

一种基于 $\mu\text{C}/\text{OS-II}$ 符合 OSEK 标准的实时系统内核设计

朱怡安¹ 魏润之² 苏世游² 黄姝娟¹

(西北工业大学计算机学院 西安 710072)¹ (西北工业大学软件与微电子学院 西安 710072)²

摘要 一些安全关键任务系统对嵌入式操作系统的实时性和安全可靠性提出了更高的要求。设计并实现了一种满足 OSEK/VDX 标准且支持时间/事件混合触发的实时操作系统内核,该内核除具有事件触发的实时性好、使用方便、灵活性高等特点外,还具有时间触发的确定性和安全性等特点。此外,还提出并实现了一种基于静态表的混合任务调度策略,并给出了时间触发任务可调度性的静态测试算法,在保证事件触发灵活性的基础上,通过中断级和任务级时间/事件触发任务的灵活切换,可确保时间触发任务的确定性和安全性,并提高系统的利用率。实验结果表明,该内核可以有效支持时间/事件混合触发的任务调度,并具有良好的实时性与安全性。

关键词 OSEK/VDX, 时间触发, 事件触发, 实时系统

中图分类号 TP316 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.4.035

Real-time Kernel Design Based on $\mu\text{C}/\text{OS-II}$ and Meeting OSEK Standard

ZHU Yi-an¹ WEI Run-zhi² SU Shi-you² HUANG Shu-juan¹

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)¹

(School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710072, China)²

Abstract Safety critical systems make more requirements on timeliness and reliability of embedded systems. This paper designed a new kernel for a mixed time-triggered and event-triggered mechanism, meeting the OSEK standard. The kernel has the features of good timeliness, high flexibility and certainty. This paper also put forward a static schedule strategy for mixed tasks and an algorithm to check the schedulability of time-triggered tasks. The flexible switch between tasks at both interrupt and task level helps to guarantee those good features while improving the system utilization as well. The experiment shows that the real-time kernel is effective and efficient with good timeliness and high reliability.

Keywords OSEK/VDX, Time-triggered, Event-triggered, Real-time system

1 引言

当前,嵌入式实时系统的不断发展,要求操作系统具备实时性与可靠性,且能同时支持事件触发与时间触发^[1]。事件触发是指实时任务只有在与之相关的特定事件发生的条件下才能开始运行;时间触发是指实时任务在预先确定的时间点上开始运行^[2]。目前常见的嵌入式实时操作系统并不满足时间/事件混合触发的要求^[1],如 $\mu\text{C}/\text{OS-II}$ 、PowerPac、FreeRTOS 等只支持事件触发而不支持时间触发。

传统的事件触发操作系统以特定的事件作为调度的触发源^[3],而事件发生的随机性会导致任务的可调度性分析极为复杂,系统的可靠性与可预测性难以保证^[4]。相比较而言,采用时间触发,设计人员能够根据任务的时间约束,静态安排任务执行的顺序,避免任务间冲突,从而保证系统的实时性与可靠性,并使系统具有良好的确定性与可预测性^[5]。然而随着实时系统复杂度的提高,一些复杂的实时应用中周期性/非周期性实时任务和非实时任务同时存在,单纯采用时间触发,系

统的灵活性和实时响应能力将被削弱。

陈曦等^[6]对 $\mu\text{C}/\text{OS-II}$ 操作系统进行了改进,使其支持时间/事件触发混合任务调度,但其时间触发任务执行的完整性难以保证,系统可靠性不高;且时间/事件触发任务的切换只在时钟中断中进行,这样,时间触发任务执行完毕到下一个时钟中断之间 CPU 将会空闲,造成系统资源的浪费。淡图南等^[7]提出一种基于时间片的混合调度策略,即根据时间触发任务的时间参数,将系统的调度分割为不同的时间片,静态安排每个时间片上任务的执行。然而这种调度完全基于时间触发任务的静态参数,系统调度缺乏灵活性,同时时间/事件触发任务的切换也只能在时钟中断中进行,系统利用率不高。

本文根据 OSEK/VDX 标准^[8,9],基于 $\mu\text{C}/\text{OS-II}$ 操作系统,设计并实现了一种支持时间/事件触发混合任务调度的实时系统内核。对于时间触发任务,根据其时间参数,静态安排任务执行的时间点,保证系统具有良好的可确定性。为提高系统利用率和调度的灵活性,将优先级较低的事件触发任务安排在空闲时刻执行,并实现在中断级与任务级时间/事件触

到稿日期:2015-03-25 返修日期:2015-06-10 本文受陕西科技研究发展计划项目(2014K05-25),陕西省科技公关项目(2015GY035),航空基金(20130753006)资助。

