文章编号：2096-1472(2016)-07-10-04

基于汽车电子实时系统的健康监控模块实现

李自清
（青海民族大学物理与电子信息工程学院，青海 西宁 810007）

摘 要：随着汽车整车安全性、经济性等方面的发展，本文结合汽车电子软件平台化的需求，建立了车控实时系统健康管理方法理论架构和流程体系，实现了面向汽车电子实时系统的健康监控管理模块。本文中的健康监控模块通过AUTOSAR/OSEK系统服务API获取车控实时系统运行时任务、资源、事件、中断等软件状态信息，通过环境和应用系统状态检测器读取运行环境信息和车控应用对象的状态信息，利用相应的hook函数实现信息的监测、收集和发送。

关键词：计算机软件；汽车电子；实时系统；健康管理；软件监控
中图分类号：TP311.15 **文献标识码**：A

Analysis of the Health Monitoring Module Based on the Automotive Electronics Real-time System

LI Ziqing
(*School of Physics and Electronic Information Engineering*,*Qinghai University for Nationalities*,*Xining* 810007,*China*)

Abstract: With the development of automobile safety and economical efficiency,considering the requirement of automotive electronics platforms,this paper constructs a theoretical framework and a processing system of real-time health management,and comes up with the real-time health monitoring and management module.Through system service APIs of AUTOSAR/OSEK,the monitoring module acquires the real-time status information of tasks,resources,events, interrupts,etc..Besides,through the environment and application system status detector,it obtains running environment information and the status information of objects under control.The Hook functions are applied to implement the information monitoring,collecting and sending.

Keywords:computer software;automotive electronics;the real-time system;health management

1 引言(Introduction)

汽车电子技术已成为当今汽车工业创新的主要手段。从信息处理的角度讲，未来汽车将是一个具有交通功能的移动信息终端。汽车的绝大多数功能将通过支持这个信息终端的汽车电子技术实现，软件在汽车价值中的比重越来越大。过去的汽车电子软件子系统相对独立，产生的故障也易于在小范围内隔离和处理。而随着新一代汽车电子的发展，更大规模的互联、更复杂的软件功能为汽车故障处理带来了巨大的挑战。软件健康管理(Software Health Management，SHM)从硬件的PHM发展而来[1]。软件健康管SHM是一个基于实时在线监测的反馈循环过程。常用的软件故障恢复策略可分为冗余策略、恢复点策略、构件修复策略、系统重构策略[2]。文献[3]阐述了一个面向安全关键系统软件和传感器集成健康管理系统，其收集硬件传感器、软件传感器、操作系统等多种来源的信息，使用贝叶斯网络进行融合分析；文献[4]认为安全系统的故障从产生阶段、系统边界现象原因、尺度、日标、意图、能力和持续八个方面分为16个基础分类：开发故障、操作故障、内部故障、外部故障、自然故障、人为故障、硬件故障、软件故障、恶意故障、非恶意故障、故意故障、非故意故障、意外故障、不适当故障、持久故障、瞬时故障。文献[5]采用状态监测器和环境监测器，分别对软件的内部状态信息和外部环境信息进行监测；本文中以面向汽车电子的实时操作系统内核为基础，对健康管理系统主要分为控制流、数据流、实时性这三个方面触，因为任何一方面的违背都会造成系统的故障，因此目前主要从这三个方面并结合全局错误处理机制进行健康监控管理的设计和实现。首先，本文对健康监控管理系统进行总体设计和描述；其次,本文对控制流、数据流、实时性和全局错误处理机制的详细设计和实现进行阐述；最后，对本文中所提到的健康管理系统进行总结。

2 健康管理系统总体架构(General framework of health management system)

健康管理是一个系统性的过程，是提高实时系统可靠性和安全性的重要保障，它通过在故障诊断基础上结合更多的信息进行故障预测、预防和恢复，提高系统可靠性并降低运行维护成本[6]。在汽车电子软件领域已经得到广泛关注和应用的AUTOSAR标准[7]需求规范文档4.0版本中已明确指出了健康管理的重要性，本文拟结合汽车电子软件标准体系

基金项目：青海民族大学2015年度校级理工自然科学项目，项目编号：2015XJZ14.

