基于 μC/OS-II 的双消息队列中断管理系统设计实现

宋丽华，张秋娟+

（北方工业大学 信息工程学院，北京 100144）

摘 要：μC/OS-II 操作系统没有中的断管理模块，该情况严重影响了其可移植性，针对该问题，设计了一种基于双消息队列的分层中断管理系统，提出了一种分层管理的中断管理设计思想，将中断处理分为紧要部分和可推迟两部分。紧要部分优先处理，满足实时性要求；可推迟部分利用 μC/OS-II 的消息队列机制，实现了具有优先级双消息队列的可推迟处理机制。实验结果表明，该系统比同等条件下的 Linux 系统具有优越的实时性，平均提高了 51%，为工业和航空等嵌入式系统等提供了实时性的解决途径。

关键词：μC/OS-II；中断管理；双消息队列；分层管理；紧要部分；可推迟部分；实时性

中图法分类号：TP302.1 **文献标识号**：A **文章编号**：1000-7024（2013）07-2377-07

Design and realization of double message queue in interrupt management system for μC/OS-II system

SONG Li-hua, ZHANG Qiu-juan+

(College of Information Engineering, North China University of Technology, Beijing 100144, China)

Abstract: The μC/OS-II's portability is blocked seriously by non-interrupt management. To solve this problem, a standpoint of a interrupt management system (IMS) is proposed, which adds IMS into μC/OS-II. Double message queue for IMS (DMQIMS) is proposed. This IMS is divided into a critical part (CP) and a delayed part (DP) with hierarchical thinking. CP will be processed preferentially, which can meet the need of real-time performance; And DP achieves deferred processing mechanism, using the priority dual message. Experiments show that excellent real-time operating is showed in DMQIMS, and the real-time performance has increased by average 51%, which is compared to the same conditions Linux system. Finally, the DMIMS can also be used in industry or aviation and so on embedded system.

Key words: μC/OS-II; disruption management; double message queue; hierarchical management; critical part; delayed part; real time

0 引 言

μC/OS-II 是一个实时操作系统，它以开源、短小精悍、稳定性高等优点得到广泛应用。但其是一个裸核操作系统，只提供一些任务调度和任务间通信等基本功能，没提供设备管理、定时器管理、中断管理等机制，还算不上一个完整的操作系统。外围设备与 CPU 交换数据一般由于速度不对等，经常需要使用中断机制来解决此问题。中断机制关系到一个系统实时性和稳定性的重要因素，所以设计实现一个可移植性好、稳定可靠地的中断管理系统是一个完整系统不可或缺的。目前，Linux 在嵌入式领域得到广泛应用，它可移植性强，已经广泛应用于多种处理器平台，但它不是实时操作系统，本文将对 μC/OS-II V2.76 添加中断管理系统，从而构建基于 μC/OS-II 的中断管理系统。在此之前，做几点限定：①本系统不支持多核处理器，简化掉与多核处理器处理相关的内容；②本系统都是在管理模式下运行，中断都是发生在特权模式下。

1 相关研究

中断处理方法中包括非嵌套中断处理、嵌套中断处理、可重入中断处理等多种方法。对于常见的以上 3 种对其优缺点比较如下：

收稿日期：2012-10-17；**修订日期**：2012-12-29

基金项目："十一五"国家科技支撑计划重点基金项目（2009BAI71B02）；北京市属高等学校人才强教计划基金项目（PHR201007121）；2011 年度市教委科研计划面上基金项目（KM201110009002）

作者简介：宋丽华（1979-），女，河北邯郸人，博士，讲师，研究方向为嵌入式技术、网络协议、VLSI 设计；+通讯作者：张秋娟（1985-），女，河南新乡人，硕士，研究方向为嵌入式技术。E-mail: zhqj06dianzi@163.com

· 非嵌套中断，即顺序地处理和服务各个中断，该方式处理简单，稳定性好，但实时性不好，对于高优先级的任务不能及时的得到执行。

· 嵌套中断，即处理多个没有优先级的中断，该方式实时性高，但中断嵌套复杂化，稳定性不高。

· 可重入中断处理，即处理多个可以有优先级的中断。该方式实时性更高，能根据优先级处理中断。但处理过程会变得复杂。

但是对于工业控制领域，实时性和稳定性的要求极为重要，所以实时性得以提高的可重入中断处理在该领域应用极为广泛。本课题的框架也主要是围绕该种处理方法进行设计与实现的。

