汽车电子嵌入式操作系统的隔离保护机制

陈丽蓉，燕立明，罗 蕾

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】描述了汽车电子嵌入式操作系统隔离保护机制的一种策略及实现。通过结合有限的硬件资源及软件机制，满足操作系统、应用、任务及中断服务例程等3个级别的隔离保护要求，并提供保护错误处理的机制，将系统的访存错误局限于一定区域内，降低系统整体失效可能性。该实现能有效减少分页的数量，提升操作系统性能及存储空间利用率。具备隔离保护机制的汽车电子嵌入式操作系统可将不同来源、不同安全完整性级别的软件部件在一个ECU系统中集成。

关 键 词 汽车电子; 嵌入式操作系统; 隔离; 分区; 保护
中图分类号 TP319 文献标志码 A doi:10.3969/j.issn.1001-0548.2014.03.023

An Isolation and Protection Mechanism of Automotive Electronic Embedded Operating System

CHEN Li-rong, YAN Li-ming, and LUO Lei

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract A strategy and implementation for the isolation and protection mechanism of automotive embedded operating system is described. Upon limited hardware resources, the software mechanism satisfies the three-level isolation and protection requirements covering operating system, applications, tasks and interrupt service routines. A protection error handling mechanism is provided with the ability of restricting memory accessing errors to limited regions, reducing the probability of the whole system's failure. The number of memory pages is reduced apparently along with the improvement of operating system performance and utility of memory space. An automotive embedded operating system with isolation and protection mechanism can integrate software components of different sources and safety integration levels into a same ECU system.

Key words automotive electronic; embedded operating system; isolation; partition; protection

随着汽车电子控制系统功能、复杂性的不断增加，当今一辆中档轿车中装备有大约 80 个电子控制单元(ECU)，它们通过多达 5 种不同的总线系统来通信与交互。出于成本、可靠性等多方面因素考虑，一种趋势是将大量ECU替换成少量但功能更强大的微控制器，原来运行在若干专有ECU上的应用共享一个微控制器。当众多功能集成在一个ECU中实现时，要系统性地考虑汽车电子应用在安全、可靠方面的需求。为了在运行时分离不同来源的软部件，从软件层面阻止一定级别的故障传播，在汽车电子嵌入式软件设计中要通过分区机制来防止各软部件之间的干扰[1]。AUTOSAR组织阐述了存储分区和处理器工作模式对实现防干扰的支持[2]；AUTOSAR OS规范[3]则提出了一系列与隔离保护相关的要求。

1 问题的提出

软件分区分为操作系统分区和应用分区。操作系统内核、存储软件、通信诊断软件、外设与I/O控制等基础软件模块执行在一个可信的、处于特权模式的操作系统分区中。应用层软件部件在逻辑上被划分到不同的应用中，一些应用属于不可信的、非特权模式的应用分区，而另一些应用与操作系统一样，也处于可信的、特权模式的操作系统分区。AUTOSAR规范要求实现不同应用软件之间的隔离保护，以及应用软件与基础软件之间的隔离保护。另外为了满足一些安全限制，该规范还进一步基于程序的不同分段(数据、代码、栈)提出了对操作系统、应用以及执行体等3个级别的隔离保护要求。AUTOSAR架构中软件分区的总体模型如图1所示。

收稿日期：2013 – 07 – 02； 修回日期：2013 – 11 – 20
基金项目：国家“核高基”重大专项(2009ZX01038-002-003)；四川省应用基础研究项目(2011JY0118)
作者简介：陈丽蓉(1972 –)，女，博士生，主要从事嵌入式实时软件方面的研究.

