资深工程师潜心之作，内容兼顾原理和动手实践。
引领读者从原理、设计、实现、应用4个层面仔细体验RTOS的精妙之处。
全书图文并茂，配合STM32硬件平台和源代码，深度践行口袋实验室理念。

华章科技

单片机与嵌入式

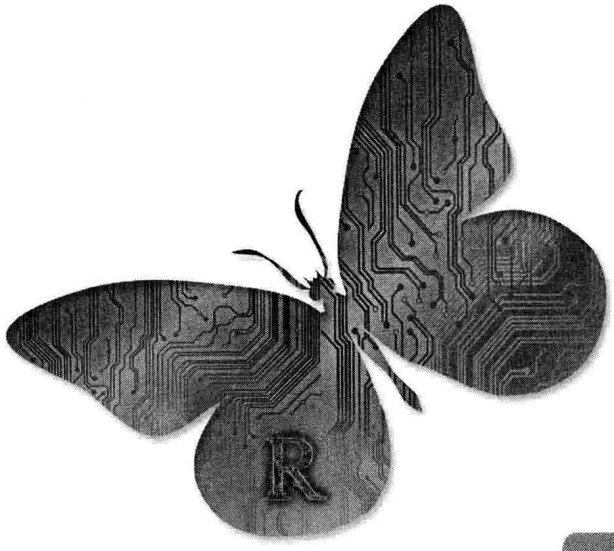
RTOS

嵌入式实时操作系统原理与最佳实践

刘旭明 著

机械工业出版社
China Machine Press

单片机与嵌入式



RTOS

嵌入式实时操作系统原理与最佳实践

刘旭明 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

嵌入式实时操作系统原理与最佳实践 / 刘旭明著 . —北京：机械工业出版社，2014.8
（电子与嵌入式系统设计丛书）

ISBN 978-7-111-47607-8

I. 嵌… II. 刘… III. 实时操作系统 IV. TP316.2

中国版本图书馆 CIP 数据核字（2014）第 182909 号

本书系统地介绍了嵌入式操作系统内核的原理、设计和实现。首先通过大量图表详细介绍了嵌入式操作系统的基本概念和原理。然后通过对内核各个功能的分析、设计和实现来加深读者对相关知识的理解。最后通过实际的应用程序来演示如何使用这些功能。全书从原理、设计、实现和使用的角度来向读者展示嵌入式操作系统。

本书可作为广大从事嵌入式系统工作的工程师以及其他相关技术人员的参考资料，也可作为相关专业本科生的辅助参考书。

嵌入式实时操作系统原理与最佳实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：陈佳媛　　责任校对：董纪丽

印　刷：蓟城市京瑞印刷有限公司　　版　次：2014 年 9 月第 1 版第 1 次印刷

开　本：186mm × 240mm　1/16　　印　张：22

书　号：ISBN 978-7-111-47607-8　　定　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

随着计算机和电子、通信技术的发展，嵌入式系统在人们的生活中变得越来越重要，相关技术发展得越来越快。特别是近几年，各类电子设备得到极快的发展。很多嵌入式系统的开发离不开其核心基础软件，即嵌入式操作系统。本书主要介绍嵌入式操作系统内核的概念与设计实现。

在内容编排上，本书首先对嵌入式操作系统内核各个模块的概念进行讲解，对可选的设计方案进行比较分析，然后采用一个具体的方案作为目标来设计实现，通过流程图、图表、示例代码等详细演示如何实现该机制。最后，还会通过实际应用程序演示这些功能的使用方法。本书从原理、设计、实现和应用各个角度完整展示嵌入式操作系统内核的相关功能。

