基于CPU使用率监测的软件容错研究

王小刚[1] 曹 东[2]
(南京航空航天大学自动化学院 南京 210000)[1] (南京航空航天大学飞行控制研究所 南京 210000)[2]

摘 要 在硬实时操作系统中,任务超时运行将会给系统带来灾难性后果。为了提高系统的可靠性和容错能力,系统设计需要采取一定容错策略。系统的CPU使用率是实时系统运行正常与否的重要指标,其可以表征系统的时间特性和任务状态。针对CPU使用率的特点以及容错监测的要求,选取机器周期作为时间信息统计的基准;分析监测周期不同对监测效果的影响;提出了嵌入式实时系统CPU使用率异常的判决条件;并结合结构冗余和时间冗余的思想,设计了4种处置方法用于CPU使用率异常处置。仿真测试表明,基于CPU使用率监测的软件容错方法可以有效提高系统的可靠性和容错能力。

关键词 实时系统,软件容错,CPU使用率,异常处置
中图法分类号 TP302.8 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.07.004

Research of Software Fault-tolerance Based on Monitoring of CPU Utilization Ratio

WANG Xiao-Gang[1] CAO Dong[2]
(College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China)[1]
(Flight Control Research Institute, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China)[2]

Abstract In hard real-time system, overtime of task running can induce a disastrous consequence of system performances. To enhance reliability and fault-tolerant ability of system, some software fault-tolerance strategy must be used. CPU utilization ratio of system is a key characteristic, which tells whether system is normal or not. Time feature and task state can be shown by CPU utilization ratio. Because of characteristic of CPU utilization ratio and demand of fault-tolerance, machine cycle was selected as measurement baseline. There are great differences when different monitor cycle is selected. Conditions of judging CPU utilization ratio anomaly in system were presented. Four measures that refer to structural redundancy and time redundancy were designed to handle this case. The simulation shows that software fault-tolerance based on monitoring of CPU utilization ratio can enhance reliability and fault-tolerant ability of system effectively.

Keywords Real-time system, Software fault-tolerance, CPU utilization ratio, Exception handling

1 概述

嵌入式实时系统是一类嵌入到其他物理设备中且对其上执行的任务有实时计算要求的特殊计算系统[1]。随着大规模集成电路技术的迅猛发展和微电子工艺水平的不断提高,以及软件工程技术的不断完善,嵌入式实时系统日益深入地被应用到对任务执行时间和失效影响有严格要求的领域,如航空电子系统、无人机飞行控制系统、机器人系统、医疗设备和汽车电子系统。用于这些领域的嵌入式实时系统一旦失效,可能会给人们的生命和财产带来巨大的损失和灾难[2]。因此,保证嵌入式系统的正常运行、消除各种潜在的安全隐患、提高嵌入式实时系统的容错能力成为了各国科研人员十分迫切的科研任务。

容错是指数字系统存在或正在发生某些允许范围内的故障时能够继续保持正常(或降级)运行的方法和策略[3]。在容错设计中,使用较多的方法是冗余技术。一般来讲,一个具有冗余结构的系统可能采取某种或多种措施实现容错功能,这些功能主要有:故障限定、故障检测、故障屏蔽、功能复执、故障诊断、部件重组、系统恢复、系统重启、系统修复、系统整合等[4]。容错系统实现的一般思路是监测系统状态,在系统出现异常时发现系统异常并进行相应的处理,以保证系统的安全正常运行[5]。为了提高系统的可靠性与容错能力,很多策略与方法已经被应用到嵌入式系统开发中,这些方法包括双机冷备份技术、余度计算机技术、数据容错、基于时间冗余的检查点技术和缓存恢复技术等,这些技术从多个角度提高软件某些方面的容错能力,在实际工程应用中发挥了较大的作用。由于软件中可能出现的故障或错误多种多样,某种软件容错技术仅能从某一个或某几个方面提高软件的容错能力,因此,软件容错技术仍然是容错技术研究的一个重要方向。

