第 35 卷　第 21 期　　计　算　机　工　程　　2009 年 11 月
Vol.35　No.21　　Computer Engineering　　November 2009
·开发研究与设计技术·　文章编号：1000—3428(2009)21—0277—03　文献标识码：A　中图分类号：TP393

基于广义表结构的 OIL 配置器设计

李银国[1]，汤卓群[1]，蒋建春[1]，盛一伦[2]
(1. 重庆邮电大学汽车电子与嵌入式系统研究所，重庆 400065；2. 重庆长安新能源股份有限公司，重庆 401120)

摘　要：分析应用于汽车电子实时操作系统的 OSEK OIL 规范，以及 OIL 配置器的工作原理和特点，结合嵌入式实时操作系统 AutoOSEK 的特点，设计基于广义表结构的 OIL 配置器。给出 AutoOSEK 嵌入式实时操作系统配置器的设计方法和说明。实验结果证明，该设计能减少类似 OIL 配置系统的重复开发，提高符合 OSEK 标准的不同嵌入式操作系统的开发速度。

关键词：嵌入式操作系统；系统配置；广义链表

OIL Configurator Design Based on Generalized Table Structure

LI Yin-guo[1], TANG Zhuo-qun[1], JIANG Jian-chun[1], SHENG Yi-lun[2]
(1. Institute of Automotive Electronic and Embedded System, Chongqing University of Posts and Telecommunications, Chongqing 400065;
2. Chongqing Changan New Energy Co. Ltd., Chongqing 401120)

【Abstract】This paper analyses OSEK specification applied in vehicle electronic real-time operating system and the working principle and characteristics of OIL configurator. It designs OIL configurator based on generalized table structure combining with the characteristics of AutoOSEK embedded real-time operating system, and introduces design technique and illustration of AutoOSEK embedded real-time operating system for configurator. Experimental result proves that the design can reduce repeatedly exploiting like OIL configuration system, enhance exploiting speed for different embedded operating system according OSEK specification.

【Key words】embedded operating system; system configuration; generalized linked table

1　概述

OSEK 标准是德国汽车工业界于 1993 年联合推出的汽车电子开放式系统及接口的工业标准，现已成为汽车电子行业标准。该标准分为操作系统、通信、网络管理以及实现语言 4 个方面，其中，OSEK 实现语言(OSEK Implementation Language，OIL)规定用户应用程序涉及到操作系统需要配置的方面，统一用抽象的 OIL 语言将整个系统用一组 OIL 对象来描述。一个 CPU 对应一组描述，CPU 是此类 OIL 对象的容器。OIL 明确为每个 OIL 对象定义所有标准属性。每个 OSEK 应用定义附加的特殊执行属性和引用，并限制每个属性的取值范围[1]。

OIL 规范中配置分为 CPU(处理器)、OS(操作系统)、Appmode(应用模式)、Isr(中断服务)、Resource(资源)、Task(任务)、Counter(记数器)、Ipdu(交互层协议数据单元)、Event(事件)、Alarm(报警)、Com(通信子系统)、Message(消息)、NM(网络管理)等对象。每个对象有多个属性，不同枚举属性参数有不同子属性。对象属性相互间关联，属性的参数定义了对象间的间接连接情况(如设置报警激活任务)。繁复的属性选项完全地描述了系统的配置情况。

在 OSEK 标准下，一个应用程序由 3 个部分代码同时编译而成，配置代码作为独立部分存在，它是通过 OIL 语言描述完成后再编译为对应系统配置的 C 代码。

因此，在 OSEK 标准下同样的 OIL 应用配置文件，通过相同操作系统配套的配置器就能得到适合该操作系统的配置文件，加大了系统应用软件的通用性和实际嵌入式操作系统的开发。配置器屏蔽了配置文件和操作系统 API 调用的细节部分，用户无须了解不同操作系统的区别，只须关注应用程序配置本身。如自主研发、符合 OSEK 标准的实时嵌入式操作系统 AutoOSEK[2]在配置文件中涉及到系统任务管理时任务优先级和名称的关系匹配问题，而配置器会对用户屏蔽此类问题。配置应用原理如图 1 所示。

