嵌入式

实时操作系统μC/OS-III

应用技术

——基于ARM Cortex-M3 LPC1788

张 勇 夏家莉 陈 滨 蔡 鹏 编著

源程序下载地址：
http://www.buaapress.com.cn的“下载中心”

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

嵌入式实时操作系统 μC/OS－III 应用技术
——基于 ARM Cortex－M3 LPC1788

张　勇　夏家莉　陈　滨　蔡　鹏　编著

北京航空航天大学出版社

内 容 简 介

本书基于 μC/OS - III 和 IAR - LPC1788 实验板讲述基于嵌入式实时操作系统进行面向任务应用程序设计的方法，阐述了 μC/OS - III 系统组件的应用技巧和开发应用程序的工作流程。全书共 14 章，包括嵌入式实时操作系统 μC/OS - III 概述，Cortex - M3 内核体系，IAR KSK LPC1788 开发板与 LPC1788 微控制器，IAR EWARM 软件和应用程序框架，μC/OS - III 移植，μC/OS - III 用户任务，μC/OS - III 系统任务，信号量、任务信号量和互斥信号量、消息队列和任务消息队列、事件标志组、多事件请求、存储管理、LCD 显示原理与面向任务程序设计实例以及 Keil MDK 程序设计方法。书中给出了 23 个完整实例，对学习嵌入式操作系统应用程序设计具有较强的指导作用，读者可在北京航空航天大学出版社网站下载源代码。

本书可作为电子通信、软件工程、自动控制、智能仪器和物联网相关专业高年级本科生或研究生学习嵌入式操作系统及其应用技术的教材，也可作为嵌入式系统开发和研究人员的参考用书。

图书在版编目(CIP)数据

嵌入式实时操作系统 μC/OS - III 应用技术 ：基于 ARM Cortex - M3 LPC1788 / 张勇等编著. --北京 ：北京航空航天大学出版社，2013.4

ISBN 978 - 7 - 5124 - 1098 - 5

Ⅰ. ①嵌… Ⅱ. ①张… Ⅲ. ①实时操作系统 Ⅳ. ①TP316.2

中国版本图书馆 CIP 数据核字(2013)第 065046 号

嵌入式实时操作系统 μC/OS - III 应用技术
——基于 ARM Cortex - M3 LPC1788

张 勇 夏家莉 陈 滨 蔡 鹏 编著

责任编辑 刘晓明

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316936

涿州市新华印刷有限公司印装 各地书店经销

*

开本：710×1 000 1/16 印张：26.5 字数：565 千字

2013 年 4 月第 1 版 2013 年 4 月第 1 次印刷 印数：4 000 册

ISBN 978 - 7 - 5124 - 1098 - 5 定价：59.00 元

前　言

➢ 本书的结构与简介

与人们熟知的通用计算机系统相对应的概念是专用集成电路系统，专用集成电路系统的特点在于面向某些方面应用、存储空间相对较小且具有特定的外设，系统的核心为 ARM 芯片、DSP 芯片或 FPGA 等可编程芯片。随着人们对智能技术提出越来越高的要求，专用集成电路系统的软件设计越来越复杂，特别是基于 ARM 核心的专用集成电路系统，往往需要加载嵌入式操作系统，例如 Windows CE、嵌入式 Linux、VxWorks、eCos、μC/OS－II 等，然后在嵌入式操作系统的基础上设计用户应用程序。

嵌入式操作系统与通用 Windows XP(或 Windows 7)系统有较大的区别，一般地，可以认为嵌入式操作系统具有体积小、实时性强、可靠性高、功能可裁剪、系统可移植等特点。www. google. cn 上关于嵌入式操作系统的定义为“为嵌入式计算机系统设计的操作系统，该操作系统被设计得非常紧凑和高效，舍弃了那些不会被用于专用场合下的非嵌入式计算机操作系统提供的函数，嵌入式操作系统往往是实时操作系统。例如，ATM、CCTV 系统、机顶盒、GPS、MP5 和机器人等设备上常使用嵌入式操作系统。”

