文章编号：1001－9081(2009)04－1128－03

UC/OS-II内核扩充时间片轮转调度算法的设计

高富强[1]，秦昌硕[2]，游纪原[1]，邹　恒[2]
（1.重庆大学 ICT研究中心，重庆 400030；2.重庆大学 自动化学院，重庆 400030）
（qcs20061302093@126.com）

摘　要：通过详细分析 UC/OS-II的内核任务调度算法机制，对该算法机制在实际应用中存在的不足做出探讨，提出了一种实现时间片轮转调度算法的扩充设计，使改进后的系统内核具备了优先级抢占调度和时间片轮转调度两种调度机制。较好地弥补原内核任务调度机制存在的不足，并在实际应用中取得了良好的效果。

关键词：UC/OS-II；时间片轮转；任务调度
中图分类号：TP316.2　　文献标志码：A

Extension of time slice circular scheduling in UC/OS-II kernel

GAO Fu-qiang[1], QIN Chang-shuo[2], YOU Ji-yuan[1], ZOU Heng[2]
(1. ICT Research Center, Chongqing University, Chongqing 400030, China;
2. Automation College, Chongqing University, Chongqing 400030, China)

Abstract: Based on the detailed analysis of UC/OS-II kernel task scheduling, this paper discussed the limitation of the scheduling algorithm in practical application, and then put forward the design that achieved time slice circular scheduling. It made the operating system have two kinds of task scheduling— task scheduling based on priorities and time slice circular scheduling, and it also preferably repaired the lack of the UC/OS-II kernel. Finally the new kernel obtained good effect in practical application.

Key words: UC/OS-II; time slice circular; task scheduling

0　引言

UC/OS-II是一种公开源代码、结构小巧、具有可剥夺实时内核的嵌入式实时操作系统。其具有占用空间小，易于移植，高度可剪裁以及开源的特性。因此在嵌入式领域，尤其在中小型嵌入式实时领域，UC/OS-II凭借其特性得到了广泛的应用。但同时在具体应用中也发现了它的不足之处。其中系统内核调度算法存在着的局限性，对采用它作为底层操作系统的嵌入式设计产生了较大的影响。于是怎样改进操作系统内核任务调度机制，以满足嵌入式应用设计要求的问题就凸显出来。

1　任务调度分析和局限性研究

UC/OS-II的内核是基于任务优先级的抢占式多任务调度，每个任务都有自己的优先级，不能和其他任务优先级相同[1]。其中每个任务都和一个任务控制块（Task Control Block，TCB）一一对应，在 TCB中包括了任务的堆栈指针，优先级，事件控制相关的属性等。内核调度的主要任务就是尽可能地保证就绪态任务中优先级最高的任务处于运行态。

1.1　内核任务调度机制简要分析

任务调度算法主要涉及了如下数据结构：

TCB(任务控制块)

OSTCBPrioTbl[]（任务块优先级表）

OSRdyTbl[]，OSRdyGrp(就绪表)

OSUnMapTbl(反映射表)

其中 OSTCBPrioTbl[]的元素为指向 TCB的指针，通过优先级数作为数组下标指向对应的 TCB。就绪表中有两个变量 OSRdyGrp和 OSRdyTbl[]。在 OSRdyGrp中，任务按优先级分组，8个任务为一组。OSRdyGrp中的每一位表示 8组任务中每一组中是否有进入就绪态的任务。任务进入就绪态时，就绪表 OSRdyTbl[]中的相应元素的相应位置位。就绪表 OSRdyTbl[]数组的大小取决于 OS_LOWEST_PRIO。OSRdyGrp和 OSRdyTbl[]的关系为：当 OSRdyTbl[i]中的任何一位是 1时，OSRdyGrp的第 i位置 1。i从 0到 7。它们的关系如图 1所示。

图 1　就绪表数据结构关系

这样每当有任务进入或退出就绪态时，都要对 OSRdyGrp和 OSRdyTbl[]进行操作，并通过 OSRdyGrp、OSRdyTbl[]和 OSUnMapTbl得出当前处于就绪态中的最高优先级任务的优先级值，继而通过该优先级值和 OSTCBPrioTbl[]得到该任务