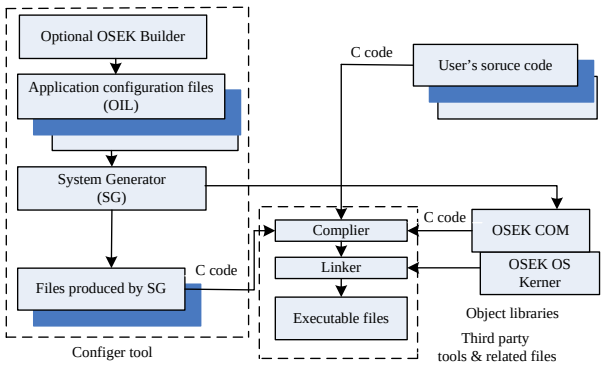

图 1　配置应用原理

用户在配置系统时实际是一个不断修改 OIL 对象、修改其繁复的对象属性参数，配置系统不断自动校验配置有效性的过程。所有 OIL 语法校验，在具体系统配置使用的规范以

基金项目：国家“863”计划基金资助项目(2006AA11A1C1)
作者简介：李银国(1955－)，男，教授、博士生导师，主研方向：智能控制理论及应用，模式识别技术，汽车电子控制系统；汤卓群，硕士研究生；蒋建春，讲师、博士研究生；盛一伦，工程师
收稿日期：2009-05-09　**E-mail：**tangzhuoqun@sohu.com

及其他配置时的相关 OIL 和具体操作系统细节部分，用户无须了解，由配置系统处理。因此，要求配置系统对 OIL 中各对象繁复的属性选项的相互关联关系能完全描述。

本文根据 OIL 实现语言规范特性，结合 OIL 配置系统的工作原理，提出将广义表结构应用于系统设计的思想，并将该设计应用于嵌入式实时操作系统 AutoOSEK 的开发。该设计方法能够满足配置中对各个 OIL 配置对象的各种属性间关联关系的完全描述，实现配置过程中对用户配置的有效性进行及时校验，很好地帮助操作系统用户进行系统配置。

2 基于广义表的设计

广义表是一种多层次结构，兼容线性表、数组、树和有向图等各种常用的数据结构，这些特点与 OIL 语言特点相对应。由于广义表中的元素可以具备不同的结构，因此通常采取链式的存储结构来存储广义表。

OIL 用一组 OIL 对象组合来描述 OSEK 应用，一个 CPU 好比一个容器对应一个 OIL 描述，并包含这些 OIL 对象。在 OIL 规范中明确定义每个对象的标准属性，以及在不同属性参数下子属性的参数设定要求。如果以 CPU 作为表头展开，那么配置描述可以看作一张描述各 OIL 对象属性间相互关系的实现语言配置表。

CPU 节点下包含 OS, Appmode, Isr, Resource, Task, Counter, Ipdu, Event, Alarm, Com, Message 和 NM 等子节点，每个子节点作为一种类型对象表的头节点。在标准允许有多个类似对象时，多个类似对象以链表结构连接。

每个节点做为一个 OIL 对象实例，包含 OIL 明确定义的各种属性。各对象属性间有相互联系，如果用广义表结构描述相互间关系并建立对应的链表，那么就能对整个配置情况作一个完整描述。当读取解析结构类似文本的 OIL 文件时，根据标准要求填入配置器设定的通用匹配值，就能生成一张完全描述某个配置情况的广义表。配置过程，如在该广义表上添加节点、删除节点或者修改节点等操作都能较快捷地进行。在配置时，通过该广义表校验对各属性间相互联系是否符合 OIL 规范，从而很好地辅助配置人员进行配置并修正配置过程中产生的语法逻辑错误。