CPU的使用率是嵌入式系统的重要性能指标,其可以反

到稿日期:2013-09-03 返修日期:2013-11-02
王小刚(1989—),男,硕士生,主要研究方向为嵌入式实时系统、飞行控制技术,E-mail:wangxg@nuaa.edu.cn;曹 东(1972—),男,副研究员,主要研究方向为嵌入式系统、飞行控制技术。

映系统运行的时间特性和任务状态。系统中每个任务都是按照开发人员设计的运行路径正常运行。在这种情况下，每个任务的执行时间会在相对应的标称范围内，对于整个系统而言，整个系统的运行时间也会处于标称范围内。如果整个系统或某些任务的CPU使用率异常，即其运行时间超出标称范围，则说明系统的整体或某个部分出现了异常，有可能导致系统失效。例如，实时系统在异常状态下，系统中部分任务可能运行超时，进而可能导致实时任务无法保证时限要求，带来无法预期甚至灾难性的后果[6]。因此，对嵌入式系统的CPU使用率进行监测十分必要，在监测到异常时，必须及时进行处置，消除系统故障或抵消故障带来的影响，提高系统对于时间相关的故障容错能力，保证系统正常运行。

2 CPU使用率的监测

系统在一段时间内，一个任务的CPU使用率是指系统运行该任务所花费的时间与该段时间长度的比值，计算方法如式(1)所示。

$$P_{cpu}=(t_{task}/t_{total})\times 100\% \quad (1)$$

其中，t_{total}是统计CPU使用率时间段的时间长度，t_{task}是在该段时间内，该任务运行所耗费的CPU时间。

2.1 CPU使用率监测周期

CPU使用率监测周期的选取对于监测效果有着重要的影响，监测周期过长，可能造成所获得的数据不够准确；监测周期过短，则会造成系统资源浪费。如图1所示，图中黑色部分表示CPU在使用中，白色部分表示CPU空闲。如果以0.5s为监测周期，假设第一个0.5s内的CPU使用率达到了90%，第二个0.5s内的CPU使用率仅为30%，则系统出现了CPU占用过高的情况。而如果以1s为监测周期，则得到的CPU使用率为60%，极有可能监测不到CPU使用率异常的情况。在嵌入式实时软件中，CPU使用率是用来监测系统状态的，因此监测的准确度较为重要。为保证监测结果的可靠，监测任务频率大于或等于系统中所有其他任务的最大运行频率，如式(2)所示。

$$f_{monitor}\geqslant\max(f_1,f_2,\cdots,f_i,\cdots f_n),i\in[1,n] \quad (2)$$

其中，n为系统中任务的个数，f_i为系统中各个任务的运行频率。

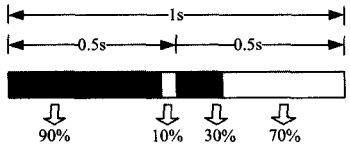

图1 CPU使用率监测周期的影响

2.2 嵌入式操作系统CPU使用率监测方法

CPU使用率的计算原理并不复杂，其计算的准确与否取决于用户程序运行时间的获取。根据需求不同，其获取的方法也是不同的。uC/OS-II和VxWorks是常见的两种嵌入式操作系统，两种操作系统均提供了CPU使用率的计算方法或工具。

uC/OS-II是常见的嵌入式操作系统，其在医疗器械、网络设备和工业机器人等诸多领域得到了广泛的应用[7]。uC/OS-II操作系统本身提供了用于统计运行时间的任务OSTaskStat()，这个任务可以统计系统的CPU使用率。该统计方法的基本思想是：在启动时，运行系统空闲任务*IdleTask*()，并记录1s内的CPU计数值；在系统运行时，记录每秒的空闲任务计数值，进而实现CPU使用率的计算。在实际使用中，统计函数首先测定在没有其他应用任务运行时，系统空闲计数器的计数值，并将其作为系统空闲计数器的最大值；然后根据空闲计数器的当前计数值和最大计数值，计算CPU使用率。在uC/OS-II系统中，该统计任务的运行频率是1Hz，精度为1%。其计算方法如式(3)所示。