朱怡安 男,教授,博士生导师,主要研究领域为嵌入式与普适计算、并行计算等,E-mail:zhuya@nwpu.edu.cn;魏润之 男,硕士生,主要研究领域为嵌入式操作系统、软件工程;苏世游 男,硕士生,主要研究领域为嵌入式操作系统、软件工程;黄姝娟 女,讲师,主要研究领域为网络与分布式计算、嵌入式计算。

发任务的灵活切换。最后,采用 EDF(Earliest Deadline First)算法对被抢占的时间触发任务进行恢复,同时还给出时间触发任务可调度性的静态测试算法,确保时间触发任务能够完整执行,以保证系统的可靠性与安全性。实验表明,该内核能有效支持时间/事件混合触发的任务调度,并具有良好的实时性能。

2 时间/事件触发机制的设计

2.1 时间触发任务模型

OSEK/VDX 标准规定,时间触发任务(Time-triggered Tasks, TT 任务)优先于事件触发任务(Event-triggered Tasks, ET 任务)。TT 任务具有 3 种状态:挂起(Suspended)、运行(Running)和被抢占(Preempted)。3 种状态之间的转换关系如图 1 所示。

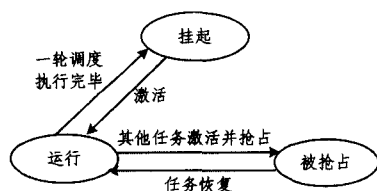


图1 TT任务状态转换关系

TT 任务创建后即处于挂起状态;任务激活后,即被切换执行并进入运行态;TT 任务若一轮调度正常结束,即被挂起并等待下一轮调度,若被其他新激活的任务抢占,则会进入被抢占态并等待恢复执行。

2.2 时间/事件触发任务调度策略

根据 OSEK/VDX 标准,支持时间/事件触发混合任务调度的实时系统内核应具备以下条件:(1)时间触发任务较事件触发任务具有更高的优先级;(2)时间触发任务在预先确定的时间点开始执行,任务由调度表激活并采用本地时钟作为基准时间^[6];(3)时间触发任务可以被其他新激活的时间触发任务所抢占;(4)时间触发任务处于空闲时,事件触发任务开始执行,事件触发任务采用基于优先级的调度策略。

根据以上条件,提出一种基于静态调度表的调度策略:

- (1)时间/事件触发任务调度可以在时钟中断和时间触发任务结束时进行;
- (2)时间触发任务按照其触发时间在时钟中断中被激活执行;
- (3)新激活的时间触发任务可以抢占正在执行的其他任务,被抢占的时间触发任务将被加入到恢复链表中,并按照 EDF 算法在空闲时刻依次恢复执行;
- (4)当时间触发任务空闲时,进行事件触发任务的调度。任务调度的示例如图 2 所示。