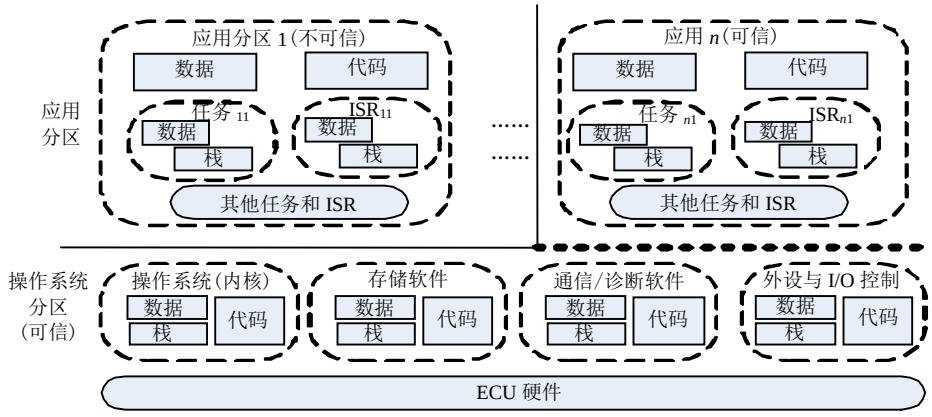

图1　AUTOSAR架构基于软件分区的隔离保护模型

1) 操作系统保护：禁止不可信应用对操作系统数据段和栈进行写操作。

2) 应用的隔离：

①禁止不可信应用对其他应用的私有数据段进行写或读的操作。

②禁止不可信应用对其他应用中任务或 ISR(中断服务例程)的私有栈或数据段进行写操作。

③允许某个应用禁止其他不可信应用执行其代码段。

3) 执行体的隔离：禁止(非强制)某个不可信应用中的任务和 ISR 的数据段或栈被该应用中的其他任务或 ISR 进行写操作。

上述隔离保护要求需要依赖特定硬件(比如MMU或MPU)的支持。而由于嵌入式处理器硬件资源(如TLB表项数量)的限制，使得规范中某些粒度的隔离保护需求难以实现，如将一个应用内部不同任务与ISR所使用的数据和栈区分开来，文献[4]也提到了该问题。如果将所有这些数据及栈的分段各放到一个独立的存储页中，显然页的数量很容易超过TLB表项的数量。而利用MMU的缺页异常处理机制解决该问题又会对系统的实时响应特性带来影响。因此基于单纯的硬件机制所实现的隔离保护在一定程度上降低了操作系统的可用性。

文献[5]提出了纯粹基于软件的存储保护机制，但它主要是面向任务的，没有达到应用级的隔离保护粒度，也没有考虑对中断服务例程的隔离保护；文献[6]对比了基于硬件和软件的存储保护方案，但未描述具体实现细节且侧重于编程语言(比如JAVA)所提供的安全特性；文献[7]申明了操作系统提供存储分区机制满足故障隔离要求的重要性，但未提供具体实现；文献[8]涉及的问题域与本文类似，但面向航空电子领域，应用环境有所差异；文献[9]采用一种off-chip的方案检测存储访问异常行为，但其不是操作系统内置机制与功能；文献[10]提出了一种较为通用的存储保护机制，但基于ITRON规范及SH3 RISC处理器进行的实现。

本文提出了一种软件与硬件机制相结合的、分层实现隔离保护的框架和机制，有效利用硬件提供的保护功能，降低对硬件资源的需求，并基于软件实现，满足细粒度的隔离保护要求，增强应用系统的安全性，满足汽车电子这一特殊应用领域规范的要求以及系统的实时性要求。

2　总体框架与基础

2.1　总体框架

多级隔离保护机制框架如图2所示。基于一定的硬件基础，隔离保护机制主要涉及以下3个层次。

1) 第一级隔离保护：操作系统保护，主要基于处理器工作模式，对应用的访存操作进行权限控制，从而实现操作系统分区与应用分区的隔离；

2) 第二级隔离保护：应用隔离，对应用的非操作系统访问进行页编号匹配检查，从而实现不同应用分区之间的隔离；

3) 第三级隔离保护：执行体隔离，通过分离应用中不同执行体(任务、中断服务程序即 ISR)所使用的栈空间，实现应用分区内部的隔离。

这3个保护级别的实现，贯穿在操作系统的系统初始化、异常处理及维护管理几大功能模块中。

2.2　硬件基础及第一、二级隔离保护

第一、二级隔离保护基于MMU或MPU硬件所提供的分页机制。MPC5634处理器的分页地址转换机制为：在指令的执行过程中产生有效地址将与MMU的PID寄存器(用于维护系统当前应用分区号)、地址空间标识相结合，产生的虚拟地址与TLB中的页表

