时间/事件触发的安全关键系统调度研究

黄姝娟，刘白林，张 雅，茹 媛

(西安工业大学计算机科学与工程学院 西安 710021)

【摘要】针对大多数实时操作系统只支持事件触发的机制，该文提出了一种时间和事件双重触发的任务调度机制，并在μC/OS-II的内核中进行了实现。在该调度机制中，针对安全关键任务模型，提出了一种简单、易操作的基于关键度(criticality degree based priority, CDBP)的调度算法，该算法不仅保证了系统处于高级别时，高关键级别任务的执行，而且还保证了系统处于低级别时紧急任务的执行,同时减少了不必要的任务切换开销。实验结果表明,该算法在提高系统效率方面优于OCBP(own criticality based priority)算法。

关 键 词 嵌入式系统； 事件触发； 混合关键； 调度算法； 时间触发

中图分类号 TP319 **文献标志码** A doi:10.3969/j.issn.1001-0548.2017.03.025

Study on Scheduling Mechanism in Time-Triggered and Event-Triggered Safety Critical System

HUANG Shu-juan, LIU Bai-lin, ZHANG Ya, and RU Yuan

(School of Computer Science and Engineering, Xi'an Technological University Xi'an 710021)

Abstract For most real-time operating systems, only the event-triggered mechanism is supported. This paper proposes a scheduling mechanism which can support not only time-triggered but also event-triggered tasks in μC/OS-II. For safety critical tasks in this embedded system, a simple and easy scheduling algorithm is also presented based on criticality degree based priority (CDBP). This algorithm ensures the execution of the emergency tasks at a low level and the execution of the higher critical tasks at the high level while reducing the unnecessary task switching overhead. Experimental results show that the proposed algorithm is better than own criticality based priority(OCBP) algorithm in improving the system efficiency and provides better support for the criticality tasks and emergency tasks.

Key words embedded system; event-triggered; mixed-criticality; scheduling algorithm; time-triggered

时间或事件触发作为操作系统设计的主流思想在很多领域中得到了广泛应用[1]，然而在航空、航天等领域，由于系统的高可靠性与硬实时性要求，单纯的事件触发方式无论在设计还是维护方面都存在较大困难[2]。而时间触发方式在某种程度上会降低系统的灵活性，难以满足系统需求。尤其在安全关键系统中，为了保证安全关键任务执行的可靠性和确定性，以及整个系统的资源利用率，需要在完成时间触发任务的同时，支持事件触发任务[3]。

1 相关工作

时间触发的操作系统通常按照时间顺序分配时间槽来调度实时任务。文献[4]提出了一种改进型的Slot-shifting算法，在为时间触发任务预留时间槽的基础上，为事件触发任务动态分配执行时间。当预留给时间触发任务的时间槽没有用完时，剩下的部分可以分配给事件触发的任务；时间触发任务的执行时间还可以在预先定义的时间和离线计算的时间间隔内前后移动，以此来提高任务调度的灵活性。然而，这种方法的任务可调度性分析高度复杂，系统实现很困难。文献[5]为了在静态调度中适当安排事件触发任务，提出了一种轮询的方式在固定的时间点为事件触发任务分配时间。文献[6]和文献[7]分别提出了基于“super loop”和“Sandwich Delay”的时间触发合作式调度机制，可以加强系统的确定性并提高系统利用率，但对于非抢占式内核显然会降低系统的灵活性，且对安全关键任务的支持不够。文献[8]以时间触发总线和时间触发协议为基础，研

收稿日期：2015 – 12 – 29；修回日期：2016 – 09 – 27
基金项目：陕西省工业科技攻关项目(2015GY031)
作者简介：黄姝娟(1975 –)，女，博士，主要从事嵌入式与分布式计算方面的研究.

