FreeRTOS内存管理方案的分析与改进

刘 林[1],朱 青[1],何昭晖[2]
LIU Lin[1], ZHU Qing[1], HE Zhaohui[2]

1.湖南大学 电气与信息工程学院,长沙 410082
2.威胜集团有限公司,长沙 410205
1.College of Electrical and Information Engineering, Hunan University, Changsha 410082, China
2.Wasion Group Ltd., Changsha 410205, China

LIU Lin, ZHU Qing, HE Zhaohui. Analysis and improvement of memory management scheme in FreeRTOS. Computer Engineering and Applications, 2016, 52(13):76-80.

Abstract: The memory management shceme of Free Real Time Operating System(FreeRTOS) has some shortcomings, such as uncertain allocation time, frequent cutting, low utilization rate as well as inadequate consolidation mechanism, this paper adopts a strategy of "precised cutting" and "delayed merging" in order to reduce memory fragment and raise utilization rate furthest. Thus it introduces TLSF(Two-Level Segregated Fit) algorithm into FreeRTOS, using two levels of bitmap index to manage its dynamic memory and improving its process of memory allocation as well as release, then conducts the experimental test on the STM32 platform which ports with FreeRTOS. At last, the results show that this method improves the speed of memory allocation and reduces memory fragmentation rate.
Key words: FreeRTOS; memory management; Two-Level Segregated Fit(TLSF) algorithm; STM32 platform; memory fragment

摘 要: 针对FreeRTOS内存管理方案分配时间不确定,切割次数较多,利用率低及合并机制不足等缺点,采用一种"精确切割"和"延时合并"相结合的策略以最大限度减少内存碎片,提高内存利用率。具体实现方法是在FreeRTOS中引入TLSF(Two-level Segregated Fit)算法数据结构,采用二级位图索引对动态内存进行管理,并改进TLSF算法的内存分配和释放过程;最后将改进的算法以及FreeRTOS移植到STM32开发平台上进行实验测试。测试结果表明该方法提高了FreeRTOS的内存分配速度,减少了内存碎片率。
关键词: FreeRTOS操作系统; 内存管理; TLSF算法; STM32开发平台; 内存碎片
文献标志码: A **中图分类号:** TP316.2 doi:10.3778/j.issn.1002-8331.1407-0557

1 引言

FreeRTOS是一个轻量级的嵌入式实时操作系统,具有源码公开、可移植、可裁减的特点,它实现的任务没有数量限制,内核支持优先级调度和轮转调度算法,用户可以很方便地将其移植到各种嵌入式控制器上进行开发设计[1]。虽然FreeRTOS的内核较小但可提供比较齐全的功能,包括内存管理、时间管理、任务管理、消息队列、信号量、记录功能等,基本上能够满足微型嵌入式实时系统的设计需求[2]。在实际嵌入式产品开发中,为了降低成本,其资源通常是受限的,因此内存资源分配是嵌入式产品开发中需要考虑的主要问题之一[3]。FreeRTOS的最新版本中提供了四种内存分配方案,包括静态与动态内存分配方式,在固定内存分配过程中其效果都较好,但对不同大小的内存分配需求会产生较多的内存碎片,同时其分配时间也难以确定。

为了改善嵌入式实时操作系统的性能,提高其内存分配效率,研究者们提出了多种动态内存分配算法,文献[4-5]对其进行了深入研究。本文分析和比较了

基金项目: 国家高技术研究发展计划(863)(No.2011AA05A120)。
作者简介: 刘林(1988—),男,硕士,主要研究领域为电路与系统、嵌入式系统及应用,E-mail:1148530988@qq.com;朱青(1968—),女,博士,副教授,博士生导师,主要研究领域为现代网络与通信技术、数字信号处理;何昭晖(1976—),男,硕士,嵌入式软件工程师,主要研究领域为智能配电、物联网技术。
收稿日期: 2014-08-08 **修回日期:** 2014-10-09 **文章编号:** 1002-8331(2016)13-0076-05
CNKI网络优先出版: 2015-03-20, http://www.cnki.net/kcms/detail/11.2127.TP.20150320.1511.014.html