项进行比对，每一个页表项则记录了对应物理分页的TID号、实际页地址及访问权限信息。在匹配的情况下进行正常的地址转换，生成实际物理地址。而发生不匹配的情况有以下几种：

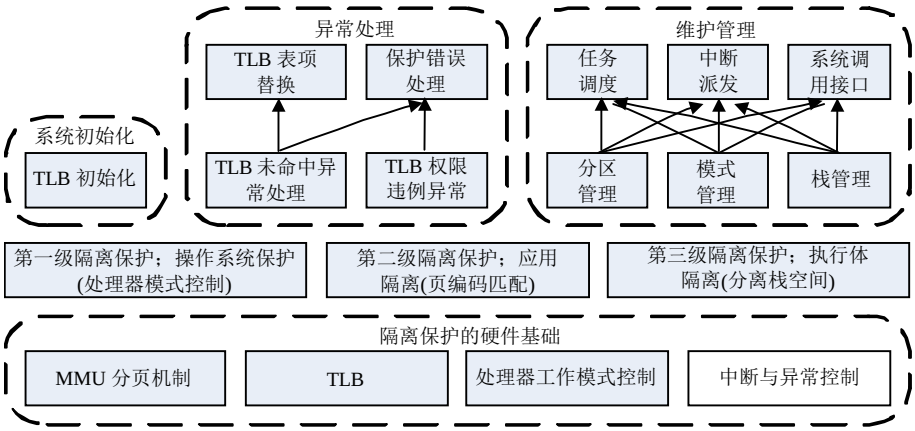

图2 多级隔离保护机制框架

1) 在TLB中未找到能够匹配的有效页地址：产生TLB未命中异常，需要在该异常处理程序中进行页面置换(即TLB表项替换)；如页面置换仍然未能找到匹配该有效页地址的页表项，则说明被访问的页不存在，进入保护错误处理；

2) 有效页地址匹配而PID与TLB表项中的TID不匹配：产生TLB未命中异常(实质是权限违例)，此异常被用于实现不同应用分区之间的隔离，意味着出现了一个应用访问其他应用私有页面的情况；

3) TID为0时PID为任何值均不产生TLB未命中异常，此特性被用来实现操作系统分区，以便操作系统向应用提供各种共享服务。为防止处于用户模式下的应用代码对操作系统代码及数据空间的非法访问，需要利用TLB表项中的权限位结合处理器当前工作模式以及将要访问空间的属性(代码或数据)进行控制。6个权限信息位是：特权模式下可读(SR)、可写(SW)、可执行(SX)；用户模式下可读(UR)、可写(UW)、可执行(UX)。若在地址转换过程中发现权限不匹配，则产生TLB权限违例异常。

基于此，本文实现第一级和第二级的隔离保护，将每一个不可信应用分区均划分为1个代码段和1个数据段，可信的操作系统分区(包含可信应用)分为2个代码段(核心代码段和系统调用接口段)和1个数据段。每个段被放置到分离的内存页面中(MPC5634允许不同大小的页面共存)，以便利用MMU硬件的分页机制实现隔离保护。表1为某系统页表项配置实例，其中应用1和应用2是可信的，应用3和应用4是不可信的。系统调用接口是可以被用户程序访问的，因此被划分为一个单独页面且访问权限为用户和系统均可执行。每个不可信应用的TLB表项具有不同的TID，以控制它们之间的访问。

表1 页表项配置实例

页内容	TID	SR	SW	SX	UR	UW	UX
内核及可信应用1和2的代码	0	0	0	1	0	0	0
系统调用接口	0	0	0	1	0	0	1
内核和可信应用1和2的数据	0	1	1	0	0	0	0
应用3代码	3	0	0	0	0	0	1
应用4代码	4	0	0	0	0	0	1
应用3数据	3	0	0	0	1	1	0
应用4数据	4	0	0	0	1	1	0