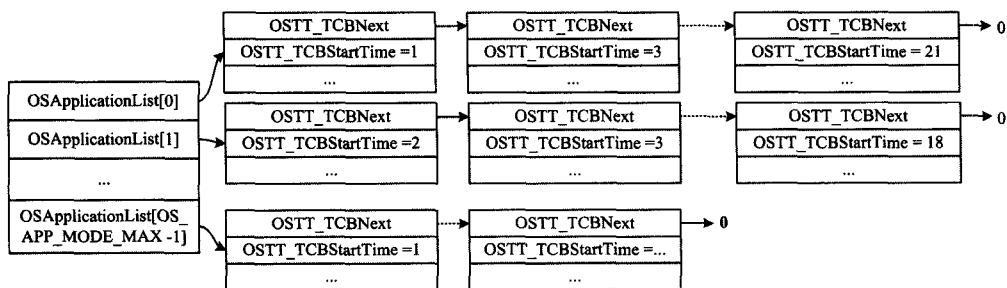


图 4 时间触发任务调度表示意图

任务恢复链表用于保存被抢占 TT 任务的`task_control_block`指针,链表中任务控制块指针按照指针所指向任务的截止时间

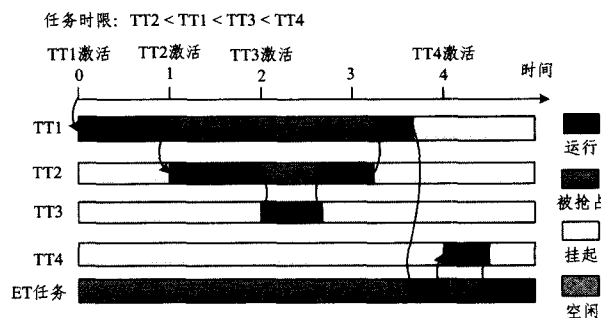


图2 时间/事件触发混合任务调度示例

3 时间/事件触发任务调度的实现

3.1 时间触发任务的相关数据结构

时间/事件触发混合任务调度设计中,ET 任务由 $\mu\text{C}/\text{OS-II}$ 原有的任务控制块 OS_TCB 描述,具有优先级和任务延迟时间等参数;TT 任务由任务控制块 OS_TT_TCB 结构体描述,如图 3 所示。

```
typedef struct os tt tcb
```

```
OS_STK* OSTT_TCBStkPtr; /* 任务堆栈指针 */
struct os_tt_tcb* OSTT_TCBNext; /* 指向下一个控制块的指针 */
ttTaskType OSTT_TCBTaskID; /* 任务标识符 */
ttAppModeType OSTT_TCBAppModel; /* 任务所属应用模式 */
INT8U OSTT_TCBStartTime; /* 任务的开始时间 */
INT8U OSTT_TCBWcet; /* 任务的最坏执行时间 */
INT8U OSTT_TCBDeadline; /* 任务的截止时间 */
ttTaskStateType OSTT_TCBState; /* 任务的状态 */
}OS_TT_TCB;
```

图3 时间触发任务控制块

系统利用硬件定时器产生精确的时钟节拍,在同一调度表中 TT 任务的触发时间须各不相同^[7]。其中 OSTT_TCB-StartTime 用于标识 TT 任务在一个周期内触发的时钟节拍刻度;OSTT_TCBWcet 是任务执行所需的最长时钟节拍数;OSTT_TCBDeadline 是任务截止的节拍刻度。

调度表是用于时间触发任务调度的数据结构,不同应用模式对应不同的调度表,每个调度表中包含具有相同周期且触发时间各不相同的一系列 TT 任务。TT 任务被创建时,系统根据任务所属的应用模式将任务关联到不同的调度表中,同一调度表中的 TT 任务按照 OSTT_TCBStartTime 从小到大链接为一个单向链表,如图 4 所示。

3.2 中断级任务调度

该混合实时系统内核中,任务调度分中断级和任务级两种情形。中断级任务调度在时钟中断服务函数中进行,主要完成是否需要时间/事件触发任务切换的逻辑判断,任务上下文切换在时钟中断退出函数中进行。中断级任务调度主要进行以下几种情形的逻辑处理。

(1)当前接收到同步信号,需要进行重新调度,其C语言形式的伪代码如下:

```
/* 接收到同步信号 */
if(OSTTReschedule==TT_RESCHED_TRUE)
{
    OSTTReschedule=TT_RESCHED_FALSE;
    OSTTTick=0; /* 时钟节拍计数器置0 */
    更新调度数据;
    清任务恢复链表;
    /* 当前时刻有 TT 任务需要执行 */
    if(OSTTNext != NULL && OSTTTick == OSTTNext ->
        OSTT_TCBStartTime)
    {
        锁定 ET 任务调度;
        OSTTCur=OSTTNext;
        OSTTNext=OSTTNext->OSTT_TCBNext;
        OSTTCur->OSTT_TCBState=TT_RUNNING;
        OSTTSw=TT_SW_TRUE;
    }
    else
    {
        解锁 ET 任务调度;
        OSTTSw=TT_SW_FALSE;
    }
}
```

OSTTSw 为 TT 任务切换标志,表示是否需要切换到新的 TT 任务执行;OSTTTick 为时钟节拍计数器;OSTTCur 指向当前正在执行 TT 任务的控制块,当 OSTTCur 为 NULL 时,说明当前无 TT 任务执行;OSTTNext 指向调度表中下一个将要执行的 TT 任务的控制块。更新调度数据包括更新当前应用模式、当前调度表指针,并将 OSTTCur 置为 NULL,OSTTNext 指向当前调度表中的第一个任务等。

(2)当前调度周期已结束,即 OSTTTick==OS_TT_TICK_MAX。这种情形的处理与(1)类似,但需首先判断 TT 任务恢复链表是否为空:若为空,则启动新一轮调度;否则,说明有 TT 未执行完毕,任务超时,需要进行超时错误处理。

(3)当前调度周期正在进行,当前时刻有 TT 任务激活,即 OSTTNext != NULL && OSTTTick == OSTTNext -> OSTT_TCBStartTime。这时需要考虑下列情形:

```
if(OSTTCur==NULL) /* 当前无 TT 任务执行 */
{
    锁定 ET 任务调度;
    更新调度指针和当前任务状态;
    OSTTSw=TT_SW_TRUE;
}
/* 正在执行恢复链表中的 TT 任务,抢占执行 */
else if(OSTTResumeFirst!=NULL && OSTTCur==
    OSTTResumeFirst->OSTT_PTCb)
{
```