FreeRTOS现有内存管理方案的优缺点,同时引入了TLSF算法数据结构对FreeRTOS内存分配进行管理,并对TLSF算法没有利用近似内存重复分配的缺陷进行改进,最后将改进的算法以及FreeRTOS在STM32开发平台上进行内存管理实时性和碎片率实验测试,通过实验数据来验证改进后的内存管理实时性和碎片率性能。

2　FreeRTOS内存管理方案比较

FreeRTOS因其开源、免费、占用资源少等优势越来越受到开发者的亲睐。目前,FreeRTOS提供了四种内存分配方案对受限的嵌入式控制器内存资源进行管理,分别为heap1、heap2、heap3、heap4,其对比结果如表1所示[6]。

表1　FreeRTOS四种内存分配方案比较

C文件	优 点	缺 点
heap1.c	分配简单、时间确定	只分配、不回收
heap2.c	动态分配、最佳匹配	碎片、时间不定
heap3.c	调用标准库函数	速度慢、时间不定
heap4.c	相邻空闲内存可合并	碎片、合并效率低

其中heap1与heap3的分配过程相对较简单,heap2与heap4是两种常用的动态内存分配算法,两者内存分配原理类似,但又有细微差别。下面对heap2与heap4的内存分配与释放过程进行详细分析与对比。

Heap2的内存分配过程中如图1所示,它采用了最佳匹配算法,即当应用程序请求内存分配时首先找出空闲内存块链表中与请求大小最接近的空闲内存块,如果内存块刚好符合,则应用程序可以直接使用,如果内存块比请求的大,则将该内存块一分为二,一块提供给需求使用,另一块则重新插入空闲内存块链表中指定的位置,同时内存空间释放新产生的空闲内存块也会插入空闲内存块链表中,并且将整个链表中的空闲内存块按照顺序从小到大排列[7]。经过整个内存分配与释放过程后,从图1中D与A对比可以看出,整块大的空闲内存块被分为了三块按照从小到大顺序排列的小内存块,依此类推,当嵌入式系统频繁的创建与释放内存时,将会产生越来越多的小内存块,以致最终会产生较多的内存碎片。

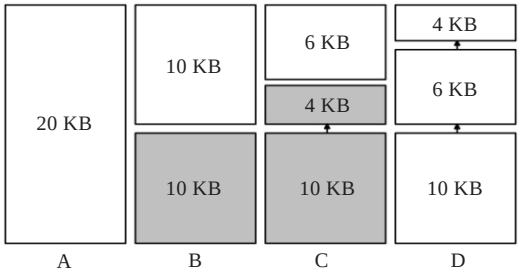

图1　heap2内存分配情况

Heap4是FreeRTOS开发团队在heap2的基础上改进的一种方案,主要是改进了空闲内存块链表的结构,即将原来以空闲内存块大小顺序排列的空闲内存块链表修改为按照以内存块物理地址大小进行排列[8]。如图2所示,heap4与heap2的分配过程基本一致,即当应用程序需要内存时,也是从空闲内存块链表中查找合适的内存块,如果有内存块分割,则将剩余的内存块插入空闲内存块链表中,只不过heap4分配的内存是按照物理地址从小到大的顺序排列,这样设计的目的是为了能够在释放过程中对相邻物理地址且空闲的内存块进行合并。从图2中D可以看出空闲的6 KB和10 KB内存块由于物理地址不相邻并不能合并,但是当分配的4 KB内存也释放后则会有4 KB与10 KB合并,然后再与6 KB合并从而最终达到E的效果。虽然从图1、图2的对比可知heap4的算法设计减少了大块内存的碎片化,但是当在嵌入式系统中频繁创建与释放内存导致空闲块物理地址相对离散时,同样也会产生较多内存碎片。

