摘 要

嵌入式系统是当前微电子技术和计算机技术中最热门的概念之一。而嵌入式操作系统则是嵌入式系统技术中最重要的一个组成部分。它整合了操作系统、网络、多媒体、文字信息处理等多项软件技术，极大地推动了嵌入式技术在各个领域的发展，给人们的工作和生活带来了极大的便利。当前主流的嵌入式操作系统有 VxWorks、Windows CE、uClinux、Nucleus、uC/OS II 等等。其中抢先式实时多任务操作系统Nucleus，以其精简的内核、强实时性和高可靠性，深受广大用户的喜爱。本文以嵌入式操作系统 Nucleus 体系结构分析为重点，详细阐明了 Nucleus 的内核结构特点并深入研究了相关算法的改进和系统功能的扩展，最后以具体实例说明了 Nucleus 在实际开发中的应用。

由于嵌入式操作系统的内核构建技术将直接影响到系统的性能；因此，Nucleus 体系结构分析以系统内核研究为中心，从任务管理、中断管理、线程调度策略、任务通信与同步、内存管理等角度深入地分析了 Nucleus 的内核结构组成。同时将 Nucleus 与其它常用的嵌入式系统 uClinux、uC/OS II 的内核进行了比较，从进程调度策略、文件系统、系统的移植性等方面分析了三者的不同点和相似之处。

在此基础之上，针对嵌入式系统任务的调度，文章提出了一种基于模糊理论的任务调度算法，它利用模糊集合来描述任务的不确定性特征；使用多层模糊综合评判和最大隶属度原理来综合考虑任务的多个特征参数并确定任务的优先级；采用动态构建多层评判模型的调度策略来减小任务优先级评判的失效率。此外，文章还讨论了一种基于三层容错技术的Nucleus操作系统功能扩充。它利用“冗余”的思想、分层检测和处理错误的方法屏蔽了故障的影响，提高了嵌入式系统的可靠性，具有非常现实的意义。

最后，论文阐述了嵌入式操作系统Nucleus的实际应用开发——手机下的人体健康检测。手机中处理器采用ARM7；嵌入式操作系统采用Nucleus。它通过串口接收采样到的人体生理数据，并经过相关算法处理分析得出人体的各项生理指标。目前，该项目正在商品化。

关键词：嵌入式操作系统　Nucleus　模糊集合　调度策略

ABSTRACT

Embedded system is one of the most favorite for Microelectronics technology and Computer technology nowadays. Embedded operation system is one of the most important constituent of embedded system. It integrates many software techniques, such as operation system, network, multimedia, characters information processing and so on. It has embedded system developed in other field and brings convenience to people's work environment and daily life. Nowadays, the popular operation systems are VxWorks, Windows CE, uClinux, Nucleus, uC/OS-II and so on. Among them, Nucleus will be very favorable because of its high real-time quality, dependability and simplified kernel. Analysis of Nucleus system structure, study of kernel arithmetic, expanding of system function and application of Nucleus will be introduced in this paper.

Technique of kernel design influences the quality of embedded system directly. Therefore, analysis of Nucleus system structure takes kernel research as center. It analyzes Nucleus kernel deeply in task management, interrupt management, scheduling policy and memory management fields. This article compares Nucleus with uClinux, uC/OS-II, and gets differences and likenesses in scheduling policy, document system and system replanting fields.

At this foundation on, Aiming scheduling of embedded system tasks, it is presented that a scheduling algorithm based on fuzzy theory. It makes use of fuzzy sets for describing uncertain characteristic parameters, multilevel fuzzy synthetic decision and the principle of maximum membership function for determining task priority level, an algorithm which make multilevel decision model dynamically for decreasing invalidation ratio of judging task priority level. In addition, this article discusses a technology of fault tolerance. It makes use of redundancy thought and layered method of handling faults to shield system breakdown. It improves the dependability of system and has realistic meaning.

In the end, this thesis elaborates an actual application of Nucleus

system, human body exam using mobile phone. Processor of mobile phone is ARM7. And embedded operation system of mobile phone is Nucleus. It gets sample data of human body physiology through UART and achieves all kinds of analytical results of human body physiology according to related arithmetic. Currently, this project is commercializing.

KEY WORDS: embedded operation system, Nucleus, fuzzy set, scheduling policy

原 创 性 声 明

本人声明，所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了论文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得中南大学或其他单位的学位或证书而使用过的材料。与我共同工作的同志对本研究所作的贡献均已在本论文中作了明确的说明。

作者签名：道理　　日期：2007年5月20日

关于学位论文使用授权说明

本人了解中南大学有关保留、使用学位论文的规定，即：学校有权保留学位论文，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以采用复印、缩印或其它手段保存学位论文；学校可根据国家或湖南省有关部门规定送交学位论文。

作者签名：道理　导师签名：　日期：2007年5月20日

第一章 绪论

1.1 研究背景

随着嵌入式技术不断地发展，嵌入式操作系统已被广泛应用在汽车电子、航空航天、工业控制、医疗仪器等众多领域。对于这些嵌入式产品而言，嵌入式系统操作的性能往往是成为产品质量保证的关键因素之一。

一个真正的嵌入式系统往往在可靠性和效率上都有着极高的要求，为此需要采用多种技术来保证。如任务调度策略、内存分配及、软件的多版本技术、错误检测与恢复等。这往往增加了系统设计的复杂性，使得这样一个系统的设计过程需要经历很长时间，投入大量的人力和物力才能达到最终的设计要求。而且这样设计出来的系统，又往往是基于专用的硬件和软件，从而使得整个系统的升级、移植以及可维护性等受到了很大的影响。

嵌入式操作系统的出现，为嵌入式系统的设计和发展提供了强大的推动力。最初的嵌入式系统并没有嵌入式操作系统的支持。在那时，由于受内存大小以及处理器能力的限制，程序设计往往用汇编代码来实现，而对于多个任务的并发执行，则完全靠设计人员的手工安排来保证。随着芯片制造技术以及程序语言的发展，渐渐涌现出许多基于高级语言的编程技术以及基于高级语言的嵌入式操作系统。而内存制造技术的提高，使得嵌入式系统不再受限于几K、几十K的小内存，而向几兆甚至几百兆的范围扩展，这也为嵌入式操作系统在实际应用场合提供了基础。嵌入式操作系统的使用提高了开发效率，缩短了开发周期。在嵌入式操作系统环境下，开发一个复杂的应用程序，通常可以按照软件工程中的解耦原则将整个程序分解为多个任务模块。每个任务模块的调试、修改几乎不影响其他模块。大量的实践经验也证明，嵌入式操作系统的引入可以极大地简化嵌入式系统设计过程，节约人力物力的投入，具有很高的应用价值。

嵌入式操作系统伴随着嵌入式技术的发展，从过去简单专用的程序发展到现在可在多个嵌入式处理器上通用的操作系统，已经取得了长足的进步。但是它仍然存在许多需要改进的地方。例如，怎样加强嵌入式系统的可靠性、实时性；如何提高系统任务执行的效率；如何提升操作系统的容错能力等。目前主流的嵌入式操作系统，无伦是通过对 LINUX 改造而成的嵌入式 LINUX，还是已有的如 VxWorks、Nuleucs、 UCOS/II 等，对于高可靠性应用中所需要的实时性能和容错机制都还不尽人所意。因此，如何将嵌入式操作系统和实时系统相结合，设计出一个可靠的、可维护的、内核结构灵活，能够满足众多应用领域所需要的强实

时性和高容错能力的嵌入式实时系统成为当前急需解决的一个问题。这也是本文研究的重点之一。通过对嵌入式操作系统 Nucleus 的内核分析以及实时系统任务调度算法、系统容错方式的研究，本文提出了一种基于模糊理论的实时系统任务调度算法和适用于嵌入式系统的三层容错技术，从而为在嵌入式操作系统在提高实时性能和容错能力上提供一个新的思路。同时，针对嵌入式操作系统 Nucleus 内核中关键技术任务管理、中断管理、线程调度策略等问题也进行了深入的研究。

1．2 嵌入式操作系统综述

1．2．1 嵌入式系统与嵌入式操作系统

嵌入式系统是当前微电子技术和计算机技术的一个重要的分支。由于其本身是面向特定应用的，因此它的概念也比较模糊。根据国际电气和电子工程师协会（IEEE）给出的定义，嵌入式系统是："Devices used to control，monitor，or assiat of operation of equipment，machinery or plants"。由此可知，嵌入式系统是软件和硬件的综合体。我们也可以广义地认为嵌入式系统是一个以应用为中心，以计算机技术为基础，并且软硬件皆可裁减，适用于对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统[1]。

嵌入式系统的架构可以分为硬件和软件两大部分。硬件部分由嵌入式处理器、存储器、输入设备、输出设备等组成[2]。这些单元与标准的计算机结构相比，是以比较特殊的形态存在的，例如说计算机的标准输入是键盘，但是嵌入式产品掌上电脑（PDA）的输入设备可能就是它的触摸屏。软件部分主要是嵌入式操作系统（RTOS）以及架设在其上的嵌入式应用软件。二者是一个紧密的结合体，通常会被一同编译连接并下载到嵌入式硬件载体中从而完成系统预定的功能。

嵌入式系统结构图如下：

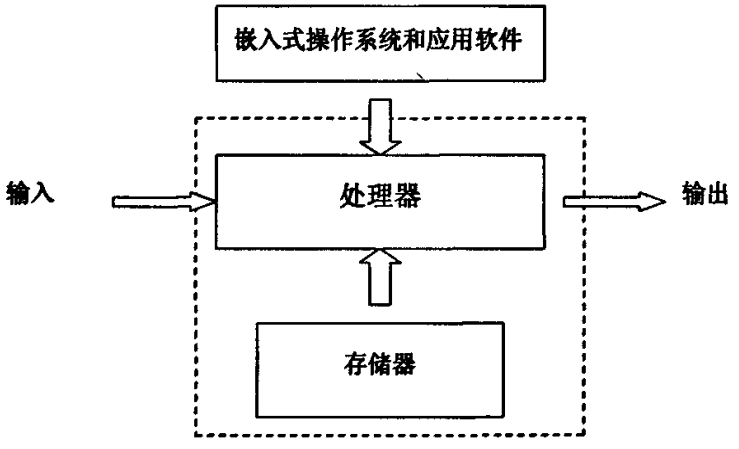

图 1.1 嵌入式系统组成

自INTEL公司推出有史以来的第一颗微处理器4004开始，作为嵌入式系统硬件核心的微处理器由最简单的4位、8位模式发展到目前的16位、32位甚至64位的模式，寻址范围、资源管理、编程方式变的越来越复杂，应用的范围也日益广泛，如果仍然按照简单的循环控制对外界的控制请求进行处理，每添加一项新的功能，都可能需要从头开始设计，没有操作系统已成为嵌入式系统发展的一个最大缺点了，为了适应嵌入式应用的复杂性和多样性，并缩短其开发周期，作为嵌入式系统的灵魂嵌入式操作系统应运而生。

嵌入式操作系统是一种支持嵌入式系统应用的操作系统软件，它是嵌入式系统(包括硬、软件系统)极为重要的组成部分，通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等。嵌入式操作系统具有通用操作系统的基本特点，如能够有效管理越来越复杂的系统资源；能够把硬件虚拟化，使得开发人员从繁忙的驱动程序移植和维护中解脱出来；能够提供库函数、驱动程序、工具集以及应用程序接口（API），使得嵌入式应用软件开发变得更加容易。与通用操作系统相比较，嵌入式操作系统在系统实时高效性、硬件的相关依赖性、软件固态化以及应用的专用性等方面具有较为突出的特点。其系统构架图如下：

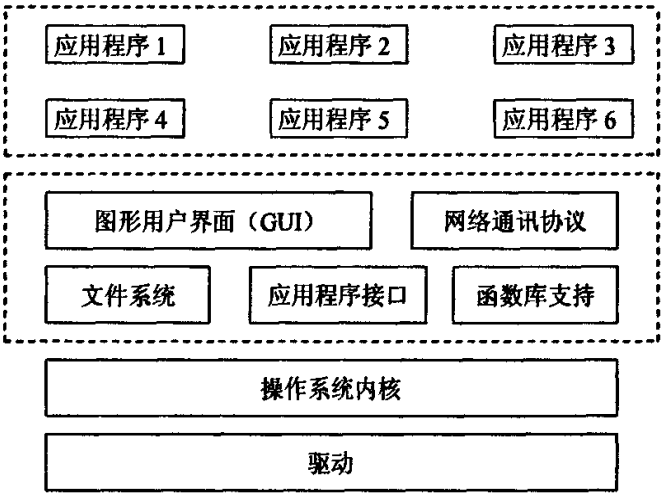

图 2-1 嵌入式操作系统构架图

嵌入式操作系统的引入，在应用系统目标软件和硬件之间架起了一座桥梁，大大减少了系统设计的复杂性。它能移植到各种不同类型的微处理器上，兼容性好；操作系统内核精小、效率高，并且具有高度的模块化和扩展性[3]。在嵌入式操作系统环境下开发应用程序使程序的设计变得更加容易，应用程序被分割成若干独立的任务模块，不需要大的改动就可以增加新的功能。同时对实时性要求苛刻的应用任务也能够得到了快速、可靠的处理[4]。总之通过操作系统提供的高效服务，各种系统资源可以得到更充分的利用。

1．2．2嵌入式操作系统的分类

嵌入式操作系统按其内核的实时性可以分为两大类：一类是面向控制、通信等领域的实时操作系统；另一类是面向消费电子产品的非实时操作系统[5]。

1.非实时式操作系统

由于早期的嵌入式系统中没有操作系统的概念，所以程序员编写嵌入式程序通常直接面对裸机及设备。在这种情况下，通常把嵌入式程序分成两部分，即前台程序和后台程序。前台程序通过中段来处理事件，其结构一般为无限循环；后台程序则掌管整个嵌入式系统软、硬件资源的分配、管理以及任务的调度，是一个系统管理调度程序。这就是通常所说的前后台系统。一般情况下，后台程序也叫任务级程序，前台程序也叫事件处理级程序。在程序运行时，后台程序检查每个任务是否具备运行条件，通过一定的调度算法来完成相应的操作。对于实时性要求特别严格的操作通常由中断来完成。

实际上，前后台系统的实时性比预计的要差。这是因为前后台系统认为所有的任务具有相同的优先级别，即是平等的，而且任务的执行又是通过FIFO队列排队，因而对那些实时性要求高的任务不可能立刻得到处理。另外，由于前台程序是一个无限循环的结构，一旦在这个循环体中正在处理的任务崩溃，使得整个任务队列中的其他任务得不到机会被处理，从而造成整个系统的崩溃。

2.实时式操作系统

实时系统（Real Time System）是一种能够在指定或者确定时间内完成系统功能，并且对外部和内部事件在同步或者异步时间内能做出及时响应的系统。在实时系统中，操作的正确性不仅依赖于逻辑设计的正确程度，而且与这些操作进行的时间有关，也就是说，实时系统对逻辑和时序的要求非常严格，如果逻辑和时序控制出现偏差将会产生严重后果。

实时系统有两种类型：软实时系统和硬实时系统。软实时系统仅要求事件响应是实时的，并不要求限定某一任务必须在多长时间内完成；而在硬实时系统中，不仅要求任务响应要实时，而且要求在规定的时间内完成事件的处理[6]。通常，大多数实时系统是两者的结合。实时应用软件的设计一般比非实时应用软件的设计困难。实时系统的技术关键是如何保证系统的实时性。

实时多任务操作系统是指具有实时性、能支持实时控制系统工作的操作系统。其首要任务是调度一切可利用的资源完成实时控制任务，其次才着眼于提高计算机系统的使用效率，重要特点是要满足对时间的限制和要求。实时操作系统具有如下功能：任务管理(多任务和基于优先级的任务调度)、任务间同步和通信(信号量和邮箱等)、存储器优化管理(含ROM的管理)、实时时钟服务、中断管理服务。实时操作系统具有如下特点：规模小，中断被屏蔽的时间很短，中断处理

时间短，任务切换很快。

1．2．3 嵌入式实时操作系统的特点

虽然嵌入式实时操作系统已经具备了许多系统功能，但是它与通用型操作系统（如 Windows、Unix 等）相比还是有很大的差别。下面我们将通过比较这两类操作系统之间的差别来描述实时操作系统的主要特点。

通用操作系统是由分时操作系统发展而来，大部分都支持多用户和多进程，负责管理众多的进程并为它们分配系统资源。分时操作系统的基本设计原则是：尽量缩短系统的平均响应时间并提高系统的吞吐量，在单位时间内为尽可能多的用户请求提供服务。由此可以看出，分时操作系统注重平均表现性能，不注重个体表现性能。如对于整个系统来说，注重所有任务的平均响应时间而不关心单个任务的响应时间，对于某个单个任务来说，注重每次执行的平均响应时间而不关心某次特定执行的响应时间。

而对于嵌入式实时操作系统，前面我们已经提到，它除了要满足应用的功能需求以外，更重要的是还要满足应用提出的实时性要求，而组成一个应用的众多实时任务对于实时性的要求是各不相同的，此外实时任务之间可能还会有一些复杂的关联和同步关系，如执行顺序限制、共享资源的互斥访问要求等，这就为系统实时性的保证带来了很大的困难。因此，实时操作系统所遵循的最重要的设计原则是：采用各种算法和策略，始终保证系统行为的可预测性(Predictability)。可预测性是指在系统运行的任何时刻，在任何情况下，实时操作系统的资源调配策略都能为争夺资源(包括 CPU、内存、网络带宽等)的多个实时任务合理地分配资源，使每个实时任务的实时性要求都能得到满足。与通用操作系统不同，实时操作系统注重的不是系统的平均表现，而是要求每个实时任务在最坏情况下都要满足其实时性要求，也就是说，实时操作系统注重的是个体表现，更准确地讲是个体最坏情况表现[7]。举例来说，如果实时操作系统采用标准的虚存技术，则一个实时任务执行的最坏情况是每次访存都需要调页，如此累计起来的该任务在最坏情况下的运行时间是不可预测的，因此该任务的实时性无法得到保证。从而可以看出在通用操作系统中广泛采用的虚存技术在实时操作系统中不宜直接采用。

由于嵌入式实时操作系统与通用操作系统的基本设计原则差别很大，因此在很多资源调度策略的选择上以及操作系统实现的方法上两者都具有较大的差异，这些差异主要体现在以下几点：

1.任务调度策略

通用操作系统中的任务调度策略一般采用基于优先级的抢先式调度策略，对于优先级相同的进程则采用时间片轮转调度方式，用户进程可以通过系统调用动

态地调整自己的优先级，操作系统也可根据情况调整某些进程的优先级。

实时操作系统中的任务调度策略目前使用最广泛的主要可分为两种，一种是静态表驱动方式，另一种是基于优先级调度方式[8]。

静态表驱动方式是指在系统运行前工程师根据各任务的实时要求用手工的方式或在辅助工具的帮助下生成一张任务的运行时间表，这张时间表与列车的运行时刻表类似，指明了各任务的起始运行时间以及运行长度，运行时间表一旦生成就不再变化了，在运行时调度器只需根据这张表在指定的时刻启动相应的任务即可。静态表驱动方式的主要优点是：