2.3 分离栈策略——第三级隔离保护

基于分页机制及处理器模式，仅仅能够实现操作系统与应用分区、不同应用分区之间的隔离。但是如果将分页应用于应用内部各执行体之间数据访问的隔离，则需要为每一个执行体的数据或栈提供独立的内存页，这样会大大增加系统的负担，影响系统性能(增加TLB未命中率，增加任务切换的负担，降低单页的利用效率)。因此，采用分离栈策略来满足分区内部细粒度隔离保护的需要，即为操作系统、不同的任务、不同的用户ISR提供独立的栈，但这些栈不占据独立的内存页，而是在其所属应用或内核的数据页面进行分配，并通过操作系统来维护，以此实现第三级的隔离保护。

运行于特权模式的可信应用具有与操作系统内核一样的特权级，其执行体在进行系统调用或响应中断时都可使用同一个栈。而对于运行于用户模式的不可信应用，当其执行体进行系统调用或保存中断上下文时需要从其当前运行栈切换到另外一个独立的系统栈以保证数据的安全和一致性。然而由于ISR的执行具有严格的LIFO特性，中断栈的使用可

以优化：可以使用一个共享的中断栈对所有嵌套中断的上下文进行保存，并且让所有的可信ISR也运行在这个栈上，而不会导致栈中内容的破坏。因此系统中栈的安排策略如下：

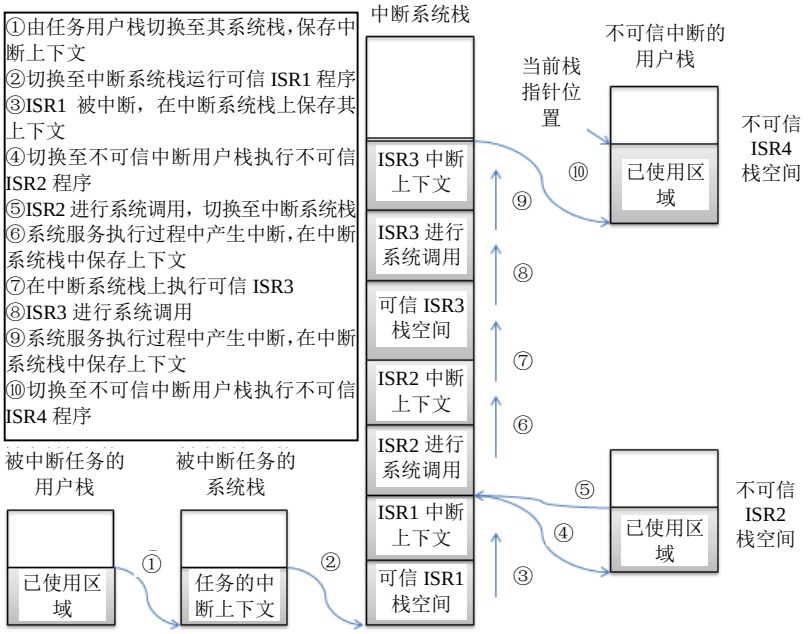

图3　中断系统栈的共享机制

1) 为每个可信应用任务和不可信应用任务安排一个独立的栈；

2) 对于每一个不可信应用中的任务，为其在可信区域中设置一个“任务系统栈”，以便该任务在进行系统调用或保存中断上下文时使用；

3) 对于每一个不可信应用中的中断服务程序，安排一个独立的“中断用户栈”；

4) 所有可信的中断服务程序共享一个“中断系统栈”，该栈也用于保存系统中所有嵌套中断的上下文，以及ISR调用系统服务时使用。

3　维护管理及保护错误处理

3.1　系统控制流及影响分析

系统控制流分析如图4所示。在系统运行过程中，由于应用对操作系统的功能调用、发生中断以及任务调度导致的任务切换等情况，系统的控制流会频繁地在应用的不同执行体与操作系统之间转换。在各种可能的转换情况中，操作系统要能够保证处理器工作模式、应用分区以及栈转换的正确性。为确保无遗漏，本文应用面向方面编程思想[11]进行程序分析与设计，以这3方面的内容为关注焦点，完成相应的系统调用处理、中断派发处理以及任务切换。