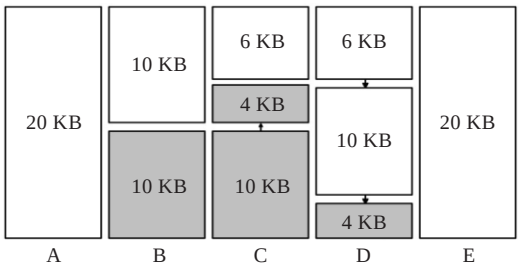

图2　heap4内存分配情况

3　TLSF算法的改进与实现

3.1　TLSF算法数据结构

TLSF算法使用离散隔离适应机制,它采用二级位图管理,是内存分配的一个最佳适应策略之一,其数据结构如图3所示。TLSF算法定义了两级链表结构,第一级链表是按照2的n次幂(n为5,6,…,31)形式将内存块进行划分,称为FLI(First-Level Segre-gated Fit);第二级链表是在第一级链表结构的基础上再进行线性划分,这个划分可以根据用户的需求决定分类的数目,并且每个类别都有与尺寸范围对应的内存块链表,每条链表对应一个相关的位图用来标记链表是否为空,用1表示链表非空即有空闲内存块可用,0则代表相反的情况,称为SLI(Second-Level Segretgated Fit)[9]。

如图3所示,TLSF算法主要由三个参数决定,FLI标识一级链表长度;SLI为二级索引,它将一级链表划分的内存区域按照线性比例再次划分,SLI的值可由用户设定,若SLI=4,则一级索引划分的内存区被划分成16段;MBS是TLSF定义的最小块大小,用户可根据实际需求设定,这里将其设定为16 Byte。假设用户通过TLSF算法请求160 Byte的内存块,由$160_D = 2^7 + 2*2^4$,则一级索引值为7,二级索引值为2[10]。

3.2　TLSF算法改进与实现

由上述分析过程可知,与FreeRTOS四种内存分配

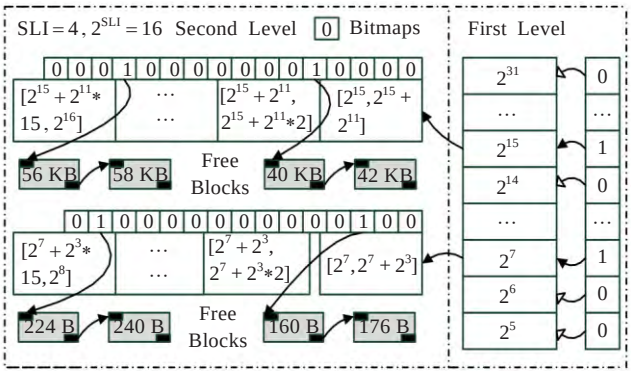

图3 TLSF算法数据结构图

算法相比,TLSF算法在实时性和碎片率方面有所改善,但在面对不同大小内存需求的嵌入式系统应用中,其实时性及内存碎片率均有进一步优化的余地[11]。TLSF算法在内存切割时采用的是“取下限”策略,即按照二级索引划分的内存块区间最小值进行切割,内存块释放后仍可以插入到原先的内存分区里,下一次再重新申请类似大小的内存时也可以再次从该内存区分配[12]。按照“取下限”策略能够保证实时性要求,但同时会造成大量的内部碎片,降低内存资源利用率。针对这种情况,本文对TLSF算法的内存分配与释放过程进行改进,并将改进的算法引入到FreeRTOS中称之为“heap5”。内存分配改进流程如图4所示,改进的原则是尽量减少内存切割碎片,即将空闲块链表中的内存块按照请求分配大小的内存块进行切割,从而可以最大限度提高内存利用率。具体实现的过程是在内存分配过程中首先判断内存是否需要切割,如果需要切割并且大于设定的切割阈值,则以“精确切割”作为内存块的分割条件,否则,按照内存块区间最小值进行切割。

