在嵌入式软件设计过程中查找缺陷的技巧 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/252/2021_2022__E5_9C_A8_E5_B5_8C_E5_85_A5_E5_c103_252969.htm 大部分软件开发项目依靠结合代码检查、结构测试和功能测试来识别软件缺陷。尽管这些传统技术非常重要，而且能发现大多数软件问题，但它们无法检查出当今复杂系统中的许多共性错误。本文将介绍如何避免那些隐蔽然而常见的错误，并介绍的几个技巧帮助工程师发现软件中隐藏的错误。 结构测试或白盒测试能有效地发现代码中的逻辑、控制流、计算和数据错误。这项测试要求对软件的内部工作能够一览无遗(因此称为"白盒"或"玻璃盒")，以便了解软件结构的详细情况。它检查每个条件表达式、数学操作、输入和输出。由于需要测试的细节众多，结构测试每次检查一个软件单元，通常为一个函数或类。 代码审查也使用与实现缺陷和潜在问题查找同样复杂的技术。与白盒测试一样，审查通常针对软件的各个单元进行，因为一个有效的审查过程要求的是集中而详尽的检查。 与审查和白盒测试不同，功能测试或黑盒测试假设对软件的实现一无所知，它测试由受控输入所驱动的输出。功能测试由测试人员或开发人员所编写的测试过程组成，它们规定了一组特定程序输入对应的预期程序输出。测试运行之后，测试人员将实际输出与预期输出进行比较，查找问题。黑盒测试可以有效地找出未能实现的需求、接口问题、性能问题和程序最常用功能中的错误。 虽然将这些技术结合起来可以找出隐藏在一个特定软件程序中的大部分错误，但它们也有局限。代码审查和白盒测试每次只针对一小部分代码，忽视了系统

的其它部分。黑盒测试通常将系统作为一个整体来处理，忽视了实现的细节。一些重要的问题只有在集中考察它们在整个系统内相互作用时的细节才能被发现；传统的方法无法可靠地找出这些问题。必须整体地检查软件系统，查找具体问题的特定原因。由于详尽彻底地分析程序中的每个细节和它与代码中所有其它部分之间的相互作用通常是不大可能的，因此分析应该针对程序中已经知道可能导致问题的特定方面。本文将探讨其中三个潜在的问题领域：* 堆栈溢出 * 竞争条件 * 死锁 读者可在网上阅读本文的第二部分，它将探讨下列问题：* 时序问题 * 可重入条件 在采用多任务实时设计技术的系统中，以上所有问题都相当普遍。 堆栈溢出 处理器使用堆栈来存储临时变量、向被调函数传递参数、保存线程“状态”，等等。如果系统不使用虚拟内存(换句话说，它不能将内存页面转移到磁盘上以释放内存空间供其它用途)，堆栈将固定为产品出厂时的大小。如果由于某种原因堆栈越出了编程人员所分配的数量范围，程序将变得不确定。这种不稳定可能导致系统发生严重故障。因此，确保系统在最坏情况下能够分配到足够的堆栈至关重要。 确保永不发生堆栈溢出的唯一途径就是分析代码，确定程序在各种可能情况下的最大堆栈用量，然后检查是否分配了足够的堆栈。测试不大可能触发特定的瞬时输入组合进而导致系统出现最坏情况。 堆栈深度分析的概念比较简单： 1. 为每个独立的线程建立一棵调用树。 2. 确定调用树中每个函数的堆栈用量。 3. 检查每棵调用树，确定从树根到外部“树叶”的哪条调用路径需要使用的堆栈最多。 4. 将每个独立线程调用树的最大堆栈用量相加。 5. 确定每个中断优先级内各中断服务程序(ISR)的最大堆

栈用量并计算其总和。但是，如果ISR本身没有堆栈而使用被中断线程的堆栈，则应将ISR使用的最大堆栈数加到各线程堆栈之上。 6. 对于每个优先级，加上中断发生时用来保存处理器状态的堆栈数。 7.如果使用RTOS，则加上RTOS自身内部用途需要的最大堆栈数(与应用代码引发的系统调用不同，后者已包含在步骤2中)。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com