3.2　中断派发

中断派发实现被中断执行体和目标ISR之间的转换。当响应中断时处理器已自动将系统置为特权模式，中断派发程序需根据被中断执行体及目标ISR的属性进行相应处理。被中断执行体包括可信或不可信的任务、可信或不可信的ISR和系统服务；目标ISR包括可信或不可信的ISR。中断派发程序的流程：

1) 中断上下文保存：确保中断上下文保存在可信栈上：可信任务和 ISR 及系统服务被中断时不进行栈切换，直接保存中断上下文；不可信任务和 ISR 被中断时，切换到任务系统栈或中断系统栈进行保存。

2) 用户 ISR 调用预处理：根据目标 ISR 的可信属性、所属应用分区，决定是否进行相应的模式切换、应用分区切换和栈切换。

①如果目标 ISR 所属应用分区号与当前 PID 寄存器中的应用分区号不同，则保存 PID 到中断上下文中，并设置 PID 为新的应用分区号；

②如果目标 ISR 为可信的，则不进行模式切换和栈切换(直接在当前中断系统栈的位置开始执行用户 ISR)；

③如果目标 ISR 不可信，则从特权模式切换为用户模式由由中断系统栈切换至目标 ISR 的用户栈；

3) 调用用户 ISR，并在其返回后做如下处理：

①如果是从可信 ISR 返回，则不进行模式切换和栈切换，直接回到中断派发程序中；

②如果是从不可信 ISR 返回，则需从用户模式切换回特权模式，并从 ISR 的用户栈切换回中断系统栈；

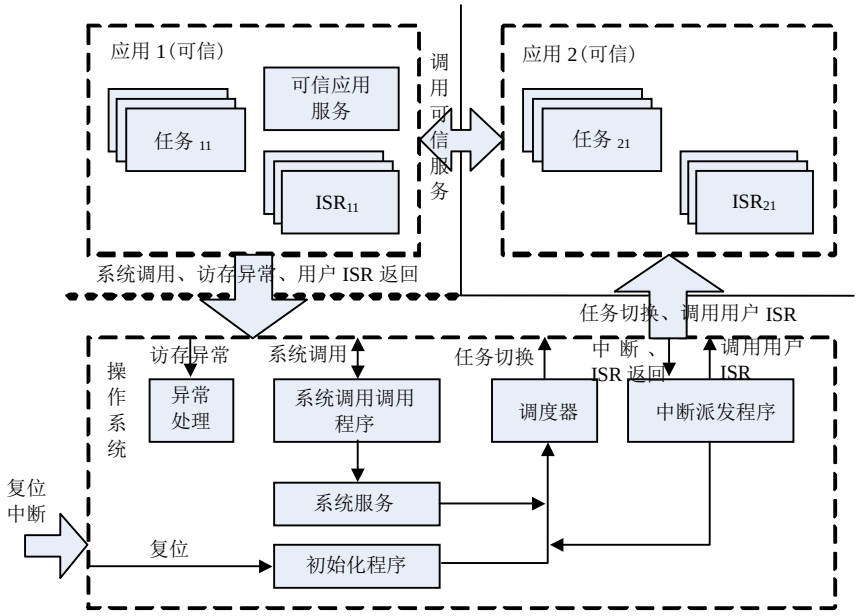

图4 系统控制流分析

4) 中断派发程序的后处理：存在 3 种可能性：

①回到上级 ISR：如果还存在中断嵌套，则恢复中断上下文，返回上一级用户 ISR 程序中；

②回到被中断任务：如果没有中断嵌套，则切换回到任务系统栈，在不需要重新调度任务或需要重新调度但调度结果仍然是原被中断任务的情况下，则从任务系统栈中恢复最外层的中断上下文，回到原任务被中断的位置继续执行；