本书重点讲述嵌入式操作系统的内核体系以及基于 Cortex－M3 架构 LPC1788 芯片进行面向任务应用程序设计的方法，由于 μC/OS－III 嵌入式实时系统是一款公开了源代码的中小型嵌入式操作系统，适合于教学、研究以及微控制内核的应用，故本书以讲解 μC/OS－III 为主线。全书共分 14 章，第 1 章介绍嵌入式实时操作系统 μC/OS－III 发展历程和系统组成；第 2 章介绍 Cortex－M3 内核体系；第 3 章介绍 IAR KSK LPC1788 开发板与 LPC1788 微控制器；第 4 章介绍 IAR EWARM 软件和应用程序框架；第 5 章介绍 μC/OS－III 在 LPC1788 微控制器上的移植；第 6、7 章分别介绍 μC/OS－III 用户任务和系统任务；第 8 章介绍信号量、任务信号量和互斥信号量；第 9 章介绍消息队列和任务消息队列；第 10 章介绍事件标志组；第 11 章介绍

多事件请求；第 12 章介绍存储管理；第 13 章介绍 LCD 显示原理并给出了一个功能全面的面向任务的程序设计实例；第 14 章讲述基于 Keil MDK 的程序设计方法。其中，第 8～12 章为 μC/OS－III 系统组件。

本书是作者已出版的《μC/OS－II 原理与 ARM 应用程序设计》和《嵌入式操作系统原理与面向任务程序设计》（西安电子科技大学出版社）的升级篇，偏重于讲述嵌入式操作系统 μC/OS－III 的内核结构与应用技术，读者可以在本书实例的基础上，修改并扩展具有个人特色的功能，对以后工程应用具有较强的指导作用。作为从事嵌入式方面教学与科研的大学教师，作者将会一直跟踪 μC/OS－III 的升级和发展，并不断充实和修订本书。

➢ 本书的自学方法

本书理论内容尽可能做到自成体系，读者只需参考一些芯片资料和本书列出的参考文献就可通读全书。但是工程实例源代码是完整的和自成体系的，即每个工程的代码都是完整的。读者仍然可以通过 Email：zhangyong@jxufe.edu.cn 或 QQ：493815991 向作者索取源代码，作者将针对 EWARM 提供 H－JTAG 或 J－Link 仿真器实现的工程，针对 Keil MDK 提供 U－Link2 仿真器实现的工程。对 LPC1788 芯片而言，建议使用 U－Link2 仿真器。

在自学过程中，读者需要同步阅读 LPC1788 和 Cortex－M3 的相关资料，特别是对于硬件设计相关专业的学生，应充分理解书中给出的原理图，对 LPC1788 芯片的片上资源和外设有全面的了解。但愿读者的记忆力足够好，一些内容是需要死记硬背的。

本书已经将 μC/OS－III 应用程序设计技术讲解得非常全面，但是仍然建议读者进一步参考 J. J. Labrosse 先生关于 μC/OS－III 的英文原著，以及经常登录 www.micrium.com 网站了解 μC/OS－III 的最新进展，自学掌握 μC/Probe 观测 μC/OS－III 系统变量和用户变量的方法。

最后，建议读者有一套 Cortex－M3 架构的 LPC1788 实验平台，自己动手，在实现书上提供的实例的基础上，能独立地进行实例设计和实现工作，这样将会有一个巨大的能力提升。

➢ 本书的教学思路

本书根据作者的讲义整理而成，理论课时为 32 学时，实验课时为 32 学时，开放实验课时为 16 学时。如用做大学本科教材，则理论课时宜为 28～40 学时，建议讲述第 1～12 章内容，讲述顺序为 2→3→1→4→5→7→6→8→9→10→11→12。当理论课时高于 32 学时时，可讲述全部内容。实验学时建议为 20～32 学时。

建议理论教学与实验教学同步进行。理论教学过程中，可设置 2 学时讨论课，或让学生分组作学习交流主题报告。实验教学可设置 3～4 个基础性实验和 1～2 个设计性实验，实验应涉及 μC/OS－III 内核以及实验平台外设驱动等方面的内容，应以学生自己动手为主。