① 运行时间表是在系统运行前生成的，因此可以采用较复杂的搜索算法找到较优的调度方案。

② 运行时调度器开销较小。

③ 系统具有非常好的可预测性，实时性验证也比较方便。

这种方式主要缺点是不灵活，需求一旦发生变化，就要重新生成整个运行时间表。

基于优先级调度方式则与通用操作系统中采用的基于优先级的调度方式基本类似。任务的优先级是在运行前通过某种优先级分配策略（如Rate-Monotonic、Deadline-Monotonic 等）来指定，越重要的任务其被赋予的就优先级越高。同时，基于优先级的调度算法处总是让处理器先运行处于就绪状态的、优先级最高的任务。这种方式的优缺点与静态表驱动方式的优缺点正好完全相反，它主要应用于一些较独立的嵌入式系统。

2.内存管理

关于虚存管理机制我们在上面已经进行了一些讨论。为解决虚存给系统带来的不可预测性，实时操作系统一般采用如下两种方式：

（1）在原有虚存管理机制的基础上增加页面锁功能，用户可将关键页面锁定在内存中，从而不会被 SWAP 程序将该页面交换出内存。这种方式的优点是既得到了虚存管理机制为软件开发带来的好处，又提高了系统的可预测性。缺点是由于 TLB 等机制的设计也是按照注重平均表现的原则进行的，因此系统的可预测性并不能完全得到保障；

（2）采用静态内存划分的方式，为每个实时任务划分固定的内存区域。这种方式的优点是系统具有较好的可预测性，缺点是灵活性不够好，任务对存储器的需求一旦有变化就需要重新对内存进行划分，此外虚存管理机制所带来的好处也丧失了。

3.中断处理

在通用操作系统中，大部分外部中断都是开启的，中断处理一般由设备驱动

程序来完成。由于通用操作系统中的用户进程一般都没有实时性要求，而中断处理程序直接跟硬件设备交互，可能有实时性要求，因此中断处理程序的优先级被设定为高于任何用户进程。

但对于实时操作系统采用上述的中断处理机制是不合适的。首先，外部中断是环境向实时操作系统进行的输入，它的频度是与环境变化的速率相关的，而与实时操作系统无关。如果外部中断产生的频度不可预测，则一个实时任务在运行时被中断处理程序阻塞的时间开销也是不可预测的，从而使任务的实时性得不到保证；如果外部中断产生的频度是可预测的，一旦某外部中断产生的频度超出其预测值就可能会破坏整个系统的可预测性。其次，实时操作系统中的各用户进程一般都有实时性要求，因此中断处理程序优先级高于所有用户进程的优先级分配方式是不合适的。

一种较适合实时操作系统的中断处理方式为：除时钟中断外，屏蔽所有其它中断，中断处理程序变为周期性的轮询操作，这些操作由核心态的设备驱动程序或由用户态的设备支持库来完成。采用这种方式的主要好处是充分保证了系统的可预测性，主要缺点是对环境变化的响应可能不如上述中断处理方式快，另外轮询操作在一定程度上降低了 CPU 的有效利用率。另一种可行的方式是：对于采用轮询方式无法满足需求的外部事件，采用中断方式，其它时间仍然采用轮询方式。但此时中断处理程序与所以其它任务一样拥有优先级，调度器根据优先级对处于就绪态的任务和中断处理程序统一进行处理器调度。这种方式使外部事件的响应速度加快，并避免了上述中断方式带来第二个问题，但第一个问题仍然存在。

此外为提高时钟中断响应时间的可预测性，实时操作系统应尽可能少地屏蔽中断。

4. 系统调用以及系统内部操作的时间开销

进程通过系统调用得到操作系统提供的服务，操作系统通过内部操作（如上下文切换等）来完成一些内部管理工作。为保证系统的可预测性，实时操作系统中的所有系统调用以及系统内部操作的时间开销都应是有界的，并且该界限是一个具体的量化数值。而在通用操作系统中对这些时间开销则未做如此限制。

5. 系统的可重入性

在通用操作系统中，核心态系统调用往往是不可重入的，当一低优先级任务调用核心态系统调用时，在该时间段内到达的高优先级任务必须等到低优先级的系统调用完成才能获得 CPU，这就降低了系统的可预测性。因此，实时操作系统中的核心态系统调用往往设计为可重入的。

1．3 本文的主要研究内容

本论文深入研究了嵌入式操作系统Nucleus的内核结构，并从任务管理、中断管理、线程调度策略、任务通信与同步、内存管理等角度去分析了系统的构建技术。同时，将操作系统Nucleus与其它常用的嵌入式系统uClinux、uC/OS II的内核进行了比较，从进程调度策略、文件系统、系统的移植性等方面分析了三者的不同点和相似之处。在此基础之上，针对嵌入式系统任务的调度，本文提出了一种基于模糊理论的任务调度算法，它使用多层模糊综合评判和最大隶属度原理来综合考虑任务的多个特征参数并确定任务的优先级。此外，还讨论了一种基于三层容错技术的Nucleus操作系统功能扩充。论文的最后讨论了嵌入式操作系统Nucleus的应用开发——人体健康检测；并总结了开发过程中的不足之处以及改进的构思。

本文主要的研究内容总结如下：

1. 分析了嵌入式实时操作系统Nucleus的内核。从内核中发各个组件出发，研究、说明了Nucleus的组织结构以及运行机理。

2. 将操作系统Nucleus与其它开发源代码的常用嵌入式操作系统进行对比；分析出不同嵌入式实时操作系统之间各自的优缺点。

3. 针对嵌入式操作系统中的任务具有特征参数多和特征参数不确定的特点，提出了一种基于模糊理论的任务调度算法。

4. 讨论了一种基于三层容错技术的Nucleus容错能力扩充。

5. 阐述了基于嵌入式操作系统Nucleus下的人体健康检测应用开发。

1．4 本文的组织

第一章为绪论，首先介绍了嵌入式操作系统的基本概念及其特点。并分析出了嵌入式操作系统目前还存在的不足从而引出了本课题的来源，说明了研究嵌入式操作系统的意义。

第二章分析了嵌入式操作系统 Nucleus 的内核结构。首先对嵌入式实时操作系统 Nucleus 进行一个简单的介绍。然后，按照 Nucleus 的组件结构对系统的内核初始化、任务管理、线程调度、任务通信和同步、内存管理等一一作了详细的说明。分析出了 Nucleus 是怎样构建一个功能强大的嵌入式实时操作系统。

第三章将嵌入式操作系统 Nucleus 与其它常用嵌入式操作系统 uClinux、uC/OS II 进行了对比。首先对 uClinux、uC/OS II 两款嵌入式系统进行了介绍说明。然后，从进程调度策略、文件系统、系统的移植性等方面说明了三者的不同点和相似之处；分析出这三款嵌入式操作系统各自的有缺点以及它们在实际开

发时所应用的领域。

第四章针对嵌入式操作系统Nucleus提出了一种基于模糊理论的任务调度算法以及容错功能的扩充。论文针对任务具有特征参数多和特征参数不确定的特点，提出了一种基于模糊理论的任务调度算法利用模糊集合来描述任务的不确定性特征；使用多层模糊综合评判和最大隶属度原理来综合考虑任务的多个特征参数并确定任务的优先级；采用动态构建多层评判模型的调度策略来减小任务优先级评判的失效率。此外，论文还讨论了一种基于三层容错技术的Nucleus功能扩充。

第五章介绍了如何利用嵌入式操作系统Nucleus开发手机下人体健康检测功能模块。首先介绍了手机的硬件和软件结构。然后在此基础上分析了人体健康检测的总体设计。最后详细说明了应用开发中通信协议、人体脉搏检测、人体体温检测的具体实现过程；并对开中的不足之处进行了总结。

第六章总结了在书写论文期间开展的相关研究工作，并分析了在该领域中需要进一步研究的相关问题。

第二章 Nucleus体系结构分析

2.1 嵌入式实时操作系统Nucleus介绍

Nucleus是为实时嵌入式应用而设计的一款抢先式多任务操作系统，其内核95%的代码是使用ANSI C写成，因此具有很强的移植性能够支持大多数类型的处理器。从实现的角度来说，Nucleus以函数库的形式链接到目标应用程序中，并形成可执行目标代码。Nucleus内核在典型的CISC体系结构上占据大约20k空间，而在典型的RISC体系结构上占据空间为40k左右，其内核数据结构占据1.5k字节的空间[9]。此外，Nucleus采用了软件组件的方法来组织整个系统。每一个组件具有单一而明确的功能，通常由几个C及汇编语言模块构成，提供清晰的外部接口，对组件的引用就是通过这些接口完成的。

下面就实时操作系统Nucleus的特点作一下简单的介绍。

第一，开放源代码。源代码清晰易读，组织有序而且注释详尽。这极大方便了用户二次开发，即无需编写和调试板支持软件包BSP，从而达到易学易用的目的。

第二，移植性好。由于Nucleus系统内核95%的代码是用ANSI C写成的并对用户开放，因此该系统非常便于移植并能够支持大多数类型的处理器。

第三，响应时间快速。对临界资源的检测时间不依赖于占有该临界资源的线程执行时间的长短，一旦低优先级线程释放掉临界资源，高优先级线程就会抢占运行。

第四，健壮的实时内核系统服务。系统提供任务控制、任务通信、任务同步、内存管理、标准输入/输出设备接口等。所有的操作系统对象（任务、邮箱、队列、管道等）都可以动态地创建和删除。创建一个对象时，要指定其控制块的内存区域和其它的数据要求（堆栈空间等）。

第五，可用性强。Nucleus系统调用名直接表明系统的各项功能。比如，你可以通过Nu_Create_Task系统调用来创建一个任务。Nucleus内核对象都不隐式地和别的对象相关。因此，用户可以利用多个Nucleus内核对象之间的结合形成混合系统调用。

第六，好的任务调度策略。对任务的调度采用优先级和时间片结合的方法，首先查看任务是否可以抢占，如果不能抢占，则一直执行到任务完成或任务放弃时间片；否则，依靠优先级进行调度，先调入优先级最高的任务，对于优先级相同的任务则分享时间片、轮流调度。因此，它具有较高的吞吐量，即随着任务数

目的增多，任务的调度时间为常数。

第七，系统配置简单。Nucleus 允许应用程序中使用或者不使用系统所提供的功能。如果你需要某个系统功能的话，可以在Nucleus.h中进行宏定义。例如定义#define NU_ENABLE_STACK_CHECK，那么每个系统调用时都要进行堆栈检查，否则就不进行堆栈检查。

第八，功能模块种类丰富。Nucleus嵌入式系统除提供功能强大的操作系统内核外，还针对不同的用户需求提供了种类丰富的其它功能模块。例如用于通讯系统的局域和广域网络模块，支持图形应用的实时化Windows 模块[10]，支持Internet 网的WEB产品模块、工控机实时BIOS 模块、图形化用户接口[11]以及应用软件性能分析模块等。各类功能模块采用软件组件的方法来组织整个系统。同时，利用现有系统功能，经过组合可以很容易地得到新的系统功能，提高了系统的可扩展性。

2.2 嵌入式实时操作系统Nucleus内核分析

2.2.1 Nucleus内核概述

内核，是操作系统的核心部分。它的主要目的是负责管理实时任务的竞争运行（共享处理器），系统内存的分配和回收，以及系统各个硬件设备的驱动控制得等等。同时，内核还需要为系统上层应用程序提供了各种便利的接口，必须能快速响应外部事件以实现实时性。所以一款操作系统内核的好坏将直接决定了这款操作系统的性能和稳定性。

在Nucleus操作系统中，内核提供了大量的内核对象和优秀的管理策略来实现了上述对内核的要求。在系统任务控制方面，任务之间可以按照优先级和时间片两种方式来共享处理器资源；通过邮箱（mailbox），队列（queue）和管道（pipe）进行通信；利用信号量（semaphore）、事件组（event group）和信号（signal）进行任务之间的同步和互斥。此外，在内存管理方面，Nucleus提供了动态内存分配和分区内存分配（dynamic memory and partition memory）两种存储器管理机制；在定时管理方面，Nucleus 提供了内核对象定时器（Timer）来处理周期性事件和任务的睡眠和挂起超时等[12]。

Nucleus系统内核结构图如下：

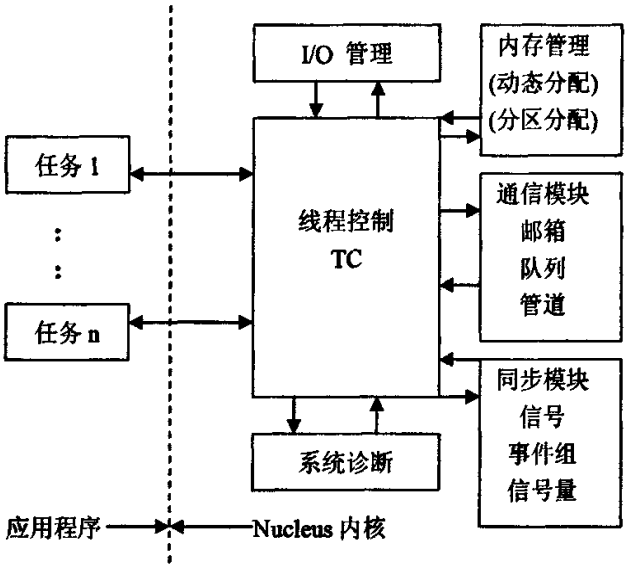

图 3-1 Nucleus 系统内核结构图

从 Nucleus 的系统结构图中可以看出线程控制是整个内核的核心。Nucleus 将其它功能机制称为软件组件（Software Component）。Nucleus 为每一个软件组件提供了一系列的系统调用，任务与 Nucleus 的交互就是在这些系统调用上进行的[13]。

2. 2. 2 Nucleus 内核组件组成

在上节中，曾一度提到嵌入式实时操作系统 Nucleus 内核又可以分成许多不同的内核对象。它们之间相互联系，有效组织在一起共同完成了内核所具备的复杂的功能。除了少数情况外，Nucleus 不允许从外部对组件内部访问；只能通过组件对外提供的接口来调用所需的功能。也正是采用了软件组件的方法，Nucleus 各个组件非常容易替换和复用，这大大提高了软件的扩展性能[14]。在 Nucleus 操作系统中总共有 16 个不同的组件，即 16 种内核对象。具体说明如下表所示：

表 3-1 Nucleus 操作系统组件表

组件名称	简写	描述
Common Service Componet	CS	通用服务组件
Initalization Component	IN	初始化组件
Thread Control Component	TC	线程调度组件
Timer Component	TM	定时组件
Active Timers List	-	活动定时队列
Mailbox Component	MB	邮箱组件
Queue Component	QM	队列组件
Pipe Component	PI	管道组件

Semaphore Component	SM	信号量组件
Event Flag Component	EV	事件组件
Partition Memory Component	PM	存储分配组件
Dynamic Memory Component	DM	动态存储组件
Input/Output DriverComponent	IO	输入输出组件
History Component	HI	历史组件
License Component	LI	许可证组件
Release Component	RL	版本控制组件

每个组件都有组件特定的功能，下面简单的介绍一下：

通用服务组件负责提供给其它组件提供链表功能。

初始化组件负责初始化 Nucleus 系统。

线程控制组件负责管理相互竞争的执行实体——系统任务和高级中断服务程序。

定时管理组件负责处理所有的 Nucleus 定时设备。

邮箱组件负责处理所有的 Nucleus 邮箱设备。邮箱是一种低级的任务间通信机制。

队列组件负责处理所有的 Nucleus 队列设备。它同样也是种在任务间传递消息的机制。

管道组件负责处理所有管道设备。管道，一种任务间通信的机制。

信号量组件负责处理所有信号量设备。信号量，任务间同步机制。

事件组件负责处理所有事件组设备。事件，确定系统事件已发生的机制。

内存分区组件负责处理内存分区设备。分区内存池包含固定大小内存池。

动态内存组件负责处理动态内存设备。动态内存池包含用户指定的字节数。

I/O 驱动组件负责处理 I/O 驱动设备。它提供了标准的 I/O 驱动接口。

历史组件负责处理历史设备，保持了不同系统活动的循环记录。

错误组件是一个公共错误处理服务例程，它解决系统遇到致命错误的情况。

许可证组件用于存储和报告客户许可证以及序列号等相关信息。

发行组件专用于存储和报告发行信息。

每一种组件都有自己的源码文件、数据结构和函数功能。它们之间相互独立，但又可以通过组件的对外接口相互调用。组件的源码文件均包含了以下一些类型文件：数据类型定义和常量定的文件、外部接口文件、函数功能实现的 c 文件或汇编文件。并且源码文件根据组件和功能的不同，统一了命名方式即：

文件名＝组件简称＋功能标识

具体说明如下表所示：

表 3-2 内核组件文档结构表[12]

文件名	功能说明
XX_DEFS.H	定义数据结构和常量
XX_EXTR.H	定义外部接口
XXD.C	定义静态和全局数据结构
XXI.C	定义组件的初始化函数
XXF.C	组件管理对象的信息
XXC.C	组件核心功能函数
XXCE.C	组件核心功能函数错误处理
XXS.C	组件补充功能函数定义
XXSE.C	组件补充功能函数错误处理

其中，在表 3-2 里，XX 表示内核组件的简称，而紧跟 XX 后面的字符就是各类文件功能的标识符号。例如，通用服务组件共包括三个文件：CS_DEFS.H、CS_EXTR.H、CSC.C；它们分别定义了通用服务组件的数据结构、外部接口和核心功能函数。同时，我们可看出并不是每个组件都必须包含上述 9 种类型文件。如通用服务组件。

2.2.2 Nucleus 系统的初始化

内核的启动是从系统低级初始化开始的。在初始化组件中包含一个汇编文件 INT.[S, ASM, SRC]，它包括了所有与目标板有关的初始化函数，系统的低级初始化入口点也在其中。由于低级初始化与具体的目标板（如嵌入式处理器、外围扩展设备）相关，因此它必须使用汇编编写，主要完成中断向量表指针初始化、寄存器基址初始化、外部 RAM 分区的初始化（包括：ram，data，init，sys_memory 等）。在低级初始化完成以后，控制就交给初始化线程 INT_Initialize。

在 Nucleus 中，INT_Initialize 是一个被系统调用的子程序。它首先完成硬件寄存器的配置，包括：存储器片选，软件看门狗(SWT)，系统周期定时器(PIT)，A 口、B 口、C 口引脚功能设定，串行通信控制的初步配置等。这些硬件配置跟目标板相关，需要用户根据自己的实际要求来配置并进行软件编写。其次，初始化系统堆栈指针。将系统堆栈指针 TCD_System_Stack 初始化为堆栈区 stack 的顶部，同时在系统内存区 Sys_Memory 中拿出 TMD_HISR_Stack_Size 大小的一片内存作为高级中断服务程序的堆栈 HISR_STACK。