③切换到新的任务：如果调度的结果是需要切换到其他任务执行，则由任务切换函数完成相应的模式转换、应用分区切换及栈切换。

3.3 系统调用接口

系统调用接口实现调用者(任务或者ISR)与操作系统服务之间的转换，调用者也分为可信和不可信两类。在进行系统调用以及系统服务执行期间，均不需要切换应用分区，系统调用处理只关注处理器模式和栈空间的变化。

1) 对于可信的任务或ISR在进行系统调用的时候，处理器已经处于特权模式，不需要进行模式切换和栈切换，系统服务可以 C 函数调用的方式，直接在调用者的栈上运行；

2) 对于不可信的任务或 ISR 在进行系统调用时，需要由用户模式切换至特权模式，并从任务或 ISR 的用户栈切换至其系统栈。模式切换需要利用处理器的系统调用指令或自陷指令完成，执行该指令时，处理器将自动切换为特权模式，并进入到相应的异常处理程序。因此，本文可进一步在此异常处理程序中完成栈的切换。当系统服务函数完成后返回到该异常处理程序中时，再将栈切换回原用户栈，并利用异常返回指令自动完成模式的切换。

3.4 任务调度与切换

任务调度的结果可能发生任务切换，也可能不发生切换。在发生任务切换的情况下，基本原则是要保存好即将离开的任务的上下文，并建立好新任务的运行环境。在这两个环节中均要考虑处理器模式、应用分区以及任务栈指针的正确维护。

1) 首先对离去任务的上下文进行保存，包括当前指令执行位置、栈指针、处理器模式以及其他需要保存的通用寄存器的内容。

2) 然后根据新任务所属应用来对PID寄存器进行更新，这样在随后任务的运行过程中PID才能与对应应用页面的TID号相匹配。

3) 根据新任务是否第一次运行进行处理；如果任务以前运行过但因某种原因被抢占，则在该任务上一次被切换离开处理器时，其栈指针、处理器模式已经作为其上下文内容的一部分被保存了下来，则此次它恢复执行的时候，只需要恢复其上下文，即可实现其栈指针、处理器模式的恢复。如果任务是第一次运行，则将栈寄存器设置为该任务的栈起始地址，并根据任务所属应用的可信属性对处理器当前的工作模式进行设置：由于当前程序还运行在特权模式的操作系统内核中，如果是可信应用，则直接调用任务函数启动任务的运行；如果是不可信应用，则需要通过内嵌汇编程序将处理器工作模式

设置为用户模式后再调用任务函数。

3.5 保护错误处理

保护错误来自两方面：在TLB未命中异常处理中发现应用进行非法访问，或是应用非法访问操作系统产生的权限违例异常。当OS检测到上述错误后：

1) OS调用保护HOOK函数并传递出错的类型；

2) 保护HOOK函数根据系统的配置反馈处理要求给OS，该处理要求与错误的严重程度相匹配；

3) OS根据反馈选择执行具体的处理：①PRO_IGNORE：什么也不做；②PRO_TERMINATETASKISR：强行终止出错的任务或ISR；③PRO_TERMINATEAPPL：强行终止出错应用；④PRO_TERMINATEAPPL_RESTART：强行终止出错应用，并重新初始化该应用；⑤PRO_SHUTDOWN：关闭OS自身。

4 实例与分析

4.1 实例验证

本文所述机制在基于MPC5634的ECU硬件平台XPC563MADPT144S上实现，并与作者团队所开发的AUTOSAR OS进行了集成。为验证已实现的隔离保护机制能有效工作，设计一个具有4个应用的系统，包含可信应用APP1和APP2及不可信应用APP3和APP4，包含任务APP1_T1，APP2_T1，APP3_T1，APP3_T2，APP4_T1和APP4_T2，任务优先级依次递增，APP1能访问所有任务。在不可信应用执行体中刻意增加非法访存操作，以验证该机制是否能对访存故障进行隔离保护。为验证TLB缺页置换功能，在系统初始化时未预置所有TLB项。保护错误处理的配置为：APP3中任务非法访存时强制终止故障任务；APP4中任务非法访存时强制终止故障应用。

