文章编号：2095-2163(2019)03-0110-04 中图分类号：TP316 文献标志码：A

基于 ARINC 653 仿真器的机载操作系统任务状态监控设计与实现

孟开元，闫 方，曹庆年，彭 寒
（西安石油大学 计算机学院，西安 710065）

摘 要：机载操作系统属于嵌入式实时操作系统，其内部内核对象的状态变迁、关键数据的值变化以及系统调度运行状况直接影响着机载应用程序的正确执行。本文以为机载应用程序更好地定位故障、发现并排除各种逻辑错误为目的，采用内嵌传感器的方法，结合任务的状态切换、触发状态切换，应用 ARINC 653 仿真器实时监控机载操作系统的状态和数据变化。经过测试，表明该系统可以得到机载应用程序的状态和行为变迁，并为机载应用程序的运行时验证、机载分区操作系统仿真器的符合性测试提供基础。

关键词：机载操作系统仿真器；机载应用程序；状态变迁；内嵌软件传感器

Design and implementation of task status monitoring for airborne operating system simulator based on ARINC 653 emulator

MENG Kaiyuan, YAN Fang, CAO Qingnian, PENG Han
(School of Computer Science, Xi′an Shiyou University, Xi′an 710065, China)

【Abstract】 The on-board operating system is an embedded real-time operating system. The state transition of internal kernel objects, the change of key data values, and the system scheduling operation directly affect the correct execution of the onboard application. This article considers that the onboard application better locates faults, discovers and eliminates various logic errors, and adopts embedded sensor method, combined with task state switching and trigger state switching based on ARINC 653 emulator, to monitor the status and data changes of the onboard operating system in real time. Tests show that the system can get the status and behavior changes of the onboard application, and provide the basis for the runtime verification of the onboard application and the compliance test of the onboard partition operating system emulator.

【Key words】 on-board operating system simulator; on-board application program; state transition; embedded software sensor

0 引 言

随着机载电子系统快速发展，机载操作系统作为学术界一个热门研究方向，也一直都是技术研发上的亮点和难点[1]。机载操作系统研究和发展起源于二十世纪中叶，最初机载实时操作系统主要与军事上需求紧密相关，但随着计算机技术应用范围不断扩大，实时操作系统已经应用于人们生活中各个领域。然而无论是在嵌入式领域、还是在通用领域，对于操作系统实时性都有严格要求。如果机载操作系统的实时性不能满足需要，将会带来不可预测后果。因此，机载操作系统实时性能是衡量机载系统优劣性的重要指标[2-3]。本文针对实时机载操作系统安全性不断提升这一项目任务，亟需对其任务状态进行监控和分析，为此专门探讨和研究了机载操作系统仿真器任务状态监控方法，设计和实现一个仿真器任务状态监控系统。

1 ARINC 653 操作系统仿真器原理

为了实现资源共享，采用 IMA（Integrated Modular Avionics system）架构的航空电子系统，软件体系结构采用了软件分层的方法[4-5]。而具有代表性的综合模块化软件体系架构之一就是 ARINC 653。如图 1 所示，ARINC 653 在应用程序、操作系统、硬件之间建立标准接口，即 APEX（Application/Executive）接口，并且提出了分区这一核心的概念[6-7]。

而分区是航空电子应用的一种功能划分。每个分区分别有 4 种模式：WAREM_START、COLD_START、IDLE、NORMAL[8]。而分区内是以进程为一个执行单元的。仿真器就是利用 Windows 线程仿真 ARINC 653 的进程，同时每个进程又各有 4 种状态，

作者简介：孟开元（1968-），男，副教授，主要研究方向：计算机网络与通信；闫 方（1995-），女，硕士研究生，主要研究方向：计算机接口技术、控制系统；曹庆年（1963-），男，教授，主要研究方向：计算机网络与通信；彭 寒（1976-），男，硕士，助教，主要研究方向：软件工程。