2 系统中断管理与设计

基于以上的分析，为了满足实时性的要求，本文将从两个方面改善可重入中断系统的实时性，其一是短中断响应时间，其二是分割中断服务程序，将复杂的中断服务程序分割成紧要、可推迟执行两部分，紧要部分完成一些紧要且耗时较短的任务，在中断级进行处理

如图 1 所示，紧要部可分为中断初始化、中断注册、中断处理 3 个部分。可推迟执行部分主要完成比较耗时且不是很紧要的任务，本课题将其放在任务级进行处理，并不占用中断时间。该部分根据紧要部分的中断源的优先级，将消息发送到对应的消息队列中，从而执行对应的中断处理。

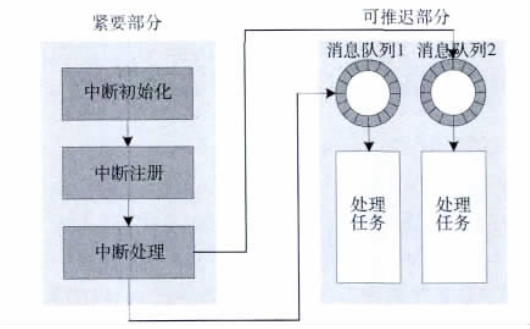

图 1 中断系统架构设计

2.1 紧要部分

2.1.1 紧要部分架构设计

本节将详细介绍中断的紧要部分，如图 2 所示将从中断的初始化、注册、处理来介绍。

在中断初始化部分，先进行有关体系结构相关的中断初始化，然后重定位异常向量表和处理程序，最后初始化中断描述符。

中断注册主要包括申请中断线、分配中断资源和向中断描述符注册中断服务程序。注册的主要任务就是把相应设备的中断处理程序添加进系统，以便在中断发生时调用

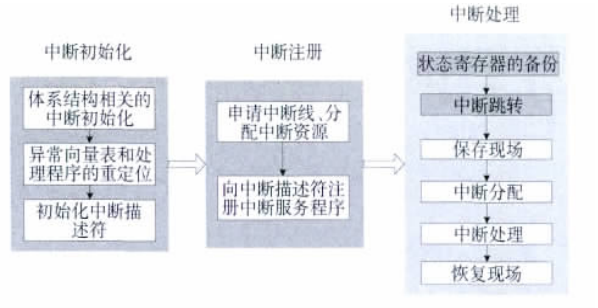

图 2 中断紧要部分

相应的中断处理程序。向中断描述符注册中断服务程序是把中断处理程序注册到中断描述符中。

在中断处理部分可以分为硬件处理和软件处理。在硬件处理部分主要完成状态寄存器的备份和中断跳转。在软件部分主要分为保存现场、中断分配、中断处理和恢复现场。

2.1.2 紧要部分核心数据结构

struct irq _desc 中断描述符，每一个 IRQ 都有对应一个 irq _desc 类型的结构体入口。

```
struct irq _desc
{
    unsigned int          irq;      // interrupt number for this descriptor
    q _flow _handler _t   handle _irq;    // highlevel irq-events handler
    struct irq _chip   * chip;    // low level interrupt hardware access
    void   * handler _data;    //   per-IRQ data for the irq _chip methods
    struct irqaction   * action;    /* IRQ action list */
    unsigned int   status;    /* IRQ status */
    unsigned int   depth;    /* nested irq disables */
    unsigned int   wake _depth;    /* nested wake enables */
    unsigned int   irq _count;    /*
};
```

2.2 可推迟部分

2.2.1 可推迟部分框架设计

在可推迟部分利用两个不同优先级的任务来专门读取对应消息队列的值。而不同优先级的两个任务又可以保证耗时可推迟部分也是有轻重缓急来处理的，这样可以很好地保持系统的实时性。每一个消息队列均是一个循环缓冲区，里面的每一个单元都是一个指针，用来指向不同的中断处理函数。μC/OS-II 系统可以管理多达 64 个任务，除了保留 4 个最高优先级和 4 个最低优先级的任务外，用户可

以使用的任务多达 56 个。我们将这两个专门处理中断的任务的优先级定为 8 和 40，可推迟部分按照任务的优先级来执行任务。具体的执行任务是首先检查消息队列中是否有消息，当有消息时，以 FCFS 方式依次执行各消息队列中所对应的任务。当没有消息时，任务挂起。当有新消息时，该任务被恢复，重新执行。