收稿日期：2008－10－30；修回日期：2008－12－04。　基金项目：国家 863计划项目（2006AA04Z104）。

作者简介：高富强（1956－），男，重庆人，副教授，主要研究方向：嵌入式数据采集、工业自动化；　秦昌硕（1983－），男，重庆人，硕士研究生，主要研究方向：嵌入式操作系统、工业 CT数据传输；　游纪原（1961－），男，重庆人，副教授，高级工程师，主要研究方向：工业计算机设计、机电控制；　邹恒（1981－），男，重庆人，硕士研究生，主要研究方向：神经网络控制系统设计。

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

对应的 TCB，最后任务切换函数使该任务进入运行态。

由于 UC/OS-II的调度算法十分精简，所以对以后支持更多优先级数提供了方便，在 UC/OS-II v2.52版本中最大只支持 64个优先级，而 UC/OS-II v2.80版就可以支持 256个优先级了，和 v2.52不同就在于 OSRdyGrp和 OSRdyTbl[]的元素长度由 v2.52中的 8位变为了 16位，那么可以以此类推，要支持 512、1 024个优先级只要改变这两个变量的元素长度就可以了，需要注意的就是和优先级变量“Prio”相关的常量和变量要由原来的 8位变为 16位。因为 8位的变量最大只能表示 256个优先级数。

1.2　任务调度机制的局限性

UC/OS-II的内核调度机制简洁、高效，但同时 UC/OS-II的任务调度算法也存在其不足的地方，比较典型的就是 UC/OS-II只能支持一个优先级只能对应一个任务，这样的调度策略有如下缺点：

1)不支持同优先级下的多任务轮转调度[2]。

在实际嵌入式设计过程中对应用系统任务的划分是存在同优先级任务的，比如网络应用中有两个处理大量网络传输数据的任务，这时就需要用同优先级任务。而 UC/OS-II不支持同优先级的多任务调度。这样只能通过任务之间的同步等动作来完成交替运行，以达到同优先级任务轮转调度的效果。这不但增加了设计划分任务时的难度，而且破坏了模块的独立性。

2)任务的数量受优先级数量的限制。

由于任务和优先级是一一对应关系，那么在实际应用中有多少任务，就要系统提供多少的优先级，但随着嵌入式系统的日趋复杂，任务数量会越来越多，如 UC/OS-II v2.52只支持 64个优先级，也就意味着支持的任务最大数量为 64个，到了 UC/OS-II v2.80，优先级数支持到了 256个，但上述的问题只是缓解了，并没有彻底解决。因为随着嵌入式系统设计的日趋复杂，总有 256个任务不够用的时候到来。

3)对于实时系统中的优先级反转问题缺乏有效的解决手段。

优先级反转问题的通常解决方法是使用优先级继承，而优先级继承的实现是需要操作系统本身支持同优先级任务的。而 UC/OS-II不支持同优先级任务，在优先级反转的问题上采用对获得信号量的任务设定一个较高的固定不变的优先级来解决，这样的解决方法并不能完美解决优先级反转问题，而且还在一定程度上影响了系统的实时性。以下举例说明：

假设一个系统中有 3个用户任务，分别为 task1、task2和 task3，其优先级分别为 6、7和 8，一个互斥型信号量其临界优先级设置为 5，task1和 task3都需要对该信号量进行操作，此时若 task1处于等待状态，但并非等待该互斥信号量(因等待其他事件或延时等而进入等待状态)，此时 task3得到该互斥信号量并处于就绪态，则 task3的优先级切换为 5，而同时 task2也为就绪态，但因为 task3的优先级比 task2高，所以 task3得以运行。而优先级高的 task2却不能运行。(此时并未发生优先级反转)而且当 task1也进入就绪状态时，同样不能运行，只有等待 task3释放互斥信号量变回原优先级，才得以运行。可以看出，在未发生优先级反转的情况下，UC/OS-II原有的解决优先级反转的机制反而打乱了任务优先级调度顺序，破坏了系统实时性。

通过对 UC/OS-II的调度算法的分析，说明了只有对系统内核调度机制做出改进，使其支持同优先级任务的时间片轮转调度，才可以较好地弥补前文所叙述的不足。

2　内核调度算法的改进设计

改进的目的：在 UC/OS-II内核基于优先级的抢占式调度算法的基础上扩展同优先级任务的时间片轮转调度算法[3]。在扩展的同时必须保证 UC/OS-II的原有的实时性，系统 API函数使用的兼容性，以及原内核的 CPU移植代码的可复用性，并尽量减少对原有内核代码的修改。