收稿日期：2019-03-06

即：运行态（RUNNING），就绪态（READY），休眠态（DORMANT）和等待态（WAITING）[9]。

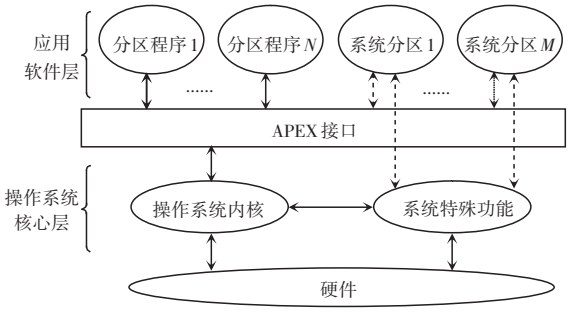

图 1　ARINC653 操作系统体系结构

Fig. 1　ARINC653 operating system architecture

本文通过分析机载仿真操作系统 ARINC653 源代码，在任务状态发生变化的关键位置处，插入监控代码，实时感知分区内任务切换，根据代码上下文判断触发任务切换事件类型；通过消息通道将此序列传输到仿真器外部显示界面并保存；完成对分区内任务状态-事件序列采集。实现了实时监控机载操作系统的状态和数据变化，实现了将机载操作系统中的分歧和进程状态切换的信息接出来并发送到界面，从而在外部对系统内部进行有效的监控和研究。

2　系统总体框架

任务状态监控工具的设计主要包括了监控器模块的设计、消息通道机制的设计和显示界面模块的设计三部分。系统的总体框架如图 2 所示。

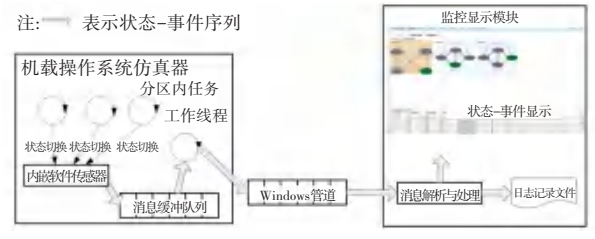

图 2　系统总体框架图

Fig. 2　System overall framework

从图 2 中可以看出，监控器模块是嵌入到 ARINC653 仿真操作系统中的，其通过内嵌软件传感器将捕获到的状态转换信息打包后写入到消息缓冲队列中，然后通过一个工作线程不断地将缓冲队列的数据写入消息通道；消息通道底层封装了 Windows 的通信机制，将监控器模块写入的数据传递到显示界面；显示界面模块接收到消息后，先是对消息进行解析处理，此后将发送到界面显示或者保存到日志文件中。

3　系统的设计

3.1　监控器模块的设计

监控器模块的设计主要包括了 4 部分内容：状态事件的设计、内嵌传感器的设计、消息队列的设计和工作线程的设计。这里对其中每一部分的功能设计可阐释解析如下。

（1）状态事件的设计。本文将分区和进程的每次状态转换当作一次触发事件来处理，并给每个转换进行了事件编号。

分区状态事件主要依据分区状态转换图，即将分区状态的每次切换进行了编号，具体的事件编号如图 3 所示。

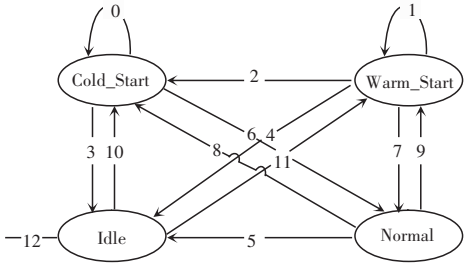

图 3　分区转换事件编号

Fig. 3　Partition conversion event number

进程状态事件主要依据进程状态转换图，将进程状态的每次切换进行了编号，具体的事件编号如图 4 所示。

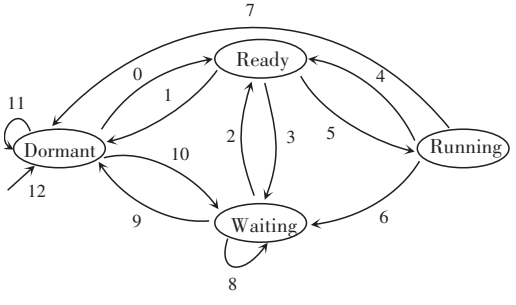

图 4　进程转换事件编号

Fig. 4　Process conversion event number

（2）内嵌传感器的设计。过程中采用了内嵌软件传感器的方法监控软件。该方法采用状态传感器代码记录分区和分区内任务的状态切换，并用事件传感器代码记录触发本次状态切换的事件；通过对机载操作系统仿真器的源代码进行分析，共找到 28 处发生分区状态以及任务状态切换的位置，在状态切换的位置上插入状态传感器代码。内嵌传感器的设计主要包括了捕获分区状态转换接口的设计和捕获进程状态转换接口的设计。