究了计算机系统架构，探讨了时间触发操作系统的特点。文献[9]通过对μC/OS-II操作系统的内核进行扩展，在μC/OS-II的任务管理机制中增加了对时间触发任务的支持。系统主要根据预先安排的时间槽进行时间触发任务的调度，在空闲时刻进行基于原μC/OS-II系统的事件触发任务的调度。这种设计很好地解决了时间/事件混合触发机制的问题，但是这种扩展只在时钟中断时进行时间/事件触发任务的切换，且时间触发任务被抢占后只能在下一调度周期才能恢复，系统利用率较低。

本文基于文献[9]设计了一种新的任务调度机制，将时间调度机制融入到μC/OS-II中，提出一种CDBP的调度算法，不仅满足了时间和事件双重触发的任务，而且利用任务结束到时钟中断之间的时间来处理事件触发任务，提高了系统的利用率，保证了安全关键任务的执行。

2 调度器原型设计

时间/事件双重触发系统要求能够同时支持TT(time triggered)任务和ET(event triggered)任务。为此，将系统调度器设计为时间触发调度模块和事件触发调度模块两大部分，并以时间触发调度部分为上层主要模块，事件触发调度部分为下层基础模块划分为层次性架构。当系统只有TT任务而没有ET任务时，系统调度器则可以关闭ET任务调度，在空闲时间执行Idle任务。

调度器作为系统内核的主要部分，时间触发模块和事件触发模块都向上提供系统服务API，向下都能和硬件通信交互。时间触发模块包括TT任务间的同步与通信、中断管理以及任务超时等错误处理。事件触发模块包括ET任务间的同步与通信、中断管理、资源管理、内存管理与警报等。系统调度器原型如图1所示。

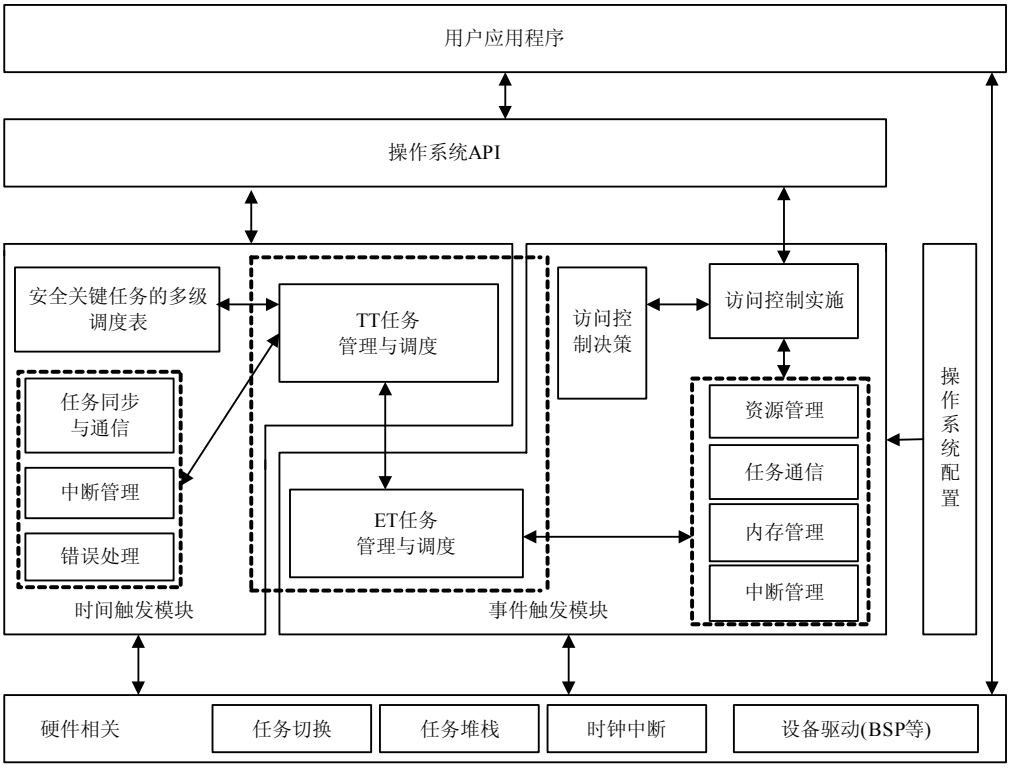

图1 时间/事件双重触发系统调度器原型

3 调度算法设计

3.1 安全关键任务模型设计

安全关键系统中的任务具有4个重要属性：发布时间、时限、关键级别和最坏执行时间。其中最坏执行时间是1个多维向量，向量值与任务的关键级别相关，各元素表示任务在各个级别下的最坏执行时间[10]。下面对多关键级别的安全关键任务模型进行形式化的定义。