如图 3 所示，在紧要部分主要完成构造、初始化和发送 MSG。需注意 μC/OS-II 与其它系统有所不同，μC/OS-II 在初始化过程中，先通过建立内存分区，然后在已经建立的内存分区上，申请一块所需要的内存块，并将指针指向该空间，这样种内存的申请与释放机制很好的解决了内存碎片，从而给程序的稳定运行提供了保证。初始化 MSG，就是定义 MSG 类型的变量。发送 MSG，即根据中断的优先级将 MSG 的指针发送到消息队列 1 或 2 中。

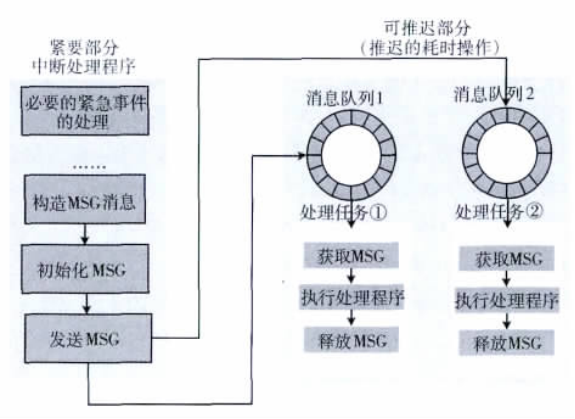

图 3 中断处理框架

在可推迟部分，μC/OS-II 内核以 FCFS 方式依次查询消息队列中的每一单元，每一单元均是挂载 MSG 中对应的处理程序。该处理程序的函数指针为 handler，其函数参数为待处理的数据，这样处理函数通过 MSG 获得所需要的参数后，开始执行对应的处理程序。具体的执行流程为首先获取 MSG 的值，根据其值调用相应的执行处理程序，最后将释放 MSG 的内存块。

2.2.2 可推迟部分核心数据结构

首先，本系统设计中核心的数据结构是消息（MSG），它的声明如下：

```
typedef struct  MSG {
    irq_handler  handler;                    //irq handler
    void *data;     // per-IRQ data for the MSG
    int  flag;
```

其中中断处理函数的定义如下

```
    typedef  void (*irq_handler) (int, void *);
```

*data 为待传输数据的指针，可初始化为待传输数据的首地址；flag 为标志位，是为了扩展用的。

3 系统的实验测试与验证

本系统的平台是基于一款工业监控系统，它的 CPU 是用基于 XScale 架构的 PXA270，我们的操作系统是 μC/OS-II2.76 版本的一个测试平台。将本课题提出的中断框架系统图移植到 μC/OS-II 系统之上，在系统的软、硬件平台搭建完成后，需要将本课题提出的框架在试验机上进行验证，并选取具有相同的硬件平台的 LINUX2.6.36 作为参照平台。实验结果表明文中设计的中断管理的方案，不但在实验条件下的测试结果性能优异，而且具有很强的工业使用价值。

在工业监控领域，系统的实时性等的重要性都是无可置疑的。在一个实时系统中，中断影响系统实时性的指标一般有：中断响应时间、中断处理时间、中断恢复时间。其中，中断响应时间可以根据中断源的优先级来确定中断时间。中断处理时间和中断退出时间一般要根据需求对不同的外设做不同的中断处理程序，所以该时间具有不可预测性。所以本实时性测试方案从中断响应时间入手。通过分析不同优先级的中断响应时间，来统计本中断框架的中断实时性。

3.1 实时性测试方案

根据以上分析，本文对实时性改善的指标为中断响应时间。操作系统的中断响应流程可分为：系统延迟时间（Tb）和保存现场时间（Tc）和中断分配程序时间（Td）。

其中，Ty（1）为最长的中断响应时间为从中断发生到开始执行用户的中断服务子程序代码的时间；Tb（0）则定义为系统中任务级临界区的时间；Tc（1）为保存 CPU 内部寄存器的最长时间；Td（1）为中断 1 最长的跳转时间。