$$P_{cpu}(\%)=100-osidlectr/(osidlectrMax/100) \quad (3)$$

其中，*osidlectr*为系统空闲计数器的当前计数值，*osidlectrMax*为系统空闲计数器的最大计数值。

VxWorks操作系统是常用的、高可靠的嵌入式操作系统，其提供了用于统计任务时间信息的组件SPY，该组件可以统计每个任务所使用的CPU时间[8]。SPY组件的原理是通过一个与辅助时钟连接的中断级函数，记录系统单位时间内的时间*tick*和各任务的时间*tick*，计算各个任务的CPU使用率[9]。SPY组件可以提供较为详细的任务时间信息，但其使用需要CPU具有辅助时钟的功能。另外，VxWorks中统计时间信息任务的频率和精度与辅助时钟有较大关系。

uC/OS-II操作系统和VxWorks操作系统提供的CPU使用率统计工具为开发人员在系统设计过程中提供了巨大的帮助。然而，以上两种操作系统提供的CPU使用率监测方法不适用于高可靠性系统的实时监控。uC/OS-II中所使用的方法，其统计的结果是系统空闲率，精度为1%。这种方法无法监测每个任务的CPU使用率，不利于对发生故障的任务进行定位。而VxWorks操作系统提供的SPY组件虽然可以对每个任务时间信息进行监测，但是其依赖辅助时钟，其计时的时间片大小与辅助时钟的最大*tick*次数有关。同时，SPY组件的运行会占用辅助时钟中断，可能会对系统其它功能设计带来不便。实现基于CPU使用率的软件容错，需要对系统中的任务进行多任务和准确的监测，并将附加的容错功能对系统的影响降到最小，因此，以上两种方式均不适合用于软件容错设计。为了达到容错的目的，需要选取更加合适的时间基准和方法。

现今，大部分主流CPU通常使用一个64位寄存器(TB)来记录CPU上电后的机器周期个数[10]。通过特定的汇编指令，可以获取这个寄存器中机器周期的计数值。基于处理器的这个功能，可以采用机器周期作为实现CPU使用率准确监控的时间基准。对于任务执行时间的统计，采用分别记录任务结束运行的机器周期计数与任务开始运行时的机器周期计数，以两者的差值计算任务运行时间的方法。VxWorks操作系统提供了任务切换的*Hook*函数[11]，在有任务切换时，调用预先挂接的*Hook*函数实现某些功能，利用这种功能，在发生任务*A*切换至任务*B*时，获取当前机器周期的计数值*Counter*，作为*B*任务本次运行开始的机器周期计数。同时，将*Counter*作为任务*A*本次运行结束的机器周期计数，计算本次运行所耗费的机器周期个数，并累加到累加器上。在系统每个周期监测点到来时，分别获取各个任务所耗费的CPU时间，并计算对应任务的CPU使用率。

2.3 CPU使用率异常判决

嵌入式实时系统在整个运行过程中，可能处于不同的工作模态，不同工作模态具有不同的CPU使用率标称曲线。因

此，判断嵌入式系统的 CPU 使用率是否异常，应区分不同模态。实际使用中的嵌入式实时系统多为多任务系统，其任务调度所产生的 CPU 使用率的曲线呈周期性曲线的形式，其周期为所有任务运行周期的最小公倍数[12]。图 2 是某嵌入式实时系统在待机状态下 2s 内的 CPU 使用率曲线，图 3 是图 2 某一部分的放大图。由图 2 可以看出，该系统中存在周期各不相同的任务，所以在待机状态下，其 CPU 使用率呈现周期性曲线形式。虽然各时刻 CPU 使用率并不相同，但是在一个周期过程的某个幅值区间内，固定范围的峰值个数应该基本稳定。