全书主要内容如下：

第 1 章介绍了嵌入式多任务系统的基本知识。图文并茂地阐述了相关的重要概念。这部分是理解后面章节的基础，初学者需要仔细理解。

第 2 章详细介绍了与任务相关的概念、设计和实现。任务是实时操作系统的重要概念，本章做了十分详细的分析。

第 3 章详细介绍了实现 IPC 机制的基础代码。本章是第 4 ～ 8 章的基础。全书介绍了用于任务间、任务和中断处理函数间的最基本的几个同步和通信机制。

第 4 章详细论述了信号量的设计和实现。

第 5 章详细论述了互斥量的设计和实现。

第 6 章详细论述了邮箱的设计和实现。

第 7 章详细论述了消息队列的设计和实现。

第 8 章详细论述了事件标记的设计和实现。

第 9 章详细介绍了定时器的机制和设计实现。

第 10 章分析了内核的移植代码，着重介绍了在意法半导体的 STM32 处理器上如何移植 Trochili RTOS。读者需要对基于 ARM Cortex-M3 的处理器有一定了解。

第 11 章重点介绍了基于 Trochili RTOS 的以太网协议应用。通过实际案例演示 Trochili

RTOS 的使用。

作者从事嵌入式工作多年，参与过多款嵌入式处理器的功能验证和固件开发工作，经常接触 RTOS，对其有浓厚兴趣。工作之余尝试编写一些任务调度代码，逐渐实现了一套较完整的 RTOS 内核。这个兴趣爱好加深了作者对嵌入式系统的理解，使得个人能力得到了提高，对本职工作也有很大帮助。目前 Trochili RTOS 并不完美，现有很多功能还需优化，作者会逐步完善 RTOS 的各项功能。

由于作者时间和水平有限，书中难免存在错误及不妥之处，敬请广大读者批评指正，如有问题，可通过 Trochili RTOS 官方网站 www.trochili.com 或者微博 www.weibo.com/trochili 和作者联系。

本书在编写和出版过程中得到了机械工业出版社华章公司编辑们的热情帮助和大力支持，在此一并表示感谢。

感谢北京航空航天大学何小庆教授对本书的指导，何老师在全书内容选取和章节安排上给了很好的建议，并对全书做了审读检查。感谢我的朋友王文东，他对全书内容进行了详细的阅读并给出了很多改进意见。