系统的最高优先级中断 1 的最长中断响应延迟如图 4 所示。当中断 1 产生时，系统刚好进入临界区（Critical），经过 Tb（0）时间以后，系统从临界区退出（图 4 中点 B）；紧着系统开始保存中断前的信息，经过 Tc（1）时间段后（图 4 中点 C），保存现场信息结束；处理器开始响应中断，经过操作系统中断跳转程序的选择，在图 4 点 D 时刻，开始中断处理子程序。则中断 1 的最长中断响应延迟 Ty（1）可以由下式所示

$$Ty(1) = Tb(0) + Tc(1) + Td(1) \quad (1)$$

但在图 4 只是显示了最高优先级的中断响应过程，对于低优先级的中断处理过程可如下描述：当低优先级中断发生时，要先判断此时是否有高优先级的中断发生，当有高优先级的中断发生时，要先进行高优先级的中断响应、处理和恢复的操作，然后才能响应低优先级的中断。

假设实时系统中有 n 个中断，其优先级从高到低的排列顺序为（中断 1、中断 2……中断 n），且中断 k（1≤k<n）能且仅能抢占比它优先级低的中断（即，中断 k+1～中断 n）。

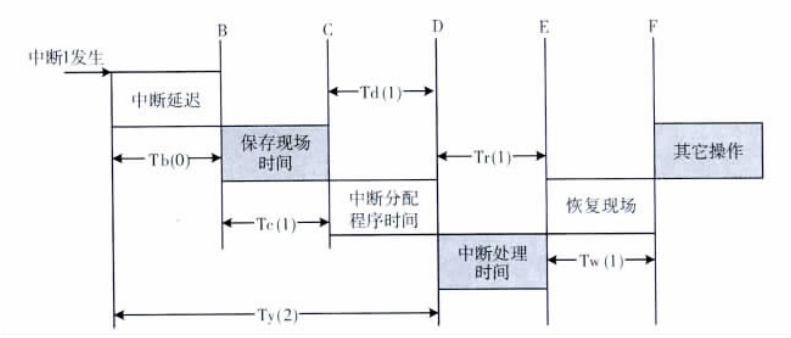

图 4 最高优先级中断的最长中断响应延迟

针对次优先级的中断 2 的最长响应延迟时间如图 5 的情况所示。当中断 2 产生时，系统刚处于临界区，当系统未退出临界区之前，有更高优先级的中断 1 产生。当系统从临界区退出后，则首先响应中断 1，由于优先级关系中断 1 优先执行。到中断 1 执行完毕后（图 5 点），中断 2 开始响应，后面的处理则与中断 1 类似。可见中断 2 最长响应延迟

$$Ty(2) = Tb(0) + Tc(1) + Td(1) + Tr(1) + Tw(1) + Td(2) \quad (2)$$

式中：Tr（1）——中断 1 的最长处理时间；Tw（1）——中断 1 的最长返回时间；Td（2）——中断 2 最长跳转时间。换一个角度，从中断 2 的响应时间看，若定义 Tx（1）为中断 2 的最长临界区时间。则 Ty（2）还可以表述为

$$Ty(2) = Tx(1) + Td(2) \quad (3)$$

而 Tx（1）则为

$$Tx(1) = Tb(0) + Tc(1) + Td(1) + Tr(1) + Tw(1) \quad (4)$$

采用递归方分析，中断 k 的最长中断响应时间如图 6 所示。其中，定义 Tx（k-1）为系统对中断 k 的中断延迟时间；Tc（k）为中断 k 最长的保存现场时间；Td（k）为中断 k 的最长中断；Ty（k）为中断 k 的最长响应延迟时间；Tr（k）为中断 k 的最长处理时间；Tw（k）为中断 k 的最长返回时间。则 Ty（k）可以表述为

$$Ty(k) = Tx(k-1) + Tc(k) + Td(k) \quad (5)$$

其中 1≤k≤n。而 Tx（k）为

$$Tx(k) = Ty(k) + Tr(k) + Tw(k) = Tb(0) + Tc(k) + Td(k) + Tr(k) + Tw(k) \quad (6)$$