更新调度指针和任务状态;

OSTTSw=TT_SW_TRUE;

}

/* 上一次激活的 TT 任务未执行完毕,将任务加入到恢复链表中,抢占执行 */

else

{

将当前任务添加到恢复链表中;

更新调度指针和任务状态;

OSTTSw=TT_SW_TRUE;

}

若当前无 TT 任务执行,则切换到当前时刻激活的 TT 任务执行;若当前正在执行恢复链表中的 TT 任务,则需再次抢占恢复链表中的任务;若上一次激活的 TT 任务未执行完毕,则需首先将该任务的控制块指针按照任务的时限插入到恢复链表中的相应位置,然后抢占该任务执行。

(4)调度周期正在进行中,当前时刻无 TT 任务激活,其逻辑处理如下:

/* 当前无 TT 任务执行 */

if(OSTTCur==NULL)

{

解锁 ET 任务调度;

OSTTSw=TT_SW_FALSE;

}

else /* 继续执行当前的 TT 任务 */

{

锁定 ET 任务切换;

OSTTSw=TT_SW_FALSE;

}

若当前时刻无 TT 任务执行,则启动 ET 任务调度;否则,继续执行当前 TT 任务。

3.3 任务级任务调度

任务级任务调度函数在每个 TT 任务执行结束时被调用,用于更新任务状态,检测 TT 任务是否超时,进行恢复链表中 TT 任务的恢复或 ET 任务切换。任务级任务调度函数首先更新当前执行完毕任务状态为挂起态,然后完成以下几种情形的逻辑处理:

(1)检测当前执行完的 TT 任务是否超时,若超时,则进行超时错误处理。

(2)任务正常执行结束,且任务恢复链表为空,当前无 TT 任务需要恢复,则解锁 ET 任务调度并进行 ET 任务切换,处理如下:

if(OSTTResumeFirst==NULL)

{

OSTTCur=NULL;

解锁 ET 任务调度;

OS_TASK_SW();

}

OSTTResumeFirst 为 TT 任务恢复链表的头指针;当前无 TT 任务执行时,置 OSTTCur 为 NULL;OS_TASK_SW()完成 ET 任务切换。

(3)TT 任务恢复链表不为空时,需要判断当前执行结束的 TT 任务是否是恢复链表中任务,以决定继续恢复其他 TT

Tag-Aided Web Service Clustering Method[J]. Chinese Journal of Electronics, 2015, 43(7): 1266-1274 (in Chinese)

田刚, 何克清, 王健, 等. 面向领域标签辅助的服务聚类方法[J]. 电子学报, 2015, 43(7): 1266-1274

[5] Wu Tao, Chen Li-fei, Guo Gong-de. High-dimensional data clustering algorithm with subspace optimization[J]. Journal of Computer Applications, 2014, 34(8): 2279-2284 (in Chinese)

吴涛, 陈黎飞, 郭躬德. 优化子空间的高维聚类算法[J]. 计算机应用, 2014, 34(8): 2279-2284

[6] Xin Yu, Yang Jing, Tang Chu-heng, et al. An Overlapping Semantic Community Detection Algorithm Based on Local Semantic Cluster[J]. Journal of Computer Research and Development, 2015, 52(7): 1510-1521 (in Chinese)

辛宇, 杨静, 汤楚衡, 等. 基于局部语义聚类的语义重叠社区发现算法[J]. 计算机研究与发展, 2015, 52(7): 1510-1521

[7] Liao Lü-chao, Jiang Xin-hua, Zou Fu-min, et al. A Spectral Clustering Method for Big Trajectory Data Mining with Latent Semantic Correlation[J]. Chinese Journal of Electronics, 2015, 43(5): 956-964

[8] Yu Xiao-dong, Lei Ying-jie, Yue Shao-hua, et al. Research on PSO-based intuitionistic fuzzy kernel clustering algorithm[J]. Journal of Communication, 2015(5): 74-80 (in Chinese)

余晓东, 雷英杰, 岳韶华, 等. 基于粒子群优化的直觉模糊核聚类算法研究[J]. 通信学报, 2015(5): 74-80

[9] Liang Hua-dong, Han Jiang-hong. Clustering and Sorting Radar Signals Based on Multi-wavelet Packets Characteristics of Bispectrum[J]. Acta Photonica Sinica, 2014, 43(3): 152-159 (in Chinese)