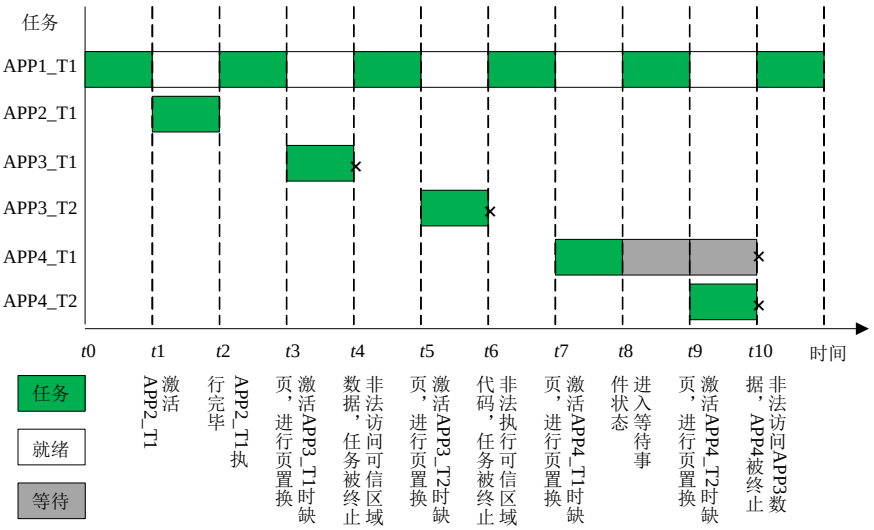

图5　应用实例运行流程

经测试表明，系统在运行过程中伴随着任务的切换可以正确地完成应用分区的转换、模式切换以及栈的切换。当发生TLB未命中异常时，能够正确完成页的置换，使相应任务能继续执行。而当发生权限违例时，能根据系统配置对发生错误的任务或应用进行终止，因而能够将错误或故障局限于一定的区域内，降低系统整体发生故障或失效的可能性。

4.2 优劣势分析

假定某系统有n个应用，其中不可信应用n_1个。每个应用包含数量不等的任务和ISR，方便起见，假设平均每个应用的任务及ISR总数为k个($k\geqslant 1$)。如果为每个应用分配1个代码页、1个被该应用所有任务及ISR共享的数据页、每个任务及ISR分配1个私有数据页和1个栈页，加上操作系统的核心代码页、数据页和接口函数页，总共需要划分出内存页共计$N=3+2n+2nk$个，假设此为方案1。采用本文的方案2，只需为每个不可信应用划分1个代码页和1个数据页，可信应用代码页与数据页分别与操作系统核心代码页及数据页划分到一起，所有的栈空间在相应的数据页内进行分配，则总共需要划分$N'=3+2n_1$个内存页。表2的数据对比说明了后者的优势。

显然方案1对于应用的数量以及应用中任务及ISR的数量变化都比方案2更敏感，它会显著增加页的数量，带来TLB命中率下降以及随之的异常处理、

TLB表项替换的系统时间开销。而另一方面，嵌入式应用中通常一个任务或ISR的规模都不是很大(可小到只需1 KB以内的空间)，如果为每个任务及ISR的数据和栈分配一个页面，则对于页内空间的利用率是很低的，这有可能造成整个存储空间不够满足应用系统的需求。

表2 两种方案的存储页数量对比

应用数量 n(假定n_1=n)	单个应用的任务和ISR数量k	方案1页数 N	方案2页数 N'
2	2	15	7
4	2	27	11
6	2	39	15
8	2	51	19
10	2	63	23
4	1	19	11
4	2	27	11
4	3	35	11
4	4	43	11
4	5	51	11