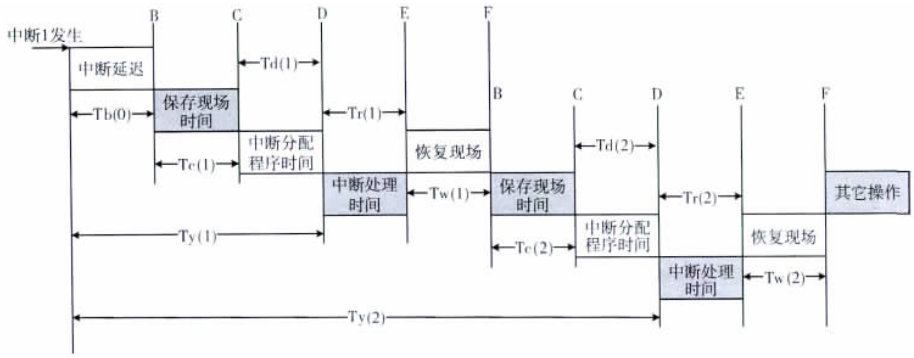

图 5 次优先级中断的最长中断响应时间

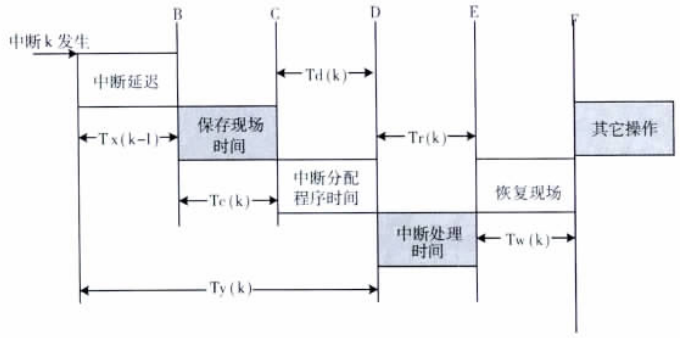

图 6 优先级为 k 的最长中断响应时间

最终得到中断 k 的最长响应时间公式

$$Ty(k) = Tb(0) + \sum_{i=0}^{k}[Tc(i) + Td(i)] + \sum_{i=0}^{k-1}[Tr(i) + Tw(i)] \quad (7)$$

由上式可见，若要缩短中断 k 的最长响应延迟时间，则需要从 Tb（0）、Tc、Td、Tr 和 Tw 这 5 个时间入手：

（1）Tb（0）中断延迟的最长时间，对于特定的系统，该时间的长短往往对共享数据采取的不同策略。最快捷的方法是关中断和开中断。

（2）Tc 是保存现场所需要的时间。即保存 CPU 寄存器的时间，对于特定的平台所用的时间是一定的。

（3）Td 是中断分配程序运行时间：中断向量表到执行具体的中断服务子程序之间的时间，该时间和具体的中断分配程序的框架的复杂程度相关。

（4）Tr 和系统的中断处理子程序复杂性相关。这里的中断处理是指我们前面提到的 框架的紧要部分的紧急处理，至于可推迟部分的中断处理是任务级的，故不计。

（5）Tw 是系统最长的中断恢复时间。对于可剥夺型内核，中断的恢复会面临一个判断高优先级中断是否就绪的过程。如图 6 所示，则执行高优先级中断，只有当所有高优先级中断执行完后，才能使低优先级任务恢复。否则，直接低优先级任务恢复。由以上分析可知 Tw 被中断的次数相关。

3.2 实时性测试结果和分析

在 C/OS-II 中，中断是响应外部的异步事件，从而引起操作系统发生任务调度的基本触发条件。因此，中断响应延迟时间，是衡量系统实时性的重要指标。

鉴于前面的分析可知系统响应时间并不是一个常量。在系统工作的各种不同条件下，其时间特性并不确定。如果采用软硬件结合通过示波器测量周期量的方法，只能大体上得到系统的平均实时性能，并不能反应系统工作的所有细节，也不能得到中断响应延迟的分布规律，更难测到最长中断响应时间。为了横向比较移植过该中断框架的 μC/OS-II 与标准 Linux 系统的实时性，本课题将采用统计学方法，对系统中断响应延迟的分布规律加以分析。

实验使用 PXA270 处理器上的定时器中断作为测试对象，通过软件记录定时器中断的中断响应延迟，最后做出统计分析。PXA270 处理器的定时器要有两个寄存器组成：