本书可作为广大从事嵌入式系统开发工作的工程师以及其他相关技术人员的参考资料，也可作为相关专业本科生的辅助参考书。

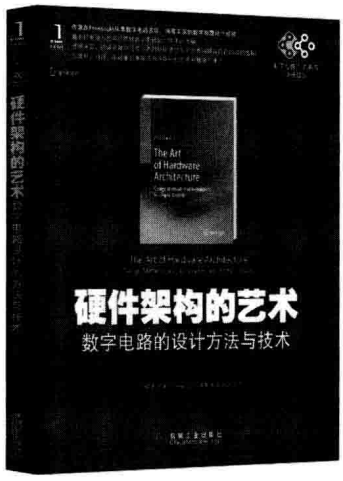
刘旭明

2014 年 4 月于北京

推荐阅读

FPGA快速系统原型设计权威指南

作者：R.C. Cofer 等 ISBN：978-7-111-44851-8 定价：69.00元



硬件架构的艺术：数字电路的设计方法与技术

作者：Mohit Arora ISBN：978-7-111-44939-3 定价：59.00元

ARM快速嵌入式系统原型设计：基于开源硬件mbed

作者：Rob Toulson 等 ISBN：978-7-111-46019-0 定价：69.00元

嵌入式软件开发精解

作者：Colin Walls ISBN：978-7-111-44952-2 定价：79.00元

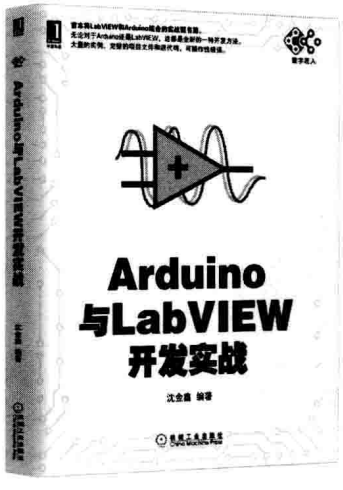
推荐阅读

Arduino高级开发权威指南（原书第2版）

作者：Steven F. Barrett ISBN：978-7-111-45246-1 定价：59.00元

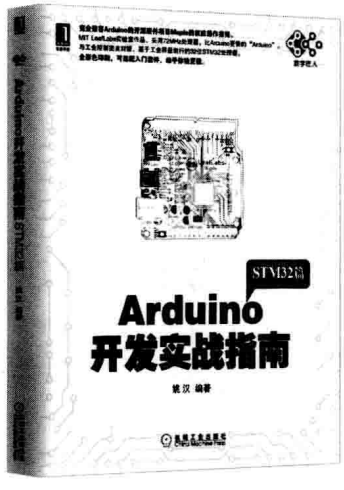
例说XBee无线模块开发

作者：Jonathan A. Titus ISBN：978-7-111-45681-0 定价：59.00元



Arduino与LabVIEW开发实战

作者：沈金鑫 ISBN：978-7-111-45839-5 定价：59.00元



Arduino开发实战指南：STM32篇

作者：姚汉 ISBN：978-7-111-44582-1 定价：59.00元

目　录

第 1 章
嵌入式操作系统基础

本章主要介绍多任务嵌入式操作系统相关的概念和整体结构，并对全书涉及的重要知识做介绍，为我们在以后各章的嵌入式操作系统的内核分析和学习做好准备。本章的内容并没有强调、区分嵌入式操作系统和通用操作系统的概念。在这两者之间，很多机制是相通的。另外本章假设读者已经对嵌入式系统有初步了解，不再介绍例如发展历史、机制特点这些基本知识。

1.1 嵌入式软件系统结构

目前常见的嵌入式软件结构可以分为轮询系统、前后台系统和多任务系统。这些方案是根据应用的具体需求提出的，各有各自的特点和适用的领域。

1.1.1 轮询系统

这是最简单的一种软件结构，主程序是一段无限循环的代码，在循环中顺序查询各个条件，如果满足就执行相应的操作。这种方案的好处是实现简单，逻辑清晰，便于开发人员掌握。但是每个事件的查询和处理时间是不能确定的。假如前面的操作时间较长，那么后面的操作必然会被延迟。

在图 1-1 中，假如步骤 1 操作需要很久，那么步骤 2 必然得不到及时处理，如果步骤 2 的工作很重要或者很紧急，那么系统的性能和响应能力就很差了。

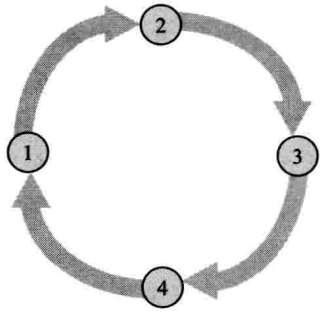


图 1-1 轮询系统结构

1.1.2 前后台系统

相对轮询系统，前后台系统对外部事件的处理做了优化。前后台系统是由中断驱动的。主程序依然是一段无限循环的代码，称为后台程序，而事件的响应则由中断来完成，称为前台程序。在后台程序执行时，如果有外部事件发生，则前台的中断程序会打断后台程序。在完成必要的事件响应之后，前台中断程序退出并通知后台程序来继续操作。由后台程序完成

事件的后继处理，比如数据的分析等操作。从代码功能上讲，事件的响应和处理分为了两个部分。因为中断自身有优先级和嵌套的功能，所以优先级高的事件能够得到及时响应。但后台程序仍然需要按顺序处理各个事件的后继事务。

前后台系统演示如图 1-2 所示：

如图 1-2 所示，在中断源之间有优先级的概念。ISR 会首先响应事件，简单的事件可以在 ISR 中直接处理，复杂的情况下则记录下必要数据和状态标记，等所有中断处理结束后，将由后台主函数按顺序处理各个事件。也就是说，事件的响应是支持优先级的，但事件的最终处理却是顺序的。使用中断来代替轮询方案中事件的查询操作，所以相对轮询方案，前后台系统对事件的响应能力有较大改善。

