 规划目录结构的意义
16.1.1 书架功能
16.1.2 意识引导
16.1.3 加速新手上手
16.2 出色目录结构的特点
16.3 一个示例
16.4 小结
第17章 平台与框架开发，高质量软件打造之路
17.1 区分系统库、平台和框架
17.1.1 系统库
17.1.2 平台
17.1.3 框架
17.2 本质和优点
17.3 确立架构模型
17.4 小结
第18章 可开发性设计，一种高效且经济的开发模式
18.1 可开发性问题一瞥
18.2 可开发性设计的内涵
18.3 引入设备抽象层

18.4 更复杂的设备抽象层
18.5 图形界面的可开发性设计
18.5.1 增强设备抽象层
18.5.2 提供可视化编辑环境
18.6 其他可开发性设计
18.7 小结
操作系统篇
第19章 引导加载器，系统启航者
19.1 功能
19.2 文件存储布局
19.3 程序加载原理
19.4 优点
19.5 小结
练习与思考
第20章 任务，软件基本调度单元
20.1 任务情景
20.1.1 情景内容
20.1.2 情景保存
20.1.3 情景恢复
20.1.4 情景切换
20.2 任务调度
20.2.1 调度算法
20.2.2 调度器
20.3 任务的生命周期
20.4 任务控制
20.4.1 任务创建
20.4.2 任务启动
20.4.3 任务删除
20.4.4 任务挂起
20.4.5 任务恢复
20.4.6 任务睡眠
20.5 竞争问题与中断控制
20.5.1 竞争问题的产生
20.5.2 通过中断控制解决竞争问题
20.5.3 中断控制的嵌套问题
20.6 任务与中断状态
20.7 任务栈溢出检测
20.8 滴答与空闲任务
20.9 多任务环境控制
20.10 任务模块管理
20.11 taskv1示例程序
20.12 任务钩子函数
20.13 任务变量
20.13.1 taskv2示例程序
20.13.2 原理
20.13.3 实现
20.14 其他概念与思考

20.14.1 抢占式任务与实时系统的关系
20.14.2 影响任务切换效率的因素
20.14.3 避免直接删除任务
20.14.4 小心多任务设计被滥用
20.15 小结
练习与思考
第21章 任务同步与通信，实现协同工作
21.1 信号量
21.1.1 应用场合
21.1.2 程序实现
21.1.3 semaphore示例程序
21.2 互斥锁
21.2.1 应用场合
21.2.2 程序实现
21.2.3 mutex示例程序
21.2.4 优先级反转与继承
21.2.5 递归锁
21.3 事件
21.3.1 应用场合
21.3.2 程序实现
21.3.3 event示例程序
21.4 消息队列
21.4.1 应用场合
21.4.2 程序实现
21.4.3 实现消息队列
21.4.4 queue示例程序
21.4.5 使用指南
21.5 死锁及预防
21.6 小结
练习与思考
第22章 内存管理，协调动态内存的使用
22.1 堆管理
22.1.1 heapv1示例程序
22.1.2 程序实现
22.1.3 设计改进
22.1.4 支持内存泄漏检测
22.1.5 实现内存溢出检测
22.1.6 内存碎片问题
22.2 内存池管理
22.2.1 mpool示例程序
22.2.2 程序实现
22.2.3 缓冲区泄漏检测
22.3 小结
练习与思考
第23章 设备管理，方便与外设交互
23.1 字符设备管理
23.2 中断管理

23.2.1 中断向量表
23.2.2 中断控制
23.2.3 中断状态管理
23.2.4 设备与中断
23.2.5 模块管理
23.3 实现设备管理
23.3.1 安装驱动程序
23.3.2 注册设备
23.3.3 打开设备
23.3.4 关闭设备
23.3.5 设备读写与控制
23.4 设备驱动程序实现
23.4.1 “滴答”设备
23.4.2 控制台设备
23.4.3 终止程序运行设备
23.5 驱动安装与设备注册
23.6 小结
练习与思考
第24章 定时器，程序闹钟
24.1 软件定时器分类
24.2 设计思路
24.3 中断回调定时器
24.3.1 程序实现
24.3.2 timerv1示例程序
24.4 定时误差
24.5 提高遍历效率
24.6 改善实时性
24.6.1 实时性分析
24.6.2 改进实时性
24.7 任务回调定时器
24.7.1 程序实现
24.7.2 timerv3示例程序
24.8 小结
练习与思考
第25章 ClearRTOS“实时”操作系统
25.1 设计原则
25.2 源程序目录管理
25.3 让Makefile体现概念
25.4 实现集中配置
25.5 改进与移植
质量保证篇
第26章 质量保证导言
26.1 软件开发的特点
26.1.1 脑力密集型工作
26.1.2 实现不具唯一性
26.1.3 隐性成本高
26.1.4 忽视的细节很容易被放大