系统时钟计数寄存器（OS timer count register，OSCR），它是一个计数器，只要定时器开启，OSCR 就以 3.25MHz 的时钟频率计数。

系统时钟匹配寄存器（OS timer match register，OSMR）。每当 OSCR 的数值等于 OSMR 的值时候，定时器就会向处理器产生中断请求。

如图 7 所示，OSMR 事先被设定为 3，OSCR 从 1 开始计数，经过两个周期以后的 A 时刻 OSCR 值为 3，这时定时器中断产生，经过一段时间以后，系统跳转到定时器的中断处理子程序。在此中断处理子程序的起始（B 时刻），本次定时器中断响应延迟时间可以通过计算当前的 OSCR 与 OSMR 的差值得到——若使用微秒单位则计算公式表述为（OSCR-OSMR）/3.25。然后，在内存中记录所得到的差值。最后，设定 OSMR 为 OSCR ＋cycle，即经过 cycle 个 OSCR 计数的周期后，定时器中断会再次发生。在所有的测试环境中，设定 cycle 为 3.25MHz＊0.01s＝32500，即每隔 10 毫秒将产生一次定时器中断。如此反复，每次产生的中断延迟都会被系统记录。该方法的测量精度主要由两个因素决定：

（1）系统定时器计数周期，即 308 纳秒；

（2）在定时器中断程序入口，读取 OSCR 的时间。根据 ARM 体系结构，这需要执行两条指令（得到 OSCR 的地址、读取 OSCR 的数值）。

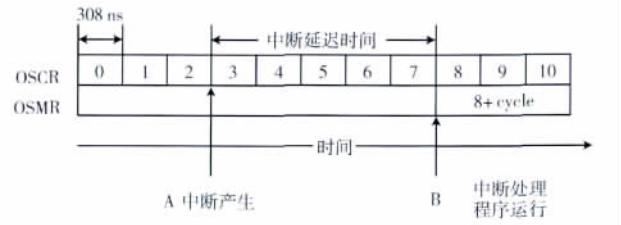

图 7 中断响应时间统计

对于工作在 520MHz 主频的 PXA270 处理器，1 为决定性因素，即可认为该方法测试中断响应延迟时间，分辨率为 308 纳秒。在 μC/OS-II 平台上，上述定时器中断处理函数被注册为实时中断。而在 Linux 系统上通过标准的中断注册函数（request_irq）注册。为便于测量和记录，本文编写了各自平台的驱动程序并注册为标准的 Linux 字符设备。这使得在 Linux 的应用程序空间可以实现控制（开启、关闭）定时器中断；输出或者清除中断延迟时间的记录等功能。

在对比试验中，使用的被测试系统有如下 2 种：标准的 Linux 系统，在后文测试统计图中简记为 Linux；移植中断框架后的 μC/OS-II 系统，在后文测试统计图中简记为 μC/OS-II。

所有的系统均运行在相同的硬件平台上，使用相同的测试软件环境，故测试结果具有横向可比性。对于中断响应的实时性测试，为了比较在以上两种环境下的性能，本课题构造了两种典型的应用环境。对于每种环境都运行 20 分钟，并分以空闲任务和运行所有任务两种典型的应用对中断响应时间的记录进行统计分析。

横坐标为中断延时时间，单位为微秒，为了便于观察数据的走势，这里使用 2 的指数坐标；纵坐标为中断延时百分比。例如坐标（1.85，98.0%）则表示在第 1.85 微

秒，有 98%的中断延时。从 0 微秒到延时分布达到 100%的时间区域为平均中断响应时间。由前面的内容知道，中断响应时间越短，说明系统的实时性越好。

（1）空闲任务时：空闲任务即除了系统以外的基本程序以外，不运行任何应用程序情况下的测试结果。该情况是考察系统在极低负载的情况下其的实时性。如图 8 所示，在 1.85 微秒时，μC/OS-II 的中断延时略短于 LINUX，其中，μC/OS-II 的中断响应时间为 12.14 微秒，而 LINUX 的中断响应时间是 25.56 微秒；具体对比结果如图 8 所示，在中断延时最长时间方面 μC/OS-II 要比 LINUX 好。这一方面是鉴于 μC/OS-II 内核为抢占式内核，中断延时小；另一方面虽然经削弱实时性，但 μC/OS-II 实时性依然比 LINUX 要好，从而验证了我们。