2.1　内核就绪任务调度相关数据结构的修改

通过对 UC/OS-II内核调度的分析可以看出，因为优先级和任务是一一对应，所以调度算法在就绪表上查找出最高优先级的同时就确定了相应任务的 TCB，这个一一对应关系的具体表现在 OSTCBPrioTbl[](任务块优先级表)，当优先级 Prio一定时，通过 OSTCBPrioTbl[Prio]就能确定唯一的任务 TCB(类似于索引功能)。而时间片轮转调度中一个优先级是可以对应多个任务的。则 OSTCBPrioTbl[]的元素应该由原来的指向单个 TCB改进为指向多个 TCB，采用双向循环 TCB链表来连接同优先级下的多个 TCB块。需要说明的是：由于 OSTCBPrioTbl[]是整个内核中所有任务 TCB的索引(事件控制等系统功能也要用到 OSTCBPrioTbl[])，所以 OSTCBPrioTbl[]改名为 OSTCBReadyTbl[]，专门用作对就绪任务 TCB的索引。另增加了一个变量为：OSTCBIDTbl[]，用做接替原 OSTCBPrioTbl[]对系统中所有任务的索引的功能。因为优先级值已经不能唯一确定一个任务了，所以采用了任务 ID号来确定(原内核中任务 TCB中默认就有任务 ID的属性)。

根据时间片轮转调度在同优先级任务中的应用，需要对任务的 TCB进行修改，增加 4个属性项：

1) OSTSLen(时间片初始值)。保存任务建立时给任务分配时间片的个数。

2) OSTSCurLen(时间片剩余值)。记录了任务运行中还剩余的时间片个数[4]。

3) OSTSPrev(TCB双向链表前驱指针)。用作组成图 2所示的 TCB循环链表。

4) OSTSNext(TCB双向链表后继指针)。用作组成图 2所示的 TCB循环链表。

OSTCBPrioTbl[]和 TCB块的关系改动如图 2所示。

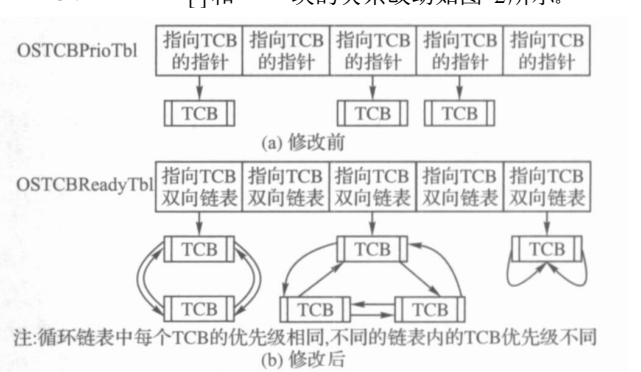

图 2　OSTCBPrioTbl[]结构修改示例

通过改动，由就绪表(OSRdyTbl[]，OSRdyGrp)就能得出进入就绪状态任务中最高优先级值，再根据最高优先级值和 OSTCBReadyTbl[]确定需要进入运行态的任务 TCB双向循环链表，而后对链表中的任务进行时间片轮转调度。

2.2　内核就绪任务调度部分的时间片轮转调度算法设计

时间片轮转调度中的时间片的消耗计数是通过系统时钟函数 OSTimeTick()来实现的。即时间片轮转中的时间片单位长度为一个操作系统节拍。OSTimeTick函数的原有功能

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

是对延时状态任务的剩余时间片数进行递减操作，如果发现有延时完成的任务就将其转出延时态。为了实现时间片轮转，需要增加 OSTimeTick函数的功能。即在函数中增加对当前运行态的任务 TCB中的时间片剩余值进行递减操作的功能[5]。若 OSTSCurLen为零，说明时间片用完，需要让循环链表中的下一个任务进入运行态。此时 OSTCBReadyTbl[]中指向当前运行态任务 TCB的指针改指向链表中的下一个 TCB，最后调用任务切换函数把下一个同优先级任务转入运行态。其基本流程如图 3所示。