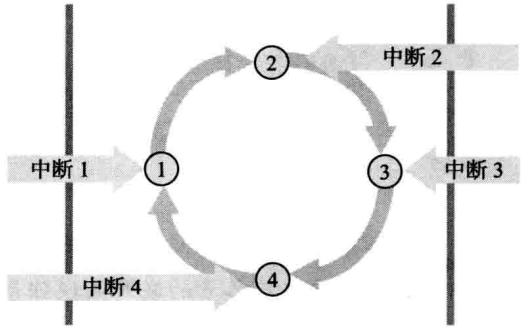


图 1-2 前后台系统结构

1.1.3 多任务系统

和前后台系统相比，多任务系统在响应事件的时候，同样是由多个中断处理程序完成的。但是对于事件的后继操作则是由多个任务来处理的。也就是说每个任务处理它所负责的事件。在基于优先级的多任务系统中，因为任务间优先级的关系，优先级高的任务可得到优先处理。这样优先级高的事件就能及时得到处理；在基于分时机制的多任务系统中，任务间则按比例轮流占用处理器。

多任务机制如图 1-3 所示。

在图 1-3 中，中断用来响应事件，事件的后续操作则由任务来完成。中断和任务都有优先级。假如其中中断 2 和任务 2 处理的事件是紧急的或者重要的，那么当中断 2 发生时，即使其他任务或者中断正在处理，也会被抢占，最终任务 2 会优先得到运行机会。

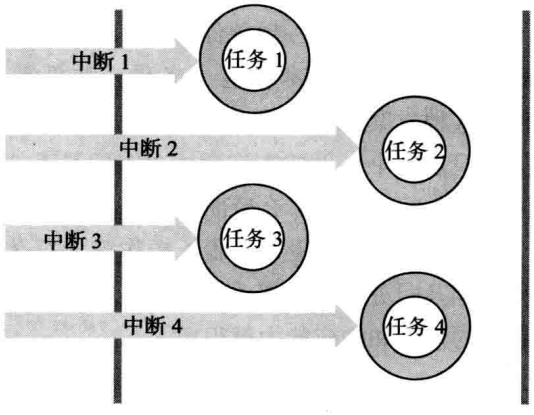


图 1-3 多任务系统结构

因为多任务操作系统允许将具体的应用系统分成若干个相对独立的任务来管理，所以多任务操作系统的使用可以简化应用程序的设计，系统也变得简洁且便于维护和扩展。对实时性要求严格的事件都能得到及时可靠的处理。不过多任务操作系统自身将占用部分处理器、存储器等硬件资源，这是引入多任务机制的必要代价。

从事件和数据处理的角度考虑，可以把整个应用流程简化为事件响应和事件处理两个阶段。从这两个阶段采用的不同技术手段出发，可以清晰合理地分析上面介绍的这三种软件结

构方案，可以看到解决问题的思路越来越清晰，结构和层次越来越合理。

表 1-1 是对三种软件结构的比较。

表 1-1 常见嵌入式软件模型

模型	事件响应	事件处理	特点
轮询系统	主程序	主程序	轮询响应事件，轮询处理事件
前后台系统	前台多个中断程序	后台单个主程序	实时响应事件，轮询处理事件
多任务系统	多个中断程序	多个任务	实时响应事件，实时处理事件

通过上面的比较，我们可以清楚地看到嵌入式软件结构上的不同和发展，但这并不是系统结构好坏的标准。每种方案都有它产生的年代、硬件资源的发展阶段和所适合的应用领域。

多任务系统是基于多任务操作系统的应用开发模型。本书介绍的就是嵌入式操作系统的核心部分：嵌入式操作系统内核的设计和实现。它的主要功能包括：任务管理、任务调度、任务同步、互斥和通信、设备管理、中断管理、时间管理等。而像图形用户接口、文件系统、TCP/IP 协议、嵌入式数据库引擎等，则可以归为嵌入式操作系统内核层之外的功能模块。多任务模型下 RTOS 组成如图 1-4 所示。

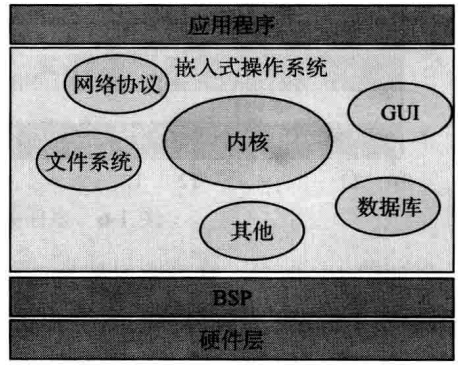


图 1-4 多任务模型下 RTOS 组成

关于嵌入式操作系统，有很多常见的技术概念，熟悉这些概念是我们学习嵌入式操作系统的基础。本书后续章节着重分析、设计和实现一个“嵌入式实时操作系统内核”，有时会使用“内核”这个简称。在内容的编排上，会把各种功能模块的概念放在各章起始，首先介绍其原理，然后分析设计和实现。

