控制理论与应用

基于 Xenomai 的 Linux 嵌入式系统在轨道交通客运车辆电气控制平台中的应用

冯　建，方　鹏，王　伟
（株洲中车时代电气股份有限公司，湖南 株洲　412001）

摘　要：轨道交通客运车辆的电气控制平台是微机化电气屏柜中的智能化控制单元，通过核心控制模块中的嵌入式操作系统完成电气逻辑控制和多种智能监测保护功能。文章通过对 Linux 与 VxWorks 这两种主流嵌入式系统的实时性、功能性、性价比和开发资源等方面进行对比分析，设计了一种基于 Xenomai 微内核移植的实时 Linux 嵌入式操作系统。该系统既具备 Linux 的多任务处理能力、丰富的网络制式通信、成熟可靠的图形处理与电能质量监测分析算法，又可实现微秒级的系统任务调度，对轨道交通客运车辆微机化电气屏柜的工程化应用具有重要意义。

关键词：Xenomai；Linux；客车电气控制平台；实时性；逻辑控制；智能监测
中图分类号：TP316.85；U264.91　　**文献标识码：**A　　**文章编号：**2096–5427(2018)02–0011–04
doi:10.13889/j.issn.2096–5427.2018.02.003

A Linux Embedded System Based on Xenomai for Coach Electric Control Platform in Rail Transit

FENG Jian, FANG Peng, WANG Wei
(Zhuzhou CRRC Times Electric Co., Ltd., Zhuzhou, Hunan 412001, China)

Abstract: Coach electric control platform is an intelligent control platform of the computerized electrical cabinet, which has electrical logic control and various intelligent monitoring and protection functions realized by the embedded operating system of the core control module. This paper compared the performance of two mainstream embedded systems, Linux and VxWorks, in terms of real-time capability, functionality, cost-effectiveness and development resources cost, and presented a Linux real-time embedded system based on Xenomai plug-ins. The proposed system not only has the multi-tasking capabilities, abundant network standard communication, mature and reliable graphics processing, and power quality monitoring and analysis algorithms of Linux, but also achieves microsecond-level system task scheduling, which has significant effects on computerizing electrical cabinet of passenger vehicles in rail transit.

Keywords: Xenomai; Linux; coach electric control platform; real-time; logic control; intelligent monitoring

0　引言

随着计算机技术和轨道交通行业的发展，铁路客运车辆电气控制系统正从传统的硬线控制向网络微机化控制发展。国内外已有多家相关企业和科研单位研制开发出多种针对内燃机车、电力机车、城轨列车的逻辑控制单元并成功应用于不同类型的车辆上，大大提高了车辆的检修效率和准确度[1–5]。轨道交通客运车辆的电气控制平台是一种智能化控制平台，其以相关车辆电气控制屏柜为应用对象，实现车辆电气控制、线路综合保护、器件状态监测、电能质量分析及设备执行与维护诊断功能的智能化应用；同时借助基于大数据与云计算服务的列车信息化技术，为用户提供运营、管理及维护保障等综合性系统解决方案。平台主要包含 3 种控制单元，即车辆逻辑控制单元、车厢设备执行单元和车厢综合继电保护单元。其采用模块化功能设计，根据各自的应用场

收稿日期：2017–12–07
作者简介：冯建（1990–），男，工程师，主要从事车辆电气控制方面的研究和设计。

景集成不同数量和规格的电源模块、输入输出模块、通信模块和核心控制模块，以实现功能多样化定制。其中核心控制模块主要负责多种控制算法和数据处理的具体实现，也是本文研究的对象。

1 核心控制模块架构

核心控制模板负责系统的主要控制功能，是数据的汇总处理与分发中心，拥有最高的控制权限，主要实现逻辑控制、线路保护、回路监测、器件状态监测、寿命预测、电能质量分析算法、通信管理、日志记录以及维护等功能。模块采用完全相同且互为冗余的 A、B 两系处理数据，每系主要由带锁步模式运行的 TMS570 型双 CPU 处理器、集成 ARM 与 FPGA 架构的 Zynq-7000 型全可编程片上系统（ALL Programmable SOC）、嵌入式多媒体卡（embedded multi media card , eMMC）和以太网交换芯片等外围器件构成，每系控制模块的架构如图 1 所示。