当 INT_Initialize 初始化完成后，控制转移到 Nucleus 内核高级初始化子程序 INC_Initialize 上，并不再返回 INT_Initialize。在上节，描述 Nucleus 内核组件组成时曾说明在系统的各个组件文档结构中有一个命名为 XXI.C 的文件，它包含了组件各自的初始化函数。INC_Initialize 函数在这里要做的事仅仅就是依次调用系统各个功能组件的初始化函数以完成系统组件的初始化。这其

中包含许可证组件、历史组件、线程控制组件、邮箱组件、队列组件、信号量组件、输入/输出设备驱动组件等等。在 INC_Initialize 调用系统组件初始化函数完成用初始化后，INC_Initialize 转而调用 Application_Initialize 函数，开始初始化由用户编写的系统应用层模块。

Application_Initialize 是 Nucleus 内核对用户提供的一个外部接口。用户通过自己编写该函数实体来完成应用程序环境的详细初始化。它包括任务、邮箱、队列、管道、事件集、内存池等的创建，中断的注册以及应用程序的初始化。在 Application_Initialize 完成用户自定义初始化后，控制返回到 INC_Initialize 函数并调用 TCT_Schedule 函数开始线程的调度。

初始化具体过程如下图所示。

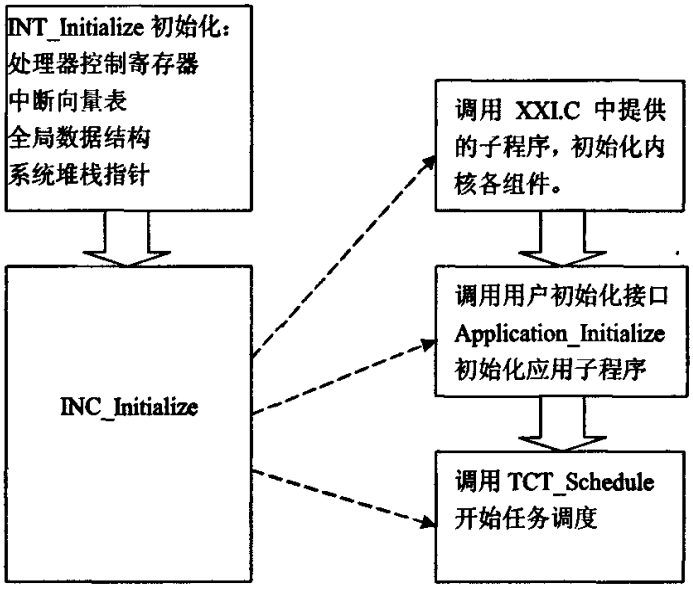

图 3-2 Nucleus 系统的初始化

2. 2. 3 任务管理

一个任务，也称作一个线程，是一个简单的程序。实时应用程序的设计过程就是如何把问题分割成多个任务的过程。每个任务都是整个应用程序的一部分，都具有一定的优先级和自己的一套堆栈空间。

在 Nucleus 中，任务优先级范围从 0 到 255，这里 0 代表最高优先级，255 代表最低优先级。当多个任务就绪时，优先级最高的任务优先执行；若此时已有一个优先级较低的任务已经执行，则判断该低优先级任务是否可抢占，如果可以抢占那么系统保存现场最高就绪的优先级立刻执行，否则就等待该低优先级任务结束后最高优先级再执行。倘若，最高优先级的任务不只一个时，它们之间采用时间片轮转法轮流使用资源[15]。系统中，任务具有五种状态：运行态，就绪态，

挂起态，完成态和终止态。运行态是指任务得到了处理器的使用权，正在运行之中。就绪态意味着任务已经准备好，可以运行，但是由于它不够条件剥夺正在运行的任务而暂时不能运行。挂起态相当于任务驻留在内存中，但是不被多任务内核所调度。终止态是指任务被禁止；一旦进入这种状态，任务直到复位之前都不能运行。完成态说明任务完成并返回到初始入口子程序。一旦进入这种状态，任务直到复位前都不能运行。

任务管理使用到了以下重要数据结构：

1. 已创建任务链表

Nucleus 能动态地创建和删除任务。每一个任务都有一个任务控制块（OS_TCB）。它是一种数据结构，用来描述任务的一些属性。当一个任务被系统动态创建时，相应的任务控制块便被初始化并被保存在一个双向的循环链表中。该链表的头指针是 TCD_Created_Tasks_List，专用于保存已经创建的任务控制块。当有新的任务被创建时，它的任务控制块被插入链表的尾部，当任务被删除时，它的任务控制块则从该链表中删除。

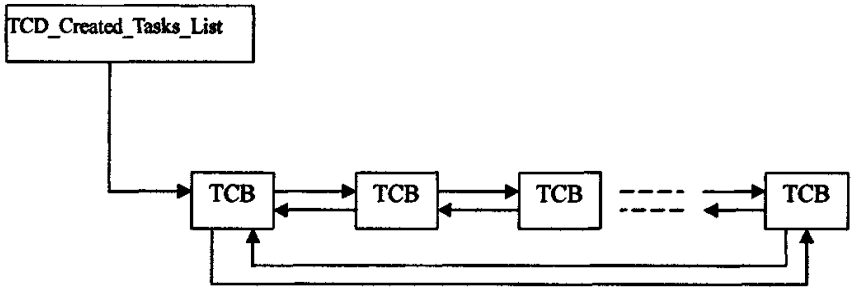

图3-3 任务控制块循环链表[12]

2. 优先级链表

系统使用一个 TCB 指针数组 TCD_Priority_List[256]来进行任务优先级管理。该数组中共有 256 个元素，代表任务 256 个优先级等级，每个元素都是对应优先级的就绪任务链表头指针。若某个任务处于就绪状态系统便会根据该任务的优先等级将其插入到对应链表之中。如果某一个元素为空，则表示在此优先级下没有任务就绪。

优先级链表 TCD_Priority_List 结构如下：

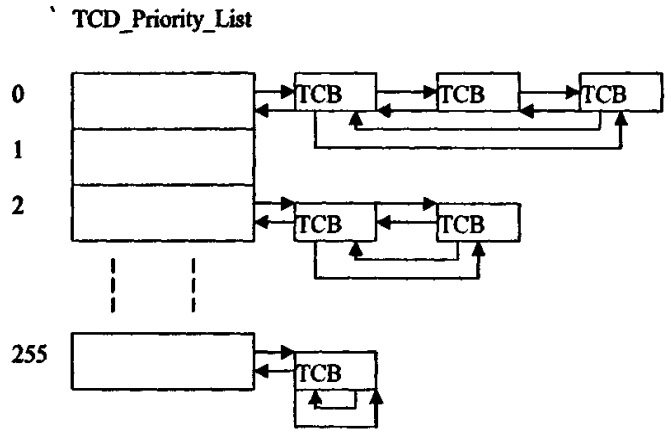

图3-4 优先级双向链表数组[12]

3. 优先级组及优先级子组

除指针数组 TCD_Priority_List，系统还使用到 TCD_Priority_Groups 和 TCD_Sub_Priority_Groups 两个数据结构来辅助管理任务的优先级。TCD_Priority_Groups 是一个 32 位的无符号长整形整数，它的每一位对应一个包含 8 个优先级的优先级组。若某一位被置位，就表示对应的优先级组中至少有一个任务已就绪。例如，如果位 0 被置位，就表示优先级为 0 到 8 的任务中至少有一个已准备好执行。TCD_Sub_Priority_Groups 是一个大小为 32 的无符号字符数组。它代表了优先级子组的位映像图，每一个元素对应一组包含 8 个优先级的优先级组，而每个八位数组元素的每一位又对应着一个优先级。比如 TCD_Sub_Priority_Groups[0]对应优先级组 0-7，bit0-bit7 分别表示优先级 0-7。

TCD_Highest_Priority 、TCD_Execute_Task 和 TCD_Current_Thread 是系统中另外三个用于任务管理的重要数据结构。TCD_Highest_Priority 是一个整形整数，代表当前就绪任务的最高优先级。TCD_Execute_Task 是一个 TCB 结构指针，它指向当前正在运行的任务。TCD_Current_Thread 是一个 void*类型指针，它指向当前正在运行的任务或是高级中断。

利用以上数据结构系统可以顺利地进行任务调度，这包括了创建任务、删除任务、终止任务、恢复任务、挂起任务等。

1. 任务创建：

任务地创建一般是在 Application_Initialize 中进行，当然也可以在其它任务中动态地创建和删除任务。系统首先调用统一的对外接口 NU_Create_Task 来创建任务。NU_Create_Task 是一个宏，它指代系统任务模块中的 TCC_Create_Task 功能函数。该函数是系统真正创建任务的地方，它包括初始化新建任务控制块（TCB）、创建任务初始堆栈、修改

已创建任务链表 TCD_Created_Tasks_List、设置任务优先级、修改 TCD_Priority_Groups、TCD_Sub_Priority_Groups 及 TCD_Execute_Task 等数据结构、设置任务状态（就绪态或挂起态）等等。

任务创建流程如下：

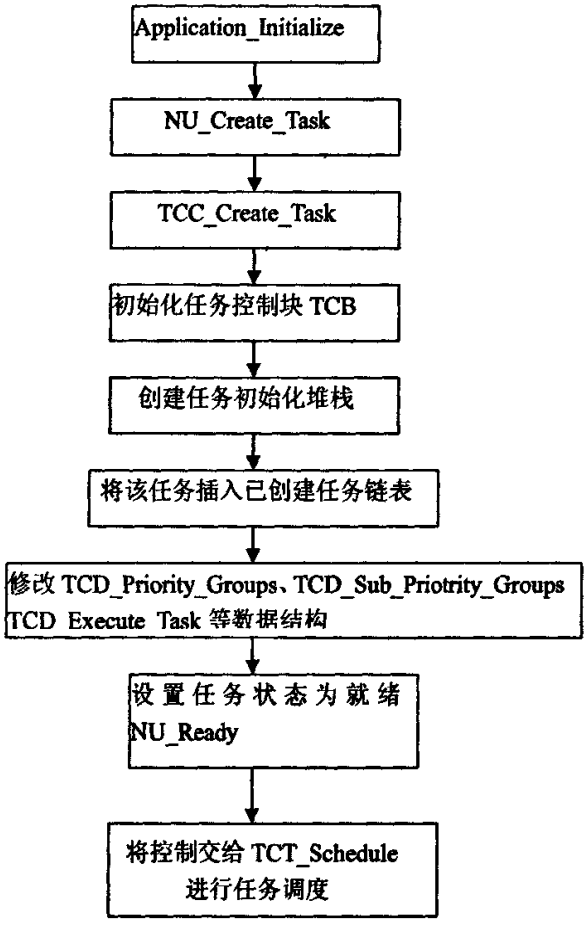

图3-5 任务创建流程图

任务的具体实现部分较简单，它通常是一个无限循环结构。但这里值得注意的是任务绝对不会返回任何值，故返回参数必须定义成 void。示意函数如下：

```
void  Task(UNSIGNED argc,  VOID *argv)
{
  While(1)
   {     /*用户代码*/
     /*调用 Nucleus 各种组件服务*/
    }
}
```

2. 任务删除

删除任务是由任务模块中 TCC_Delete_Task 功能函数完成。该函数假定要删除的任务处于完成或终止状态，并将其移出已创建任务列表 TCD_Created_Tasks_List。但是值得注意的是该函数不释放任务控制块和堆栈所占内存。

3. 复位任务

任务模块中 TCC_Reset_Task 功能函数用于复位一个任务。但是只有当任务处于完成或终止状态时，才能对该任务执行复位，否则返回 NU_NOT_TERMINATED，表示任务没有结束或终止。任务复位主要是将任务控制块中的数据成员重新赋初值，重新创建任务堆栈，将任务状态设置为无条件挂起状态 NU_PURE_SUSPEND。

4. 终止任务

如果要终止的是当前任务，则直接将任务挂起，将任务状态设置为终止态 NU_TERMINATED。

如果要终止的不是当前任务，则要对任务状态进行判断。如果任务已经处于完成或终止状态，则什么也不做。如果任务处于就绪状态，则直接将任务挂起，将任务状态置为终止态 NU_TERMINATED。如果任务处于挂起状态，则必须释放和该任务相关的所有保护结构后才能将任务终止。

5. 恢复任务

如果任务可以获得执行所需要的系统资源，比如：对于向管道发送消息的任务如果消息管道已有空余空间，或者等待接收管道消息的任务如果管道中有消息，那么挂起在该管道上的任务就会恢复。

如果任务挂起类型与请求恢复类型一致，则把任务状态置为就绪态，将任务插入就绪任务优先级链表 TCD_Priority_List，设置优先级组 TCD_Priority_Groups 和子优先级组 TCD_Sub_Priority_Groups 中本任务优先级对应的位，指示本优先级有任务就绪。如果要恢复的任务优先级比当前最高优先级 TCD_Highest_Priority 要高，且当前任务 TCD_Execute_Task 可以抢占，则将要恢复的任务置为当前任务，同时会产生任务抢占，返回 NU_TRUE，否则返回 NU_FALSE。

6. 挂起任务

如果任务不能获得执行所需要的系统资源，那么任务就会被挂起。任务挂起首先判断要挂起的是不是当前任务，如果不是挂起当前任务，则要释放任务的当前保护结构。

其次，如果任务的状态为就绪态且该任务优先级就绪任务链表中只有这一个任务就绪，则要清空该优先级就绪任务链表，同时要清除子优先级组 TCD_Sub_Priority_Groups 和优先级组 TCD_Priority_Groups 对应的位。如果要挂起的任务具有最高优先级，则要根据优先组和子优先组重新搜索最高优先级，如果其他组中没有任务就绪，则设置当前就绪任务的最高优先级 TCD_Highest_Priority 为 255。然后根据最高优先级，重新调整任务运行，如果最高优先级为 255，则设置当前正在运行的任务为空。

如果任务的状态为就绪态且该任务优先级就绪任务链表中不只这一个任务就绪，则将该任务从优先级就绪任务链表中删除，不用修改子优先级组和优先级组，另外也不用调整前就绪任务的最高优先级 TCD_Highest_Priority，只是利用最高优先级重新调整任务的运行。

如果要挂起的是当前线程 TCD_Current_Thread，则调用功能函数 TCT_Control_To_System 将控制转交给系统，在 TCT_Control_To_System 中给当前线程创建一个 solicited 类型的堆栈帧，线程入口是调用 TCT_Control_To_System 的下一条指令，任务恢复时从这条指令开始继续执行。TCT_Control_To_System 随后又将控制交给 TCT_Schedule，TCT_Schedule 根据 TCD_Execute_HISR 或 TCD_Execute_Task 开始下一轮的任务调度，如果调度过程中发现挂起任务需要的系统资源可以满足，就会把任务恢复，按照优先级重新调度。

2.2.4 中断管理

中断是一种为外部和内部事件提供立即响应的机制。在 Nucleus 系统中，中断可以分为两类：一类是低级中断（LISR），另一类是高级中断（HISR）。

低级中断服务子程序主要功能是完成硬件中断处理以及激活高级中断服务程序。它和正常的中断服务子程序一样，在调用之前系统保存上下文，在调用返回之后恢复上下文。低级中断服务子程序可以被其它子程序调用，然而它能够访问 Nucleus 的服务却很少。因此如果中断处理需要附加较多的 Nucleus 服务必须激活一个高级的中断服务处理子程序。

高级中断类似于任务同样支持动态创建和删除，并且每个高级中断都有各自的堆栈空间和控制块（HCB）。高级中断能够访问大多数的 Nucleus 服务，并且具备三个优先等级。在一个低优先级的高级中断服务处理期间，如果一个更高优先级等级的高级中断被激活，低优先级的高级中断以与任务抢占方式相同的方法进行抢占。相同优先级的高级中断以它们最初激活的顺序运行。所用激活的高级中

断在正常任务调度恢复之前运行。

中断管理将使用到以下重要的数据结构：

1. 已注册低级中断和低级中断函数指针数组

Nucleus 维护一个叫做 TCD_Registered_LISRs[NU_MAX_VECTORS+1]的数组用于保存已经注册的低级中断。数组的每一个元素对应着一个中断向量，其中 NU_MAX_VECTORS 为系统定义的最大中断向量数。如果某元素值为 0 表示该中断向量没有注册，操作系统不能处理；若为非 0 表示该中断向量已经注册，且其值为在低级中断函数指针数组 TCD_LISR_Pointers[NU_MAX_LISRS+1]中的索引下标。其中 NU_MAX_LISRS 表示系统低级中断数，TCD_LISR_Pointers 保存当中断产生时要调用的低级中断服务程序的入口函数。

2. 已激活高级中断头链表指针数组及已激活高级中断尾指针链表数组

在前面曾经提到过高级中断拥有三等优先级 0～2；0 代表最高优先级，2 代表最低优先级。Nucleus 按照优先级方式维护一个记录所有已激活高级中断链表的头指针数组，这个数组叫做 TCD_Active_HISR_Heads[3]。同样，系统也维护了一个类似的数组来记录所有已激活高级中断链表的尾指针，它叫 TCD_Active_HISR_Tails[3]。其中这两个数组的下标代表了高级中断的优先级。

优先级首尾链表图如下：

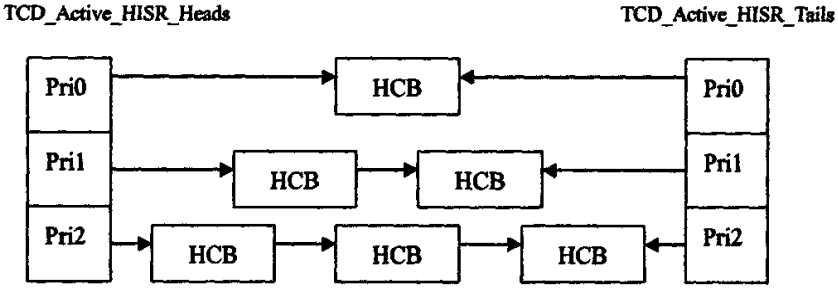

图3-6 优先级首尾链表图[12]

3. 已创建高级中断链表

和已创建任务链表一样，系统使用一个双向链表来维护已经创建的高级中断，指向这个双向链表的是 TCD_Created_HISRs_List。当一个高级中断被系统动态创建时，相应的高级中断控制块（HCB）便被初始化并被保存这个双向的循环链表中。如果 TCD_Created_HISRs_List 为空 NU_NULL，那么就表示当前没有高级中断被创建。

此外 TCD_Interrupt_Count 和 TCD_Execute_HISR 是系统用于中断管理的另两重要数据结构。TCD_Interrupt_Count 是一个整形整数记录系统正在进行处理的中断服务程序个数。其值为 0 表示目前没有中断处理；为 1 表示只有一个中断