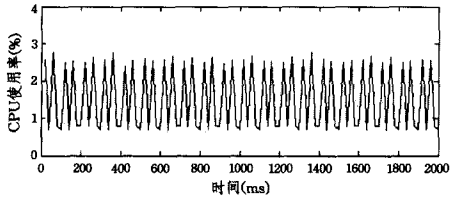

图 2　某嵌入式实时系统待机状态的 CPU 使用率曲线

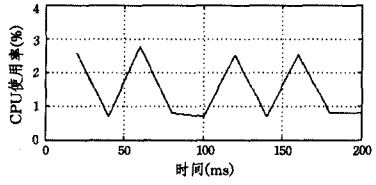

图 3　某嵌入式实时系统待机状态 CPU 使用率曲线局部放大图

在图 3 中，CPU 使用率在 $t=20$ms、60ms、120ms 时刻幅值分别约为 2.56%、2.76%和 2.5%。在 2s 内，CPU 使用率在 2%～3%范围内出现了 20 次，即 10 次/s。因此，对于此类嵌入式实时系统，CPU 使用率正常判决条件为式(4)和式(5)：

$$\sum_{k=1}^{k} f(i\cdot\Delta t,p)=\delta \tag{4}$$

$$f(i\cdot\Delta t,p)=\begin{cases}1, & p\in[p_{\min},p_{\max}]\\ 0, & p\notin[p_{\min},p_{\max}]\end{cases} \tag{5}$$

其中，i 为单位时间内监测点的计数，Δt 为监测周期，p 为 CPU 使用率，$p_{\min}$ 和 $p_{\max}$ 分别表示 CPU 使用率某个区间的上下限，$f(i\cdot\Delta t,p)$ 表示单位时间内 CPU 使用率处于监测区间内的计数，δ 表示正常情况下单位时间内 CPU 使用率处于监测区间内的计数。例如，上例中，系统监测周期为 20ms，在 1s 内，使用率在 2%～3%之间的监测点出现了 10 次，则判决条件可以用式(6)与式(7)表示：

$$\sum_{k=1}^{k} f(i\cdot\Delta t,p)=10 \tag{6}$$

$$f(i\cdot\Delta t,p)=\begin{cases}1, & p\in[2\%,3\%]\\ 0, & p\notin[2\%,3\%]\end{cases} \tag{7}$$

上述公式构成了[2%，3%]区间内的判决条件，而整个系统正常的判决条件为各个类似区间条件的组合。考虑到上述条件在实际应用中过于严格，我们将 CPU 使用率正常条件修正为式(8)和式(9)：

$$\sum_{k=1}^{k} f(i\cdot\Delta t,p)\in[\delta-\Delta,\delta+\Delta] \tag{8}$$

$$f(i\cdot\Delta t,p)=\begin{cases}1, & p\in[p_{\min},p_{\max}]\\ 0, & p\notin[p_{\min},p_{\max}]\end{cases} \tag{9}$$

其中，Δ 是根据实际情况设置的单位时间允许的误差次数。

若系统在某个峰值区间内出现的频率高于预期，则需要首先采取降低 CPU 使用率的手段，然后查找异常任务，并进行相应处置。如果系统在某个峰值区间内出现的频率低于预期，则应直接查找异常任务，并进行相应处置。

3　CPU 使用率异常处置

嵌入式实时系统出现 CPU 使用率异常的情况，可能是由多种原因造成的。从硬件角度考虑，当运行的环境不同时，硬件器件运行性能存在差异，从而可能造成 CPU 使用率的不同；从软件角度考虑，软件运行过程中，可能存在执行了非预期操作的可能性，如非预期的循环、任务的异常调度等。

CPU 使用率异常处置的方法主要有两类：主动纠错和被动容错。主动纠错是指在监测到异常时，首先主动分析导致故障的原因，然后根据故障的不同，采取不同的方式进行处置，排除故障，使系统恢复正常；被动容错是指在监测到异常时，并不去查找故障原因，而是针对系统可控部分进行处置，以减少或抵消故障对系统带来的影响，从而保证系统的正常运行。嵌入式实时系统的 CPU 异常处置最理想的方案是主动纠错方案，但在实际应用中，由于导致嵌入式系统 CPU 使用率异常的原因较多，而且并不容易区分，通常要结合被动容错来达到整个系统容错的目的。CPU 异常处置根据不同任务的特征，采用不同的处置方法，这些方法主要有运行备份任务、任务降频运行、异常任务重启和系统重启。