定义 1 在1个有K个关键级别的安全关键系统中，关键级别最低为1，最高为K，任务表示为J_i，则有$J_i=(X_i,A_i,D_i,\boldsymbol{C}_i)$。其中，$X_i$表示任务$J_i$的关

键级别；A_i表示任务的发布时间；D_i表示任务的时限；$\boldsymbol{C}_i$表示任务的最坏执行时间，$\boldsymbol{C}_i$是一个向量，$\boldsymbol{C}_i=(C_i(1),C_i(2),\cdots,C_i(K))$，$C_i(1)$表示任务$J_i$在关键级别为1时的最坏执行时间，$C_i(2)$表示任务$J_i$在关键级别为2时的最坏执行时间，$C_i(K)$表示任务$J_i$在关键级别为$K$时的最坏执行时间。若$K>X_i$时，有$C_i(K)=C_i(X_i)$。本文的方法是基于状态切换的，研究的对象只限定在两个关键级别的系统中，即任务只具有一高一低两种安全关键级别。在一个具有2个关键级别的系统中，低关键级别为1，高关键级别为2，任务表示为$J_i=(X_i,A_i,D_i,C_i(1),C_i(2))$。

设想系统中有两个任务，其中J_1的关键级别为1，J_2的关键级别为2。当系统处于1级关键级别而J_2未能在$C_2(1)$时间内执行完时，系统会提升至2级关键级别，以保证J_2在$C_2(2)$时间内能完成执行，这时J_1的执行情况则不再重要。因此，对于低关键级别的任务J_1来说，系统不会允许它的执行时间超过$C_1(1)$。

由上述分析，可以得出两点结论：

1) 对于所有的任务J_i，都有$C_i(2)\geqslant C_i(1)$；

2) 如果任务J_i的$X_i=1$，那么$C_i(2)=C_i(1)$。

对于两级安全关键任务的系统I，需要设计关键级别不同的两张调度表，低关键级别状态下的S_1和高关键级别状态下的S_2。两张调度表的时间均从第一个任务的发布时间开始，到最后一个任务的时限结束，即$[\min_{J_i\in I}\{A_i\},\max_{J_i\in I}\{D_i\}]$。对于任意时刻$t\in[\min_{J_i\in I}\{A_i\},\max_{J_i\in I}\{D_i\}]$，$S_1(t)$和$S_2(t)$分别表示调度表中$t$时刻应该运行的任务。在系统运行过程中，对任务的调度执行遵循下列规则：

1) 系统当前所处的关键级别用Γ表示，而系统刚开始运行时，$\Gamma=1$。

2) 当$\Gamma=1$时，在每一时刻t，任务$S_i(t)$执行。

如果当前运行的任务J_i在$C_i(1)$时刻还未结束，那么系统在低关键级别下的调度表已经无法按计划完成了，这时系统需要提升关键级别，令$\Gamma=2$，也就是说此时将进行状态切换。

3) 当$\Gamma=2$时，在每一时刻t，任务$S_2(t)$执行。

如果对于两级安全关键任务系统I，能设计出满足上述规则的两张调度表S_1和S_2，则说I是时间触发可调度的(TT schedulable)[11]。

现有一个两级混合关键系统I，系统中有3个任务，其相关属性如表1所示。

对于这个安全关键系统来说，图2为一种可行的任务调度表。开始系统处于低关键级别，根据调度表S_1来进行任务调度：时刻0，任务J_1到达，即可触发调度器进行调度；时刻1，任务J_2触发，此时任务J_1尚未执行完毕即被J_2抢占，J_2开始执行；时刻2，任务J_3触发，如果J_2完成执行(即J_2运行时间小于等于$C_2(1)$)，那么J_3即可按时执行；到时刻3，当J_3执行完时，此时系统处于空闲，可恢复之前因被抢占而挂起的任务J_1。所有任务均正常执行，没有超过时限。

表1　示例任务属性表

任务	X_i	A_i	D_i	$C_i(1)$	$C_i(2)$
J_1	2	0	4	2	2
J_2	2	1	2	1	2
J_3	1	2	3	1	1

这是系统运行顺利情况下的任务调度。如果到时刻2，J_2运行时间已等于低关键级别的最坏执行时间$C_2(1)$，仍未执行完时，系统会将关键级别从1提升至2，并进行状态切换，调度器使用的调度表由S_1切换到S_2。J_2会保持运行直到时刻3。在系统处于高关键级别时，不再对低关键级别的任务J_3提供任何保障。到时刻3，J_2运行时间达到高关键级别下最坏执行时间$C_2(2)$，完成运行后，系统恢复之前因被抢占而挂起的任务J_1。