正在处理；大于1时则表示有中断嵌套。TCD_Execute_HISR是一个HCB结构指针，它指向当前正在执行或要执行的具有最高优先级的的高级中断。

中断的调度包括：中断向量的注册、上下文的保护与恢复、低级中断的执行、高级中断的创建与删除、高级中断的激活以及高级中断的调度等。

4.注册中断向量

嵌入式操作系统 Nucleus 要管理一个中断，首先要通过系统功能函数 NU_Register_LISR(INT vector , VOID (*new_lisr)(INT) , VOID (**old_lisr)(INT))注册其中断向量。该功能函数的第2个参数是低级中断的入口函数指针，也就是中断产生后要执行的低级中断服务程序。

注册中断首先要判断该中断向量是否已经注册过。如果该中断向量没有注册过，则在低级中断函数指针数组中找出一个没有使用的单元，将中断函数入口地址填入该单元，同时将该单元的下标填入该中断向量在已注册低级中断数组中对应的单元。如果该中断已经注册过，则利用已注册低级中断数组索引到该中断向量在低级中断函数指针数组中的下标，然后更新的中断服务函数入口地址。

5.执行低级中断

中断发生时系统根据中断向量，在低级中断指针数组TCD_LISR_Pointers中索引到本中断向量的中断服务入口函数指针，然后执行中断服务函数，在执行完毕后，系统恢复上下文并重新开始进行系统调度。通常低级中断要做的只是处理硬件中断及激活高级中断。

6.创建高级中断

高级中断是中断产生后由低级中断激活的高级中断服务程序。高级中断服务程序类似任务，但它的创建要比任务的创建简单，不用进行设置任务状态、恢复任务等操作，只需为其创建一个高级中断控制块HCB和堆栈即可。最后将该高级中断插入到已创建高级中断链表TCD_Created_HISRs_List 中。

7.删除高级中断

高级中断的删除，仅仅是将高级中断从已创建高级中断链表TCD_Created_HISRs_List中删除，并清除相应的标志，但不释放与HISR相关的内存（控制块、堆栈等），同时也不影响高级中断的激活。

8.激活高级中断

激活高级中断首先要判断该高级中断是否已被激活。如果高级中断已被激活，则只将该高级中断激活次数加1。若该高级中断未被激活，则将该高级中断插入到本优先级激活链表上。同时，系统检测该优先级链表在插入操作之前是否为空，如果为空则必须根据当前所有已激活的高级中断的优先级决定是否修改TCD_Execute_HISR。

值得注意的是：虽然高级中断是在低级中断中被激活的；但系统不会立刻去执行高级中断服务函数，而是必须由系统调度（TCT_Shedule）来地统一进行中断调度执行。

9. 调度高级中断

高级中断的调度是通过循环调度当前的 TCD_Execute_HISR 来实现的，也就是循环执行高级中断创建时用户指定的入口函数，直至其激活次数等于 0 为止。如果 TCD_Execute_HISR 只被激活了一次，则 HISR 的入口函数只会执行一次。

当 TCD_Execute_HISR 调度完毕（激活次数为 0），如果 TCD_Execute_HISR 所在优先级的激活链表只有这一个高级中断，则将本优先级激活链表清空，然后从激活链表头指针数组 TCD_Active_HISR_Heads 中按优先级顺序搜索到一个已被激活的最高优先级的高级中断，由此来修改 TCD_Execute_HISR，如果没有其它高级中断被激活，则 TCD_Execute_HISR 为空指针。

最后，系统将控制权交给 TCT_Shedule 进行重新调度，如果还有其它的高级中断被激活，则重复上面的过程；否则进入任务的调度，或在 TCT_Shedule 死循环，等待高级中断被激活或任务就绪。

2. 2. 5 线程调度策略

线程控制事用来管理实时任务和高级中断服务的执行，它是 Nucleus 嵌入式实时操作系统的最核心部分。系统使用 TCD_Current_Thread 指针来指向当前执行的线程控制块。如果当前系统执行的线程是任务，TCD_Current_Thread 就指向 TCB 结构；如果当前线程是中断该指针则指向 HCB 结构。它可以向被保存在变量中正如上文所述，为了在控制执行过程，任务和高级中断通常被分配了一个优先级，并采用包括按优先级高低算法和时间片轮转算法进行调度。

在嵌入式操作系统 Nucleus 中，线程调度可以简单看成下面这种循环模式：

```
do
  {
   TCT_Schedule_Loop
   } while ((!TCD_Execute_HISR) && (!TCD_Execute_Task));
```

在前面几节已经提到 TCD_Execute_HISR 指向当前系统已经就绪的优先级最高的高级中断，而 TCD_Execute_Task 则指向当前系统已经就绪的优先级最高的任务。系统调度不断循环检测是否有高级中断或任务就绪；如有则进行线程调度执行相应的程序，如果没有高级中断或任务就绪，TCD_Execute_HISR 和 TCD_Execute_Task 的指针值就都为空，系统不进行线程调度操作。若高级中断和任务都同时就绪那么系统优先调度高级中断。

在系统确定就绪任务中最高优先级时，Nucleus内核能够通过几个简单的数据结构较快地找到该级别。首先系统使用掩码 0xFF 与 32 位的任务优先级组（TCD_Priority_Groups）从低到高分四个段依次进行与操作。在任务优先级组中，位越低代表地任务优先级越高，如下图所示：

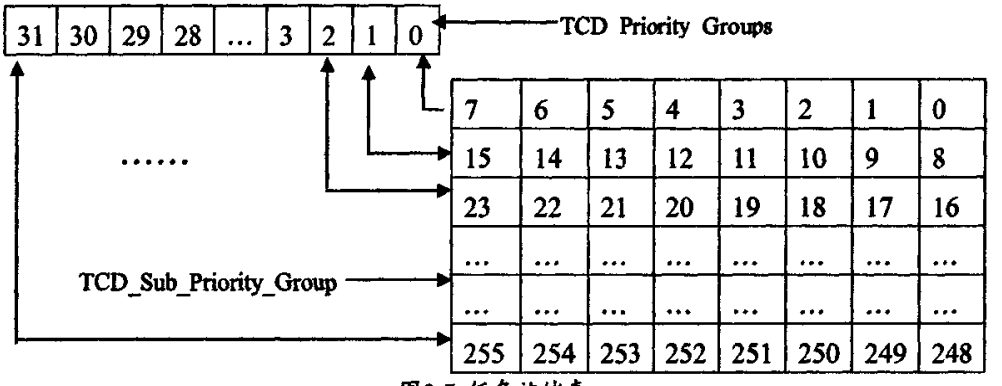

图3-7 任务就绪表

当 0xFF 和 TCD_Priority_Groups 分段进行与操作结果首次不为零时，系统便匹配最高优先级任务所在优先级子组的范围。该范围涉及到八个任务优先级子组，因此 Nucleus 内核采用一个大小为 256 的查询表 TCD_Lowest_Set_Bit 来精确查出最高优先级任务所在的子组。子组计算公式如下：

子组号＝（0xFF 匹配次数－1）×8＋查询表[TCD_Priority_Groups 匹配段]

在求出子组号后，根据子组号在 TCD_Sub_Priority_Groups 中的值再利用查询表计算出就绪任务中的最高优先级。计算公式如下：

最高优先级＝子组号×8＋查询表[子组号所在任务优先级子组中的值]

在系统确定就绪任务中的最高优先级后，通过查找优先级链表 TCD_Priority_List 可以很快地找到具备该优先级的任务，并确定 TCD_Execute_Task 指向的任务控制块 TCB。

2．2．5 操作系统数据结构的保护

由于 Nucleus 操作系统的线程是可以抢占的，高优先级的任务可以抢占低优先级的任务，HISR 可以抢占任务，HISR 之间也可以抢占。

如果某个低优先级的任务正在通过系统调用对操作系统的某个数据结构进行操作，比如：正在修改已创建任务链表 TCD_Created_Tasks_List，或正在往某个消息管道 Pipe 中填消息（要修改消息管道的数据成员），这时发生了任务抢占，如果发生抢占的高优先级任务也要修改同一数据结构，就必须等待低优先级的任务完成修改数据结构的系统调用后，再让高优先级的任务运行，否则就会破坏操作系统的数据结构。

这一机制是通过操作系统结构保护（Protect）实现的。任务运行时如果要修改操作系统的数据结构，就要通过系统调用 TCT_Protect (TC_PROTECT *protect) 把该数据结构的保护结构保护起来，当任务对该数据结构的操作结束就会通过系统调用 TCT_Unprotect (void) 或 TCT_Unprotect_Specific (TC_PROTECT *protect) 来释放保护结构。

如果低优先级的任务没有释放保护结构之前，发生了任务抢占，高优先级的任务抢占了低优先级的任务，如果高优先级的任务也要修改同一系统数据结构，那么在做系统调用 TCT_Protect 时，就会发现该数据结构的保护结构已经被另一个线程拥有，当前的线程就会暂时被挂起，将控制交给拥有保护结构的线程，等待拥有该保护结构的线程释放掉保护结构后，也就是拥有保护结构的线程在做 TCT_ Unprotect 时，高优先级的任务才能真正地把控制权抢占过来。

Nucleus 共有四种跟线程调度有关的比较重要的保护结构：

TCD_List_Protect：已创建任务链表的保护结构；

TCD_System_Protect：系统保护结构，用于任务调度；

TCD_LISR_Protect：用于 LISR 的创建和删除；

TCD_HISR_Protect：用于 HISR 的创建和删除；

保护结构的结构体是这样定义的：

```
typedef struct TC_PROTECT_STRUCT
{
    TC_TCB          tc_tcb_poiter;    /* 拥有保护结构的线程指针 */
    UNSIGNED        tc_thread_waiting;/* 有线程在等待该保护结构的标志 */
}TC_PROTECT;
```

1.数据结构的保护（TCT_Protect）

当线程在操作的数据结构不想因为线程的抢占而破坏时，就要申请对该数据结构的保护，比如：在创建 HISR 时，就要申请 HISR 链表的保护 TCT_Protect(&TCD_HISR_Protect)；在链表插入完成后，就要调用 TCT_Unprotect( )释放当前线程拥有的保护结构。

另外，如果线程正在执行的系统调用不想因为线程的抢占而中断时，就要申请系统保护结构的保护 TCT_Protect(&TCD_System_Protect)，比如：任务的恢复、任务的挂起、消息管道的收发等，这些操作都要申请系统保护结构，当系统调用结束后，就要调用 TCT_Unprotect( )释放当前线程拥有的保护结构。

TCT_Protect 首先判断要保护的结构是否被其它线程拥有，如果没有其它线程拥有要保护的结构，则将当前线线程 TCD_Current_Thread 赋给

protect->tc_tcb_pointer，表示当前线程要占用该保护结构，清除保护结构的线程等待标志，将保护结构的指针送给TCD_Current_Thread->tc_current_protect，表示当前线程拥有一个保护结构，然后返回。

如果其它线程拥有要保护的结构，则要调用TCT_Schedule_Protectd，直至其它线程释放掉该保护结构，然后重复上面的处理，表示当前线程拥有该保护结构。

2.保护结构的释放（TCT_Unprotect）

当拥有保护结构的线程处理完系统调用后，就要释放被保护的结构。保护结构的释放有两个系统调用：TCD_Unprotect 释放的是线程当前的保护结构tc_current_protect；TCD_Unprotect_Specific 释放的是用户或操作系统指定的保护结构。

TCD_Unprotect 首先判断当前线程是否拥有一个保护结构，再判断保护结构的线程等待标志，看看是否有其它线程在等待该保护结构，如果这些条件都成立，则将控制权交给TCT_Control_To_System。TCT_Control_To_System为当前线程创建一个Solicited类型的堆栈帧，清除当前线程拥有的保护结构以及保护结构上挂的线程，清除 TCD_Current_Thread，将堆栈从线程堆栈切换到系统堆栈，然后将控制权交给TCD_Schedule重新进行调度，从而引起TASK或HISR的竞争。

如果没有其它线程在等待该保护结构，则直接清除当前线程的tc_current_protect，以及保护结构的tc_tcb_pointer，然后返回。

TCD_Unprotect_Specific 首先清除指定保护结构上挂的线程tc_tcb_pointer，如果没有其它线程在等待该保护结构就直接返回。如果还有其它线程在等待该保护结构，则为当前线程建立一个Solicited类型的堆栈帧，清除当前线程 TCD_Current_Thread，将堆栈从线程堆栈切换到系统堆栈，然后将控制权交给TCD_Schedule重新进行调度，从而引起TASK或HISR的竞争（线程的抢占）。

3.保护的调度（TCT_Schedule_Protected）

保护结构的调度也就是调度拥有保护结构的线程，直至其释放保护。

TCT_Schedule_Protected 首先为想得到保护结构的当前线程建立一个Solicited 类型的堆栈帧，将拥有保护结构的线程置为当前线程TCD_Current_Thread，将堆栈切换至系统堆栈，将控制权交给TCT_Control_To_Thread，这里将恢复运行因线程抢占而挂起的拥有保护结构的线程，等其处理完相关数据结构后，就会调用TCD_Unprotect来释放保护结构，释放时就会发现还有另一个线程在等待该保护结构，然后控制权就会交给

TCD_Schedule 重新进行调度，引起线程之间的竞争。

数据结构保护的处理示意图如下：

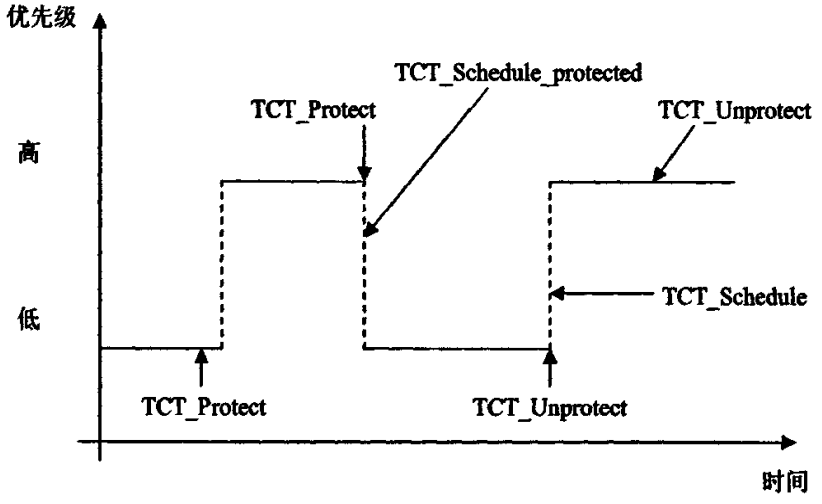

图3-8 数据结构保护示意图

注意：TCT_Schedule_Protected 并没有修改 TCD_Execute_Task 及 TCD_Execute_HISR，只是临时地将拥有保护结构的线程置为当前线程，当拥有保护结构的线程释放保护后，TCD_Schedule 又会恢复运行想得到保护结构的 TCD_Execute_Task 或 TCD_ Execute_HISR。

Nucleus 操作系统不具备优先级自动逆转的功能，它有自己的解决方法。当高优先级的线程不能获得临界系统资源时，高优先级的任务并没有挂起，所有在此优先级之下的线程就都不能运行，Nucleus 会把拥有临界系统资源的低优先级线程临时地升为当前线程 TCD_Current_Thread（相当于临时把优先级升为最高），当低优先级的线程释放临界系统资源后，高优先级的任务马上会恢复运行。

2. 2. 5 任务间通信

Nucleus 为任务间通信提供了邮箱（mailboxes）、队列（queues）、管道（pipes）。邮箱、队列、管道是独立的公共设备。任务之间和其他系统设备之间的联系由应用程序确定。这些通信设备之间主要差别是数据通信的类型。下面分别说明：

· 1. 邮箱

Nucleus 的邮箱（mailboxes）为传输简单数据提供低消耗方案。每个邮箱可以保持 4 个 32 位字大小的单一消息。消息以值的方式发送和接收。发送消息是将消息拷贝到邮箱；接受消息则是将消息从邮箱拷贝出来[16]。

任务在邮箱内可能因为几种原因挂起。一个试图从空邮箱中接受消息的任务可以被挂起。同样，一个试图发送消息至满邮箱的任务可以被挂起。当邮箱能够确保任务请求时挂起的任务恢复。例如，假设一个任务在邮箱等待接受消息时挂起。当一个消息发送到邮箱时，挂起的任务就恢复了。多任务能在一个邮箱上挂起。依靠创建邮箱的任务可以以 FIFO 或是优先级次序被挂起。如果邮箱支持 FIFO 挂起，任务按他们挂起的顺序恢复。另外，如果邮箱支持优先级挂起，任务从高优先级到低优先级顺序恢复。

邮箱可以动态地创建和删除，并且数目不受限制。

2. 队列

Nucleus 的队列（queues）提供了传输多个消息的机制。消息以数值形式发送和接收。发送消息是将消息拷贝到队列；接收消息是从队列上拷贝消息出来。一列消息包括一个或多个 32 位字，并且支持长度固定和长度可变两种消息。消息类型的格式在队列创建的时候定义。长度可变的消息队列需要为队列中的每一个附加一个 32 位字。另外，在接收消息长度可变消息队列时，要求指定队列最大消息尺寸。

任务能在队列内因为几种原因挂起。一个试图从空队列中接受消息的任务可以被挂起。同样，一个试图发送消息至满队列的任务可以被挂起。当队列能够确保任务请求时挂起的任务恢复。例如，假设一个任务在队列等待接受消息时挂起。当一个消息发送到队列时，挂起的任务就恢复了。多任务能在一个队列上挂起。依靠创建队列的任务可以以 FIFO 或是优先级次序被挂起。如果队列支持 FIFO 挂起，任务按他们挂起的顺序恢复。另外，如果队列支持优先级挂起，任务从高优先级到低优先级顺序恢复。

队列也可以动态地创建和删除，队列的数目不受限制。

3. 管道

Nucleus 的管道（pipes）提供了另一种传输多个消息的机制，消息以数值形式发送和接收。发送消息是将消息拷贝到队列；接收消息是从队列上拷贝消息出来。一列消息包括一个或多个 32 位字，并且支持长度固定和长度可变两种消息。消息类型的格式在队列创建的时候定义。长度可变的消息队列需要为队列中的每一个附加一个 32 位字。另外，在接收消息长度可变消息队列时，要求指定队列最大消息尺寸。

任务能在管道内因为几种原因挂起。一个试图从空管道中接受消息的任务可以被挂起。同样，一个试图发送消息至满管道的任务可以被挂起。当管道能够确保任务请求时挂起的任务恢复。例如，假设一个任务在管道等待接受消息时挂起。当一个消息发送到挂起时，挂起的任务就恢复了。多任务能在一个管道上挂起。

依靠创建管道的任务可以以 FIFO 或是优先级次序被挂起。如果管道支持 FIFO 挂起，任务按他们挂起的顺序恢复。另外，如果管道支持优先级挂起，任务从高优先级到低优先级顺序恢复。