采用本文方法能够大大降低分页的数量，提升操作系统性能及存储空间利用率，但对于同一个应用区域内的栈溢出不能够很及时地进行检测。栈溢出有刻意或非刻意两种情形。非刻意溢出缘于在系统设计中未能分配足够的栈空间，尤其是突发情况(通常为出现了多个嵌套的中断)下栈空间的超正常使用。不过由于系统只允许响应比当前中断优先级更高的中断，中断嵌套的最大层数是有限的。在本文所述方案中，用于嵌套中断处理的栈已经与系统中其他的栈分离，可单独为这个栈划分足够大的空间。这样既有效避免了非刻意性栈溢出的发生，又在最大程度上减少了栈空间的分配。刻意的栈溢出操作可能来自于黑客软件，其意在破坏系统的正常工作。由于本文至少在可信与不可信区域之间设置了基于物理页的隔离机制，对于可能非法进行栈操作的应用是作为不可信的部分集成到系统中的，因此此类栈溢出可以被及时检测出来。综上，本文所用机制已较好地规避了该问题。

5 结 论

应用具有隔离保护机制的嵌入式操作系统，能够在一个ECU系统中容纳来自不同渠道、具有不同安全完整性级别的软件部件，能够降低整个软件系统满足高安全等级的难度，满足日益增长的汽车电子控制系统的安全需求。对于硬件资源受限且有实时性需求的汽车电子嵌入式系统，该机制的实现具有一定的挑战性。本文工作通过有效结合硬件资源与软件策略，降低了对硬件资源的需求，并提升了性能，以满足在高安全完整性级别的汽车控制系统中应用此操作系统的要求。

参 考 文 献

[1] ISO. ISO/IS 26262-6 road vehicles-functional safety-part 6: product development: software level[S/OL]. [2013-06-12]. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=46752.

[2] AUTOSAR GbR. Technical Safety Concept Status Report V1.1.0 R4.0 Rev 2[S/OL]. [2013-06-12]. http://www.autosav.org.

[3] AUTOSAR GbR. Specification of Operating System V4.1.0 R4.0 Rev 2[S/OL].[2013-06-12]. http://www.autosav.org.

[4] 张吕红. 参照AUTOSAR标准的SmartOSEK OS4.0的设计与实现[D]. 杭州: 浙江大学, 2010.
ZHANG Lü-hong. Design and implementation of smartOSEK OS 4.0 consulting AUTOSAR[D]. Hangzhou: Zhejiang University, 2010.

[5] 邓俊, 李红, 方正, 等. AUTOSAR OS存储保护方案的改进与实现[J]. 仪器仪表学报, 2011, 32(9): 2146-2152.
DENG Jun, LI Hong, FANG Zheng, et al. Improvement and implementation of AUTOSAR OS memory protection mechanism[J]. Chinese Journal of Scientific Instrument, 2011, 32(9): 2146-2152.

[6] STILKERICH M, LOHMANN D, SCHRÖDER-PREIKSCHAT W. Memory protection at option[C]// Proceedings of the 1st Workshop on Critical Automotive applications: Robustness & Safety. New York, USA: ACM, 2010: 17-20.

[7] XI Chen. Requirements and concepts for future automotive electronic architectures from the view of integrated safety [D]. Germany: University Karlsruhe(TH), 2008.

[8] GUI S L, LUO L, TANG S S, et al. Optimal static partition configuration in ARINC653 system[J]. Journal of Electronic Science and Technology, 2011, 9(4): 373-378.

[9] Dinh-Duc A V, HO N. A run-time detector for violated memory access in embedded systems[C]//Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on. [S.l.]: IEEE, 2010: 217-223.

[10] YAMADA S, NAKAMOTO Y, AZUMI T, et al. Generic memory protection mechanism for embedded system and its application to embedded component systems[C]// Proceedings of the 8th International Conference on Computer and Information Technology Workshops. Los Alamitos, CA, USA: IEEE, 2008: 557-562.

[11] LOHMANN D, HOFER W, SCHRÖDER-PREIKSCHAT W, et al. Aspect-aware operating system development[C]// Proceedings of the tenth international conference on Aspect-oriented software development. [S.l.]: ACM, 2011: 69-80.

编 辑 税 红