AUTOSAR/OSEK[8]等规范，研究面向汽车电子的实时系统健康管理；文献[9]的健康监控通过对三个健康监控表的管理来实现：系统健康监控表，模块健康监控表和分区健康监控表。

本文针对汽车电子系统的领域需求，参照AUTOSAR/OSEK规范，提出了一种面向汽车电子实时操作系统的健康监控管理方法[10]。其架构的健康管理系统的总体设计思路如图1所示。

图1 健康管理系统总体架构

Fig.1 General framework of health management system

该健康管理系统基于汽车电子领域主流处理器平台，面向汽车电子的实时操作系统内核进行扩展，主要包括健康监控、故障处理、自主容错等部分。本文重点对健康监控模块进行了设计实现，通过对实时系统的数据流、控制流、实时性的监控，并结合全局错误处理机制实现有效管理，实时监管汽车电子软件系统的健康状态，降低系统运行的错误风险。

3 健康监控管理设计思路(Design of health monitoring management)

面向汽车电子的实时操作系统的健康管理模块主要从控制流、数据流、实时性，以及全局错误处理模块进行阐述，健康管理模块的功能逻辑设计如图2所示。

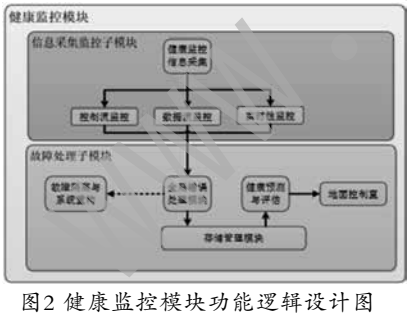

图2 健康监控模块功能逻辑设计图

Fig.2 Functional logic design of health monitoring module

3.1 数据流监控

对于一个函数来说，传入参数的正确与否直接影响着该函数的执行结果，因此在函数执行前对传入参数的检测是必不可少的。由于每个函数需要检测的内容不同，同时每个函数的参数各异，本文通过分析，确定将函数检测部分放在函数执行代码之前，由用户来进行检测。通过提供一种全局的错误处理机制，用户检测到错误后，向ErrorManager报告错误类型，由ErrorManager进行统一处理。

函数参数的检测目前没有和函数的执行分离开来，主要是考虑到参数的作用域问题。除了参数检测之外，通过提供对全局关键数据的实时检测，提供统一的函数入口，具体检测代码由用户来实现。这个功能的好处就是用户可以在任何时间任何地方检测全局关键数据，不需要记住相关函数，只需要向全局关键数据检测函数中传入变量即可。

在全局错误处理机制和全局关键数据的检测外，同时提供关键数据的存取服务。可以实现在程序运行的过程中动态的保存某一个变量的值，随时访问保存的数据。保存时用户需要按照固定的结构体进行封转(包括：函数名、变量名、变量类型、变量值)，读取时只需提供变量名和函数名(之所以要求同时提供变量名和函数名是为了防止变量名的重复)。数据流监控相关API描述如表1所示。

表1 数据流监控API描述

Tab.1 API description of data flow monitoring

API名称	API描述
Viod ReportError(uint16 VarNameID)	监测到错误后，将错误报告给错误管理器
Void HandleSystemCountError (void)	由用户进行定义，对检测到的数据流错误进行处理
Void CheckSystemoCount(void)	由用户进行定义，检测数据流错误
Void CheckGlobalVar(uint16 VarNameID)	全局关键变量检测函数

3.2 控制流监控

从实时操作系统运行的角度来观察，任何任务的实质就是若干函数的顺序执行。保证函数按顺序执行是完成任务的首要工作，因此控制流健康监控的主要作用就是防止不合理任务调度，保证任务按序执行。

本文将任务(包括多干函数)中函数的执行看作是一个控制流，在运行之前将每一个任务中的函数注册到一个控制流(ControlFlow)中，该功能也可以通过配置工具来实现(因为任务中函数的执行顺序是事前确定的)。在一个函数执行之前，调用Hook函数检测该函数的前驱是否已执行，若已执行就执行该函数，否则报错；执行完该函数之后调用Hook函数将其标记为已执行。控制流监控相关API描述如表2所示。