表 1　CPU 使用率异常处置方式

序号	处置方式	处置方式类型
1	运行备份任务	被动容错
2	任务降频运行	被动容错
3	异常任务重启	主动纠错
4	系统重启	被动容错

3.1　运行备份任务

在嵌入式系统中，某些任务的实现可以采取不同的方式，其所消耗的时间也不相同。对于此类任务可以编写备份任务。在时间资源充裕情况下，运行正常版本的任务；当时间资源不足时，运行备份版本任务。适合构造备份的任务主要有：

· 计算类任务

计算类任务主要包含需要迭代运算和需要较高运算精度的任务。需要迭代计算的任务运行时，迭代次数越多，其运算结果越精确；需要较高运算精度的任务运行时，采用的数据精度越高，其运算结果越精确。

· 功能可精简类任务

任务的执行时间依赖于所执行功能的情况，比如大数据量通讯任务，其传输的数据通常并不完全必要，部分数据的精简可能对系统影响较小。

在设计嵌入式系统时，对于计算类任务可以编写迭代次数较少或数据精度较低的备份任务，在输出结果可接受的前提下，降低精度或减少迭代次数，以减少对系统资源的占用；对于功能可精简类任务，在保证系统关键功能的前提下，对原有任务进行简化，以减少对系统资源的占用。比如在飞行控制系统中，对于导航信息计算等大运算量任务，可以编写精度较低的备份任务；对于遥测数据发送，可以对数据帧进行裁剪。在系统实际运行中出现 CPU 占用率过高的情况下，主动将此类任务切换至备份任务，可以使 CPU 较繁忙时刻的 CPU 使用率峰值减小。

3.2 任务降频运行

嵌入式系统中，部分任务的执行效果依赖它的执行频率，执行的频率越高，其运行结果越好。适合设计可降频的任务主要有：

·监测类任务

此类任务主要用于某些状态信息的监测。其监测频率越高，越能及时反映某些状态的变化，监测效果越好，如飞行控制系统中，传感器监测任务及其他监测任务。这些任务监测的准确性与其监测频率相关，监测任务的周期越短，监测效果越好，但系统消耗越大。

·通信类任务

此类任务的功能通常是与外界进行交互，其任务频率越高，数据的通讯延迟越小，如飞行系统中，传感器数据接收和遥控遥测通信。

在设计嵌入式系统时，可降频任务的任务执行频率可以设计为可变形式。如果出现 CPU 使用频率过高，在保证其任务功能正常实现的情况下，降低其运行频率，可以降低 CPU 负担，保证系统中关键任务的正常运行。

3.3 异常任务重启

异常任务重启包含两种方式：单任务重启和用户任务全部重启。CPU 占用率过高的较为常见的原因是任务执行了非预期的执行顺序或路径，或者资源的申请或释放出现异常等导致意外的阻塞。单任务重启是指在某个任务的 CPU 占用率出现异常时，检查是否有任务状态出现异常。若有任务出现异常，则将此任务以及相关任务重启，并进行任务状态恢复，重新执行，以期降低 CPU 使用率，保证系统正常运行。用户任务全部重启的流程是，首先保存系统关键运行状态信息，然后删除非系统级任务，删除所有信号量等变量。之后重新初始化任务，初始化信号量等。系统重启后，由于所有任务均重新初始化，内存堆栈重新分配，使得所有任务重新回到正常状态的可能性大大增加。