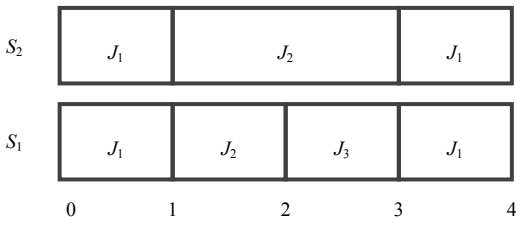

图2　示例任务调度表

3.2　调度表的创建

本文研究的是时间/事件触发的安全关键任务的调度，时间触发要求系统确定性好，使用资源预留的方法提前创建调度表，有利于提高系统的确定性。而事件触发则希望调度过程可以发生任务抢占，以此来保证高关键级别任务和紧急任务。为此，本文研究的调度方法，将时间预留和优先级抢占方法结合起来。创建调度表的时候首先根据任务在该级别下的最坏执行时间，为各个任务分配好时间槽；然后再给各个任务分配优先级，允许高优先级任务对低优先级任务进行抢占。

OCBP算法[12]的思想是如果调度可以顺利进行，每次找出剩余任务中优先级最低的任务，当所有任务都被指派优先级或者剩余的任务在当前的优

先级都不能被调度时，算法终止。该算法的缺陷有两个：一是任务分配过程中主要考虑任务的关键级别，与任务优先级无关；二是如果找不到最低优先级任务时，该算法就失败了。而本文提出的CDBP算法则全面考虑了任务的关键级别和紧急程度，且调度表的创建不会因为找不到最低优先级任务而终止。

定义 2 相对关键度 ρ_i，是指在具有n个任务的系统I中，任务J_i相对于其他任务的关键级别的重要程度。用 $\rho_i=\dfrac{x_i}{\sum_{j=1}^{n}x_j}$ 来表示，其中 x_i 和 x_j 是各任务在系统某个统一时刻的关键级别[13]。

假设系统中有4个任务，关键级别分别为1，2，1，2，则 ρ_i 分别为 $\frac{1}{6}$，$\frac{1}{3}$，$\frac{1}{6}$，$\frac{1}{3}$。

定义 3 关键额度 δ_i，是指在具有K个关键级别的系统I中，J_i当前所处级别在当前系统的关键级别k下所具有的关键份数。用 $\delta_i=\dfrac{x_i}{k}$ 来表示，其中 x_i 是任务的当前关键级别。假设系统中有两个任务J_1和J_2，其中X_1=1，X_2=2，则K=2。当系统关键级别k=1时，任务J_1和J_2的当前关键级别 x_1 和 x_2 也都是1，则 $\delta_1=\delta_2=1$；当系统关键级别k=2时，x_1=1和 x_2=2，则 $\delta_1=\dfrac{1}{2}$，$\delta_2=1$。

定义 4 时限紧急度 d_i 是指任务时限的先后对任务的紧急程度的影响，用 $d_i=\dfrac{1}{D_i^2}$ 来表示。在同等情况下，时限更早到来的任务应该具有更高的优先级。

定义 5 K关键级别下的关键度 θ_i^k 是指在K关键级别下，任务J_i的一次执行相对于其他任务的一次执行在所处关键级别的重要程度以及利用率的一种度量。用 $\theta_i^k=\rho_i\delta_i d_i$ 来表示。

从关键度 θ_i^k 的定义可以看出，它充分体现了任务J_i的关键级别相对于其他任务的重要程度和它在系统处于某一关键级别时在整个系统中的关键程度，以及该任务的时限对紧急程度的影响。以它为基础来为系统处于某一关键级别时的各任务分配优先级，能充分体现出任务优先级的特点，在系统处于低关键级别时，也减少了不必要的任务切换，提高了系统利用率。