图 2 描述了整体结构，用 Task 对象举例，对其中的属性和其他对象中属性的关系给出基于广义表的结构描述，其语法结构参考文献[3-4]。

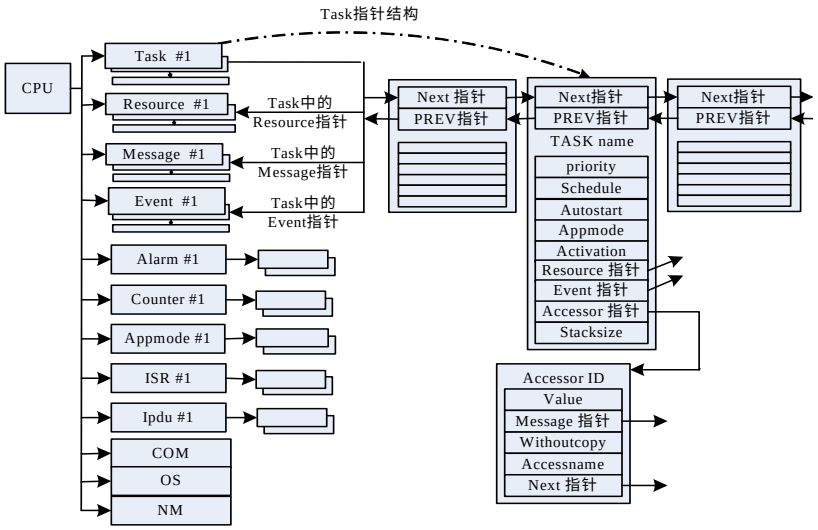

图 2 设计结构示意图

3 OIL 配置器的设计方法

3.1 OIL 配置器的需求分析

符合 OSEK 标准的嵌入式实时操作系统配置工具需要具备如下功能：

(1)自动解析 OIL 文件；

(2)自动校验配置是否符合 OIL 语法规范；

(3)自动辅助配置者进行有效的配置，屏蔽规范中的细节部分，如资源优先级的设定；

(4)自动辅助配置者修正配置过程中的语法错误；

(5)自动生成 OIL 配置文件；

(6)自动根据 OIL 配置文件翻译为与操作系统对应的 C 配置文件。

3.2 功能模块设计

整个配置系统按照配置流程划分为 6 个模块，如图 3 所示。

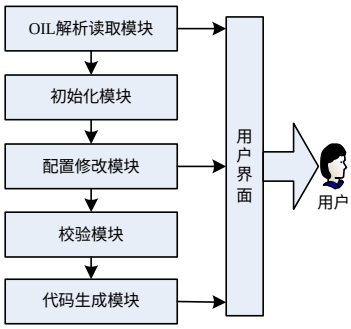

图 3 配置系统功能模块设计

在图 3 中，OIL 解析读取模块根据对象和属性关键字解析 OIL 文件，并读取相应数据交给初始化模块处理；初始化模块根据默认框架初始化新配置的各个 OIL 对象(如 OS, COUNTER)，或者对象指针队列(如 TASK, EVENT, ALARM)，或配置修改过程中新添加对象的默认属性项；配置修改模块根据用户的配置修改操作对存储该配置数据的广义链表进行相应的修改，如插入某个对象实例指针或修改某个对象的属性值或属性指针等；校验模块负责统一完成该配置系统的校验工作，自动校验用户当前配置是否符合 OIL 语法规范，辅助配置者有效地进行配置并屏蔽规范中的细节部分，同时辅助用户修正配置过程中的语法错误；代码生成模块根据广义表各项接合配套的嵌入式实时操作系统生成配置文件。

3.3 代码重用设计

该系统的模块设计考虑到符合 OSEK 标准的不同配置系统对于 OIL 配置和校验标准都是一样的，从而将代码生成模块独立出来作为一个模块。针对不同操作系统除了代码生成模块对应不同的嵌入式实时操作系统配置器，其余模块可以重复使用。

如应用于 AutoOSEK，配置文件之一的 cfg.h 中任务定义代码必须同任务的优先级对应，并对应 cfg.c 中的任务入口指针数组和属性数组里的排列顺序。代码生成模块通过遍历广义链表中所有任务节点优先级的属性值，得到一个优先级的排列顺序，根据该顺序生成对应任务的定义代码和其他需要考虑该顺序的代码，而非按照任务在链表中的顺