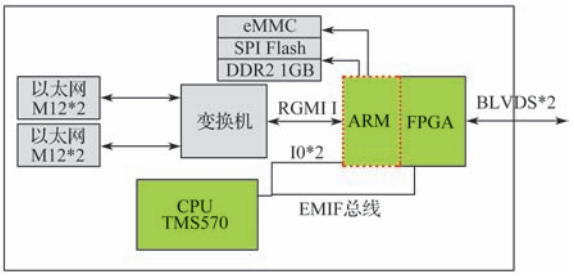

图 1 核心控制模块架构

Fig. 1 Architecture diagram of the core control module

TMS570 处理器运行工业控制软件 CoDeSys Runtime 无操作系统程序，从而确保逻辑控制功能的实时性，以满足车辆的安全功能需求。CoDeSys 支持所有符合标准 IEC61131-3 *Programmable controllers - Part 3*: *Programming languages* 的编程语言。TMS570 根据输入的 DI 信号数据，按照预定逻辑控制要求，在设定的时间内实现逻辑常开、常闭、与、或、非、延时、自锁和互锁等功能，从而完成逻辑算法计算，并将计算结果通过通信总线发送给 Zynq-7000 型 SOC。

SOC 内部集成了 ARM 处理系统、FPGA 可编程逻辑系统和其他丰富的外设资源，结构紧凑、集成度高、功能强，能帮助设计者使用工业标准的工具在单个平台上实现高性能和低成本的应用，因而非常适合于客车电气控制平台使用。其主要完成以下功能：

（1）数据传输和校验——通过总线实时与 DI，DO 板进行数据传输和校验；

（2）数据及故障日志存储——实时记录列车的输入、输出通道数据、运行状态数据和故障记录，并可通过维护端口下载存储数据信息；

（3）高低压交直流供电回路的电能质量分析——采用优化的继电保护算法技术，以实现配电控制、线路保护、电能质量分析等功能；

（4）图像智能监测——结合屏柜内的红外影像仪对柜内的器件进行状态监测，实现对柜内器件功能状态的实时监测，当器件发生形变、温度异常或者功能状态异常时，可进行报警提示，严重时可直接对相关回路进行断开保护。

基于以上核心控制模块的功能需求分析，需要选择一个实时性能良好、算法功能强大、支持多任务进程、支持多种网络制式通信、图像处理能力强、有丰富的设备驱动资源和兼容第三方功能组件库的嵌入式操作系统。

2 嵌入式操作系统选择

嵌入式软件控制系统具有集成度高、功耗低、可靠性高以及可定制化开发等优点，被广泛应用于信息家电、工业控制、电信设备、轨道交通等领域。VxWorks 和 Linux 系统是当前被广泛使用的两种嵌入式系统[6–8]。

2.1 VxWorks 系统和 Linux 系统比较

VxWorks 嵌入式操作系统是一款付费实时操作系统，具有高性能内核、良好的中断管理机制、内存管理机制和任务调度管理等优点，可确保系统任务的实效性和可靠性，被广泛应用于轨道交通、航空、军工等高端领域。

Linux 嵌入式系统是一款高度模块化、可裁减的类 UNIX 操作系统，支持多任务、多线程，且内核任务管理与调度性能优异，程序设计师可在 GNU 公共许可权限下免费获得并进行定制化开发。Linux 属于开源项目，网络开发资源丰富，且各硬件厂商一般都提供 Linux 驱动程序以方便用户进行系统开发。另外，市场上很多第三方公司开发了很多算法先进的功能组件，包括图像视频处理、网络通讯管理、综合继电保护等功能，便于为嵌入式系统增添新功能。

综上可知，VxWorks 实时性好、可靠性高，因此被广泛应用于高端领域，但 VxWorks 源代码作为商业秘密而不公开，图像处理网络资源和第三方设备驱动种类和数量少；Linux 作为一个高度模块化的开源项目，能提供的设备驱动种类和数量远超过 VxWorks 的，也有成熟的优化电力继电保护和图像处理算法，便于后续软件开发，且 Linux 系统开源免费使用，开发成本低。因此针对核心控制电路的功能需要，本设计选择使用