➢ 本书的特色

本书具有以下四个方面的特色：

其一，很细致地讲解了嵌入式实时操作系统 μC/OS－III 的基本结构和工作原理，是目前可查的为数不多的 μC/OS－III 方面的中文图书；

其二，基于 μC/OS－III 讲解了其在 Cortex－M3 内核 LPC1788 芯片上的应用，重点讲述了 LPC1788 芯片设计、加载和运行嵌入式操作系统和应用程序的方法；

其三，实例丰富，书中给出了 23 个代码完整的实例，通过这些实例详细阐述了 μC/OS－III 各个组件的使用方法和技巧；

其四，对硬件平台和软件设计均有详细的描述，详细介绍了 Cortex－M3 内核、LPC1788 芯片和面向任务的程序设计方法，对嵌入式系统设计具有较强的指导作用。

➢ 本书的配套源码

实例丰富是本书的特色，书中给出了 23 个实例，后面章节的实例是在前面章节实例的基础上扩充或修改而来的，使得所有 23 个实例均为代码完整的实例。根据本书内容，读者能还原本书的所有实例工程，作者强烈建议读者按书中方法和源代码重构各个实例，加深学习认识；此外，读者可通过 Email：zhangyong@jxufe.edu.cn 向作者索取所有工程源代码，务请来信时告知使用的仿真器和平台；此外，在北京航空航天大学出版社网站上也可以下载到所有工程代码。

➢ 致　谢

感谢恩智浦半导体公司(NXP)提供了 IAR－LPC1788 实验板和 EA－LPC1788 实验板，本书所有实例均基于这两个实验板，这些实例也支持集成了 LPC1788 芯片的类似实验平台，甚至稍作修改即可适用于基于 Cortex－M3 内核的其他 LPC 系列芯片。同时，感谢 NXP 公司金宇杰、王朋朋、周荣政、梅润平、张宇和宋岩等领导的大力支持，特别感谢张宇工程师为本书编写提出了大量建设性意见。感谢北京博创公司陆海军总经理对本书编写的关心与支持。

同时，感谢那些将阅读本书并提出宝贵意见的读者，这些意见对于作者编写新书或修订再版已出版的书籍将有实质性的帮助。由于作者水平有限，书中难免会有纰漏之处，敬请专家和读者批评指正。

➢ 免责声明

作　者

2012 年 12 月于江西财经大学

目　录

第1章

嵌入式实时操作系统 μC/OS-III概述

本章将介绍μC/OS-III的发展历程、特点、应用领域和系统结构，重点在于通过与μC/OS和μC/OS-II对比，阐述μC/OS-III的特色，并详细讨论μC/OS-III的系统组成、文件结构、配置文件、用户应用程序接口(API)函数和自定义变量类型等。

1.1 μC/OS-III发展历程

自1992年μC/OS诞生至2012年μC/OS-III开放源码，20年来，这款嵌入式实时操作系统在嵌入式系统应用领域得到了人们广泛的认可和喜爱，特别是在教学领域，由于其开放全部源代码，且对教学用户免费，因此受到了广大嵌入式相关专业师生的欢迎。

μC/OS内核的雏形最早见于J. J. Labrosse于1992年5—6月发表在*Embedded System Programming*杂志上的长达30页的实时操作系统(RTOS)。J. J. Labrosse可称为“μC/OS之父”。1992年12月，J. J. Labrosse将该内核扩充为266页的书*μC/OS the Real-Time Kernel*。在这本书中，μC/OS内核的版本号为V1.08，与发表在*Embedded System Programming*杂志上的RTOS不同的是，书中对μC/OS内核的代码作了详细的注解，针对半年来用户的一些反馈作了内核改进，解释了μC/OS内核的设计与实现方法，指出该内核是用C语言和最小限度的汇编代码编写的，这些汇编代码主要涉及与目标处理器相关的操作部分。μC/OS V1.08最大支持63个任务，凡是具有堆栈指针寄存器和CPU堆栈操作的微处理器均可以移植该μC/OS内核。事实上，当时该内核已经可以和美国流行的一些商业RTOS相媲美了。

