[toc]

实时操作系统

嵌入式软件设计的演变

顺序程序设计

- 顺序调用任务，执行完一个任务后再执行下一个任务
- ==若任务长时间使用cpu，其他任务无法响应==
- 改进

> 从延时处分段，分成小片段后插入另一个任务中

基于状态机的程序设计

- **占用资源少，执行效率高，容易测试**

> 用于资源较少的嵌入式系统

- 任务分散，流程不直观
- 每个任务可用==延时和状态==两个变量来表示

基于简易任务调度器的程序设计

- 任务切换与任务调度

> - 与顺序执行不同，在执行每个任务后，任务释放cpu，调度器分派下一个任务接管cpu
> - 与状态机不同，状态机及主程序是任务调用者，是主动调用者，任务片段是受调用者，而任务调度器中，任务调度是被调用者

- 任务调度的核心：**堆栈迁移**
- 调度器种类
  - 合作式
  - 抢占式
  - 混合式

基于操作系统的程序设计

- 实时与分时系统
- 基本概念
  - 时钟节拍
    - 任务切换和始终调度：2%~5%

- 实时中的重要概念
  - 系统响应时间
  - 任务切换时间
  - 中断延迟
- 代码的临界段
  > - 不可分割的代码称为临界段代码
  > - 临界段代码影响中断响应时间(需要关中断)
- 资源
  > - 任何为任务所占用的尸体
  > - **共享资源**
  >   - 被一个以上任务使用的资源
- 任务
  > - 任务状态
  >   - 休眠态
  >   - 就绪态
  >   - 运行态
  >   - 挂起态
  >   - 被中断态
  > - 任务切换
  >   - 任务切换过曾增加了应用程序的额外负荷
- 内核
  > - 任务调度
  > - 任务通信
  > - 对cpu的占用时间一般在2-5%之间

嵌入式软件设计

嵌入式软件架构与层次

- 硬件层
- 驱动层
- 系统层
- 应用层

代码优化

- 变量优化技巧
  - 统一清零
  - 定义时赋值
- 算法优化

    - 查表代替运算
    - 求余运算
    - 平方运算
    - 移位代替乘除法
    - 公共子表达式

- 内嵌汇编
    - 解决时间效率问题
    - 8-2原则
        > 20%的程序消耗80%的时间，最主要考虑改进20%的代码
    - 循环优化

代码可靠性

代码实现的安全性

guide

- 读研或者工作
    - 从会做 -> **快**速做
        - 快才有机会
        - 积累是基础