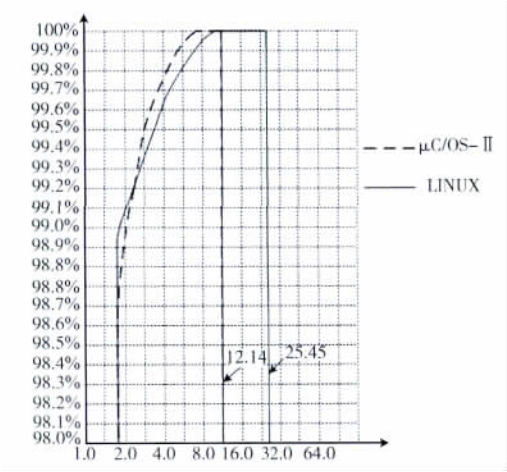

图 8 空闲任务时中断延时比较

实时性提高了（25.45－12.14）/25.45＊100%＝52.30，故实时性提高了 52.30%。

（2）所有任务运行时：本情况主要说明本系统在有非实时型任务运行时，实时中断的延迟情况。如图 9 所示，对 μC/OS－II 系统，98%的中断都小于等于 8.53 微秒，其中断响应时间为 58.15 微秒；对 LINUX 系统，98%的中断延时都小于等于 10.29 微秒，其中断响应时间为 98.23 微秒。延时该结果验证了经移植后的 μC/OS-II 在多任务调度运行的情况下，对系统实时中断的保证情况。

实时性提高了（98.23－48.34）/98.23＊100%＝50.79%。

4 结束语

本文在 μC/OS-II 系统上设计和实现中断管理系统，针对对紧要部分处理程序的简化与精简，从而大大缩减了中断响应时间。在中断的可推迟部分又提出利用两个消息队列来处理不同优先级的中断处理，从而能很好的处理可推迟部分。目前本中断管理机制在 ARM 平台上得到很好的

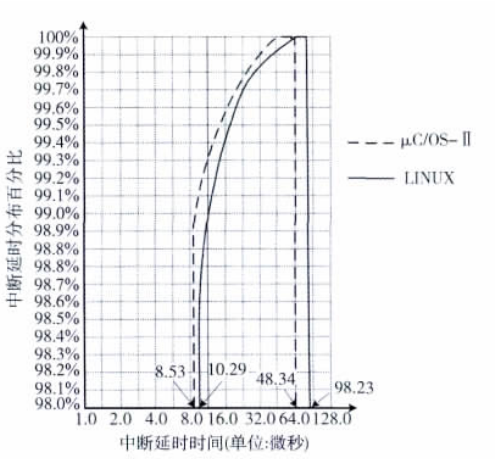

图 9 所有任务运行时中断延时比较

移植和应用，但还未在其它硬件平台和操作系统下进行移植和验证性实验，这将是下一步工作的重点。

参考文献：

[1] Wald Nicholas J，Bestwick Jonathan P，Huttly Wayne J. Effect of interrupting prenatal down syndrome screening due to a large nuchal translucency [J]. Prenatal diagnosis，2012，32 (7)：655-661.

[2] Bertogna M，Cirinei M，Lipari G. Schedulability analysis of global scheduling algorithms on multiprocessor platforms [J]. IEEE Transactions on Parallel and Distributed Systems，2009，20 (4)：553-566.

[3] YAN Liping，SONG Kai，DENG Hubin. Research on improvement of real-time capacity of embedded-based Linux kernel [J]. Compu-ter Engineering and Design，2010，32 (1)：121-125 (in Chinese). [严丽平，宋凯．邓胡滨．基于嵌入式应用的 Linux 内核实时性改进研究 [J]．计算机工程与设计，2010，32 (1)：121-125.]

[4] LI Yan，WANG Xianshan. Hardware implementation of real-time operating system task scheduling algorithm [J]. Computer Engineering and Applications，2010，46 (35)：52-56 (in Chinese). [李岩，王显山．实时操作系统任务调度算法的硬件实现 [J]．计算机工程与应用，2010，46 (35)：52-56.]

[5] MEI Hong. Design of real-time distributed interrupt system based on spacewire [J]. Aerospace Control，2011，29 (2)：93-97 (in Chinese). [梅洪．基于 Spacewire 的实时分布式中断系统设计 [J]．航天控制，2011，29 (2)：93-97.]

[6] Shaopo Wang，Jingjie Yu，Tianlan Wei，et al. Applying real-time control for achieving nitrogen removal via nitrite in a lab-scale CAST system [J]. More Information Environ Technol，2012，33 (10)：1133-1140.