μC/OS内核发展到V1.11后，1999年，J. J. Labrosse出版了*MicroC/OS-II The Real Time Kernel*，正式推出了μC/OS-II，此时的版本号为V2.00或V2.04(V2.04与V2.00本质上相同，只是V2.04在V2.00的基础上对一小部分函数作了调整)。同年，J. J. Labrosse成立了Micrium公司，研发和销售μC/OS-II软件；这一年初，J. J. Labrosse还出版了*Embedded Systems Building Blocks, Second*

Edition: Complete and Ready-to-use Modules in C，这本书当时已经是第 2 版，针对 μC/OS-II 详细阐述用 C 语言实现嵌入式实时操作系统各个模块的技术，并介绍了微处理器外设的访问技术；然后，在 2002 年出版了 *MicroC/OS-II The Real Time Kernel Second Edition*（第 2 版），在该书中，介绍了 μC/OS-II V2.52 内核。μC/OS V2.52 内核具有任务管理、时间管理、信号量、互斥信号量、事件标志组、消息邮箱、消息队列和内存管理等功能，相比 μC/OS V1.11，μC/OS-II 增加了互斥信号量和事件标志组的功能。早在 2000 年 7 月，μC/OS-II 就通过了美国联邦航空管理局（FAA）关于商用飞机的符合 RTCA DO-178B 标准的认证，说明 μC/OS-II 具有足够的安全性和稳定性，可以用于与人性命攸关、安全性要求苛刻的系统中。

张勇在 2010 年 2 月和 12 月出版了两本关于 μC/OS-II V2.86 的书：《μC/OS-II 原理与 ARM 应用程序设计》和《嵌入式操作系统原理与面向任务程序设计》。当时 μC/OS-II 的最高版本就是 V2.86，相比 V2.52，其重大改进在于自 V2.80 后，由原来只能支持 64 个任务扩展到支持 255 个任务，自 V2.81 后支持系统软定时器，到 V2.86 支持多事件请求操作。J. J. Labrosse 的书是采用“搭积木”的方法编写的，读起来更像是技术手册，这对于初学者或入门学生而言，需要较长的学习时间才能充分掌握 μC/OS-II；而张勇的书则从实例和应用的角度进行编写，特别适合于入门学者。对于那些对硬件不太熟悉的初学者，还可以参考一下张勇 2009 年 4 月出版的《ARM 原理与 C 程序设计》。后来，J. J. Labrosse 对 μC/OS-II 进行了极其微小的改良，形成了现在的 μC/OS-II 的最高版本 V2.91。

现在，μC/OS-II 仍然在全球范围内被广泛使用，但是早在 2009 年，J. J. Labrosse 就推出了 μC/OS 第三代 μC/OS-III。最初的 μC/OS-III 仅向授权用户开放源代码，这在一定程度上限制了它的推广应用。直到 2012 年，新的 μC/OS-III 才面向教学用户开放源代码，此时的版本号已经是 V3.03。伴随 μC/OS-III 的诞生，Labrosse 还针对不同的微处理器系列编写了大量相关的应用手册，目前面世的就有 *μC/OS-III: The Real-Time Kernel for the Freescale Kinetis*、*μC/OS-III: The Real-Time Kernel for the NXP LPC1700*、*μC/OS-III: The Real-Time Kernel for the Renesas RX62N*、*μC/OS-III: The Real-Time Kernel for the Renesas SH7216*、*μC/OS-III: The Real-Time Kernel for the STMicroelectronics STM32F107*、*μC/OS-III: The Real-Time Kernel for the Texas Instruments Stellaris MCUs*。令人欣慰的是，这 6 本书均可以从 Micriμm 官方网站 http://www.micrium.com 上免费下载全文电子稿阅读。实际上，这 6 本书的每一本都包含两部分内容，即均分为上下两篇，上篇是以 μC/OS-III 为例介绍嵌入式实时操作系统工作原理，下篇是针对特定的芯片或架构介绍 μC/OS-III 的典型应用实例，因此，这 6 本书的上篇内容基本相同；而下篇内容则具有很强的针对性，不同的手册采用了不同的硬件平台，而且编译环境也不尽相同，有采用 Keil MDK 或 RVDS 的，有采用 IAR EWARM 的。北京航空航天大学出版社也正在对其中一些书的中译本进行紧张的出版工作。