表2 控制流监控 API 描述

Tab.2 Control flow monitoring API description

API名称	API描述
Uint16 CreateControlFlow(void)	根据用户配置的函数执行顺序创建控制流
Uint16 ControlFlowPreHook(uint16 ControlFlowID,uint16 FuncID)	位于函数调用之前判断是否需要进行前驱是否已执行的检测，如果需要则调用相应处理函数
Void ControlFlowPostHook(uint16 contronlFlowID,uint16 FuncID)	位于函数调用之后，对相应的控制流中对应的func执行标记为已执行，最后检测当前控制流是否都已经执行过
Uint16 ClearControlFlow(uint16 ContronlFlowID)	内部函数，清除整个控制流中的flag标记，表示整个任务已按规定顺序正确执行
Void Handle_Control_Flow(uint16 ControlFlowID)	用来处理控制流错误

3.3 实时性监控

从实时操作系统的高安全性角度考虑，仅仅执行顺序正确以及得到正确的运算结果还不够，仍需保证每一个任务必须在规定的时间内(deadline之前)得到执行完毕，这样才能保证整个系统的安全性。对于实时性的监控，可利用该实时操作系统实现的时间保护机制完成。

在操作系统配置工具中，针对每一个Task对应生成一个Alarm，alarm名称跟任务名一一对应。同时配置好Alarm的回调函数(callback)，该回调函数的参数中设置了TaskID，以及对应的错误等级，发生错误时由回调函数将发生错误的TaskID，以及错误等级报告给错误管理器(ErrorManager)。实时性监控的API描述如表3所示。

表3 实时性监控API描述

Tab.3 API description of real time monitoring

API名称	API描述
RealtimeMonitorPreHook(AlarmID，MaxLimitTime)	放在函数执行前，设置Alarm
RealtimeMonitorPostHook(AlarmID)	放在函数执行后，若函数执行时间没有超时，就会调用该函数取消对应的Alarm

3.4 全局错误管理机制

前面分别对控制流监控、数据流监控和实时性监控三方面做了阐述，本小节将具体阐述全局错误管理机制。错误类型分为系统级错误和用户级(控制流、数据流、实时性)错误，错误码为四位十进制正整数组成：第一位是错误类型，后面三位为具体的错误代码。比如1002代表第一类错误2号错误。目前规定1代表系统级错误、2代表控制流错误、3代表数据流错误、4代表实时性错误。在启动健康监控之前，需要定义好各种错误的解决方案，当在系统运行过程中错误管理中心接收到错误报告时，根据相应的错误类型调用不同的解决方案。

全局错误管理的好处在于将以前分散的错误处理集中起来，提高错误处理的效率，同时便于管理。全局错误管理机制的示例API如表4所示。例如当需要更新某种错误类型的解决方案时，只需要更新全局错误管理中对应错误的解决方案。

表4 全局错误管理API描述

Tab.4 Global error management API description

API名称	API描述
ReportToErrorManager(ErrorCode)	检测到错误后，将错误报告给错误管理器
HandleSystemError(ErrorCode)	处理系统错误函数
HandleControlFlowError(ErrorCode)	处理控制流错误函数
HandleDataFlowError(ErrorCode)	处理数据流错误函数
HandleRealTimeError(ErrorCode)	处理实时性错误函数

4 健康监控管理功能实现及应用分析(Realization and application analysis of health monitoring and management function)

健康监控管理模块的功能实现依据上述的健康监控管理模块的设计思路。在实现中采用了C语言实现，在本文利用伪代码描述了数据流监控、控制流监控、实时性监控、全局错误管理机制的实现框架。

4.1 数据流监控

以全局关键数据为例，在数据流监控实现中，为需要监控的全局变量进性统一编号。本文中的全局关键数据编号是在操作系统配置时，将需要检测的全局变量全部定义成宏。在需125要检测指定的全局关键数据时，需要将关键数据的ID作为参数传入函数CheckGlobalVar，该函数根据不同的ID对关键数据监测或调用回调函数进行检测和检测结果反馈。CheckGlobalVar函数的实现如下所示：

```
Void checkGlobalVar(Unint16 VarNameID)
{
    Switch(VarNameID)
{
    Case X：
……；//这里可以是检测代码，也可以是函数调用
                    break；
    Caes Y：
……；
break；
……；
default：
……；
break；
}
}
```

4.2 控制流监控

控制流的监控主要监控函数的执行顺序，为此需要为每一个函数执行节点分配如表5所示结构，记录整个函数的执行序列。在执当前结点时，首先检测整个执行流中该函数的前驱节点(第一个函数节点除外)是否已经成功执行，若成功执行继续执行后续节点，否则向系统发出执行流紊乱警告，整个执行流成功运行完成后，需要对执行流中的函数节点进行释放和复位。