Linux 操作系统。

尽管 Linux 系统实时性处于不断改进中，最新版本内核状态下可以达到百微秒级，但因系统需要实现高阶继电保护算法和屏柜智能图像监测等对实时性有较高要求的功能，所以必须改善 Linux 的实时性以达到更高精度。

2.2　Linux 实时化技术

当前针对 Linux 系统实时性弱的不足，主流改进方案有修改内核和移植微内核两种。

（1）修改内核

修改 Linux 内核的任务调度方式、中断管理、数据结构等部分。这相当于重建一个 Linux 系统，修改后的系统实时性较好，但是工作量大且繁琐，可能会降低系统的可靠性。典型代表有 Kurt-Linux。

（2）移植微内核

对 Linux 内核移植一个同时运行的实时微内核以专门响应和处理实时任务，而一般的非实时任务则交给标准的 Linux 内核来处理。其以一种简便的方法改善了 Linux 系统的实时性，并支持原应用程序在新实时系统上运行，具体代表有 Xenomai 和 RTLinux。Xenomai 提供了精确的定时中断来响应调度实时任务并具备良好的实时管道和共享内存通信机制，其在用户层中实现大部分实时模块，只有和硬件相关以及强实时性的模块才在内核层中实现；而 RTLinux 则在内核层开发所有的实时模块来改进系统实时性。

移植后的 Xenomai/Linux 操作系统既有 Linux 强大的多任务管理能力又具有 Xenomai 良好的实时性。基于以上优点，本系统对 Linux 系统进行 Xenomai 实时性改造。

3　Xenomai 简介

Xenomai 是一个完全遵守 GNU/Linux 自由软件协议的项目，其架构如图 2 所示，Xenomai 实时内核运行于系统自适应域环境（adaptive domain environment for operation system，adeos）的中断管道架构之上。抽象实时操作系统则运行在硬件抽象层 (hardware abstraction layer，HAL) 之上，并具有多种实时程序运行需要的应用程序编程接口（application programming interface，API）[9-11]。

Adeos 的设计目标是为了实现在单个硬件设备之上运行多个操作系统。它轮流创建的实时域可保证抽象实时操作系统既能与其他域共享相同的硬件底层，同时又可优先接收中断管道。运行于 Adeos 之上的每个域都有一个静态的优先级，中断管道会按照域的优先级依次传递事件任务，从而实现高低优先级的区别，以提高实时性。

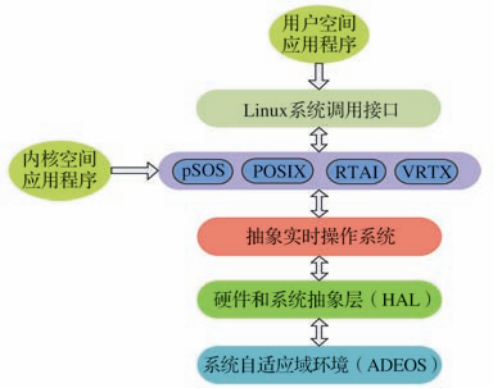

图 2　Xenomai 架构

Fig. 2　Architecture of Xenomai

中断管道传递 Adeos 域之间的中断可通过改变中断优先级机制实现 Linux 的实时性扩展，改造后的 Xenomai / Linux 实时性系统架构如图 3 所示。Xenomai 与 Linux 内核作为 Adeos 中 2 个独立的域，当有中断任务事件时，因 Xenomai 域的优先级高于 Linux 内核域，所以 Adeos 先将中断交给 Xenomai 处理；当 Xenomai 域不需要处理中断任务时，普通任务则由 Linux 内核处理，从而保证 Xenomai 的调度和中断的实时性。