尽管 μC/OS - III 的工作原理与 μC/OS - II 有相同之处，但是，专家普遍认为 μC/OS - III 相对于 μC/OS - II 是一个近似全新的嵌入式实时操作系统，本书第 1.2 节将对比这两个操作系统的特点，来进一步说明 μC/OS - III 的优势。关于 μC/OS - II 的详细内容，请读者首先参考张勇的《μC/OS - II 原理与 ARM 应用程序设计》和《嵌入式操作系统原理与面向任务程序设计》，然后深入学习 J. J. Labrosse 的 *MicroC/OS - II The Real - Time Kernel　Second Edition*，其中译本《嵌入式实时操作系统 μC/OS - II(第 2 版)》已经于 2003 年 5 月由北京航空航天大学出版社出版了。因此，本书中仅介绍 μC/OS - III 的工作原理和应用实例，不再涉及 μC/OS - II 的方方面面，同时考虑到避免与 J. J. Labrosse 先生已出版的系列手册内容重复，本书中没有对 μC/OS - III 的工作原理全面展开论述。本书重点考虑的读者为入门初学者和嵌入式相关专业大学本科学生，同时兼顾这方面的工程师、嵌入式爱好者和研究生对嵌入式实时操作系统的学习。

Micriμm 网站 http://www.micrium.com 上有大量关于 μC/OS - III 的应用手册和资料以及不断更新的 μC/OS - III 最新源代码供读者下载，显然，μC/OS - III 是一个不断发展和进化的嵌入式实时操作系统，初学者应经常浏览该网站，并获取最新的 μC/OS - III 应用信息。需要强调指出的是，尽管 μC/OS - III 是开放源代码的，但是 μC/OS - III 不是自由软件，那些用于非教学及和平事业的商业场合下的用户，必须购买用户使用许可证。

1.2　μC/OS - III 特点

嵌入式实时操作系统 μC/OS - III 是具有可裁剪和方便移植的抢先型实时内核，具有以下特点。

(1) 开放源代码

μC/OS - III 系统内核包括 20 个文件，约 17 098 行代码，所有这些代码采用右对齐的方式被完美地注释了，使得这些代码本身就是一部良好的学习 C 语言和实时操作系统的手册。

(2) API 函数命名规范

μC/OS - III 提供了 81 个用户可调用的应用程序接口(API)函数，需要掌握的只有 55 个函数。所有这些函数的命名非常规范，都是以 OS 开头，达到了见名知义的目的，例如，OSVersion 函数用于返回当前 μC/OS - III 的版本号；OSTimeDly 函数允许一个任务延时一定的时钟节拍数等。

(3) 抢先型多任务内核

μC/OS - III 系统的任务具有 5 种状态，即就绪态、运行态、等待态、中断态和休眠态。任何时刻处于运行态的任务都只能是处于就绪态的优先级最高(优先级号最

小)的那个任务,当有多个就绪的任务优先级相同且同时是就绪态所有任务的最高优先级任务时,μC/OS－III 将按时间片循环方法依次执行这些同优先级的任务。

(4) 高性能中断管理

μC/OS－III 提供了两种保护临界区代码的方法,其一是与 μC/OS－II 相同的关闭中断的方法,其二是新的通过锁住任务调度器防止任务切换的方法。在后者中,执行临界区代码时,没有关闭中断,因此,与 μC/OS－II 相比,μC/OS－III 的总的中断关闭时间大大减小,使得 μC/OS－III 可以响应快速中断源。