管道也可以动态地创建和删除，管道的数目不受限制。

Nucleus 的队列和管道是十分类似的，不同之处在于：队列的消息按长字来访问，管道的消息按字节来访问。

2.2.6 任务的同步

Nucleus 提供了信号量（semaphores）、事件组（event groups）和信号（signals）来解决任务的同步问题。其中信号量和事件组是独立的公用设备。任务和其它系统设备的联系由应用程序来决定；而信号只和指定任务相关。下面分别说明。

1.信号量（semaphores）

信号量提供了一种系统临界资源分配的机制，它的值计算范围是 0～4294967294。信号量的两个基本操作是获取和释放，获取操作会消耗信号量，释放操作会增加信号量。通常一个信号量的大小代表着系统中某个临界资源的多少。一个任务获得了信号量就意味着它申请到某个临界资源，与此同时该信号量值就会减少，表示着系统中对应的可用临界资源减少。任务释放信号量意味着它释放了原先使用到的临界资源，因此可用临界资源增加，信号量就增加。

当一个任务试图去获取一个值为零的信号量时就会挂起，直到该信号量被释放任务才有可能恢复。当多个任务试图获得一个信号量时，任务就会根据信号量创建时支持的是 FIFO 挂起方式还是优先级挂起方式来决定任务挂起的顺序。

Nucleus 避免死锁的方法是当应用程序使用信号量时强加一些规则给应用程序，如禁止任务在同一时间占用多个信号量等。

2.事件组（event groups）

事件组提供一个机制来描述一个指定系统事件的发生。事件由事件组中单个位来描述。这位叫事件标志。每个事件集有 32 个事件标志。事件标志可以通过逻辑 AND/OR 组合被设置和清除。事件标志也可以以逻辑 AND/OR 组合接收。另外，事件标志可以在接收完后自动复位[17]。

当一个任务试图得到事件标志位未置位的事件时将会被挂起。当置位事件标志操作满足任务中事件请求组合时任务恢复。

Nucleus 可以动态地创建和删除事件组。

3.信号（signals）

信号从某种程度上说和事件组类似，但是它们在操作上有几点重要的区别。事

件组是用来同步的，信号则是以异步的方式运行。当信号出现时，任务中断并转而执行任务预先定义的信号处理子程序。每个任务可以处理32 个信号，每个信号用一个描述位来表示。

信号处理程序类似于高级中断服务程序，它不会被新的信号中断。所有新信号的处理都要在当前的信号处理完成之后才可以运行。

2. 2. 7 内存管理

嵌入式操作系统Nucleus提供两种内存管理方式，一种是分区内存管理，另一种是动态内存管理。分区内存管理具有确定性但是灵活性差，动态内存管理相反灵活性好但是确定性差。大多数应用程序对两种类型的内存管理都需要。下面分别说明。

1.分区内存池（Partition Memory Pools）

分区内存池包含一定数量的定长内存分区。内存池在内存中的位置，内存池的字节数，内存池中每个分区的字节数等都是由应用程序来确定的。各个分区在内存池中分配和回收。

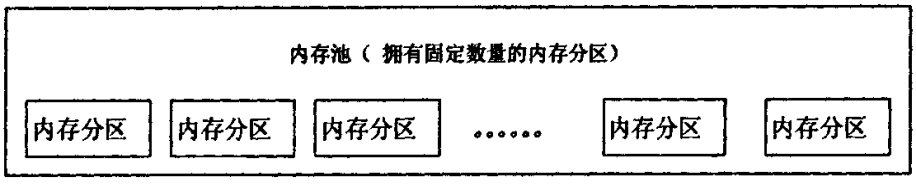

图 3-9 内存池

当一个任务试图从一个空的池中分配一个区时就会被以被挂起。直到内存池回收到内存分区时任务才可能恢复。一个分区内存池可以挂起多个任务。依据分区内存池创建时的设置，任务可以以 FIFO 或是按优先级顺序被挂起。如果分区内存池支持 FIFO 挂起，任务可以按照他们挂起的顺序恢复。同样，如果分区内存池支持优先级挂起，任务按照从高优先级到低优先级顺序恢复。

Nucleus 可以动态地创建和释放分区内存池。一个应用程序可以拥有的分区内存池个数不受限制，每个分区内存池需要一个控制块和指向内存区的指针。

2.动态内存池（Dynamic Memory Pools）

一个动态内存池包含用户指定的字节数。内存在池中的位置由应用程序决定。Nucleus PLUS 为动态内存池提供可变长度的分配和释放服务。分配以最适合的方式（first_fit_manner）运行。例如，满足要求的第一个有效内存被分配。如果分配的块比要求的大得多，没有使用的内存将被返回到动态内存池。

当一个任务试图从当前没有足够有效内存的池中分配动态内存时会被挂起。直到池中拥有足够的可分配内存时任务才可能恢复。在单个动态内存池上可以挂

起多个任务。依据动态内存池的创建方式，任务即可以 FIFO 顺序挂起也可以优先级顺序挂起。如果动态内存池支持 FIFO 挂起，任务以他们被挂起的顺序恢复；如果动态内存池支持优先级挂起，任务从高优先级到低优先级顺序恢复。

Nucleus 可以动态地创建和删除动态内存池。一个用程序可以拥有的动态内存池个数不受限制，每个动态内存池需要一个控制块和指向内存区的指针。

2.2.8 输入/输出设备驱动

大多数的实时应用程序需要不同外围设备的输入输出。这些输入输出的管理通常由一个 I/O 器件驱动器来完成。

Nucleus PLUS 为初始化、赋值、释放、输入、输出、状态和中止请求提供一个标准的 I/O 驱动器接口。这个接口由通用控制结构来实现。每个驱动器有一个唯一入口。控制结构确定被请求的服务和所有必须的参数。如果指定的驱动器需要附加参数，控制结构提供一个机制为它连接一个追加的控制结构。有一个标准接口使能应用程序来解决多种多样类似或不一样的外围器件[18]。

一个 I/O 驱动器通常处理初始化、分配、释放、输入、输出、状态和中止请求过程。如果 I/O 驱动器由中断驱动，中断处理子程序也是必须的。

Nucleus PLUS 设备可以被 I/O 驱动器使用。队列、管道和信号量通常可以被 I/O 驱动器利用。除了大多数 Nucleus PLUS 服务的可用性之外，I/O 驱动器也提供服务保护内部数据结构不被同步高优先级 ISR 访问。低优先级 ISR 同步访问的保护通过禁止适当的中断来保护。

I/O 驱动器可以被系统中不同的线程调用。如果一个 I/O 驱动器被一个任务线程调用，与其他 Nucleus PLUS 设备相关的挂起设备有效。另外，一个服务被提供用来挂起和清除 HISR 同步保护。

Nucleus I/O 驱动器可以动态创建和删除。一个应用程序拥有的 I/O 驱动器数量没有预先限定。每个 I/O 驱动器需要一个控制块。控制块内存由应用程序提供。创建和删除驱动器子程序并没有真正的调用驱动器。完成不同的调用进行初始化和中止驱动器。

第三章 Nucleus与其它嵌入式操作系统的比较研究

随着嵌入式系统领域不断的发展，越来越多的 IT 组织、公司都开始进行商用嵌入式操作系统和专用操作系统的研发并陆续推出了自己的产品。除 Nucleus 嵌入式操作系统外，商业中常用的著名嵌入式操作系统还有 VxWorks、uClinux、Windows CE、uC/OS II 等[19]。其中 uClinux、uC/OS II 和 Nucleus 都是性能优良、源码公开且被广泛应用的嵌入式操作系统。同时，它们也是作为研究嵌入式操作系统的典范。

3.1 几款嵌入式操作系统简介

uCLinux 是一款优秀的专为微控制系统而设计的嵌入式 Linux 操作系统。它在标准的 Linux 基础上进行了适当的剪裁和优化，形成了一个高度优化的、代码紧凑的系统[20]。虽然，uCLinux 的体积很小，但仍然保留了标准 Linux 操作系统的大多数优点：稳定性、出色的网络功能、完备的对各种文件系统的支持，以及标准丰富的 API 等[21]。但是由于 uCLinux 没有 MMU(内存管理单元)模块，因而它不能使用处理器的虚拟内存管理技术，而采用是实存储器管理策略[22]。uCLinux 具备良好的移植性，可被移植到多个平台；但是需要注意的是它本身不具备实时性，需要靠 RT 补丁解决[23]。

嵌入式实时操作系统 uC/OS II (micro-controller operating system) 是一款基于优先级的抢占式实时多任务操作系统。它适合于小型控制系统，具有执行效率高、占用空间小、实时性能优良和可扩展性强等特点[24]。其内核包含有任务管理、时间管理、任务间通信同步（信号量，邮箱，消息队列）和内存管理等功能模块并提供可确定性的服务调用。内核绝大部分代码用 ANSI C 语言编写，与硬件相关部分用汇编语言编写且已压缩到最低限度[25]。同样，uC/OS II 拥有良好的移植性，可以移植到绝大部分平台上。

Nucleus 嵌入式实时操作系统在前面章节中已经进行了详细介绍，这里不再重复。就这三款嵌入式操作系统综合性能而言，Nucleus 操作系统介于 uCLinux 操作系统和 uC/OS II 操作系统之间：uCLinux 性能>Nucleus 性能>uC/OS II 性能。

3.2 进程调度策略比较

uClinux 的进程调度沿用了 Linux 的传统，系统每隔一定时间挂起进程，同

时系统产生快速和周期性的时钟计时中断，并通过调度函数(定时器处理函数)决定进程什么时候拥有它的时间片[26]。然后进行相关进程切换，这是通过父进程调用 fork 函数生成子进程来实现的[27]。uClinux 系统 fork 调用完成后，要么子进程代替父进程执行(此时父进程已经 sleep)，直到子进程调用 exit 退出；要么调用 exec 执行一个新的进程，这个时候产生可执行文件的加载，即使这个进程只是父进程的拷贝，这个过程也不可避免。当子进程执行 exit 或 exec 后，子进程使用 wakeup 把父进程唤醒，使父进程继续往下执行。 uClinux 由于没有 MMU 管理存储器，其对内存的访问是直接的，所有程序中访问的地址都是实际的物理地址。操作系统队内存空间没有保护，各个进程实际上共享一个运行空间[28]。因此在多个进程运行时要注意数据保护；同时用户程序使用的空间也可能占用到系统内核空间，这些问题在应用开发时都需要多加注意，否则容易导致系统崩溃。

作为实时操作系统，uC/OS II 是采用的可剥夺型实时多任务内核。可剥夺型的实时内核在任何时候都运行就绪了的最高优先级的任务。uC/OS II 中最多可以支持 64 个任务，分别对应优先级 0~63，其中 0 为最高优先级[29]。调度工作的内容可以分为两部分：最高优先级任务的寻找和任务切换。 其最高优先级任务的寻找是通过建立就绪任务表来实现的。uC/OS II 中的每一个任务都有独立的堆栈空间，并有一个称为任务控制块 TCB(Task Control Block)数据结构，其中第一个成员变量就是保存的任务堆栈指针。任务调度模块首先用变量 OSTCBHighRdy 记录当前最高级就绪任务的 TCB 地址，然后调用 OS_TASK_SW()函数来进行任务切换，执行优先级最高的任务[30]。

嵌入式实时操作系统 Nucleus 的内核和 uC/OS II 的内核属于同一种类型的内核，均为可剥夺实时多任务型，因而它们在内核结构上有一定的相似之处：

1.数据结构方面

Nucleus 和 uC/OS II 都采用任务优先级组（用符号_GRP 表示)和任务优先级子组（用符号_SUB_GRP 表示)来记录系统当前就绪任务的优先级。由于 uC/OS II 系统最多支持 64 个等级的任务优先级，_GRP 在 uC/OS II 中只需要一个八位的数据表示；然而 Nucleus 支持的任务优先级最多可以达到 256 级，因此_GRP 在 Nucleus 中需要一个 32 位的数据来表示。同理，_SUB_GRP 在 uC/OS II 中长度为 8 而在 Nucleus 中长度为 32。uC/OS II 操作系统每一个任务优先级下只有允许有一个就绪任务存在因此它最多支持 64 个任务同时运行；Nucleus 对任务的数量却没有限制，因此它需要使用任务优先级链表 TCD_Priority_List[256]来记录每一级优先级下的任务。

2.任务调度策略方面

Nucleus 与 uC/OS II 在任务调度上都是按就绪任务的优先级高低次序调

用。然而由于uC/OS II在每个任务优先级下只允许一个就绪任务存在而使得它不支持时间片任务调度算法；Nucleus 显然在这一方面更具优势，它不仅仅支持任务的优先级调度，并且在多个就绪任务优先级相等的情况下它采用时间片调度的算法从而保证任务的实时运行。此外，Nucleus 与uC/OS II相比任务还多一个标志性属性，它表示着该任务是否允许被其它高的就绪任务所抢占；这样在Nucleus中任务的灵活性就大大增加。

3.中断方面

Nucleus 和uC/OS II在中断上最大的不同就是：Nucleus 内核将中断分为高级中断和低级中断。高级中断由低级中断激活并像其它普通任务一样具有优先等级并参与和普通任务的竞争调度。

4.组件结构方面

Nucleus 和uC/OS II在内核中都具有信号量，邮箱，队列等功能模块；但是二者结构上却很有很大的区别。从文件上看uC/OS II 每个功能模块使用一个文件编写完成，而 Nucleus 采用软件组件的方法来管理各个功能模块，每个模块都是多个文件构成，但是它们拥有统一的文件命名方法、统一的组件结构、统一风格的对外接口。Nucleus 功能组件大多数都有自己的控制块和任务挂起列表；而uC/OS II在这些功能模块组成上则有所不同，其内核提供的服务必须和事件控制块挂钩，利用事件控制中的任务挂起表来进行任务挂起、恢复等操作。

由上述分析可以得知，Nucleus 和uC/OS II内核是针对实时系统的要求设计实现的，可以满足较高的实时性要求。而uClinux则在结构上继承了标准Linux的多任务实现方式，仅针对嵌入式处理器特点进行改良。其要实现实时性效果则需要使系统在实时内核的控制下运行。

3.3 支持文件系统比较

所谓文件系统是指负责存取和管理文件信息的机构，也可以说是负责文件的建立、撤销、组织、读写、修改、复制及对文件管理所需要的资源(如目录表、存储介质等)实施管理的软件部分。

Nucleus 和uC/OS II是面向中小型嵌入式系统的，它们包含全部的内核功能(信号量、消息邮箱、消息队列及相关函数)，但是其系统本身并没有对文件系统的支持。但是Nucleus 和uC/OS II都具有良好的扩展性能，并且二者的产品公司也各自推出了自己的文件系统组件，如果实际开发当需要的话可以自行添加文件系统。

uClinux则是继承了Linux完善的文件系统性能。其采用的是romfs文件

系统，这种文件系统相对于一般的ext2文件系统要求更少的空间。空间的节约来自于两个方面，首先内核支持romfs文件系统比支持ext2文件系统需要更少的代码，其次romfs文件系统相对简单，在建立文件系统超级块(superblock)需要更少的存储空间[31]。Romfs文件系统不支持动态擦写保存，对于系统需要动态保存的数据采用虚拟ram盘的方法进行处理(ram盘将采用ext2文件系统)。uClinux还继承了Linux网络操作系统的优势，可以很方便的支持网络文件系统且内嵌TCP/IP协议，这为uClinux开发网络接入设备提供了便利[32]。

由上诉操作系统对文件系统的支持可知，在复杂的需要较多文件处理的嵌入式系统中uClinux是一个不错的选择。而uC/OS II则主要适合一些控制系统；Nucleus则介于二者之间。

3.3 系统的移植比较

嵌入式操作系统移植的目的是指使操作系统能在某个微处理器或微控制器上运行。Nucleus、uC/OS II和uClinux都是源码公开的操作系统，且其结构化设计便于把与处理器相关的部分分离出来，所以被移植到新的处理器上是可能的。　以下对三种系统的移植分别予以说明。

1. uC/OS的移植

要移植uC/OS，目标处理器必须满足以下要求[33]：

- 处理器的C编译器能产生可重入代码，且用C语言就可以打开和关闭中断；
- 处理器支持中断，并能产生定时中断；
- 处理器支持足够的RAM(几K字节)，作为多任务环境下的任务堆栈
- 处理器有将堆栈指针和其他CPU寄存器读出和存储到堆栈或内存中的指令。

在理解了处理器和C编译器的技术细节后，uC/OS的移植只需要修改与处理器相关的代码就可以了。具体有如下内容：

- OS_CPU.H中需要设置一个常量来标识堆栈增长方向；
- OS_CPU.H中需要声明几个用于开关中断和任务切换的宏；
- OS_CPU.H中需要针对具体处理器的字长重新定义一系列数据类型；
- OS_CPU_A.ASM需要改写4个汇编语言的函数；
- OS_CPU_C.C需要用C语言编写6个简单函数；
- 修改主头文件 INCLUDE.H，将上面的三个文件和其他自己的头文件加入。

2. Nucleus的移植

Nucleus 嵌入式操作系统内核代码95%是ANSI C编写，因此移植起来非常地方便。依赖与目标板和开发工具的 Nucleus 文件仅有独立的四个文件。其中三个是使用汇编语言编写的。这些文件为目标环境提供底层的基本运行平台。另一个文件是Nuclues.h，它直接或间接地被系统所有文件包含。该文件定义了大量的数据类型和处理器、开发工具相关的信息。

三个与目标板相关的汇编文件分别是：INT.[S，ASM，SRC]、TCT.[S，ASM，SRC]、TMT.[S，ASM，SRC]。

（1） INT.[S，ASM，SRC] 文件

在前面章节我们曾说明 INT.[S，ASM，SRC]文件是系统的启动文件，它负责系统底层的初始化包括定时中断、可用存储空间和其它处理器或着目标板的特定实体。

（2）TCT.[S，ASM，SRC]文件

文件 TCT.[S，ASM，SRC]主要负责在系统中进行线程控制权的转移。线程被定义为一个Nucleus任务或一个Nucleus高级中断。这个文件包含在任务和高级中断进行上下文切换的所有必须的代码。另外该文件也包含竞争冲突和任务信号量的所有需要的代码。

（3）TMT.[S，ASM，SRC]文件

文件TMT.[S，ASM，SRC]主要负责处理Nucleus的定时中断，包括定时器中断处理函数。在许多端口中，当没有时钟发出时，中断定时处理程序用于处理频繁的低级操作。

（4）Nucleus.h文件

NUCLEUS.H 是NUCLEUS 操作系统头文件。以NUCLEUS 操作系统服务或数据类型为参考编写的应用文件必须包含它。该文件定义变量数据类型、中断关闭或开放值、中断向量数目、系统控制块大小以及其他目标指定信息。