1.2 多任务机制概述

在前面我们介绍了多任务系统是如何演化的。和前后台系统相比较，多任务可以理解为有多个后台程序的前后台系统。每个任务都专注于自己处理的问题。下面将详细介绍一下和多任务系统相关的一些基本概念。

1.2.1 时钟节拍

时钟节拍是多任务系统的基础，它指明了把处理器时间以多大的频率分割成固定长度的时间片段。作为多任务系统运行的时间尺度，时钟节拍是通过特定的硬件定时器产生的。硬件定时器会产生周期的中断，在相应的中断处理函数中，内核代码得以运行，从而进行任务调度和定时器时间处理等内核工作。

处理器的时钟节拍如图 1-5 所示。

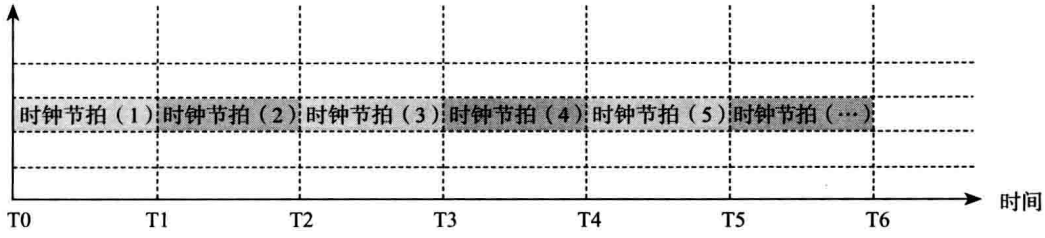


图 1-5 处理器时钟节拍

硬件定时器中断的时间间隔取决于不同的内核设计，一般是毫秒级的。时钟节拍越快，内核函数介入系统运行的几率就越大，时钟节拍中断响应次数越多，内核占用的处理器时间越长。相反，如果时钟节拍太慢，则导致任务的切换间隔时间过长，进而影响到系统对事件的响应效果。

图 1-6 演示了多任务系统中，中断处理程序和任务在时间上的关系。

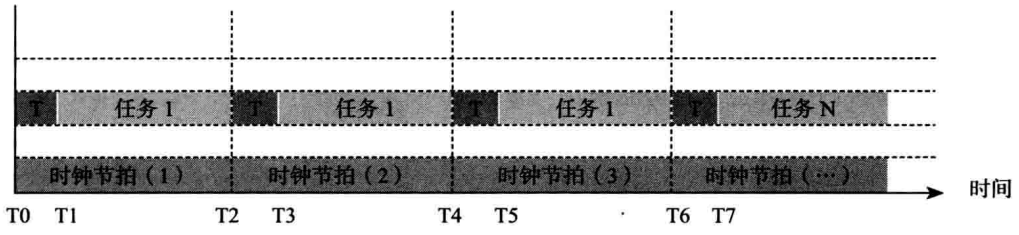


图 1-6 多任务系统中的中断和任务的时间关系

如图 1-6 所示，硬件定时器按照固定的时间间隔产生中断，然后在时钟节拍中断 ISR 中（图中以 T 标记）处理内核的工作。T0 ～ T1 这段时间是内核占用的时间（时钟节拍处理程序），T1 ～ T2 这段时间是任务占用的时间。而 T0 ～ T2 则是一次时钟节拍的全部时间。从图 1-6 中可以看出，任务 1 的本轮执行占用了 3 个时钟节拍。