表5 控制流数据结构描述

Tab.5 Control flow data structure description

Struct ControlFlow {
Unsigned char FunID；
Unsigned char flag；
}
	控制流数据结构
	FuncID：函数ID，通过宏定义实现
	Flag：是否执行标志位(1表示执行，0表示未执行，默认为0)

4.3 实时性监控

实时性监控主要针对任务或者函数的执行时间的监控，该功能的实现需要借助实时系统的Alarm机制，因实时系统的Alarm机制是建立在硬件定时器之上，保证了硬实时的要求。为了监督任务或函数的执行时间，需在任务或函数的执行开始和结束位置分别调用相应的Hook函数，如下所示，最终通过计算前后时间的差值，检测任务或函数的执行是否在指定时间范围内正常结束。

```
//Task_X函数执行前，设置对应函数的Alarm
RealtimeMonitorPreHook(AlarmId,MaxLimitTime);
Task_X
{
    Func_1();
……;
Func_X;
}
//若函数在规定的时间执行完，则取消对应的Alarm;
RealtimeMonitorPostHook(AlarmID);
```

4.4 全局错误管理机制

全局错误管理机制将将数据流错误、控制流错误、超时错误、系统错误进行分类，针对不同的错误类型和错误代码采用不同的错误处理策略，能够尽早发现、并提交错误报告给实时系统，降低后续因脏数据、执行流紊乱等错误带来系统运行风险的进一步放大。以下以伪代码方式描述了全局错误管理机制的策略选择过程。针对具体的错误的应对策略可以根据需求进行完善。

```
ReportToErrorManager(ErrorPackage EP)
{
    Switch(EP.ErrorType)
{
    case SYSTEM_ERROR:
            HandleSystemError();
            break;
caseCONTROLFLOW_ERROR;
HandleControlFlowError(EP.ControlFlowId,);
break;
case DATAFLOW_ERROR();
HandleDataFlowError(EP.VarNameId);
break;
case REALTIME_ERROR();
HandleRealtimeFlowError(EP.FUNCID);
break;
default:
未定义的错误;
break;
}
```

5 结论(Conclusion)

本文提出了一种面向汽车电子实时系统健康管理理论架构和流程体系，阐述了基于数据流、控制流、实时性和全局错误处理的健康监控管理模块功能方案，实现了对汽车电子实控系统的健康监管，在一定程度上降低了实时系统的错误运行风险。后续的健康管理相关算法和应用研究，如基于人工智能的故障预测推理、基于服务的汽车CPS应用等，都可以在此基础上进行有效的开展。

参考文献(References)

[1] FaezeEshragh,Mehdi Kargahi.Analytical architecture-based performability evaluation of real-time software systems[J].Journal of Systems and Software,2013,86(1):233-246.

[2] 李宁波,等.嵌入式RTOS健康监控技术研究[J].计算机工程,2009,35(3):260-262.

[3] 陈光,等.服务软件系统的健康管理综述[J].计算机科学与探索,2013,7(7):577-591.

[4] J Schumann,OJ Mengshoel,T Mbaya.Integrated Software and Sensor Health Management for Small Spacecraft[C].Fourth IEEE International Conference on Space Mission Challenges for Information Technology(SMC-IT 2011),2011:77-84.

[5] A Avizienis,et al.Basic Concepts and Taxonomy of Dependable and Secure Computing[J].IEEE Transactions on Dependable and Secure Computing,2004,1(1):23.

[6] Kai Wang,et al.A Prognostics and Health Management Based Method for Refurbishment Decision Making for Electromechanical Systems[J].IFAC-Papers OnLine,2015,48(3):454-459.

[7] J Park,G Yoo,E Lee.A Reconfiguration Framework for Self-Healing Software[C].International Conference on Convergence and Hybrid Information Technology,2008:83-91.

[8] AUTOSAR.AutomotiveopenSystemARchitecture[DB/OL].http://www.autosar.org,2014.

[9] OSEK/VDX.OSEK/VDX Operating System Specification Version 2.2.2[S].http://www.osek-vdx.org,2014.

[10] Michael Short,Michael J.Pont.Assessment of high-integrity embedded automotive control systems using hardware in the loop simulation[J].Journal of Systems and Software,2008,81(7):1163-1183.

作者简介：

李自清(1975-)，男，本科，讲师.研究领域：计算机应用技术.