3.uClinux的移植

由于uClinux其实是Linux针对嵌入式系统的一种改良，其结构比较复杂，相对 uC/OS，uClinux 的移植也复杂得多。一般而言要移植 uClinux，目标处理器除了应满足上述 uC/OS 应满足的条件外，还需要具有足够容量(几百K字节以上)外部ROM和RAM。

uClinux的移植大致可以分为3个层次：

（1）结构层次的移植

如果待移植处理器的结构不同于任何已经支持的处理器结构，则需要修改 linux/arch 目录下相关处理器结构的文件。虽然 uClinux 内核代码的大部分是独立于处理器和其体系结构的，但是其最低级的代码也是特定于各个

系统的。这主要表现在它们的中断处理上下文、内存映射的维护、任务上下文和初始化过程都是独特的[34]。这些例行程序位于linux/arch/目录下。由于Linux所支持体系结构的种类繁多，所以对一个新型的体系，其低级例程可以模仿与其相似的体系例程编写。

（2）平台层次的移植

如果待移植处理器是某种uClinux已支持体系的分支处理器，则需要在相关体系结构目录下建立相应目录并编写跟踪程序(实现用户程序到内核函数的接口等功能)、中断控制调度程序和向量初始化程序等相应代码。

（3）板级移植

如果你所用处理器已被uClinux支持的话，就只需要板级移植了。板级移植需要在 linux/arch/platform/中建立一个相应板的目录，再在其中建立相应的启动代码crt0_rom.s或crt0_ram.s和链接描述文档rom.ld或ram.ld就可以了。板级移植还包括驱动程序的编写和环境变量设置等内容。

第四章 Nucleus 系统改进研究

4．1一种新调度算法的提出

在嵌入式实时系统设计中，任务的调度算法无疑是一个研究的重点。Nucleus操作系统内核是一种基于优先级的抢占式实时多任务内核。任务的优先级往往是用户根据截止期最早优先或价值最高优先等算法来确定。在一般的情况下，这些任务的优先级确定方法都表现出了较好的性能。但是某些时刻如系统过载，这些算法的性能就会急剧下降，甚至出现多米诺现象[35]。原因是经典算法中任务的优先级仅仅是由某一个特征参数来确定的。然而在实际问题当中，任务的优先级通常要由多个特征参数来共同确定，如截止期、价值等[36]。因此，人们又进一步研究了如何利用多个特征作为任务的独立特征参数建立任务的优先级。如文献[37]提出一种加权的最早截止期价值密度优先算法；文献[38]提出了截止期一价值优先级表算法。这些方法的局限性主要在于参考的任务特征参数只能是两到三个，很难向采用多特征参数方向扩展。

此外在实际问题中，还常常遇到的一个困难是任务的许多特征常常是不确定的、模糊的。例如任务的执行时间通常是不准确的，从而空闲时间也是不准确的；任务的关键性是相对的，通常不太适合用某个确切的数值来区分不同任务之间的关键程度。所有这些特征都可以看成是模糊不确定的。为此，文献[39]提出了一种静态的扩展EDF 模糊实时调度算法，文献[40]提出了一种扩展LSF、动态和非抢占式的模糊实时调度算法。文献[41]提出了一种基于单层模糊综合评判的模糊动态抢占算法。同时这些算法也有着各自的不足。前两种调度算法中除了考虑的任务参数少之外，它们描述任务特征的函数也多是确定性的，这并不能很好的体现特征参数的模糊性。模糊动态抢占算法有所改进；但是当任务特征参数较多时；一方面使权重系数的决定较难，另一方面，由于要满足“归一化”（即$\sum w_i = 1$）的要求，使每一特征参数分得权重系数值很小，不易确定任务的优先等级。再加上合成运算中，取大(∨)、取小(∧)这两个运算往往会丢失许多有价值的信息，结果导致任务优先级分辨率很差，甚至有时评判不出任务优先级。

为了解决上述的问题，本文提出了一种基于模糊理论的调度算法。该算法针对任务中多个模糊不确定的特征参数，利用多层模糊综合评判模型和最大隶属度原理来确定任务的优先级。不失一般性，本文在讨论基于模糊理论的任务优先级设计方法时采用了任务的价值、空闲时间和关键性三个特征参数来分析说明。这种设计思想可以推广到其它任意多种特征参数进行任务优先级的综合设计中。本

文通过举例仿真与其它的几种算法进行了比较分析。

4．1．1 任务模型

在描述具体的调度算法之前，首先给出后面讨论中涉及到的有关任务特征参数的定义与说明。不失一般性，假设第 i 个任务T_i具有一下参数：

①a_i表示任务的到达时间；

②d_i表示任务的绝对截止期；

③$WCET_i$表示任务的最坏执行时间；

④p_i表示任务的优先等级；

⑤ V_i表示由任务的各个特征参数隶属度值向量所构成的矩阵；

当任务到来时，记任务为T_i（a_i，d_i，$WCET_i$，V_i）。此外，算法还使用到一下参数来掌握任务调度的整体情况：

①R 表示任务的特征参数集合；

②COUNT 表示任务的总数量；

③INVAL 表示任务优先级评判失效的个数；

④AVE 表示任务优先级评判失效率；

⑤LINE 任务优先级评判失效率上限。

4．1．2 任务模型模糊调度算法

1．任务特征参数模糊化处理

在前面曾提到任务的特征参数常常是不确定的、模糊的；因此需要将其进行模糊化处理。假设任务的某个特征参数R_j的论域为X，并在论域上设有n个模糊集合：

$$A_1, A_2 ..., A_n \in F(x) \tag{4-1}$$

每个模糊集合的隶属函数可以根据实际情况取不同的函数，如三角函数、梯形函数、样条函数等；有时候出于简单考虑，也可以让n个模糊集合隶属函数直接采用三角正交函数。这样可以得到任务在特征参数R_j上一组隶属度值；用向量表示如下：

$$R_j(T) = (u_{j1}, u_{j2}, ...u_{jk}...u_{jn}) \tag{4-2}$$

其中u_{jk}表示任务T第j个特征参数上第k个模糊集合的隶属度值。在此，约定任务

所有的特征参数在其论域上所划分的模糊集合数目相同，为n个；并且单从任务的某一个特征参数来考虑，A1表示优先级最高，An表示优先级最低。不难发现，在这里每一个模糊集合实际上表示了任务一个方面的优先等级。这样任务的优先级其实就已经划分成了n个等级；因此，n的取值应该事先由系统所需的优先级等级个数来确定。

2．任务优先级的确定

当对任务T各个特征参数进行模糊化处理后，便可以利用多层次模糊综合评判对任务进行模糊综合分析确定其优先级。分层法的要点就是将任务的特征参数按某些属性分成几类并再它们之间确定权重分配，于是可以进行综合评判。而这里的每一方面的评价又是低一层次的多特征综合的结果。同样，低一层次的评价也可以是更低层次的多因素的综合。由此，可以得到多特征参数的任务优先级模糊综合评价模型。

多层次模糊综合评判的步骤如下：

（1） 将任务的特征参数集R={$r_1,r_2,\cdots,r_t$}分成s个子集

$$R_i\{r_{i1},r_{i2},\cdots,r_{im}\},\quad i=1,2,\cdots s \tag{4-3}$$

满足条件：

① $\sum_{i=1}^{s} m_i = t$，

② $\bigcup_{i=1} R_i = R$ ，

③ $R_i \cap R_j = \varnothing$，$i \neq j$

（2）对每一个因素R_i，i=1，2，…s ，分别做出综合决策。记

$$A_i = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & & \vdots \\ u_{q1} & \cdots & u_{qn} \end{bmatrix}$$

再考虑到各个不确定特征在任务优先级综合评判中的重要程度，分配权重向量为

$$w=(w_1 w_2 \ldots w_q) \tag{4-4}$$

同时权系数满足：①$0 \le w_k \le 1$，k=1，2，…q；　②$w_1+w_2+\cdots+w_q=1$。

这样就得到一级评判向量：

$$B_i = w \circ A_i = (b_{i1},b_{i2}\cdots b_{in}),\ i=1,2,\cdots,s \tag{4-5}$$

（3 ）将每一个R_i视为一个因素，记R_i={$R_1,R_2,\cdots,R_s$}，这样R又是一个因素

集，

$$C=\begin{bmatrix}B_1\\B_2\\\vdots\\B_s\end{bmatrix}=\begin{bmatrix}b_{11}&b_{12}&\cdots&b_{1n}\\b_{21}&b_{22}&\cdots&b_{2n}\\\vdots&\vdots&&\vdots\\b_{s2}&b_{s2}&\cdots&b_{s2}\end{bmatrix}$$

每个R_i作为R的一部分，反映了R的部分属性，可以按它们的重要性给出权重分为

$$w=(w_1w_2...w_s),\quad w_1+w_2+...+w_s=1$$

于是得到二级评判向量：

$$w\circ C=(c_1,c_2,\cdots c_k\cdots c_n) \tag{4-6}$$

任务的优先级即为该向量中最大值的下标值$p_i=\max_k\{c_k\}$。如果$c_1,c_2,\cdots c_k\cdots c_n$中出现几个相等的最大值时，称该任务优先级评判失效，并用优先级失效率AVE来表示系统任务优先级评判失效的整体情况。若AVE值过大，应该重新构建多层模糊综合评判，使模糊评判层数更加多一些。

出于简单考虑在此只讨论了二级评判模型。如果一级因素集R_i，i=1，2，…s，仍含有较多的因素，还可以将R_i继续细分，同理推倒出三级模型、四级模型等等。

3．调度策略

（1）任务按照优先级从高到低的顺序排列。优先级越高的任务越先得到调度；等级越低的任务越在最后调度。系统每次调度任务队列中的排在最前面的一个任务。

（2）当几个任务优先级相等时任务按照截止期从小到大排列。

（3）当新任务到来时，如果新任务的优先等级比正在执行的任务优先等级高或是与正在执行的任务等级相同但是截止时间不同时，则新任务抢占当前执行的任务。

4．算法步骤

STEP 1 确定任务需要考虑的不确定特征参数、优先等级个数、每个不确定特征参数在论域上的模糊集合以及相应的隶属函数。任务优先级多层模糊综合评判的层数为一。

STEP 2 确定一层或多层模糊综合评判中任务各特征参数权重分配；初始化AVE为零、LINE为0.15。

STEP 3 当新任务到来时，确定任务的优先等级。

STEP 3.1 在优先级判定过程中，如果出现任务优先级评判失效跳转至STEP 3.2，否则将新任务送入到相应的优先等级队列中。

STEP 3.2 重新计算优先级评判失效率(ave)并检查是否大于等于上限(line)，若ave≥line则重新构建多层模糊综合评判模型，其层数在原有层数上增加一层并跳转至STEP 2并重新评判所有任务优先级；若ave<line则取评判向量中几个最大值所对应优先级中最高的优先级作为新任务的优先级，并将它送入到相应的优先等级队列中。

STEP 4 如果CPU空闲，则调度队列中排在最前面的一个任务执行；如果已有任务在执行，则查询是否有更高优先级的任务或是有同等级的任务但是其截止期更早；若存在这样的任务，就让它抢占当前正在执行的任务；若没有，当前执行的任务则继续执行。

STEP 5 跳转至STEP 3，进入任务调度循环。

5. 评价指标

①截止期错误率(MDP)=截止期错误的任务数/任务的总数。

② 重要任务的截止期错误率(CMDP)=截止期错误的重要任务的个数/重要任务总个数。

③ 任务优先级评判失效率(AVE)= 任务优先级评判失效数(INVAL)/任务总数量(COUNT)。

4.1.3 任务模型模糊调度算法性能分析

记有5个任务，分别是：$T_1(0,9,4,V_1)$、$T_2(1,6,1,V_2)$、$T_3(1,10,2,V_3)$、$T_4(2,8,5,V_4)$、$T_5(2,7,3,V_5)$。为了简便起见，在此将系统任务优先级只划成三等；并只考虑任务的三个不确定特征参数：关键性、价值和空闲时间，它们定义域上的模糊集合隶属度函数图1所示：

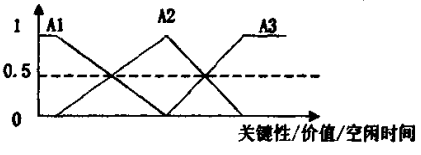

图4-1 不确定特征的隶属函数

其中，任务中向量数组的隶属值如下：

$$V_1=\begin{bmatrix}0 & 0.6781 & 0.3219\\ 0 & 0.9443 & 0.0559\\ 0 & 0 & 1\end{bmatrix} \qquad V_2=\begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0.8235 & 0.1765 & 0\end{bmatrix}$$

$$V_3=\begin{bmatrix}1 & 0 & 0\\ 0.2100 & 0.7900 & 0\\ 0 & 1 & 0\end{bmatrix} \qquad V_4=\begin{bmatrix}0.5972 & 0.4028 & 0\\ 0.7834 & 0.2166 & 0\\ 0.9150 & 0.0850 & 0\end{bmatrix}$$

$$V_5=\begin{bmatrix} 0 & 0.4231 & 0.5769 \\ 0 & 0 & 1 \\ 0.5024 & 0.4976 & 0 \end{bmatrix}$$

下面用三种调度方案来调度这五个任务：

方案一：采用参考文献[41]的模糊动态抢占算法。它是利用单层模糊综合评判和最大隶属度原理来判别任务的优先等级。取分配权值为w=(0.25　0.35　0.4)则有：

$$p_1=\max_k(w\circ V_1)=\max_k(0 \quad 0.500 \quad 0.500),$$

此时按方案一无法判断出任务T1的优先级，它的等级既可以是二级也可以是三级。因此，方案一失效。

方案二：采用本文提出的模糊调度算法，由于在一级综合评判中任务优先级失效率AVE=1/5=0.2>LINE；所以这里直接采用二级综合评判并在二级评判中分别采用权值分配 $w_1=(0.46 \quad 0.54)$、$w_2=(0.57 \quad 0.43)$。由计算可得 p_1=3、p_2=1、p_3=2、p_4=1、p_5=3。

由此，可以判断出：优先级为一等的有任务T2和T4；为二等有T3；为三等的有T1和T5。

方案三：采用EDF策略，不考虑任务特征参数的模糊不确定性。则按任务截止期由早到晚的顺序排列为：T2、T5、T4、T1、T3

图2给出了三种方案的调度过程，如下：

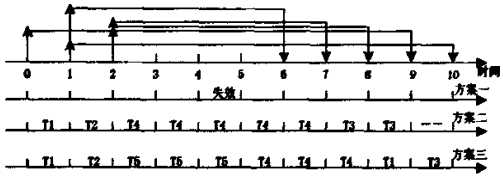

图4-2 三种方案的调度过程

针对任务特征参数具有多和不确定性的特点，利用多层模糊综合评判模型和最大隶属度原理等模糊技术提出了一种动态的多层综合评判算法，较好地解决了任务在特征参数多和不确定条件下的实时调度问题。仿真表明，该算法与传统的EDF 算法相比大大提高了重要任务调度的成功率；与使用单层模糊综合评判任务优先级的算法相比，很好地降低了任务优先级评判失效率。

4．1．4 随机任务调度试验

记Ran(x, y)表示取在x和y之间的随机整数。可以获得一组随机任务，任务总数COUNT=120，取a=Ran (0, 3)、d=Ran (5, 300)、WCET=Ran (1, 6)。任务级数的

划分仍为三级，并且各个域上的模糊集合隶属函数仍然采用正交三角函数；则可得表1，它分别给出了三种方案性能测量指标的比较，如下：

表 1 性能测量指标

调度方案	性能指标		
	MDP (%)	CMDP (%)	AVE (%)
方案一	22.10	1.95	9.2
方案二	21.17	1.89	0
方案三	26.35	27.16	—

从表中可以得知三个方案在平均任务截止期错误率这个指标上相差不大；但是在重要任务截止期错误率这个指标上，方案一、二则比方案三小了很多。这是因为方案一和方案二在权值分配时给任务关键性特征参数以较高的权值，强调了任务的重要性。同时不难看出，方案一、二在前三个指标值上都很接近，这是因为二者的算法本质一样都是采用模糊综合评判来确定任务的优先级别。而方案二算法除此之外还解决了方案一算法所遇到的任务优先级评判失效的问题；因而在指标AVE上的值达到了零。

4. 2 Nucleus 的容错功能扩充

Nucleus嵌入式操作系统和其它大多数商用嵌入式实时操作系统一样，不具有容错功能。而在某些特殊领域，系统的容错能力却是必不可少的。因此研究系统的容错能力，并利用Nucleus的可扩充性实现一种容错操作系统具有非常现实的意义。

4. 2. 1 容错技术设计

软件容错技术反映软件规避系统错误的能力。它是指当系统运行时出现未预料到的错误或故障时，为避免系统故障或损坏而在软件上采取的捕捉、处理错误的方法[42]。

容错计算技术是利用“冗余”思想屏蔽故障影响，提高计算机系统可靠性的重要手段[43]。系统中采用的FTM（fault-tolerance mechanism）是利用容错计算技术进行故障处理的重要机构。计算机系统按抽象级别的不同，可描述为由若干层次构成的多级层次结构，如由低至高可抽象为硬件、系统软件、应用软件3 层[44]。 其中每一级中均可能发生故障。系统中所采用的每一种FTM 均可视为位于该层次结构的某一级上，功能是阻止该级中产生的某些故障向上一级传播，或至少检测到该故障的发生，并向较高层次传递有关信息，以备在较高级上处理[45]。FTM所实现的故障处理功能包括检测、屏蔽、重试、重组、恢复、重构等。 编码

技术、监视定时器(watchdog)、比较、表决、备份、卷回、切换、多样性设计等是典型的FTM。

这里对Nucleus容错功能的扩充也是采用三级容错的管理模式[46]。第一级容错通过硬件的自检测、自切换， 防止CPU 故障，为系统提供软件基本正常工作的环境。第二级容错进行I/O 测试，三机同步、仿作表决，选择正确的输出，屏蔽故障机的输出，同时根据故障情况进行故障处理，系统的降级与重构、升级、恢复等工作。第三级容错指软件的容错功能，对出错的函数进行分离和重建，对关键的数据进行多重备份，多机比较，对关键的任务进行时间差多数表决等。

4. 2. 2 Nucleus 容错功能实现方式

Nucleus 三级容错功能扩充首先定义一组全局系统变量，用于记录系统状态(如系统硬件状态、软件状态)，该状态变量由巡检任务或故障检测中断处理服务程序设置， 由系统调用函数读取并传递给应用程序[47]。具体实现如下：

1.第一级容错

该层中通过一组周期性的容错处理任务负责操作系统的第一层容错功能，为系统提供一个正常的工作环境。周期性的巡检任务包括：

（1）存储器检查任务；

（2）周期自检任务，完成CPU 自检、存储器单位错、双位错、时钟正确性检查、电源状态检查等检查；

（3）本机状态检查任务。

2.第二级容错

在I/O 处理程序，中断处理程序ISR 增加容错处理功能，提供系统第二级的容错功能，完成系统降级与重构、升级、恢复等工作。故障检测中断处理程序：许多故障是由中断的形式反映出来的， 各中断处理程序负责将故障状态报告故障记录与处理软件。I/O 处理程序，中断处理程序ISR，应对数据采样进行严格的软件滤波，根据需要可采用三取二或更严格的10 取10，并进行三机比对表决，充分利用硬件的冗余资源进行软件容错，进而实现故障屏蔽和故障隔离。