1.2.2 多任务机制

在单处理器的计算机系统中，在某一具体时刻处理器只能运行一个任务，但是可以通过将处理器运行时间分成小的时间段，多个任务按照一定的原则分享这些时间段的方法，轮流加载执行各个任务，从而从宏观上看，有多个任务在处理器上同时执行，这就是单处理器系统上的多任务机制的原理，如图 1-7 所示。

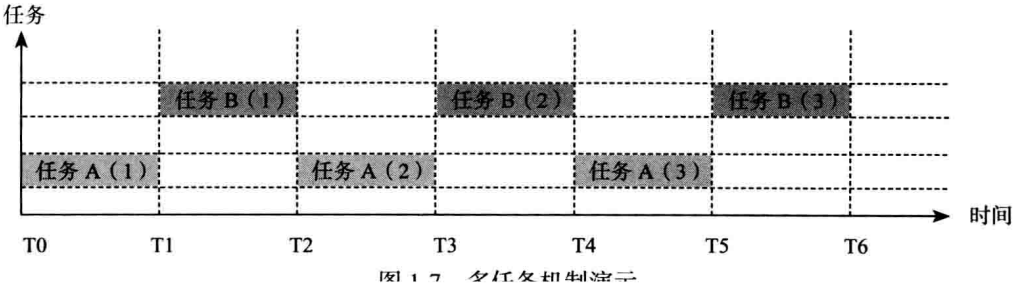


图 1-7 多任务机制演示

在图 1-7 中，任务 A 和任务 B 按照等长时间轮流占用处理器，在单处理器上造成多个任务同时运行的假象。

另外，因为不同任务的运行路径不同，在某一时刻有些任务可能需要等待一些资源，这时可以通过某种方案，使当前任务让出处理器，从而避免因为任务等待资源而长期占有处理器而使其他任务无法运行。这样多任务机制可以使处理器的利用率得到提高，并提高了系统的处理能力。

在多任务操作系统内核中必须提供解决并发任务的机制。通用操作系统一般以“进程”、“线程”等为单位来管理用户任务。在相关资料中，也会明确指出“进程”与“线程”的区别。但在很多嵌入式操作系统中，并没有区分进程和线程，只是把整个操作系统当作一个大的运行实体，其中运行着很多任务。任务通常作为调度的基本单位。

1.2.3 任务上下文

任务可以看作是用户程序在处理器等硬件上的运行，是一个动态的概念。任务在处理器上运行的某一时刻，有它自己的状态，即处理器所有的寄存器的数据，这个叫作任务的上下文，可以理解为是处理器的“寄存器数据快照”。通过这些数据，操作系统可以随时打断任务的运行或者加载新的任务，从而实现不同任务的切换运行。

任务上下文是跟处理器密切相关的概念，不同的处理器有不同的处理器上下文定义。比如在 Cortex-M3 处理器中的寄存器如下所述：

- 拥有 R0 ～ R15 寄存器组。
- R0 ～ R12 是通用寄存器。
- R13 作为堆栈指针（设置有两个，但在同一时刻只有一个指针起作用。）
- R14 为连接寄存器。
- R15 为程序计数器，指向当前的程序地址。

另外还有特殊功能寄存器：

- 程序状态寄存器组（PSR）
- 中断屏蔽寄存器组（PRIMASK、FAULTMASK、BASEPRI）
- 控制寄存器（CONTROL）

在 RTOS 设计任务上下文时经常会把大部分硬件寄存器作为任务上下文的内容，这点在介绍操作系统移植时会做详细介绍。

1.2.4 任务切换

任务切换又叫作任务上下文切换。当操作系统需要运行其他的任务时，操作系统首先会保存和当前任务相关的寄存器的内容到当前任务的栈中，然后从将要被加载的任务的栈中取出之前保存的全部寄存器的内容并加载到相关的寄存器中，从而继续运行被加载的任务，这个过程叫作任务切换。