3.4 系统重启

系统重启是利用看门狗定时器实现的。看门狗定时器是较为常见的防止程序“跑飞”的手段[13]，其主要实现思路是，程序正常运行时，需要给看门狗定时器定时“喂狗”；当超过看门狗的刷新时间后，看门狗将产生一次系统复位或关键中断。利用看门狗定时器的这一特性，在系统发生 CPU 占用率过高时，不对看门狗定时器进行“喂狗”，主动激发系统重启，使所有任务和变量重新初始化，程序重新运行[14]。系统重启后，其目标不是将程序运行状态恢复到重启前，而是保存系统控制的关键状态，在重启后保证系统控制的平稳和连续，进而屏蔽掉故障的发生条件，使系统继续执行控制功能。

CPU 使用率异常处理流程如图 4 所示。4 种 CPU 使用率异常处理方式各不相同，备份任务和任务降频运行的方法在不定位发生异常任务的状态下，仅对系统运行做适当的调整或精简，在系统异常情况下，提供一定的故障容忍能力。如果 CPU 使用率自行恢复正常后，系统也可以重新恢复正常状态。这两种处理方式对系统运行干扰较小，应作为首选方式。异常任务重启就是定位异常任务，以进行处理，其对于解决 CPU 使用率异常有较好效果。系统重启是在 CPU 使用率异常时，对系统进行全面的重启处理，其作为解决问题的最后手段，在带来较好效果的同时对系统运行的影响也较大。

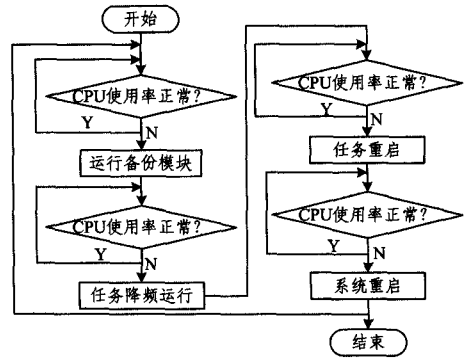

图 4 CPU 使用率异常处理流程图

在 CPU 使用率异常时，优先使用备份模块和任务降频运行的方式。如果未能降低 CPU 使用率，则判断各个任务的状态是否正常。如果有不正常的任务，则重启该任务。如果没有不正常的任务，则考虑进入下一步系统重启。

4 CPU 使用率异常处置的仿真验证

CPU 使用率异常处置的仿真验证是基于某飞行控制系统验证环境进行的，飞行控制系统验证环境包含 3 个子系统，分别为飞行控制系统、仿真系统与测控系统。飞行控制系统包含飞行控制计算机(嵌入式计算机)及对应机载飞行控制软件；仿真系统包含仿真计算机(PC 机)及对应仿真软件；测控系统包含测控计算机(PC 机)及对应测控软件。其中飞行控制计算机是基于 MPC5554 开发的嵌入式计算机，机载软件是基于 VxWorks 操作系统开发的飞行控制软件；仿真系统与测控计算机均在 PC 上运行，辅以相应的接口转换模块，对应软件均为 VC6.0 环境下开发的 WinForm 软件。飞行控制软件为主要验证对象，其基于 VxWorks 操作系统设计，软件包含周期任务 21 个，其中最小周期任务的周期为 20ms；CPU 使用率监测任务与异常处置任务作为系统级任务具有应用层任务最高优先级。根据上述 CPU 使用率监测周期的选取要求，选取监测周期为 20ms。由于仿真验证中涉及每个任务的 CPU 使用率判断，每个任务在不同模态下，具有不同区间段的 CPU 使用率判决条件。限于篇幅有限，无法将仿真结果详细罗列，敬请见谅，仿真验证部分仅给出系统的 CPU 使用率曲线，其它各个任务的 CPU 使用率仅作为监测信息供系统内部使用。

4.1 备份模块和任务降频运行仿真验证

备份模块和任务降频运行分别对 CPU 繁忙点的峰值和繁忙点的出现频率有效果。仿真验证中，以遥测数据发送任务作为样例任务，该任务既可对数据帧长度进行裁剪以实现备份模块功能的仿真验证，同时也可以降低其运行频率，实现任务降频运行的仿真验证。为避免其他任务调度的影响，测试中仅运行系统基本任务和遥测数据发送任务。该模态下 CPU 使用率曲线如图 5 所示，图 5 中，纵轴为 CPU 使用率百分比，横轴为时间，单位为 ms。