3.第三级容错

增加一组系统调用函数供应用程序调用， 对关键的数据进行多重备份[48]，多机比较， 对关键的任务进行时间差多数表决等来提高软件的容错功能，实现第三级容错。系统调用接口函数供应用程序调用， 应实现以下功能：

（1）系统运行状态查询函数；

（2）系统控制函数：当系统出现难以检测的故障时， 应用任务通过调用该函数改变机器的运行状态或控制机器进入安全模式或降级运行；

（3）三机表决接口函数：设置一个系统调用来实现三机之间关键任务的表决与同步。

关键任务在表决点进行表决时， 首先形成某种形式的表决值，系统将本机的表决信息块传送给另外两机， 并接收另外两机传来的表决值，通过比较，形成表决结果返回给调用者。在传送表决值的同时，也将本机的自检状态传送给其他两机，用以进行辅助表决。在三机互不相同或另外两机表决信息超时时，通过自检状态确定本机结果的正确性。

第五章 基于 Nucleus 的应用开发

5.1 开发简介

通过上述章节，我们知道嵌入式操作系统 Nucleus 具有功能模块化、代码重用性和开放性高、简单易用等特性。也正是因为这些优良的特性使得 Nucleus 成为嵌入式应用开发的理想平台。Nucleus 能够支持 x86、MIPS、PowerPC、ARM 系列[49]、Strong ARM、TMS320CXX 等多种处理器，主要的应用领域有，网络数据通讯产品（如路由器、桥接器等）、蜂窝电话、多媒体、汽车、医疗仪器、工业控制，以及航天航空和军事等。其中在蜂窝电话制造领域，嵌入式操作 Nucleus 得到了很好的应用。

另一面，随着社会的进步，生活水平的提高；人们越来越关心自身的健康问题，特别是一些由工作压力等因素所导致的常见疾病，如高血压、心律不齐等。治疗这些疾病的关键是要早发现早治疗。所以，人们需要一种使用起来方便并能实时测量人体生理指标（如脉搏和体温）的设备。全新开发一个专用于此的仪器当然可行，但是如果我们能通过改进现有电子产品，向其中添加人体检测功能，将可以大大节省全新开发的费用并提高现有产品的使用率。嵌入式产品——手机因其普及率高、使用率高和便于随身携带很好地满足了需求中对方便性和实时性的要求。因此，完全可以利用优秀的嵌入式操作系统 Nucleus 来开发手机中的人体健康检测功能，以满足人们的这种需求。

在本次应用开发中，开发小组选择了台湾联发科技公司的一套手机方案作为手机端的开发。同时开发小组还设计了一套小型的手机附件仪器用于采集人体生理参数。附件通过与手机相链接，利用手机现有的通信功能将测量值传送至手机之中。最后在手机端，通过数据分析、比较获得人体生理参数测量结果以及人体综合健康评估。

5.1.1 手机基带芯片简介

传统手机硬件可以分为两大模块：一类是基带模块和另一类是射频模块。其中，基带模块主要由 DSP、微控制器、内存（如 SRAM、Flash）等单元组成，主要功能是基带编码/译码、声音编码及语音编码等。而射频模块，简单地说就是与无线电波发送或接收相关的硬件电路。

MT6219 是台湾联发科技公司推出的一款基于 32 位 ARM7EJ-S 内核的 RISC 基带处理器。它不仅支持 GSM、GPRS、WAP 等基本无限通信功能；而且内置了 1.3M

摄像头处理IC，具备了MP3和MP4等多媒体功能。通过16位的Host接口，MT6219最多可支持8个扩展存储设备。每个扩展存储设备可是8位数据接口也可以是16位的数据接口，最大存储容量为64M字节。扩展设备的类型可以多种多样，MT6219支持burst Flash、page Flash、page SRAM、Pseudo SRAM等等。为了减少电流消耗和降低噪音，MT6219该接口的I/O电压被设计成可变电压，它的最低供电电压为1.8v。MT6219整合了所有必须的外围模块以用作用户的接口。如用于键盘扫描器、SIM控制器、警报器、时钟、脉宽调制、LCD控制器等。而为了满足连接传输和数据存储的要求，MT6219支持UART、IrDA、USB1.1等连接传输方式，并支持MMC、SD、MS等存储卡。同时该处理器还提供了高效DMA工作方式和硬件流控制器来实现大量数据传输。

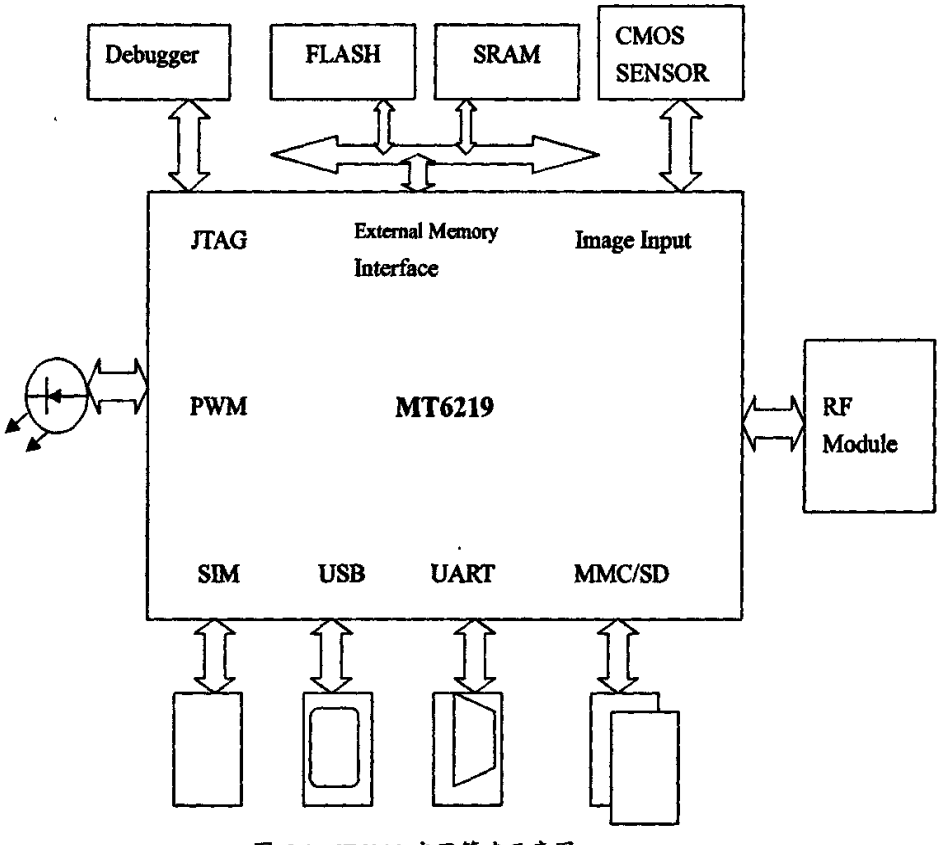

图5-1 MT6219应用简略示意图

5.1.2 手机软件简介

手机软件可以按照软件层次结构从底层到高层依次划分为：实时操作系统及硬件驱动、手机协议栈软件、手机应用程序、人机接口（MMI）。功能上包括了操

作系统封装接口管理、文件系统管理、历史管理、图片管理、通讯功能、时钟、多媒体播放、数码摄像等等。

实时操作系统及硬件驱动层：主要包括操作系统内核、目标环境的基本设置、各种硬件设备的驱动（如LCD、摄像头、Flash）等等。

手机协议栈层：包含数据链路管理、无线资源管理、移动性管理、呼叫控制、短消息和附加业务等协议栈功能。

手机应用层：该层在借助底层提供的功能服务上，为用户设计了各种基于手机的应用程序，如闹钟、日历、手机游戏等等。

人机接口（MMI）层：MMI的全称是Man Machine Interface。它是用户与手机之间交互的界面。用户通过键盘、显示屏、话筒、耳机等向手机发出指令，手机作出相应的应答。因此MMI层包含了各种手机界面、多类菜单显示、功能快捷键等等。

软件结构图如下：

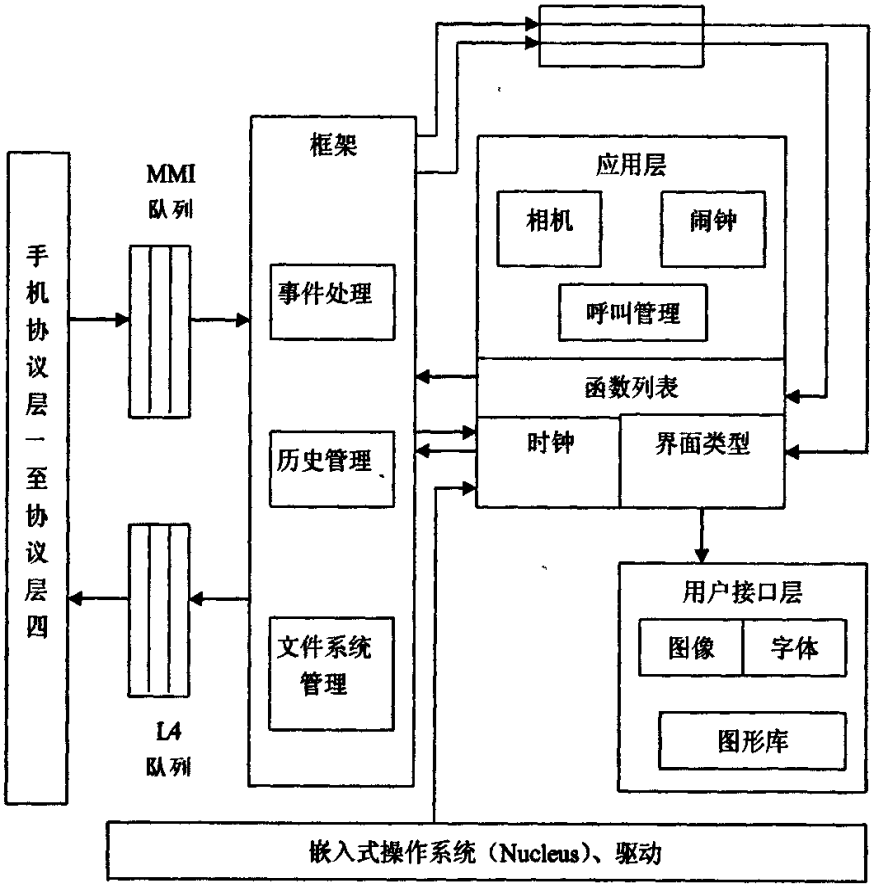

图 5-2 总体软件结构图

软件整体上是按层次结构进行模块管理的。每一层之间提供了接口函数，并通过信息队列进行数据传输。

5.2 应用开发

5.2.1 整体设计

为了不影响原有手机的外形与功能，可以将人体生理参数检测装置以手机附件形式设计。这样在实际使用过程中用户可以进行选择，当不需要人体健康检测时用户可以单独使用手机；而需要进行检测时插上手机附件立即就可以进行各项生理参数的测量。因此，项目应用研究包括两大部分：一方面是手机平台上的应用软件开发，另一方面是手机附件的开发。

手机平台上的开发其实质就是基于 Nucleus 的应用开发；只不过在新功能开发的同时要注意与原有系统任务的协调，并要充分利用原有系统所提供的各项功能接口。手机端的应用软件设计如下：

1. 数据通信。手机对外提供多种数据通信方式，如蓝牙、红外线、uart 串口等。其中 uart 串口是一种经济实用的数据通信接口，设计中使用它来与附件进行数据传输。此外，还必须设计一套数据通信的协议来有效的管理整个数据通信的过程。利用手机已有的 uart 串口通信功能和项目定义的传输控制协议，手机便可以向附件发送控制命令操纵附件进行不同的人体生理参数测量。同理，手机也可以接收附件所发送过来的采样数据并保存在指定的缓存之中。
2. 数据分析比较。在正确接收到附件发送来的数据后，手机必须对数据进行一定的预处理和计算。如测量人体脉搏时，由于外界震动或时人体的运动必然导致附件采集到的数据有毛刺现象，因此在正确计算人体脉搏之前必须先进行平滑处理，然后再通过计算得出脉搏测量值[50]。分析出生理参数测量值之后，程序通过与医学临床经验值进行比较便可以得出人体健康状况评估结果[51]。
3. 数据显示与保存。在数据分析比较之后便可以得到人体生理指标的检测值了。但是，除了得到测量值之外还必须将检测到的结果以人们可以理解的方式显示出来。如体温，生活中人们常常是以摄氏温度作为参考，因此我们就不能以华氏温度方式显示。另外一个问题就是数据的保存，在手机中我们经常用文字信息是短信；因此在设计中优先考虑的是将测量结果以短信的格式保存在手机之中。以短信格式保存的另外一个好处是当需要时可以很方便的将测量的结

果发送给用户的亲友。

手机附件采用 SOC 型单片机 C8051F350 进行开发。它的功能相对简单，通过传感器采集人体的生理信号，再由信号处理模块放大、滤波和 A/D 转换后按照数据通信协议经通信电缆传送至手机中

依据手机测量的人体生理指标不同，附件采用了 PVDF 压电传感器和热敏电阻。其中 PVDF 压电传感器是用来测量脉搏和血管弹性。可以将该传感器制成类似手表的腕带式血压计并放在手臂桡动脉处，这样不需要充气加压就可以方便地测量到人体脉搏。热敏电阻则是用来测量人体的体温。只要将其与人体测量的部位体表接触并保持足够的时间便可以得到较为精确的结果。

整体结构图如下：

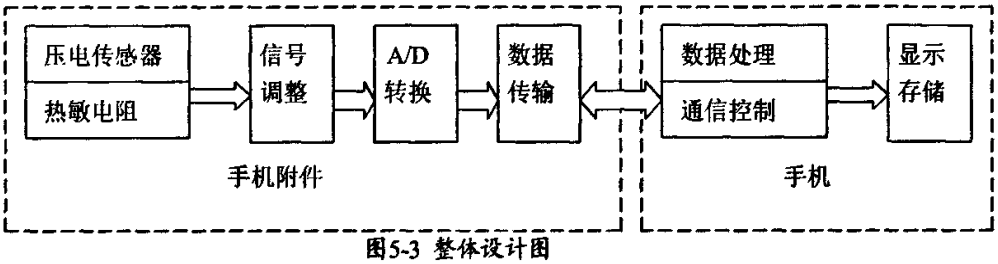

图5-3 整体设计图

5．2．2 通信协议

一份良好的通信协议是手机与附件正常通信的先决条件，它直接关系到应用软件质量的好坏。协议必须考虑到各种可能会发生通信状况，如链接断开、数据发送错误、控制命令解释等等。具体通信协议如下。

1. 附件自检协议

手机附件在上电或是在接受到复位命令以后进行自我检查。其主要任务是测试脉搏传感器、温度传感器是否存在，电池电压是否过低等异常情况。

手机向附件发送检测脉搏命令为：0xAA，10，(BCC)。附件软复位后进入自检（不超过 3 秒），完成后作出应答。其中 0xAA 是引导码，(BCC)是指除了引导码之外的数据经 BCC 效验而得到的字节数据。

附件如果自检后一切正常则向手机发送：0xAA，0x55，0xC3，0x69，0x10，(BCC)；反之则向手机发送：0xAA，0x55，0xC3，0x69，0x12，(BCC)。其中 0xAA，0x55，0xC3，0x69 为引导码。

2. 脉搏波数据检测协议

手机向附件发送检测脉搏命令为： 0xAA，'P'，(BCC)。

附件在正确接收倒脉搏检测命令后开始采集脉搏波形数据，并在采集到 8-12 秒时开始分析该脉搏波形。如果该波形能较好地符合正常人体脉搏波形则

认为此次脉搏波形数据采集有效，否则就应该认为该波形差。若测得脉搏波形差，这主要是因为用户在测量脉搏时没有将传感器正确放在身体合适的位置或是没有将检测装置牢固地扣紧。

因此通信协议也要根据上面所说的两种情况，在采集到脉搏波数据异常时发送响应数据：0x55，'Y'，(BCC)；在正常检测到脉搏波形时回答：0x55，"#####"，Data1， Data2，……其中 0x55 是引导码，其后紧随 5 个 '#'，然后是数据。其中 Data_n 数据为脉搏波形的二进制字节数据，附件将无限量发送。

手机在正确收到附件采集到的脉搏波数据后将其存放在指定缓存之中。直到缓存已装满数据为止，手机开始向附件发送停止检测脉搏命令： 0xAA，'S'，(BCC)。此时，不管附件处于什么状态，它都会停止检测脉搏、停止向手机发送数据并作出应答：0x55，'S'，(BCC)。

3.体温数据检测协议

手机向附件发送检测体温命令：0xAA，'T'，(BCC)。

附件在接收到体温检测命令开始采集体温数据，并响应手机发出数据为：0x55，TH，TL，(BCC)。此处 0x55 为引导码；TH 为检测到的体温高位字节；TL 为为检测到的体温低位字节。例如，TH=0x01，TL=0x81，则人体体温值应该是 0x181，即 385。它表示测量到的体温是 38.5 摄氏度。应答中的 BCC = TH ^ TL。

5.2.2 脉搏检测

脉搏是指随着心脏的节律性的收缩和舒张，动脉内的压力也发生周期性的变化，这种周期性的压力波动引起动脉血管发生扩张与回缩的搏动，这种搏动在浅表的动脉上可触摸到，临床上简称脉搏。因此脉搏率和心率是一致的。在医学中通常用脉搏率来代替心率。常用的直接测量方法有在桡动脉、颈动脉处人工触压测量。间接测量法则主要是用脉搏描记仪和血压脉搏监护仪等替代人工触压计数测量。具体方法视所用仪器不同而不同。目前从临床到家用的脉搏或心率自动测量仪采用不同传感器检测脉搏波周期变化的信号，以电子计数替代人工计数。这种方式方便有效，已为广泛被人们所接受。不同的是所用传感器及其检测部位有不同。

在本开发方案中采用的是 PVDF 压电传感器，并将它放置在人体桡动脉处来采集脉搏波周期变化信号。然后通过手机附件按上节所述的传输协议与手机通信完成对脉搏的测量分析。

在脉搏检测开始之前，系统需要保存手机前一个状态中的串口传输参数并按通信协议重新调整该串口的通信参数。这是因为在前一个手机状态中，串口通信参数很有可能与附件通信协议参数不一至，从而导致手机附件不能正确接收到手

机发送的控制命令，手机也不能正确接收到手机附件发送的采样数据。同时，保存手机串口原始通信参数则是为了在检测完脉搏之后可以还原手机串口参数从而保证不影响其手机其它功能模块的使用。