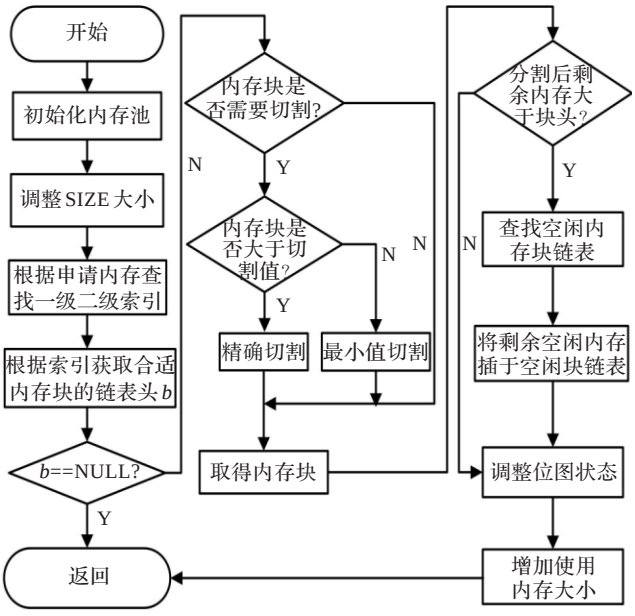

图4 TLSF内存分配改进流程

TLSF算法在内存释放时采用的是“立即合并”策略,即若释放的相邻物理地址内存块空闲,就进行“立即合并”[13]。“立即合并”策略增大了空闲内存块的大小,但同时在释放时也增大了内存开销,并且没有利用处理近似大小内存重复分配的特点,导致其性能下降,降低了实时性。内存释放改进流程如图5所示,即当释放的内存满足合并条件时并不进行立即合并,而是定义了一个合并阈值,并计算当前系统内存碎片率。比较当前系统内存碎片率与合并阈值的大小,当内存碎片率大于合并阈值时才对相邻的空闲内存块进行合并,否则只是更新空闲链表数据,这样延长了空闲内存块合并的时间,增大了重复利用相同空闲内存块的可能性。

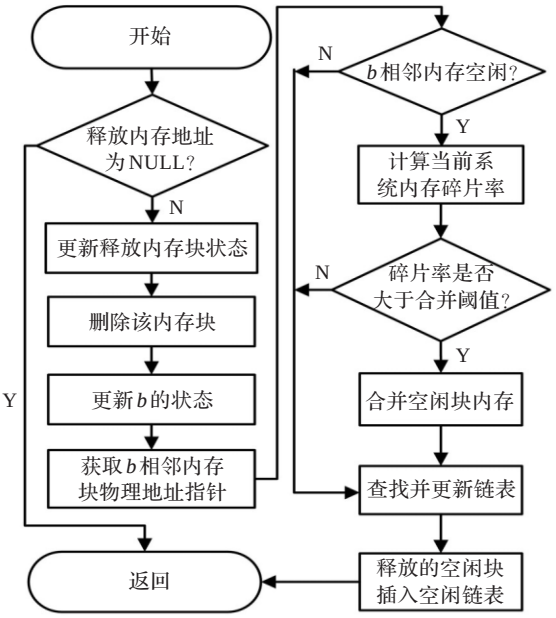

图5 TLSF内存释放改进流程

从上面改进的流程图4和图5可以看出TLSF算法没有循环递归,即它的内存分配和释放时间复杂度并没有随着空闲内存块的数量增加而变化,都是$O(1)$。同时采用“精确切割”的内存分配方式可以减少内存切割次数,从而减少了大块内存的碎片化,并最大限度地提高了内存利用率;“延时合并”策略减少了内存释放时间,在处理近似内存大小重复分配时具有更好的实时性。下面将整个改进的算法以及FreeRTOS移植到STM32开发平台上进行实验测试。