[7] Sun SY，Qin HB，Cui JD，et al. Porting μC/OS-II to cortex-M3 and optimization [J]. Computer System and Application，

2010，19 (4)：208-211.

[8] Chang Wen-Yi，Tsai Whey-Fone，Lai Jihn-Sung，et al. Development of a real-time monitoring system as a decision-support system for flood hazard mitigation [J]. Journal of the Chinese Institute of Engineers，2012，35 (7)：827-840.

[9] José Ricardo da Silva Junior，EstebanW Gonzalez Clua，Anselmo Montenegro，et al. A heterogeneous system based on GPU and multi-core CPU for real-time fluid and rigid body simulation [J]. International Journal of Computational Fluid Dynamics，2012，26 (3)：193-204.

[10] Keqiu Li. A method to improve interrupt latency in real-time OS kernels [J]. Journal of Embedded Computing，2010，4 (1)：37-45.

[11] Sun SY，Qin HB，Cui JD，et al. Porting μC/OS-II to cortex-M3 and optimization [J]. Computer System and Application，2010，19 (4)：208-211.

[12] Wischik D，Handley M，Braun M B. The resource pooling principle [J]. SIGCOMM Comput，2008，38 (5)：47-52.

(上接第 2327 页)

[14] Cheng Jen-Po，Lee Chia-han，Lin Tzu-Ming. Prioritized random access with dynamic access barring for RAN overload in 3GPP LTE-A networks [C] //Houston，Texas，USA：GLOBECOM Workshops，2011：368-372.

[15] Man Hon Cheung，Mohsenian-Rad H，Wong V W S，et al. Utility-optimal random access for wireless multimedia networks [J]. IEEE Wireless Communications Letters，2012，1 (4)：1-4.

[16] TS 25.331，Radio resource control (RRC) protocol specification [S].

[17] Sakakibara K，Sasaki M，Yamakita J. Backoff algorithm with release stages for slotted ALOHA systems [J]. ECTI Transactions on Electrical ENG，Electronics，and Communications，2005，3 (1)：553-558.

[18] Rivero-Angeles M E，Lara-Rodriguez D，Cruz-Perez F A. Access priority for throughput sensitive and delay sensitive users in S-ALOHA using different backoff policies [C] //Dallas，TX，USA：Vehicular Technology Conference，2005：201-205.

[19] Mario E Rivero-ángeles，Felipe A Cruz-Pérez. Differentiated backoff strategies for prioritized random access delay in multi-service cellular networks [J]. IEEE Transitions on Vehicular Technology，2009，58 (1)：381-397.

[20] Dehbi Y，Mikou N，Benaboud H. An analytical study of mixed backoff schemes for QoS differentiation in wireless LAN [C] //Moroccan：International Conference on Multimedia Computing and Systems，2009：187-192.

[21] LU Xianqi，ZHOU Wen'an，ZHU Chaoping，et al. A random access scheme based on CoMP [J]. Telecommunication Engineering，2012，52 (5)：619-623 (in Chinese). [**卢宪祺，周文安，朱超平，等. 一种基于 CoMP 的随机接入方案 [J]. 电讯技术**，2012，52 (5)：619-623.]

[22] JIANG Qing，LIU Zhangmao，XU Zewen，et al. Research on priority backoff algorithm of TD-SCDMA trunking system [J]. Application Research of Computers，2012，29 (4)：1500-1503 (in Chinese). [**蒋青，刘彰茂，许泽文，等. TD-SCDMA 集群系统中优先级退避算法研究 [J]. 计算机应用研究**，2012，29 (4)：1500-1503.]

[23] TS36.300，Evolved universal terrestrial radio access (E-UTRA) overall description [S].

[24] WU Renyong. Study on call admission control and intelligent resource allocation for QoS support in wireless mobile networks [D]. Wuhan：Huazhong University of Science and Technology，2006 (in Chinese). [**伍仁勇. 支持 QoS 的无线移动网络呼叫接入控制和智能资源分配研究 [D]. 武汉：华中科技大学**，2006.]

[25] HUANG Fu，WANG Gaocai，LAI Mingxing. Study on real-time QoS based on priority schedulability in multimedia networks [J]. Computer Engineering and Applications，2011，47 (34)：107-110 (in Chinese). [**黄福，王高才，赖明星. 多媒体网络中基于优先级调度的实时 QoS 研究 [J]. 计算机工程与应用**，2011，47 (34)：107-110.]