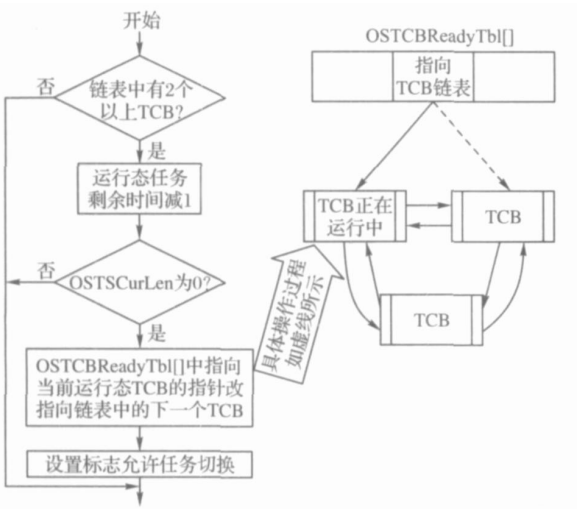

图 3 时间片轮转调度算法程序流程

由于 TCB链表是循环链表，所以每一个在链表中的任务都会依次循环执行。这样就实现了时间片轮转算法在内核部分的扩展。

2.3 内核事件控制部分扩展设计

内核就绪任务调度部分实现了时间片轮转调度的扩展后，UC/OS-II原有的事件控制部分的代码已经不能适用于新内核了。通过阅读 UC/OS-II事件控制部分的源代码可以清楚的了解：UC/OS-II对事件控制部分的任务调度处理和内核中就绪表的任务调度处理机制几乎完全一样[6]。就绪表中的 OSRdyTbl[]，OSRdyGrp和结构体 OS_EVENT中的 OSEventGrp和 OSEventTbl[]完成相同的功能，只是任务就绪时（等待 CPU时）插入就绪表，而当任务需要等待事件时要插入 EVENT等待表，反之亦同。

扩展设计采用了单一的 TCB双向循环链表结构的等待事件 TCB队列设计。即在事件控制结构体 OS_EVENT中删除 OSEventGrp和 OSEventTbl[]。增加一个指向 TCB双向循环链表头的指针 OSEventWaitList，该指针总是指向本事件等待任务队列中优先级最高任务的 TCB。

链表中的 TCB按照任务优先级的高低进行排列，最高优先级的在前，最低的在末尾。相同优先级的按照先进在前后进在后的规则进行排列。当有新 TCB进入等待链表时则按照上述规则插入链表中。若等待事件发生（信号量的释放等）则选择等待队列中最高优先级的 TCB（即链表的第一个 TCB）脱出链表。也就是 OSEventWaitList指向的 TCB，然后把 OSEventWaitList指向下一个 TCB。

具体修改主要集中在 OS_EventTaskRdy()，OS_EventTaskWait()，OS_EventWaitListInit()3个函数中。

基本结构如图 4所示，虚线表示最高优先级脱出链表后链表的连接关系。

采用这样的数据结构设计大大节约了内存空间，相比原来的事件控制调度设计，只有在有新的任务 TCB加入链表时会对链表成员依次进行检索。由于单个事件控制块等待队列中的 TCB一般只有若干个，和修改 OSEventGrp和 OSEventTbl[]相比只增加了少量的处理时间。但在 TCB脱出链表的操作中不需要依次访问 OSEventGrp和 OSEventTbl[]，OSUnMapTbl，OSTCBPrioTbl[Prio]等数据结构，而是直接得到了 TCB，明显减少了处理时间。因此新的事件控制算法在时间、空间上和原算法相比有较大优势，且支持同优先级调度。

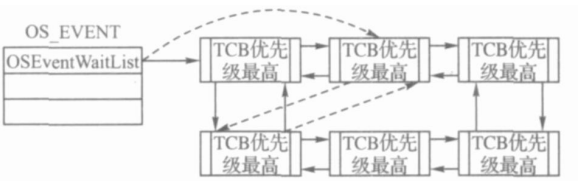

图 4 事件控制块结构

2.4 优先级继承算法的实现

因新内核支持同优先级下的多任务调度，对互斥信号量产生的优先级反转问题可以采用优先级继承解决。将互斥信号量事件结构体中 OSEventCnt成员高 16位为拥有该互斥信号量任务当前所处在的优先级，低 16位用作保存该任务原有的优先级。

当一个任务进入 OSMutexPend函数时，若互斥信号量已被占用，则比较该任务的优先级和拥有该互斥信号量任务当前所处的优先级。若该任务优先级高，则发生优先级继承，改变拥有该互斥信号量任务的优先级值为该任务的优先级。并刷新 OSEventCnt成员的高 16位。再进行下面的操作。