序生成。如应用于其他嵌入式操作系统配置器，如OSEK-Turbo,依次读取任务节点在 cfg.h 中生成相应的任务定义代码，并按照同样的顺序在 cfg.c 中生成记录任务优先级的数组，OsTask-BasePrio[]。如果是生成 OIL 文件，那么无论是 AutoOSEK 还是 OSEKTurbo 代码生成子模块都是一致的。

该设计方法使不同嵌入式实时操作系统配套的配置器开发时只要替换代码生成模块即可。这样提高了软件开发的重用性，减少了代码重复开发的浪费，同时由于配置器的快速开发而加快了嵌入式实时操作系统的开发。

3.4 数据结构设计

一次配置就是一个实例化的 OIL_object 对象。每添加一个对象就实例化一个该类对象，并添加到相应的节点上，赋值对应的指针。下文是 OIL_object 和 Task 对象结构定义的代码，Task 对象实力间采用双向链表连接，便于快速查找、添加、删除和修改。

```
Class OIL_object
{
public：
void    * P_H_TASK;
void    * P_ H _RESOURCE;
void    * P_ H_MESSAGE;
void    * P_ H_EVENT;
void    * P_ H_ALARM;
…
void    * P_NM;
void    * P_IPDU;
};
Class TASK
{
public：
String          NAME;
Integer         PRIORITY;
BOOL            SCHEDULE;
…
void    * P _T_ACCESSOR;
//指向新建立的 ACCESSOR_T_Point 对象
void            * P_EVENT ;
integer         STACKSIZE;
TASK   * NEXT;
TASK   * PREV;
};
TASK 中的 ACCESSOR：
Class ACCESSOR_T_point
{
public：
BOOL            ACCESSORVALUE;
//为 0 时是 SEND，1 时为 RECEIVE
void            * P_MESSAGE;
BOOL            WITHOUTCOPY;
String          ACCESSNAME;
ACCESSOR_T_point                *NEXT;
};
```

针对 OIL 语言的对象配置描述，模块化各个对象使之成为独立的模块，因此，当某类对象相关的规范发生变化时，能方便程序的修改。

3.5 配置系统校验设计

配置系统的校验统一由校验模块完成。其基本思路是：根据配置要求设置相应的校验函数，当用户配置时激活对应部分的校验函数，包括输入类型的基本校验函数和根据完整的广义链表动态地进行对象间相联属性校验的校验函数。例如，当用户在配置 Task 对象时，填入的 Resource 属性没有给出资源定义，此时，系统会给用户 2 种选择：添加未定义资源或在已定义的资源中做出选择替换错误项。整个配置过程软件能防止每一个错误的发生，保证得到符合 OIL 规范的 OIL 配置文件，从而得到正确的 C 配置文件。用户只须了解应用程序要用到的配置，无须了解具体的 OIL 语法，至于操作系统如何调用配置文件中的定义以及 OSEK 中的应用规定细节(如资源优先级定义必须要符合天花板协议)由系统自动处理和校验。

该模块分为 2 个部分：

(1)针对 OIL 语法规范的各校验函数组成；

(2)当操作系统对配置有特别限定时(如嵌入式操作系统要求配置 C 文件中任务名字和优先级对应),针对各个操作系统自身特殊设计结构规定的该部分校验函数组成。

3.6 用户界面设计

本文对于配置系统的界面设计,更多地从用户角度出发,结合用户的使用习惯和配置中易犯的错误，给出人性化、便捷的用户操作界面。

整个用户界面主要由 2 个树型控件和 1 个列表组成。在添加针对配置的对象情况查看时，用户通过对象树能清晰地观察到整体配置情况。属性树型控件能针对一个对象的属性查看，观察该对象的具体某个属性是否有子属性，这样就能更容易地观察使整个对象属性情况。图 4 显示了界面从左到右呈包含关系，层次分明，配置过程思路清晰。其中，右边的值列表是根据属性树的具体激活选项(是整个对象或是有子属性的某个属性)判断显示一个对象属性的具体值或某个子属性的具体值。

图 4 配置系统用户界面

（下转第 282 页）