3.3 FreeRTOS的移植

将FreeRTOS移植到STM32开发平台上,需要修改实现与硬件相关的文件有portmacro.h、portasm.s、port.c。Portmacro.h主要定义临界区管理函数、内核调度函数、编译器相关的数据类型、架构相关定义等[14];Portasm.s主要完成开关中断、调用第一个任务、SVCALL服务函数以及PENDSV异常处理等事件;Port.c是C语言接口部分,用于实现任务堆栈初始化、系统节拍定时器的管理和任务切换请求等。将上面三个文件修改移植后,调用heap5的内存处理过程就可以进行实验测试。如图6所示为FreeRTOS在STM32开发平台的内核第一次启

动流程,其中Cortex-M3内核使用了双堆栈指针,图6中MSP为异常处理中断专用指针,PSP为任务指针。

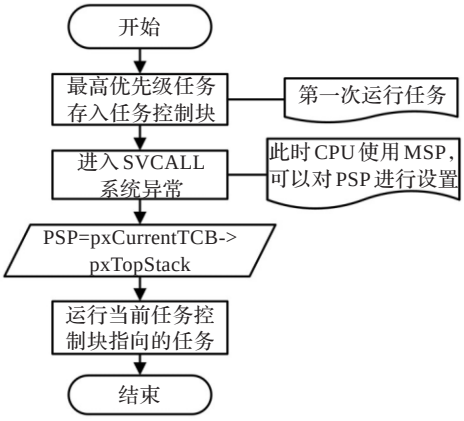

图6　FreeRTOS第一次内核启动流程图

4　实验设计和结果分析

4.1　实验设计

嵌入式实时系统动态内存分配算法主要的性能指标有三个方面:实时性、碎片率、可靠性。从前面理论分析可知改进的TLSF算法heap5比FreeRTOS现有的四种内存分配方案,在实时性和碎片率方面都有所改善。由于heap1只是简单的静态分配内存方式,故选取heap2、heap3、heap4与heap5在内存实时性和碎片率等性能指标方面行比较,设计如图7所示流程图的测试程序。

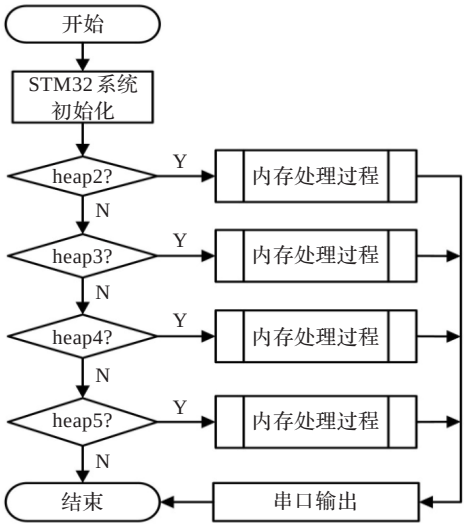

图7　测试程序流程图

如图7所示,本实验具体的硬件测试平台采用处理器主频配置为72 MHz,SRAM为64 KB的STM-32F107VCT6,软件环境是基于FreeRTOS内核的操作系统。测试主程序采用状态法设计,在每种内存管理算法间切换状态,状态内部之间进行如图8所示的内存处理过程。