根据CDBP算法为任务分配优先级的流程如图3所示。

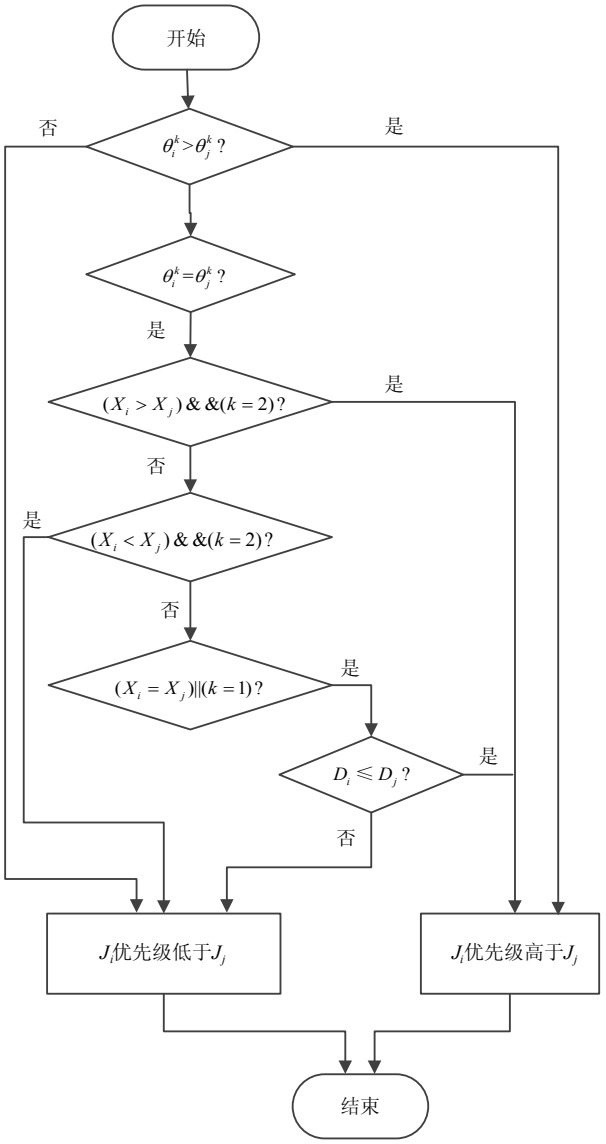

图3 CDBP算法优先级分配流程图

在CDBP的算法中，在K关键级别下关键度越大的任务具有的优先级越高。在两个任务关键度相同的情况下，若系统处于低关键级别或者两个任务的关键级别相同时，时限越早的任务优先级越高；若两个任务关键级别不同且系统处于高关键级别，则关键级别高的任务优先级高。

4 实验结果

本文是在嵌入式应用的开发平台Xilinx Virtex-5 FXT FPGA ML507上进行的实验，所设计的调度器也最终移植到该平台上进行了测试。

测试用例任务集有3个ET任务和4个时间触发的安全关键任务，其中ttTask2和ttTask4是高安全关键级别任务，ttTask1和ttTask3是低关键级别任务。

图4和图5显示了往ttTask2添加额外负载前后的运行情况对比，从图中可看出，开始ttTask2能够在低关键级别下的1个单位的时间内执行完毕，添加额外负载后，ttTask2无法按时完成，这时为了保证ttTask2能正常执行完，系统提升关键级别，ttTask1和ttTask3的执行不再得到保证，最终系统保证ttTask2、ttTask4和etTask1、etTask2、etTask3顺利执行完毕。该实验说明实现的多级调度表达到了设计要求，调度表成功切换，为高关键级别任务提供了保障。