任务基本的切换过程如图 1-8 至图 1-11 所示。

假设系统中有两个任务 A、B。当前处理器正在运行 A 任务。此时任务 A 的栈顶在变量 An 处：

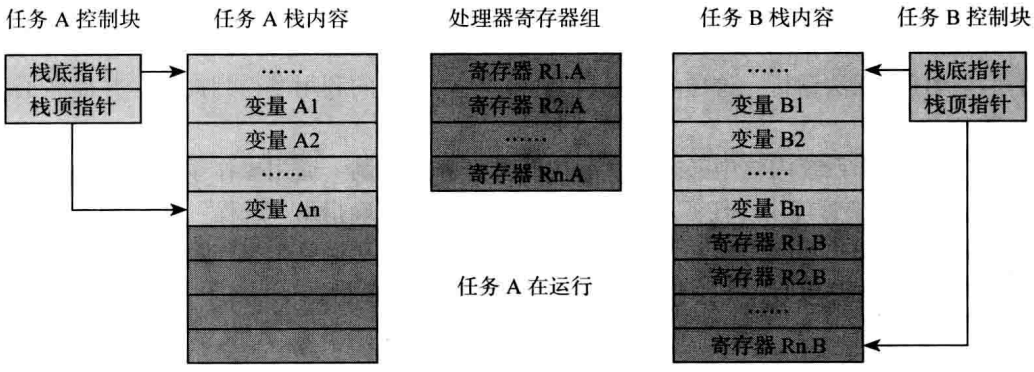


图 1-8 任务 A 运行时的上下文情况

然后发生任务调度，需要首先保存当前的处理器寄存器组的内容到任务 A 的栈中：

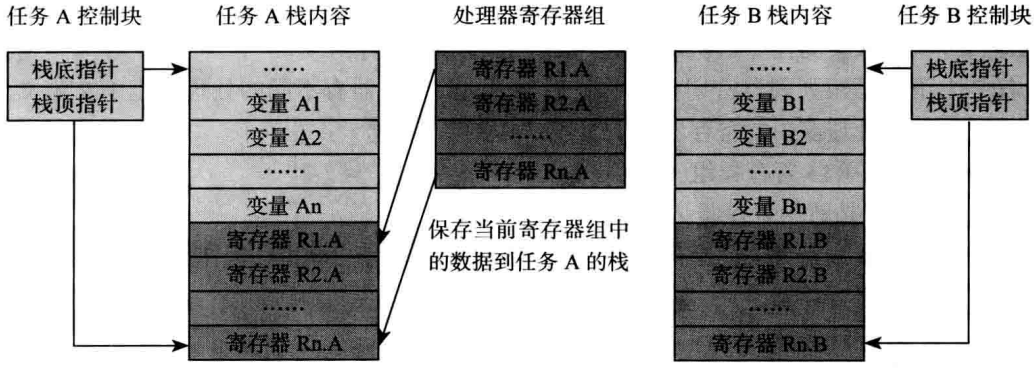


图 1-9 任务 A 上下文保存

接下来的操作就是把保存在任务 B 栈中的处理器寄存器组的内容调入到处理器寄存器组中：

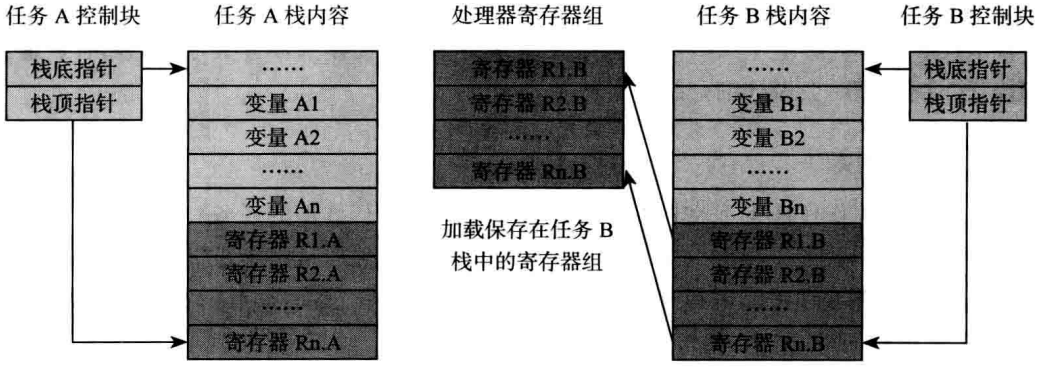


图 1-10 任务 B 上下文恢复