当拥有该互斥信号量任务进入 OSMutexPost函数时，则比较拥有该互斥信号量任务当前所处的优先级值和 OSEventCnt成员的低 16位（保存着该任务原有的优先级值）。若二者不等，表明发生过优先级继承，则使拥有该互斥信号量任务的优先级值恢复为原来的优先级。再进行下面的操作。

具体修改主要集中在 OSMutexPend和 OSMutexPost函数中，主要操作流程如图 5。

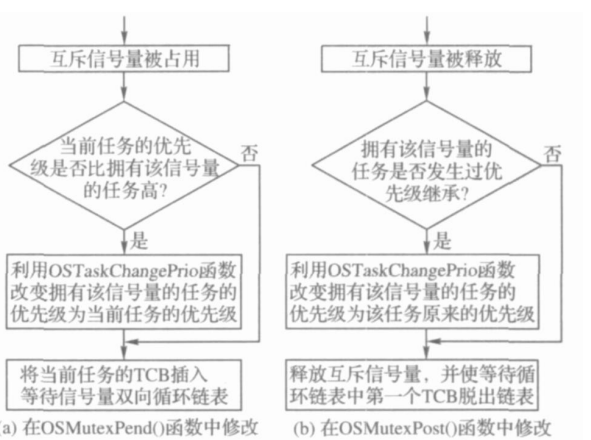

图 5 优先级继承操作流程

3 结语

通过前文对 UC/OS-II源码的修改，主要修改部分已完成，其余有关任务服务函数（OSTaskDel，OSTaskSuspend等）的修改，主要是针对这些服务函数中有关插入和脱出 TCB双向循环链表的代码实现进行修改，用作和已修改的系统内核相接合，保证 UC/OS-II的稳定运行。因修改过程较简单，在此不一一详述。修改后新内核和原内核的部分参数比较如表 1所示。

（下转第 1142页）

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

首先是交互模型设计。本设计阶段指定了用例和用例的具体操作对象——汇集，以及它们之间的操作，并对用例进行详细建模，是逻辑操作设计的核心部分。如图 1所示的用例“出院”，包含了两个基本操作：添加记录 Add_Patient_Out()和删除记录 Delete_Patient_In()。同时在这一步指定了用例的前置条件和后置条件。

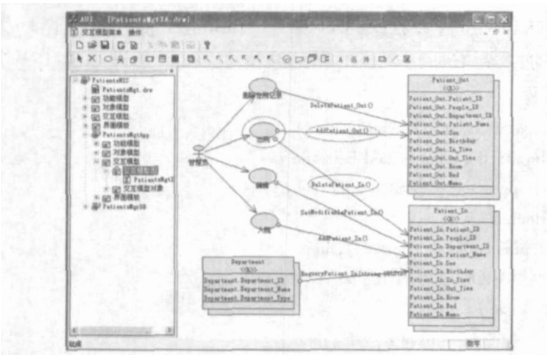

图 1 交互模型设计

在完成模型的设计之后，整个工程被序列化为一个 XML格式的文件进行保存。这个文件中严格按照上面的形式化描述对建模的用例进行记录，下一步的代码生成过程根据此 XML文档的相应记录完成用例的代码生成。

最后，由图 2、3可以看到利用建模进行代码自动生成后得到的 ASP.NET平台代码的运行结果，以及对其中的一条记录执行“出院”操作后，该记录被从在院病人记录中删除，并在曾住院病人历史记录中显示出来后的系统运行效果。

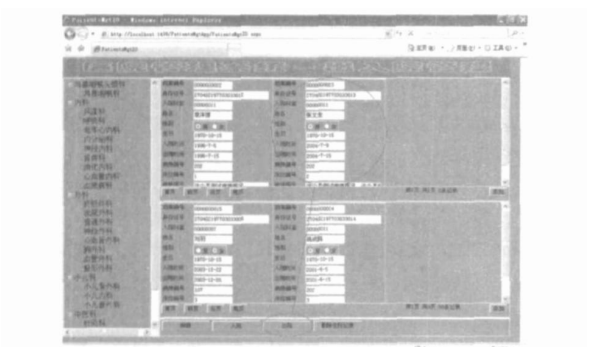

图 2 系统初始运行状态

5 结语

为了加强和完善目前基于模型开发方法的功能模型设计，加强其建模能力，本文提出了以谓词逻辑为基础的复杂行为的形式化建模方法。本方法关注于对原功能模型设计方法的改进，通过对原 FMP模型提供的基本抽象操作进行逻辑组合来完成对复杂行为的建模。逻辑组合方式主要是顺序、选择和循环。