在手机发送完检测命令后，程序开始循环接收附件发送来的数据。如果在12 秒钟内手机未能接收任何数据则表明附件没有和手机相链接，程序向用户给出提示信息："无法链接"。若手机能够正常接收到附件发送来的数据，则将数据存放到指定缓存中并同时计数，直到接收到3500个有效数据为止。

脉搏数据分析过程包括以下几个部分：基线和倍率（增益）的调整检查、平滑中值滤波、脉搏波形特征检查、脉搏特征处理[52]（包括周期节点计算、脉搏率计算和K值的计算）。K值是一个以脉搏波波形面积变化为基础的脉搏波特征量，它在一定程度上反映了人体血管弹性等生理参数的变化。

1. 平滑滤波

从传感器采集的脉搏数据，不可避免的存在很多干扰信号，使得所测波形曲线上有许多毛刺，影响测试结果，同时给后续处理极为不便，所以，必须对脉搏信号进行平滑滤波。

要平滑效果好，每一点所参与的平滑点数就会比较多，但是又要求平滑后不使波形失真，所以综合考虑、比较之后采用了五点一次加权平滑滤波的方法，实践证明此种滤波方法可以达到很好的效果。

2. 基线和倍率（增益）的调整

在正式采集数据前必须调整基线、调整倍率，以便于观察脉搏波以及确保真正采集到脉搏波信号。这是由于在人体生物信息测试中，生物电信号受人体的其它因素如呼吸、姿态等的影响，总是存在基值飘移和幅值变化[53]。

在测量的过程中，基线漂移是不可忽视的现象，基线太高，测量到的数据波峰处可能被削掉，基线太低，谷值会被从"0"处截断，因此，将基线调整到一个合适的位置，不仅观察起来比较方便，而且也能够最大限度地反映被测者的真实情况。此外，选取一个合适的放大倍率，充分利用检测装置的测量范围，可以使测量结果更加有效可靠。所以，在设计中，预采了一定数量的脉搏数据进行基线和倍率的调整。

3. 脉搏形状检查

为了保证采集数据的有效性，需要预采集一些数据，提取出脉搏形状参数，将其与脉搏波信号所应该具备的基本特征进行比较[54]，如果符合脉搏波形状，则表明已经检测到了脉搏波信号，若没有通过形状检查，则说明没有检测到脉搏波，可能是硬件或者测量位置有偏差，给出错误提示。

4. 脉搏特征处理

通过了基线、倍率的调整和形状检查以后，就可以正式采集数据。对于这些数据，需要做以下几方面的计算：周期节点计算、脉搏率计算。

首先，要找出脉搏信号的各个峰值点和谷值点。极值点的查找是非常重要的一个环节，它将被用来确定脉搏信号的各个周期，所以找准极值点关系到整个数据处理结果的准确性。在寻找各个极值点的时候，应该注意不平滑、毛刺比较多的情况，容易将极值点附近的非极值误判成极值点，因此必须适当地增加一些限定条件。找出了全部的极值点以后，就可以计算出各个脉搏波的周期了，以此为依据计算出脉搏率。

为了增加所得结果的可靠性，在数据处理过程中进行了一些可靠性处理，将周期与平均值偏差较大的脉搏信号值剔除掉，最后得出的结果更加能反映真实的人体状态。

5. 2. 2 体温检测

体温又称体核温度，是指身体内部胸腔、腹腔和中枢神经的温度，其特点相对稳定且比皮肤温度高。皮肤温度又称体表温度，易受环境温度等因素的影响。正常人体温度一般会有变化，但是升降幅度有限，不超过 1.0℃。保持一定的体温是人和高等动物进行新陈代谢和维持正常生命活动的重要条件。

手机下的体温检测和脉搏检测类似，串口参数等调整这里不再重复。附件在正确接收手机发出的体温检测命令后开始检测人体体温。由于测量体温的探头使用的是热敏电阻，因此体温的检测过程需要一定的温度稳定时间。因为热敏电阻刚一接触到人体时，它的温度并不会立刻由室温温度升高到人体体温温度。温度的变化需要一个过程，这和使用体温计测量体温时需要等待一段时间是同一个道理。所以在体温检测过程中附件发送来的数据是一个不断变化的值，直到热敏电阻的温度趋于稳定为止。热敏电阻温度变化简略示意图如下：

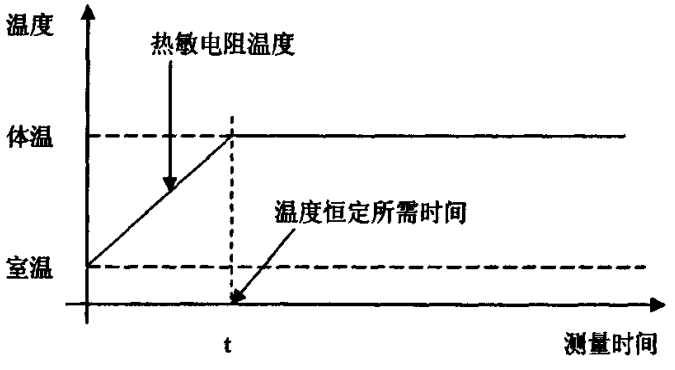

图5-4 热敏电阻温度变化简略示意图

考虑到热敏电阻的这个特性，手机不能简单地处理附件发来的数据；而是需要不断地检查体温测量数据是否趋于稳定。在体温检测的开始阶段附件发送的数据必然不准确，但是却有必要将数据保存起来。程序中使用两个无符号整形型变量 data1、data2 来记录附件发送来的体温数据，但是要注意的是记录二者的时间必须有一定的间隔。然后，分析 data1 和 data2 的差值并用一个无符号字符型变量 availcount 记录分析的情况。Availcount 值的大小表示数据趋于稳定的程度；它的初始值为零。如果两个数据之差不等于 0 体温检测继续，availcount 清零；若差值为 0，availcount 自增 1，同时判断 availcount 值是否达到系统预先设定的数值（例如 10 次），如果其值等于预先设定值表示热敏传感器温度已经趋于稳定体温检测结束，反之体温检测继续。

体温检测结束时，data1 或者 data2 所保存的值就是所测得的人体体温值。正常情况下，人体体温随着年龄和运动状态的不同而有所不同，但是浮动范围不大。成年人正常体温范围大致是 35.8 摄氏度到 37.6 摄氏度，当手机检测出的体温值不在这个温度区间时应在显示体温的同时给予用户健康警告。

5．3 本章小结

经过反复实验证明，该设计方案较好地完成了预期目标，实现了手机下人体部分生理参数的检测。其中脉搏率检测所花时间为 1～2 分钟，测量精度为±5%；血管弹性检测费时 1～2 分钟，所测结果具有非常高的参考价值；体温检测需时 60～75 秒，测量精度达到±0.2 摄氏度。装置具有测量精确高、成本较低、功耗低、使用维护方便、具有较强的可扩展性等特点。

人体检测功能还可以进一步扩展。中医有较完整的人体脉象诊断理论和方法。已进行的研究表明传统上由中医把脉获得被诊者的脉象可通过采用压力或力传感器的脉象仪来获得，并且所得结果更为客观。根据对不同脉搏波的波形特征的识别、分析，可以分辨出不同的脉象，从而判断出不同的健康状况和疾病诊断。因此只要获得了相应质量的脉搏波波形数据，就可根据脉像理论方法，通过数据分析判断人体的健康状态，为被诊者提供更多更专业的指导意见和信息，也能使普通用户完成由专业人员才能进行的脉象诊断。

对现有的人体检测方法可以进一步改进。如测量体温时本方案使用的是热敏电阻并取得了较好的效果，但是随着技术的进步可以使用红外线来测量人体体温。红外体表测温的原理是通过红外线辐射迅速测出人体表面温度，具有非接触、快速测温、减少传染概率的优点。但是这种测量的方法所花费的成本高且受体表下血液循环及周围环境导热状况的影响极大，与最准确的腋下测温相比，温差可达 1 至 3 摄氏度。

第六章 结束语

嵌入式软件是嵌入式产品的核心，作为嵌入式软件的基础，嵌入式实时操作系统在产业发展过程中扮演着越来越重要的角色，在各种嵌入式产品中的应用也越来越普及。在嵌入式实时操作系统的平台上开发嵌入系统软件是嵌入式系统发展的必然趋势。Nucleus 是一个实用、健壮的嵌入式实时操作系统，是专门为嵌入式应用而写的实时内核，具备可移植，可固化，可裁减等优先。对 Nucleus 内核及应用的研究具有非常现实的意义。

本文主要完成了以下几个方面的工作：

1. 介绍了嵌入式系统的基本概念。
2. 深入研究了开放源代码的嵌入式实时操作系统 Nucleus 的内核结构。分析了其中的任务管理、中断管理、操作系统数据结构保护、任务间通信、内存管理等模块。
3. 将 Nucleus 与其它常用嵌入式操作系统 uCLinux 和 uC/OS II 进行了对比。重点从进程调度策略、文件系统和移植性方面分析该三款操作系统的各自的特点。
4. 对 Nuclues 系统的改进进行了研究。提出一种基于模糊理论的任务调度算法和基于三层容错技术的 Nucleus 系统功能扩充。
5. 讨论了一种使用 Nucleus 嵌入式操作系统实现在手机下的人体健康检测功能的应用。介绍了手机方案中硬、软件开发环境。设计了手机与检测装置的通信协议。详细说明了人体脉搏、血管弹性、体温检测的过程。并对现有方法的不足提出改进意见。

通过对本课题的研究，本人对嵌入式实时操作系统的理论、算法、应用有了更深的认识，但本文对嵌入式操作系统的内核分析仍存在着许多不足，需要进一步研究，例如：如何设计并利用更适合的任务调度算法来改进系统性能；另外，除了内核研究，嵌入式系统还有大量的可扩展功能组件有待去研究分析，如文件组件、网络组件、GUI 组件等。在操作系统 Nucleus 的应用方面，它所涉及的领域越来越广泛，本文所讨论的应用只是其中之一。虽然手机下人体健康检测已达到预期要求，但是其在生理参数测量的方式和方法上都还有可以改进之处，如尝试进行红外测温等。以上所述的不足点也正是下一步研究工作的方向；相信通过对嵌入式系统不断学习和研究，这些不足都将迎刃而解。

参考文献

[1] 探矽工作室著,嵌入式系统软件开发圣经[M].北京:中国青年出版社, 2002

[2] 张大波.嵌入式系统原理、设计与应用[M].北京:机械工业出版社,2004

[3] 王田苗.嵌入式系统设计与实例开发[M].清华大学出版社,2003

[4] 师明珠.嵌入式应用系统软件设计技术研究[J]. 计算机工程与应用,2002, 38(7):127-130

[5] 唐寅编著,实时操作系统应用开发指南[M]. 北京:中国电力出版社, 2002

[6] 严猛等.ARM7强实时嵌入式系统设计.计算机工程与应用,2005,第21期: 110-112

[7] Arnold Berger.嵌入式系统设计[M].电子工业出版社,2002

[8] Stallings W.Operating System Internals and Design Principles [third Edition]. 北京:清华大学出版社,1998

[9] Nucleus Plus Reference Manual[M]. Accelerated Technology Inc. 1998.

[10] Nucleus GRAFIX Rendering Services Reference Manual[M]. Accelerated Technology Inc. 2000.

[11] Nucleus GRAFIX Windowing Toolkit Reference Manual[M]. Accelerated Technology Inc. 2000.

[12] Nucleus PLUS Internals Manual[M]. Accelerated Technology Inc. 1998.

[13] 贺磊.一种嵌入式实时操作系统Nucleus[J] .信息工程大学学报, 2000,1(4):53-54

[14] 朱丽英等.基于Nucleus的嵌入式系统的软件设计.计算机应用与软件,2005, 22(2):59-61

[15] 魏振华,洪炳熔等.嵌入式实时操作系统Nucleus中线程控制部件的实现方法. 计算机应用研究,2003,第四期:9-99

[16] 彭俊杰等.嵌入式实时操作系统Nucleus邮箱通信机制研究.计算机应用研究, 2004, 第五期:220-221

[17] 韩明锋.万向阳等.Nucleus事件组在变电站自动化系统中的应用.计算机工程与设计,2005, 26(6):1662~1663

[18] 阳富民等.基于Nucleus的IC卡设备驱动.计算机工程与设计, 2004,27(13):2463-2466

[19] 郁发新.常用嵌入式实时操作系统比较分析.计算机应用,2006,17(10):761-764

[20] 秦贵和,徐华中,王磊等. ARM9嵌入式技术及Linux高级实践教程[M].北京:北

京航空航天大学出版社,2005
[21] 施笑安等.Linux内核支持服务质量的改进.计算机工程,2005,31(20):88-90
[22] 陆超,朱贺飞等.针对Linux操作系统的MMU设计,小型微型计算机系统,2007,28(4):738-741
[23 梁丁,李迅勇等.应用中的嵌入式Linux实时优化,计算机工程,2007,33(1):77-79
[24] 周立功.ARM嵌入式系统基础教程[M].北京:北京航空航天大学出版社,2005
[25] Jean J.Labrosse著,邵贝贝译.嵌入式操作系统uC/OS -II [M].北京:北京航空航天大学出版社,2005
[26] 张文君,李磊等.基于嵌入式Linux系统的时间统一平台.计算机工程,2005,31(17):200-202
[27] 许先超.减少TLB失效开销提高64位Linux系统性能的方法,计算机工程,2006,32(2):70-72
[28] 梅岩,王力生.嵌入式系统中基于数据复用的进程调度.计算机工程,2005,31(21):78-80
[29] 伊叶丹等.UC/OS-II在电动车电池管理系统中的应用研究.计算机工程应用,2004，第30期:205-208
[30] 毛德操、胡希明.嵌入式系统采用公开源代码和StrongARM/XScale处理器[M].杭州:浙江大学出版社,2003
[31] 周立功. ARM微控制器基础与实践[M].北京航空航天出版社,2005
[32] 毛德操、胡希明.Linux内核源代码情景分析[M].杭州:浙江大学出版社,2003
[33] 楚红雨,李磊民等.实时操作系统uc/os-II在ARM9上移植的实现.计算机工程,2005,31(20):226-228
[34] 孙天泽.嵌入式设计及Linux驱动开发指南[M].北京:电子工业出版社,2005
[35] 程斌,金海等.一种自适应的分布式调度策略.小型微型计算机系统,2005,26(10):1793-1798
[36] 邹勇,李明树等.开放式实时系统的调度理论与方法分析,软件学报,2003,14(1):83-90
[37] Buttazzo G, Spuri M, Sensini F. Value VS. deadline scheduling in overload conditions. In: Proceedings of the 16th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1995. 90～99
[38] 王永炎、王强等.基于优先级表的实时调度算法及其实现.软件学报,2004,15(3):360-370
[39] Terrier F, Chen Z. Fuzzy calculus applied to real time scheduling. In: Yen J, ed. Proc. of the 3rd IEEE Conf. on Fuzzy Systems,Vol 3. Piscataway: IEEE Computer

Society. 1994. 1905～1910

[40] Lee J, Tiao A, Yen J. A fuzzy rule-based approach to real-time scheduling. In: Yen J, ed. Proc. of The 3rd IEEE Int'l Conf. onFuzzy Systems, Vol 2. Piscataway: IEEE Computer Society. 1994. 1394～1399

[41] 金宏,王宏安等.模糊动态抢占算法.计算机学报,2004,27(6):812-818

[42] 庞茂等. 基于模块化的容错技术在测控系统软件开发中的应用研究.传感技术学报，2006， 19(4):537-540

[43]陈国林、章立生.一种基于FPGA的容错嵌入式系统设计.计算机应用,2005,25(8):1916-1922

[44] 刘云生等.异构分布式实时仿真系统的容错调度算法.软件学报,2006,15(3):2040-2047

[45] 吴百锋等.一种基于监测的嵌入式系统设计技术.计算机学报,2003,26(12):1728-1733

[46] 袁成军等. Nucleus Plus容错功能扩充研究.微电子学与计算机，2005，22(8):155-157

[47] 陈文智等.一个构件化嵌入式操作系统的精确控制内核.计算机学报.2006,29(6):867-874

[48] 李海泉,李健等. 计算机系统安全技术[M].北京:人民邮电出版社,2001

[49] 杜春雷.ARM体系结构与编程[M].北京:清华大学出版社,2003.

[50] 陈春晓,刘建业等.心血管功能诊断议的研制.重庆医科大学学报,2002,27(4)

[51] 罗志昌等.脉搏波波形特征信息的研究.北京工业大学学报.1996, 22(1):71-78

[52] Chen W, Kobayashi T, Ichikawa S. Continuous estimation of systolic blood pressure using the pulse arrival time and intermittent calibration[J], Medical & Biological engineering &Computing, 2000;38(5): 569-574.

[53] 吴水才,刁越等.基于脉搏波的新型血流参数检测仪的研制.北京工业大学学报,2005,31(2):189-193

[54] 孔谙,白净等.心血管系统参数变化对脉搏波波形影响的数字仿真研究,航天医学与医学工程,1999,12(4):288-292

致 谢

本篇论文自始至终是在我的导师刘建成副教授的指导下完成的。刘老师学识渊博，治学严谨，在计算智能及模糊建模方面，特别是模糊信息处理方面有很深的造诣。在研究生这三年学习中，刘老师为人师表，言传身教，对本人关怀备至。在学术上，给予耐心的指导，使本人在理论水平和实践能力上都上了一个新的台阶；在生活上，处处严格要求，培养了本人严谨的态度与执着的追求精神。能得到刘老师的指导，必将使本人受益终身；在此，本人表示衷心地感谢！

感谢信息科学与工程学院的各位教授和老师！在本人读研期间，各位教授和老师在课堂上教授本人知识；在平时给予本人许多指导与帮助。在此，本人表示最诚挚的谢意！

感谢我的父母及家人，是他们伟大而无私的爱给予了我积极进取的动力和坚忍不拔的毅力，是他们的言传身教使我懂得了做人要勤奋、诚实、善良与勇敢。

感谢我的各位同窗好友和师兄弟们，他们让我感到了兄弟姐妹般的温暖。他们在生活上给了我极大的关怀，在学习上给了我很大的帮助，在精神上给了我无限的勇气和力量。是他们使我的研究生生活快乐而充实！

最后，感谢所有在我的求学之路上给予指导和帮助的人们，谢谢你们！

攻读硕士学位期间发表论文及科研情况

攻读硕士期间发表论文情况：

[1]道理，刘建成. 多特征参数的任务模糊调度算法. 计算机工程与应用，2006.12.

攻读硕士期间参加科研项目情况：

1. 株洲电力机车厂自动办公系统（横向项目）
2. 手机下人体状态指标测试系统（横向项目）
3. 湖南省自然科学基金 面向实值样本数据的模糊模型学习方法研究。2004 年份 ，基金编号 04JJY6036