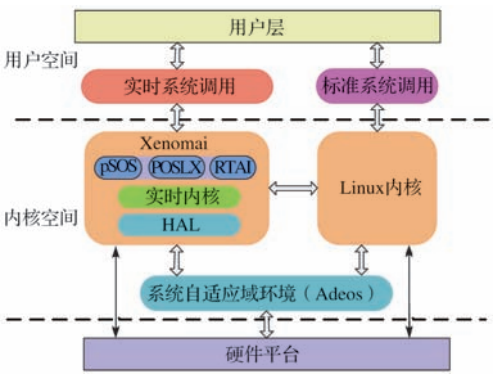

图 3　Xenomai/Linux 系统架构

Fig. 3　Architecture of the Xenomai/Linux system

4　实时 Xenomai/Linux 系统构建

项目的目标是在 Zynq-7000 型 SOC 上搭建基于 Xenomai 的 Linux 嵌入式系统，让系统具有优异实时响应能力，其软件系统结构如图 4 所示，包含 U-boot 启动文件、Xenomai/Linux 内核镜像、rootfs 根文件系统和用户应用程序。

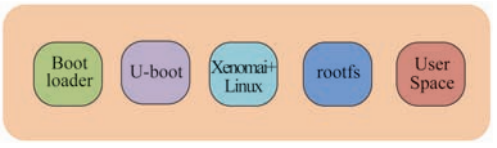

图 4　Xenomai/Linux 实时软件系统结构

Fig. 4　Structure of the Xenomai/Linux system

Xenomai 是一个基于 Linux 系统的实时子系统。通过将 Xenomai 作为 Linux 内核的一部分进行编译，可将

Xenomai 移植入系统内核。根据使用的 SOC 型号选用相匹配的 linux-xlnx-xilinx-v14.5/Xenomai-2.6.3 构建实时操作系统。具体实现方式如下：

（1）运行 Shell 脚本程序 prepare-kernel.sh，为内核安装 Adeos 补丁；

（2）修改内核根目录下的 Makefile 文件，并设置交叉编译环境，移植成功的 Linux 内核会增加“Real-time sub-system”菜单栏选项，显示 Xenomai 对 Linux 实时化的相关配置；

（3）编译内核；

（4）将 U-boot 启动和内核镜像程序烧录到 Flash 存储器中，若系统启动时串口终端显示图 5 信息，表明 Xenomai 成功移植并启动。

```
hw perfevents: enabled with ARMv7 Cortex-A9 PMU driver, 7 counters available
I-pipe: head domain Xenomai registered.
Xenomai: hal/arm started.
Xenomai: scheduling class idle registered.
Xenomai: scheduling class rt registered.
Xenomai: real-time nucleus v2.6.3 (Lies and Truths) loaded.
Xenomai: debug mode enabled.
Xenomai: starting native API services.
Xenomai: starting POSIX services.
Xenomai: starting RTDM services.
bounce pool size: 64 pages
```

图 5　Xenomai 启动打印信息

Fig. 5　Print information of Xenomai start

5　Xenomai/Linux 系统实时性测试与分析

操作系统一个最主要的实时性指标是测试系统的任务响应延迟时间，即从触发一个任务事件开始到该任务被实际执行的间隔时间。Xenomai 提供了一个测试调度延时的 latency 程序，其以一定的频率对自身程序进行循环调用，然后计算出每次程序调度的期望运行时间和实际时间之间的延迟时间。实际测试时，运行命令：./latency -tx -T60 -p400 -h -s。各参数含义为：tx 为运行任务，其中 t0 表示运行用户空间任务，t1 表示运行内核空间任务，t2 表示运行定时器中断任务；T60 表示测试程序运行时间为 60 s；p400 表示程序采样周期为 400 μs；h 表示列出延迟时间的最小值、平均值和最大值；s 表示显示测试结果统计，终端显示的测试数据如图 6 所示。

```
# ./latency -t0 -T60 -p400 -h -s
== Sampling period: 400 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01  (periodic user-mode task, 400 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      7.152|     10.293|     21.147|       0|     0|      7.152|     21.147
RTD|      6.762|     10.308|     20.784|       0|     0|      6.762|     21.147
RTD|      6.882|     10.326|     21.585|       0|     0|      6.762|     21.585
RTD|      6.765|     10.245|     20.424|       0|     0|      6.762|     21.585
RTD|      7.116|     10.257|     21.201|       0|     0|      6.762|     21.585
RTD|      7.305|     10.305|     21.483|       0|     0|      6.762|     21.585
RTD|      6.168|     10.350|     21.444|       0|     0|      6.168|     21.585
RTD|      7.254|     10.281|     22.401|       0|     0|      6.168|     22.401
RTD|      7.398|     10.278|     21.681|       0|     0|      6.168|     22.401
RTD|      7.434|     10.245|     21.057|       0|     0|      6.168|     22.401
RTD|      6.693|     10.239|     21.456|       0|     0|      6.168|     22.401
RTD|      7.188|     10.236|     21.360|       0|     0|      6.168|     22.401
RTD|      6.576|     10.281|     21.159|       0|     0|      6.168|     22.401
RTD|      7.281|     10.287|     21.516|       0|     0|      6.168|     22.401
RTD|      6.906|     10.311|     21.555|       0|     0|      6.168|     22.401
RTD|      6.312|     10.245|     20.907|       0|     0|      6.168|     22.401
RTD|      7.059|     10.290|     21.156|       0|     0|      6.168|     22.401
RTD|      6.795|     10.302|     21.633|       0|     0|      6.168|     22.401
RTD|      7.259|     10.245|     21.195|       0|     0|      6.168|     22.401
RTD|      7.359|     10.260|     20.847|       0|     0|      6.168|     22.401
RTD|      7.374|     10.281|     21.924|       0|     0|      6.168|     22.401
RTT|  00:00:22  (periodic user-mode task, 400 us period, priority 99)
```