由图 5 可以看出，在运行备份任务中，遥测数据帧的数据字节数由 39 字节/帧变为 20 字节/帧后，CPU 使用率峰值由 2.76%降低至 1.75%，考虑系统本身调度的消耗为 0.66%。CPU 使用率峰值降低了 1.01%，遥测任务 CPU 使用率最多减少约为 48%。而在该任务降频运行之后，系统峰值出现频率降低一半。另外，在多任务运行时，任务频率的降低意味着系统任务调度次数的减少，系统调度消耗也会降低。由仿真

结果可知，备份任务与任务降频对降低 CPU 使用率有效，其效果取决于对任务所进行功能裁剪的多少或降频处理的倍数。在系统作异常处置时，对于可运行备份模块的任务，所需要保留的功能越少，该方法的处置效果越好；对于可降频运行的任务，频率降低越多，该方法的处置效果越好。

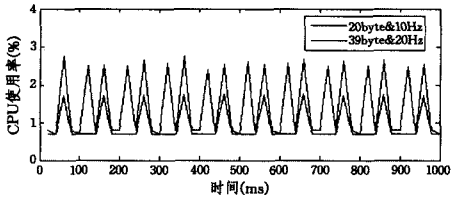

图 5　CPU 使用率异常处置——备份任务和任务降频

4.2　异常任务重启仿真验证

异常任务重启对 CPU 使用率异常的处置具有很好的效果。仿真验证中，设计一段实验测试代码，其运行时占用 CPU 资源较多，其运行与否可以通过指令触发，仿真结果如图 6 所示，在 t=400ms 时，触发实验代码，如故障状态曲线所示；在触发测试代码之后，CPU 使用率出现过高的情况，在该种模态的正常运行状态下，20ms 内 CPU 使用率超过 20%的监测点为 0，对应正常状态判决条件为：

$$\sum_{k=1}^{k} f(i\cdot\Delta t,p)=0 \tag{10}$$

$$f(i\cdot\Delta t,p)=\begin{cases}1, & p\in[20\%,100\%]\\0, & p\notin[20\%,100\%]\end{cases} \tag{11}$$

而在 $t\in[0,1s]$时，CPU 使用率超过 20%的监测点为 12，在 t=1200ms 时，确认 CPU 使用率异常，并进行任务重启处置，清除导致异常的条件，然后 CPU 使用率回到正常状态。仿真测试中，采用指令触发式进行故障模拟，但其故障产生的机制与偶发性逻辑故障类似。偶发性逻辑故障是指在某种状态下，不期望的外界环境或操作导致程序逻辑进入非预想状态。此类故障只要清除故障发生的触发条件，即可使系统恢复正常。任务重启方式，通过将任务状态按照预期模式进行恢复，使逻辑状态从非预期状态恢复到预期状态，以达到整个系统功能恢复正常的目的。

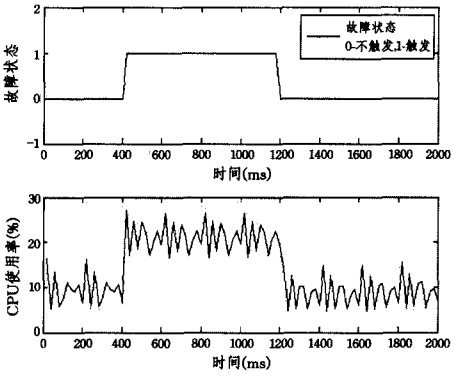

图 6　CPU 使用率异常处置——异常任务重启

4.3　系统重启仿真验证

CPU 使用率异常的系统重启仿真验证参照任务重启的仿真验证方式，编写测试代码，触发后增加 CPU 资源的占用，并将触发条件设置为全局，使任务重启无法清除故障触发条件，以模拟非偶发性逻辑故障。测试结果如图 7 所示，在触发测试代码之后，CPU 使用率出现过高的情况，在该种模态的正常运行状态下，20ms 内 CPU 使用率超过 20%的监测点为 0，对应正常判决条件为：

$$\sum_{k=1}^{k} f(i\cdot\Delta t,p)=0 \tag{12}$$

$$f(i\cdot\Delta t,p)=\begin{cases}1, & p\in[20\%,100\%]\\0, & p\notin[20\%,100\%]\end{cases} \tag{13}$$