（3）消息队列的设计。由于仿真操作系统的分

区和进程状态切换比较快,而消息通道是一个进程间的通信机制,在传送过程中会消耗大量的时间,这样就导致了监控器发送数据快,消息通道传输速度慢的情况。为了防止监控到的消息被覆盖或者丢失,专门设计了一个消息缓冲队列来延缓这个速度差。同时,又考虑到重复利用缓冲队列的空闲资源,本文使用环形队列来实现消息缓冲队列。消息队列的设计图如图 5 所示。

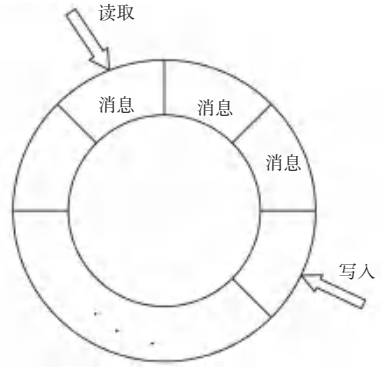

图 5 消息缓冲队列

Fig. 5 Message buffer queue

(4)工作线程的设计。工作线程的主要作用是从消息缓冲队列中读取数据,并将数据按照设计好的格式进行打包,且将打包好的数据写入消息通道中。工作线程的工作流程如图 6 所示。

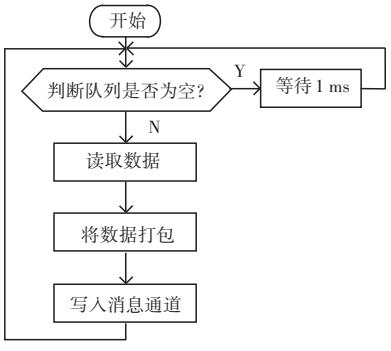

图 6 工作线程流程图

Fig. 6 Work thread flow chart

3.2 消息通道机制的设计

消息通道的底层使用了命名管道的方式进行进程间通信,为了节省系统资源,于是采用复用的方式,一个模块只对应一个命名管道(以模块名命名),模块中各个分区复用一个通道。由于在一个模块中,同一时刻只会有一个分区在运行,就使得在某时刻,命名管道是被某一分区单独使用,如此设计既节约了资源,又不会引起多个分区同时发送数据而引发的冲突。

3.3 显示界面模块的设计

显示界面用来完成监控信息的显示、历史信息的保存和回放等功能,其控件的布局如图 7 所示。

图 7 显示界面布局

Fig. 7 Display interface layout

从图 7 可以看出,显示界面主要分为 3 部分,即:上部分是界面的标题和控制按钮部分;中间部分是显示状态转化部分;下部分是事件显示列表。

4 测试

本系统的测试平台包括 Microsoft Visual Studio2008 和 Microsoft Visual Studio2015,针对是否能够成功监控到机载操作系统仿真器任务执行轨迹来展开研究,并且以状态转换图的形式直观地显示在界面上进行测试。系统测试效果如图 8 所示。

在一个模块中配置一个分区,在该分区中创建 2 个进程,启动分区后,这 2 个进程的状态不断切换,界面成功显示预期的状态-事件序列。

图 8 系统效果图

Fig. 8 System rendering

从测试结果来看,能够成功监控到机载操作系统仿真器内部的状态转化,并且通过界面可以直观地观察到机载操作系统仿真器内部的状态迁移,及时了解到系统内部发生的一系列状态迁移,可以更加方便地通过任务状态轨迹对系统的内部进行监控。

(下转第 117 页)