图 6　latency 测试结果

Fig. 6　The results of latency experiment

图 7~ 图 9 分别为在用户态、内核态、定时器中断 3 种模式下不同采样周期时的调度延迟时间。由图可知，在用户态时，系统任务调度平均延迟时间小于 15 μs，最小延迟时间小于 10 μs，最大延迟时间小于 30 μs；内核态时，系统任务调度平均延迟时间小于 10 μs，最小、最大延迟时间分别小于 6 μs 和 20 μs；定时器中断时，系统任务调度平均延迟时间小于 4 μs，最小、最大延迟时间分别小于 2 μs 和 10 μs。测试数据表明，3 种模式下调度延迟都是微秒级别的，满足系统功能要求。

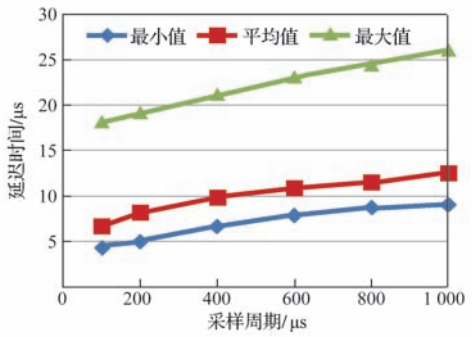

图 7　用户态任务调度延迟

Fig. 7　Task latency in the user mode

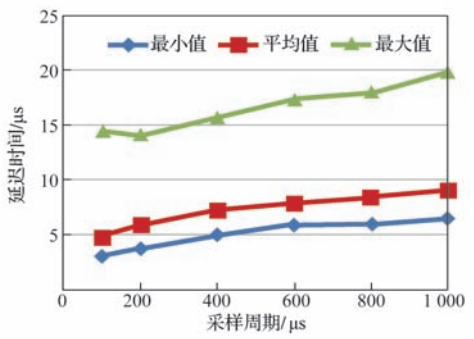

图 8　内核态任务调度延迟

Fig. 8　Task latency in the kernel mode

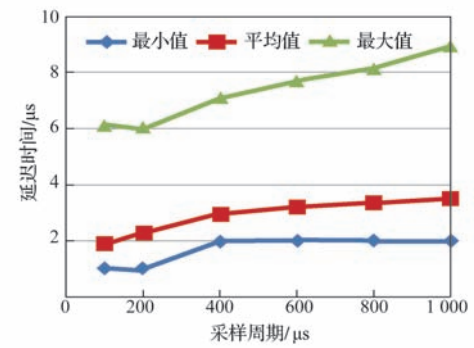

图 9　定时器中断调度延迟

Fig. 9　Dispatch latency in the time interrupted mode

由实验结果可知，Xenomai/Linux 双内核实时操作系统极大地改善了单一 Linux 系统的实时性，可以根据任务实时性要求选择在哪种模式下实现相关功能，整体提升了系统的稳定性和可靠性。

（下转第 19 页）