(5) 中断和服务执行时间确定

在 μC/OS－III 中,中断响应时间是确定的,同时,绝大多数系统服务的时间也是确定的。

(6) 可裁剪

从 Micriμm 官网上下载 μC/OS－III 系统,其文件名为 KRN-K3XX-000000.zip,其解压目录\Micrium\Software\uCOS－III\Cfg\Template 下,有一个名为 os_cfg.h 的配置用头文件,通过设置其中宏定义的值为 0u 或 1u 等,可以启用或关闭 μC/OS－III 的某些系统服务,例如,os_cfg.h 文件中的第 56 行代码:

```
#define    OS_CFG_FLAG_EN        1u
```

如果上述代码的最后设为 0u,则关闭 μC/OS－III 中事件标志组的服务功能,即不允许使用事件标志组相关的所有 API 函数,那么事件标志组就被从 μC/OS－III 中裁剪掉了;如果设为 1u,如上述代码所示,则启用 μC/OS－III 中事件标志组的服务功能,即允许用户使用事件标志组的某些相关 API 函数,例如创建、请求或释放事件标志组函数。

(7) 移植性强

μC/OS－III 内核可以被移植到 45 种以上的 CPU 架构上,涉及到单片机、DSP、ARM 和 FPGA(SoPC 系统)等,所有 μC/OS－II 的移植代码稍做修改就可以用于 μC/OS－III 的移植。那些具有堆栈指针寄存器和支持 CPU 寄存器堆栈操作的 CPU 芯片均可以移植 μC/OS－III,这使得只有极少数芯片例如 Motorola 68HC05(该芯片不具备用户堆栈操作指令)等不能移植 μC/OS－III 系统。

(8) 可固化在 ROM 中

μC/OS－III 内核可以与用户应用程序一起固化在只读存储器(ROM)中,实际上,μC/OS－III 内核是作为用户软件系统的一部分存在的。这点与 Windows CE 嵌入式实时操作系统不同,Windows CE 系统本身具有文件系统和用户界面,使得它与桌面 Windows 系统的操作相类似。当移动设备安装了 Windows CE 系统后,用户只需要在该系统上安装用户应用程序即可,即在 Windows CE 系统下,绝大多数应用程序与 Windows CE 系统是相对独立的。μC/OS－III 没有文件系统,也没有用户界面,它主要是提供了操作系统的任务调度功能,因此,它必须作为用户应用软件的一部分存在于嵌入式系统中,与用户应用软件一起移植固化到移动设备的 ROM 中。

(9) 运行时可配置组件

μC/OS－III 内核是一个模块化或组件化的嵌入式实时操作系统，它具有任务、信号量、互斥信号量、事件标志组、消息队列、内存分区、系统软定时器等组件服务，这些组件的数量可以在 μC/OS－III 系统运行时配置其数量，甚至可以配置其占用堆栈的大小。

(10) 支持无限多任务数

μC/OS－II 内核最多可支持 255 个任务，μC/OS－III 则可以支持无限多个任务。事实上，在 μC/OS－III 中，任务的调度算法(此处特指任务优先级解析算法，例如任务的优先级号因任务就绪、执行或挂起等原因进出“任务优先级表”的算法)没有 μC/OS－II 高级，由于 μC/OS－III 是基于 32 位的内核且支持无限多个任务调度，所以，其优先级号解析算法就是简单的数组元素中的位元判断。对于不支持硬件指令统计变量中第一个 1 位所在位置的处理器而言，尽管基于这类优先级号解析的调度算法的时间确定性仍可以保证，但是对于不同优先级的调度时间，做不到完全相等。

(11) 支持无限优先级数

在 μC/OS－III 中，任务的优先级号可以使用从 0 至无穷大正整数中的任意整数，即在 μC/OS－III 中，任务的优先级号的数目是无限的。

(12) 支持同优先级任务

在 μC/OS－II 中，各个任务的优先级号不能相同；而在 μC/OS－III 中，允许具有相同优先级号的多个任务存在，当这些任务同时就绪且是就绪的所有任务中优先级最高的任务时，μC/OS－III 内核将使用时间片轮换方法(Round－Robin Scheduling)进行这些任务的运行调度，并且，每个任务占有的 CPU 时间片大小可以由用户设定。

(13) 新的组件(或服务)