然后本文给出了一个医院管理信息系统出入院管理的实例，展示了从行为建模到最终代码生成实现的过程。

目前的页面功能仅是以管理信息系统作为行为总结和行为建模的对象，如何对所有应用网页的行为进行系统总结以及建模，并对代码生成进行指导是下一步的工作。

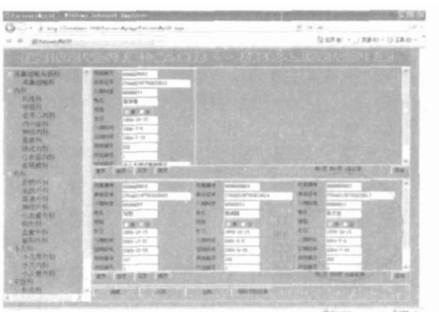

图 3 用例“出院”执行后系统状态

参考文献：

[1] PUERTA A R. A model-based interface development environment [J]. IEEE Software, 1997, 14(4): 40—47.

[2] ELKOUTBI M, KHRISS I, KELLER R K. Automated prototyping of user interfaces based on UML scenarios [J]. Automated Software Engineering, 2006, 13(1): 5—40.

[3] da SILVA P P, PATON N W. A UML-based design environment for interactive applications [C]// Proceedings of the 2nd International Workshop on User Interfaces to Data Intensive Systems: UDIS'01. Washington, D.C: IEEE Computer Society, 2001: 60—71.

[4] 秦严严，田丰，王晓春，等. 以交互为中心的 PostWIMP界面模型 [J]. 软件学报，2006, 17(4): 691—702.

[5] 冯仕红，鹿旭东，万建成. 基于模型的多设备用户界面设计 [J]. 通信学报，2006, 27(11): 55—59.

（上接第 1130页）

表 1 新内核和原 UC/OS-II内核的部分参数比较

参数	新内核	UC/OS-II v2.52
支持优先级数	1024	64
支持任务数	2^{16}	64
支持同优先级任务	支持（可选择为不支持）	不支持
支持优先级继承	支持	不支持
支持时间片调度	支持（可选择为不支持）	不支持

通过扩展设计，UC/OS-II在具有强实时性，结构小巧，高剪裁性，易移植的特点前提下，实现了优先级抢占和时间片轮转的相互协作调度。对 UC/OS-II内核调度机制存在的不足提出了一个良好的解决办法，新内核弥补了前文所提到的 UC/OS-II内核的不足。并且修改后的内核对上层用户应用程序保证了良好的兼容性，几乎所有的系统 API和原 UC/OS-II的 API在函数名称，功能上完全一致（只有若干个 API函数在输入参数上有所不同）。同时新内核的 CPU移植代码和 UC/OS-II完全兼容，UC/OS-II的 CPU的移植代码不需要任何修改就可以应用到新内核上。在实际产品设计中在任务划分，模块设计，资源分配，实时性等方面和原内核相比表现出了明显的优越性。

参考文献：

[1] LABROSSE J J. 嵌入式实时操作系统 UC/OS-II [M]. 邵贝贝，译. 2版. 北京：北京航空航天大学出版社，2003.

[2] TANENBAUM A S, WOODHULL A S. 操作系统设计与实现 [M]. 陈渝，谌卫军，译. 2版. 北京：电子工业出版社，2004.

[3] SILBERSCHATZ A, BAER P, GAGNE G. 操作系统概念 [M]. 郑扣根，译. 6版. 北京：高等教育出版社，2008.

[4] 吴平. UC/OS-II中任务调度算法的改进 [J]. 单片机与嵌入式系统应用，2006(3): 23—26.

[5] 成后发，杨春金. UC/OS-II操作系统内核的改进 [J]. 通讯和计算机，2006, 3(6): 52—55.

[6] LABROSSE J J. UC/OS-II the real-time kernel V2.83 release notes [EB/OL]. [2008—08—11]. http://www.Micrium.com.

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net