梁华东, 韩江洪. 采用双谱多类小波包特征的雷达信号聚类分选[J]. 光子学报, 2014, 43(3): 152-159

[10] Sun Yan-wei, Peng Zhi-ming, Li Jian-bo. Community detection algorithm based on particle swarm optimization and fuzzy clustering[J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2015, 27(5): 660-666 (in Chinese)

孙延维, 彭智明, 李健波. 基于粒子群优化与模糊聚类的社区发现算法[J]. 重庆邮电大学学报(自然科学版), 2015, 27(5): 660-666

(上接第 176 页)

行。经过多次实验,测得该实时内核的主要时间参数及第一个调度周期内任务的执行结果,分别如表 1、图 5 所示。

表 1 混合实时系统内核主要的时间参数

内核时间参数	时间(μ s)
TT 任务切换时间	2.32
ET 任务切换时间	2.35
TT/ET 切换时间	2.44
中断响应时间	0.66
时钟中断总时间	10.92

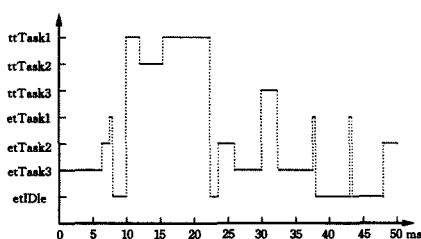


图 6 混合任务调度实验结果

由表 1 可知,该系统内核的时间特性满足大多数实时应用的需要。其中,时钟中断中由于要完成系统与时间有关的一些更新并进行 TT/ET 任务间的调度与切换,因此时间较长。

由图 6 所示的任务执行结果可知,TT 任务 ttTask1、ttTask2、ttTask3 在预定的时间点立即执行,体现出内核的实时性和确定性。0 时刻 TT 任务空闲,同时就绪的 ET 任务 etTask3、etTask2 和 etTask1 按照优先级依次执行,系统具有良好的确定性与可靠性。中断级及任务级 TT 任务与 ET 任务的灵活切换,使得系统调度更加灵活,系统利用率高。

结束语 本文根据 OSEK/VDX 标准,基于 μ C/OS-II 系统内核,扩展了其对于时间触发任务调度的支持,实现了系统对于时间/事件触发混合任务调度的支持,并给出了时间触发任务的可调度性测试算法。实验证明,该实时内核具有良好的实时性、确定性与可靠性;中断级与任务级的时间/事件触发任务调度使得内核更具灵活性,并能提高系统的利用率;该内核具有较大的理论研究与应用价值。

参考文献

[1] Van Den Heuvel M M H P, Bril R J, Lukkien J J, et al. RTOS support for mixed time-triggered and event-triggered task sets [C]//Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering. IEEE Computer Society, 2012: 578-585

[2] Kopetz H. Event-triggered versus time-triggered real-time systems[M]//Operating Systems of the 90s and Beyond. Springer Berlin Heidelberg, 1991: 86-101

[3] Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard-real-time environment[J]. Journal of the ACM (JACM), 1973, 20(1): 46-61

[4] Baruah S, Fohler G. Certification-cognizant time-triggered scheduling of mixed-criticality systems[C]//2011 IEEE 32nd Real-Time Systems Symposium (RTSS). IEEE, 2011: 3-12

[5] Itami Y, Ishigooka T, Yokoyama T. A Distributed Computing Environment for Embedded Control Systems with Time-Triggered and Event-Triggered Processing[C]//14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2008(RTCSA'08). IEEE, 2008: 45-54

[6] Chen Xi, Lv Wei-jie, Liu Lu-yuan. Design and implementation of real time kernel supporting event/time mixed trigger[J]. Computer Engineering and Applications, 2008, 44(16): 87-89 (in Chinese)

陈曦, 吕伟杰, 刘鲁源. 事件/时间触发嵌入式操作系统内核的设计[J]. 计算机工程与应用, 2008, 44(16): 87-89

[7] Dan Tu-nan, Zhu Li-ping, Yan Ji-xun. A Mixed Trigger Schedule Method Based on Time-Triggered Safety Critical Operating System[C]//China Aviation Science and Technology Conference. 2013: 1-5 (in Chinese)

谈图南, 朱立平, 颜纪迅. 一种基于时间触发的安全关键操作系统混合调度策略[C]//2013 首届中国航空科学技术大会论文集. 2013: 1-5

[8] OSEK Group. OSEK/VDX Operating System Specification[S/OL]. [2005-02-17]. <http://www.osek-vdx.org>

[9] OSEK Group. OSEK/VDX Time-triggered Operating System Specification, Version 1.0[S/OL]. [2005-02-17]. <http://www.osek-vdx.org>