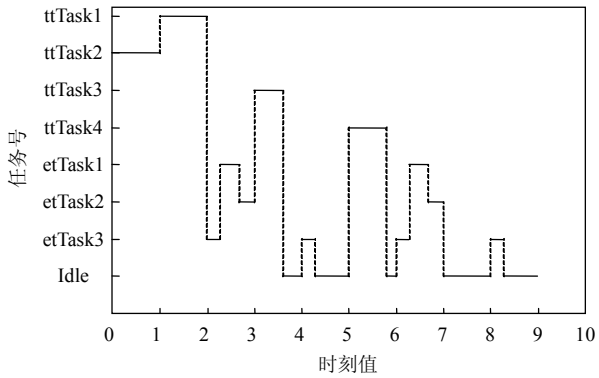

图4　低关键级别调度示意图

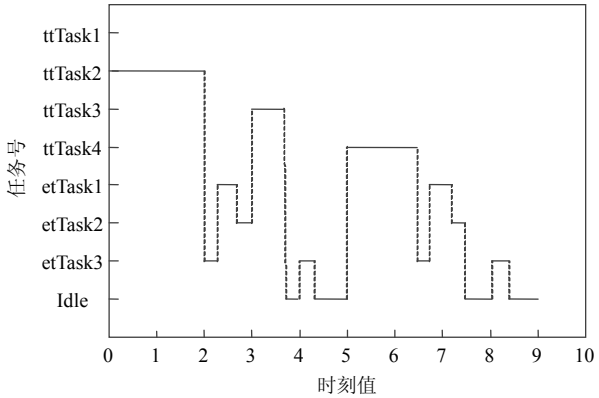

图5　高关键级别调度示意图

参 考 文 献

[1] HEEMELS W, DONKERS M C F, TEEL A R. Periodic event-triggered control for linear systems[J]. IEEE Journal of Transactions on Automatic Control, 2013, 58(4): 847-861.

[2] KOPETZ H. The complexity challenge in embedded system design[C]//The 2008 IEEE 11th International Symposium on Object OrientedReal-Time Distributed Computing (ISORC). Orlando: IEEE Computer Society, 2008, 5: 3-12.

[3] MARTIJN M H P, BRIL R J, LUKKIEN J J, et al. RTOS support for mixed time-triggered and event-triggered task sets[C]//The 2012 IEEE 15th International Conference on Computational Science and Engineering. Washington, USA: IEEE Computer Society, 2012, 12: 578-585.

[4] BARUAH S K, BURNS A, DAVIS R I. Response-time analysis for mixed criticality systems[C]//The 2011 IEEE 32nd Real-Time Systems Symposium (RTSS). Vienna: IEEE Computer Society, 2011, 11: 34-43.

[5] THEIS J, FOHLER G. Transformation of sporadic tasks for off-line scheduling with utilization and response time trade-offs[C]//The 19th Conference on Real-Time and Network Systems. Nantes, France: [s.n.], 2011, 9: 119-128.

[6] NAHAS M. Employing two ‘sandwich delay’ mechanisms to enhance predict-ability of embedded systems which use time-triggered co-operative architectures[J]. Journal of Software Engineering and Applications, 2011, 4(7): 411-417.

[7] PONT M J, KURIAN S, BAUTISTA Q R. Meeting real-time constraints using “Sandwich Delays”[M]. Lecture Notes in Computer Science, 2009(5770): 94-102.

[8] 苏罗辉, 牛萌, 刘坤. 时间触发系统体系结构研究[J]. 计算机工程与设计, 2014, 35(6): 1956-1961.
SU Luo-hui, NIU Meng, LIU Kun. Study on time-triggered system architecture[J]. Computer Engineering and Design, 2014, 35(6): 1956-1961.

[9] 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, 2009, 44(16): 87-89.
CHEN Xi, LÜ Wei-jie, LIU Lu-yuan. Design of event-triggered and time-triggered embedded operating system kernel[J]. Computer Engineer and Appllications, 2009, 44(16): 87-89.

[10] VESTAL S. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance[C]//The IEEE 28th International on Real-Time Systems Symposium (RTSS). Washington, USA: IEEE Computer Society, 2007, 12: 239-243.

[11] BARUAH S, FOHLER G. Certification-cognizant time-triggered scheduling of mixed-criticality systems[C]//The 2011 IEEE 32nd Real-Time Systems Symposium (RTSS). Vinenna: IEEE Computer Society, 2011, 11: 3-12.

[12] GU C, GUAN N, DENG Q, et al. Improving OCBP-based scheduling for mixed-criticality sporadic task systems[C]// The 2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). Seoul: IEEE Computer Society, 2013, 10: 247-256.

[13] 朱怡安, 黄姝娟, 段俊花, 等. 新的混合关键任务调度算法的研究[J]. 电子科技大学学报, 2014, 43(2): 268-271.
ZHU Yi-an, HUANG Shu-juan, DUAN Jun-hua, et al. New scheduling algorithm for mixed-criticality real-time task sets[J]. Journal of University of Electronic Science and Technology of China, 2014, 43(2): 268-271.

编　辑　蒋　晓