在 μC/OS－II 的基础上，μC/OS－III 内核提供了新的任务级别的组件(或服务)，例如，任务信号量和任务消息队列，这两种组件分别实现了任务直接向其他任务请求或释放信号量或消息(队列)，或者中断直接向任务释放信号量或消息(队列)，而无需创建“全局”的信号量和消息队列。因此，μC/OS－III 共有以下组件(或服务)：任务、信号量、任务信号量、互斥信号量、事件标志组、消息队列、任务消息队列、系统软定时器、内存分区等。

(14) 互斥信号量

互斥信号量用于保护共享资源，且能避免任务间竞争共享资源而死锁。在 μC/OS－III 中，对互斥信号量采用内置的优先级继承方法，即当低优先级的任务请求了互斥信号量在使用共享资源时，被更高优先级的就绪任务抢占了 CPU 使用权，而当这个更高优先级任务请求互斥信号量使用共享资源时，原先的占用互斥信号量正在使用该共享资源的任务的优先级将上升到与后者任务相同的任先级，后者任务进入到互斥信号量等待列表中。与 μC/OS－II 相比，在 μC/OS－III 下互斥信号量不占用新的更高的优先级号，因此，μC/OS－III 避免了没有约束的优先级继承优先级号

的增加。同时,互斥信号量的请求释放可以做最大 250 次的嵌套使用。

(15) 任务挂起嵌套

与 μC/OS - II 相似,μC/OS - III 中一个任务调用 OSTaskSuspend 函数可以将自身或其他任务挂起。如果是挂起其他任务,则在 μC/OS - III 中可以有 250 级嵌套,而 μC/OS - II 不具有该能力。被嵌套的挂起必须调用与 OSTaskResume 函数同样多的级数才能把任务从挂起状态恢复出来。

(16) 支持系统软定时器

与 μC/OS - II 相似,μC/OS - III 支持系统软定时器,即 μC/OS - III 系统定时器任务 OS_TmrTask 产生的软件定时器,包括两种,其一为单拍(One - Shot)型,其二为周期型。前者定时一次后即自动关闭,后者按指定的定时周期不断循环。这两种定时器的共同点在于,当定时值减到 0 后,自动调用该定时器关联的回调函数;不同点在于,对于周期型定时器,当定时值减到 0 后,自动重新装入定时周期值。

(17) 支持任务级别寄存器

用户可以在任务级别的寄存器保存任务调用出错码、任务 ID 号、每个任务的中断关闭时间等。

(18) 任务调用出错检查

μC/OS - III 中,每个 API 函数被调用后均返回一个出错码,当该出错码为 OS_ERR_NONE(该常量在 os.h 文件中被宏定义为 0u)时,常常表示该 API 函数被正常(或正确)调用。其他的出错码则或多或少地对应着一些不正常调用,一般地在程序中需要用 switch 语句检测这些出错码的出现,以进行相应的异常处理。

(19) 内置性能检测

在调试基于 μC/OS - III 内核的应用程序中,可以借助 μC/OS - III 内置的性能检测函数,实时掌握每个任务的执行时间、每个任务的堆栈占有情况、任务的切换次数、CPU 利用率、中断到任务和任务间切换的时间、中断关闭和调度器关闭时间、系统内部各种数据结构(记录任务或事件的各种列表、数组或链表等)的最大利用个数等(例如,对于长度为 200 的系统数组,在程序运行过程中,峰值占用个数为 120,则 120 为该数组的最大利用个数)。

(20) 易于优化

μC/OS - III 的易于优化特性表现在两个方面:其一,针对不同字长的 CPU,可以将 μC/OS - III 优化为对应的系统,尽管 μC/OS - III 本身是 32 位的,但是若目标系统是 8 位或 16 位的,那么可以很容易地将 μC/OS - III 按目标机的字长优化;其二,某些与硬件密切相关的函数可以用汇编语言实现,从而进一步提升 μC/OS - III 的运行速度,例如,优先级号解析算法,对于具有定位变量中第一个不是 0 的位元位置的汇编指令的处理器,可以用汇编语言实现优先级号解析算法,这将大大加快 μC/OS - III 的任务调度过程。