26.1.5 质量难以评估
26.2 保证质量的关键要素
26.2.1 完备的需求分析
26.2.2 高质量的设计
26.2.3 编程好习惯
26.2.4 充分的验证
26.2.5 必要的流程
26.2.6 合适的工具
26.2.7 言简意赅的文档
26.3 质量保证需要系统性的方法论
26.3.1 方法论=流程+工具
26.3.2 构建有效方法论的核心手段
26.4 走出质量困境的指导性思想
26.4.1 从管理者的角度
26.4.2 从工程师的角度
26.4.3 从组织的角度
26.5 小结
第27章 编程好习惯，质量保证的基本条件
27.1 终生受用的编程好习惯
27.1.1 判断失败而非成功
27.1.2 采用sizeof减少内存操作失误
27.1.3 屏蔽编程语言特性
27.1.4 恰当使用gto语句
27.1.5 合理运用数组
27.1.6 以逆序方式释放资源
27.1.7 在模块对外接口中防范错误
27.1.8 避免出现魔数
27.1.9 利用编程语言特性提高效率
27.1.10 复用代码提高维护性
27.1.11 借助隐式初始化简化程序逻辑
27.1.12 青睐小粒度锁
27.1.13 精确包含头文件
27.1.14 让模块的对外头文件保持简洁
27.1.15 只暴露必要的变量和函数
27.1.16 清除编译器报告的所有警告
27.2 小结
第28章 单元测试，被忽视的质量保证方法
28.1 警惕单元测试无用论
28.2 一个简单但不完善的单元测试例子
28.3 构建单元测试框架
28.4 无缝整合单元测试
28.4.1 维护规则
28.4.2 目录规划
28.4.3 更改Makefile
28.4.4 检查整合效果
28.5 几个实施问题
28.6 桩函数和打桩

28.7 错误注入，一种可测试性设计
28.8 平台开发与单元测试
28.9 被测行为的确定性
28.10 测试用例的有效性
28.11 小结
第29章 代码覆盖，单元测试效果的衡量指标
29.1 了解代码覆盖工具
29.2 无缝整合代码覆盖
29.2.1 更改Makefile
29.2.2 检查整合效果
29.3 三个代码覆盖程度指标
29.4 小结
第30章 静态分析，防止将失误带给用户
30.1 认识静态分析工具
30.2 无缝整合静态分析
30.2.1 更改Makefile
30.2.2 检查整合效果
30.3 小结
第31章 动态分析，使程序更健壮
31.1 结识动态分析工具
31.2 无缝整合动态分析
31.2.1 更改Makefile
31.2.2 检查整合效果
31.3 小结
第32章 性能分析，让优化程序有的放矢
32.1 初探性能分析工具
32.2 无缝整合性能分析
32.2.1 更改Makefile
32.2.2 检查整合效果
32.3 小结
第33章 qBench，一个开发高质软件的工作台
参考资料

章节摘录

版权页：插图：make是以从上到下的顺序读入Makefile中的内容的。
然而，处理Makefile中的语句却并非完全从上到下。
大体上，make处理一个Makefile分两个阶段。
第一个阶段包含：（1）make读入Makefile，以及Makefile中所包含的其他Makefile。
（2）make分析并获得变量名、变量值、隐式规则和显式规则。
（3）构建所有目标的关系树，以及它们的先决条件。
在第二个阶段，make基于第一个阶段所建立的内部结构分析哪些目标需要重新构建，以及需要执行哪些规则的命令来构建这些目标。
理解make处理Makefile的两个阶段对于熟练地编写Makefile非常重要。
就作者的学习经验来看，也曾经因为不了解这两个阶段而产生过不少困惑。
make在处理Makefile中的语句时，存在立即展开和延迟展开两种类别。
立即展开是指语句在第一个处理阶段就被展开。
变量和函数就是立即展开的。
延迟展开则是指当make处理它时并不立即展开，其展开动作发生在第二个阶段。
对于不同的语句，make将采用不同的展开策略。
图3.155示例说明了与赋值相关的展开策略。
注意，左边的变量名总是立即展开的，而右边的变量值却未必。
其中“ += ”的左边有可能采用立即展开也有可能采用延迟展开。
当左边的变量名在使用“ +_ ”之前如已被设置为简单扩展变量（即采用“ ：= ”赋值）时，则采用立即展开的方式，否则采用延迟展开的方式。

<<专业嵌入式软件开发>>

编辑推荐

《专业嵌入式软件开发:全面走向高质高效编程》是一本全面讲解实时操作系统实现原理的书为了使读者获得最佳的学习效果，作者为《专业嵌入式软件开发:全面走向高质高效编程》量身打造了可在Windows和Linux操作系统上直接运行的Clear RTOS“实时”操作系统，并在书中详解了所有实现细节。

这是一本介绍嵌入式软件开发所需掌握工具的书作者从实用的角度介绍了gcc编译器、binutils工具集、Id链接器和gdb调试器，并花了较大的篇幅帮助读者成为Makefile的专家。

这是一本带领读者实践如何构建高质高效软件开发方法的书 书中通过展示如何将单元测试框架、静态分析、动态分析和性能分析整合到开发环境中这种方式，阐述了作者的“以单元测试为中心”和“无缝整合”思想。

这是一位饱尝自学嵌入式软件开发痛苦的工程师在软件行业积累了12年后，与读者分享心得的一本书中就软件设计、编程习惯和质量保证等内容与读者进行了交流。

<<专业嵌入式软件开发>>

版权说明