如图8所示,内存分配与释放时间等性能采用STM32的滴答时钟配置编写的时间测量函数来测量,

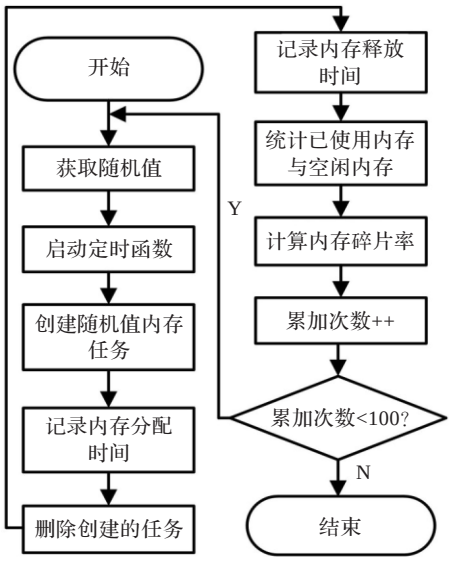

图8　内存处理流程图

它可以精确到微秒级,通过FreeRTOS频繁创建与删除任务来仿真内存分配和释放过程,并在内存分配和释放过程中分别计算出各种内存管理算法的碎片率,最终通过串口输出如图9所示的结果图。

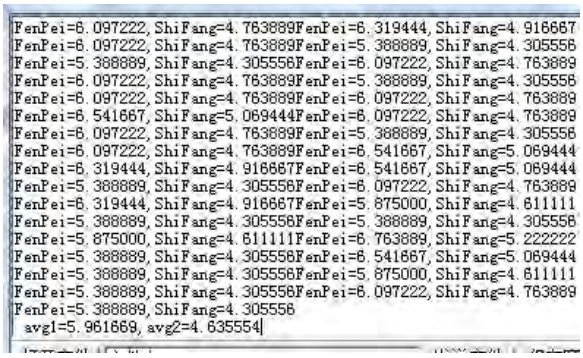

图9　实验结果图

4.2　实时性对比实验

实验将用户请求的内存大小分为6个区间(单位是Byte,序号为1,2,3,4,5,6),分别为[0,128],[128,256],[256,512],[512,1 024],[1 024,2 048],[2 048,4 096],测试程序在每个区间范围内产生随机值,并频繁地分配和释放随机值大小的内存。每组测试100次,调用设计的滴答时间测量函数,测量内存调用及内存释放时间(单位为μs),并取平均值,对比heap2、heap3、heap4、heap5的实时性能,测试结果如表2、表3所示。

表2　内存分配时间测试结果　　μs

算法	区间1	区间2	区间3	区间4	区间5	区间6
heap2	5.33	5.69	6.03	7.08	8.63	9.63
heap3	7.27	7.46	7.68	8.23	8.67	9.37
heap4	5.22	5.41	5.67	6.07	6.72	7.76
heap5	5.25	5.51	5.65	5.96	6.23	7.36

4.3　内存碎片率对比实验

在实时性实验中划分的区间范围内进行100次的

表3 内存释放时间测试结果 μs

算法	区间1	区间2	区间3	区间4	区间5	区间6
heap2	4.64	4.70	4.91	5.36	6.61	7.07
heap3	5.03	5.12	5.29	5.57	6.78	7.32
heap4	4.42	4.69	5.24	5.43	6.13	7.00
heap5	4.26	4.63	4.85	5.11	5.82	6.58

内存分配和释放请求,参照文献[15],内部碎片以 $(m-n)/n$ 来衡量,其中 m 代表动态内存申请大小,n 代表分配算法返回的空闲块大小。由其定义可知 $(m-n)/n$ 越大,表明内部碎片越大;$(m-n)/n$ 越小,表明内部碎片越小,当 $m=n$ 时,内部碎片为0。求取实际使用的最大内存和分配器所使用的最大内存的比值,计算并比较各种分配方案的内存碎片率,测试结果如表4所示。

表4 碎片率测试结果 %

算法	区间1	区间2	区间3	区间4	区间5	区间6
heap2	6.1	6.6	7.5	8.5	9.7	10.5
heap3	6.2	6.7.	8.0	8.7	9.9	10.6
heap4	5.9	6.4	6.9	8.3	9.4	9.8
heap5	5.2	5.3	6.1	7.5	7.9	8.2