而在 $t\in[0,1s]$时，CPU 使用率超过 20%的监测点为 12，在 t=1200ms 时，确认 CPU 使用率异常，进入系统重启。由于重启过程中，CPU 和系统变量重新初始化，CPU 使用率在此过程中的监测结果为 0%；最后，在重启完成后，系统控制功能恢复，CPU 使用率也恢复正常。

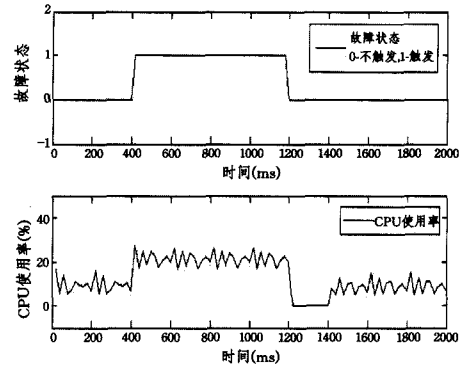

图 7　CPU 使用率异常处置——系统重启

结束语　CPU 使用率是嵌入式实时系统性能的重要体现之一，当其出现异常时，必须给予足够的重视。本文分析了嵌入式实时系统 CPU 使用率的特点，设计了合适的监测策略，并提出了 4 种异常处置方法。在不增加系统额外硬件配置的情况下，从软件的角度，提高嵌入式实时系统的容错能力，对于保证嵌入式系统的正常运行有着重要的作用。

参 考 文 献

[1] Krishna C M, Shin K G. Real-time systems[M]. McGraw Hill Education, 1997: 1-3

[2] 桂盛霖. 安全关键嵌入式实时系统的关键非功能属性分析研究[D]. 成都：电子科技大学，2007

[3] 徐拾义. 容错计算系统[M]. 武汉：武汉大学出版社，2010：314-315

[4] Cai K Y. Censored. Software-Reliability Models [J]. IEEE Trans. Rel. , 1997, 46(1): 69-75

[5] Shooman M L. Software Engineering: Design, Reliability and Management[M]. McGraw-Hill, New York, ch. 5, 1983

[6] Liu J W S. 实时系统(翻译版)[M]. 姬孟洛，等译. 北京：高等教育出版社，2003：25-26

[7] Jean J L. 嵌入式实时操作系统 uC/OS-II(第 2 版)[M]. 邵贝贝，等译. 北京：北京航空航天大学出版社，2003：97-103

[8] 张扬，于银涛. VxWorks 内核、设备驱动与 BSP 开发详解(第 2 版)[M]. 北京：人民邮电出版社，2011：58-62

[9] River W. VxWorks Programmers' Guide[DB/OL]. Wind River System Inc. Mar. 2003

[10] Soja R, Bannoura M. MPC5554/5553 微处理器揭秘[M]. 龚光华，等译. 北京：北京航空航天大学出版社，2010：70-74

[11] 曹桂平，等. VxWorks 设备驱动开发详解[M]. 北京：电子工业出版社，2011：23-25

[12] Bruyninckx H, Leuven K U. Real-time and Embedded Guide [M]. Mechanical Engineering, Leuven, Belgium, 2001

[13] Freescale. MPC5553/5554 Microcontroller Reference Manual [M]. April 2007

[14] 张靓，刘光明. RTEMS 嵌入式系统中的软件容错设计[J]. 计算机工程与科学，2007，29(5)