4.4 实验结果分析

由于heap5采用的是二级隔离链表管理,时间复杂度为常数,从实验测试结果表2、表3可以看出,它相对于采用单链表管理的内存分配算法整体实时性能要好。从表2中heap4与heap5的内存分配时间对比可以看出前4个区间的时间差别很小而在区间5和6的差别逐渐明显,这是因为heap4需要执行搜素操作,并且随着内存块大小的增加所消耗的时间也增加,heap5改进的主要原则是尽量不切分整块内存,而更多地切分释放后合并的内存块,因此heap5在小内存分配过程中与单链表算法实时性相当,但在大内存分配时,实时性能改进效果明显。在内存释放过程中,从表3中可以看出由于heap4与heap5具备内存合并机制比其他算法释放效果整体性能要好,虽然在这里两者测试的效果相差不大,这其实是与实际应用中heap5的合并阈值选取有关。在这里两者的差别主要体现在heap4采用的是对相邻空闲块进行“立即合并”,不能实现部分空闲块内存能够重复分配与利用,改进后的heap5采用的是“延时合并”减少了内存切割次数,从区间4和区间5的内存碎片率测试结果可以看出,改进后的内存碎片率明显减少。

5 结束语

本文分析了FreeRTOS现有的几种内存管理方案,比较并总结了其优缺点,在此基础上引入了改进后的TLSF算法,采用一种“精确切割”和“延时合并”策略改进了其内存分配和释放过程。实验测试结果证明改进的内存管理算法提高了内存分配的实时性,减少了内存碎片率,将会为实际资源受限的嵌入式系统开发提供很大的应用价值。

参考文献:

[1] 李志明,檀永,徐石明.STM32嵌入式系统开发实战指南[M].北京:机械工业出版社,2013.

[2] 吕成兴,刘军礼,刘波,等.基于Contex-M3和FreeRTOS的数据采集系统设计[J].中国水运,2011,11(12):86-87.

[3] 刘忠仕,戴金海,桂先洲.实时操作系统SACOS的内存管理[J].计算机工程与应用,2006,42(5):37-39.

[4] 姜艳,曾学文,孙鹏,等.实时嵌入式多媒体系统模糊阈值合并内存管理算法[J].西安电子科技大学学报,2012(5):174-180.

[5] Soto M,Rossi A,Sevaux M.Two iterative metaheuristic approaches to dynamic memory allocation for embedded systems[J].Computer Science,2011,6622:250-261.

[6] 杨继兰,白丽娜,董渊.嵌入式实时操作系统FreeRTOS在x86上的移植[D].西安:西安电子科技大学,2011.

[7] 刘滨,王琦,刘丽丽.嵌入式操作系统FreeRTOS的原理与实现[J].单片机与嵌入式应用,2005(7):8-11.

[8] 张龙彪,张果.基于实时操作系统FreeRTOS的LwIP协议的移植研究[D].昆明:昆明理工大学,2013.

[9] 郭振宇,桑楠,杨霞.一种嵌入式系统内存管理的延迟合并伙伴机制[J].电子科技大学学报,2007,36(3):555-558.

[10] 闫广明.嵌入式系统内存空间域隔离技术的研究与实现[D].哈尔滨:哈尔滨工程大学,2011.

[11] 李志军,王铮,王帅.嵌入式系统的自适应动态内存分配算法[J].计算机工程,2007,33(20):91-93.

[12] 吴文峰.嵌入式实时系统动态内存分配管理器的设计与实现[D].重庆:重庆大学,2013.

[13] Masmano M,Ripoll I,Crespo A,et al.TLSF:A new dynamic memory allocator for real-time systems[C]// Proceedings of the 16th Euromicro Conference on Real-Time Systems(ECRTS'04),2004.

[14] 王兵,杨光永.FreeRTOS在水质监测无线传感器网络中的移植与改进[D].昆明:云南民族大学,2012.

[15] Zorn B,Grunwald D.Evaluating models of memory allocation[J].ACM Transactions on Modeling and Computer Simulation,1994,4(1):107-131.