家居物联实时终端设计与实现

熊光彩[1]，王志峰[1]，张新家[1]，计国君[2]
(1. 西北工业大学 自动化学院，陕西 西安　710072；2. 厦门大学 管理学院，福建 厦门　361005)

摘要： 物联网已成为当今各国重点科技发展战略。家居物联网因与人类生活密切相关，具有广阔的市场空间，在业界引起了广泛关注和积极研究。立足于实际家居物联网需求，特别考虑到实时性能和成本因素，设计了一套针对家居物联网的终端系统。该系统采用 eCos(嵌入式可配置操作系统) 作为嵌入式操作系统，开展了系统移植和驱动移植工作，利用 eCos 的实时功能设计来满足终端的实时性需求，以其开源性节约成本，为家居物联终端普及提供条件。此外结合家居物联网结构特点，通过在 eCos 系统上移植 OLSR 路由协议来实现家居网络的路由，增强网络的鲁棒性，为家居物联网的实时传输应用提供了良好支撑。总体看来，该家居物联终端具有优异的系统实时性和较高的经济适用性。

关键词： 家居物联；实时终端；eCos 系统移植；驱动移植；OLSR；路由协议
中图分类号： TP393　　**文献标识码：** A　　**文章编号：** 1000－8829(2012)03－0080－06

Design and Implementation of Real-Time Terminal for Household Connection

XIONG Guang-cai[1], WANG Zhi-feng[1], ZHANG Xin-jia[1], JI Guo-jun[2]
(1. School of Automation, Northwestern Polytechnical University, Xi'an 710072, China;
2. School of Management, Xiamen University, Xiamen 361005, China)

Abstract: The IOT (Internet of things) has become the worldwide key strategy. Intelligent household, which is closely related to human life, has broad market space and gives rise to comprehensive attention and active research. A suit of terminal for intelligent household usage is designed and implemented based on the application requirements of IOT in intelligent households. In the design, real-time performance and cost are taken specially consideration. eCos (embedded configurable OS) is adopted to make the best of the real-time design of OS, and the corresponding drivers are transplanted. Meanwhile, open source development can promote the IOT popularity in intelligent households. On considering the network structure of intelligent household, OLSR protocol is transplanted for route, which enhances the network robustness. As a result, the platform that supports the real-time transmission is constructed. As a whole, the suit of terminal is economic and good in real-time performance.

Key words: IOT of intelligent household; real-time terminal; eCos system transplant; driver transplant; OLSR; route protocol

互联网、传感技术和嵌入式技术的不断发展迎来了物联网产业朝气蓬勃的新局面。物联网作为一种新的产业形式，将许多高新技术进行了融合，其应用范围已渗透到智能交通、智能物流、智能电网、智能城市、环境监控、公共安全、智能家居等领域，展现了巨大的发展潜力和价值[1]，并将成为新的经济增长点。如今物联网已被列为国家重点发展计划，各研究机构和企业都在这方面加大了研究力度。不久的将来，物联网将会对人类社会生活产生深远的影响。作为物联网研究的一个重要方向，智能家居应用研究正日益成为热点。

在物联网的智能家居应用研究方面，文献[2]对其趋势进行了分析和预测；文献[3]给出了一种智能家居的网络系统模式并总结了相关技术；文献[4]则采用 ZigBee 无线传输技术给出了一种实际家居物联传输方案。已有研究给出了智能家居的应用优势和技术的可实现性，但是少有考虑系统 QoS 提升问题。作为 QoS 提升的重要方面，采用实时嵌入式操作系统可以提高设备的实时性能，引入优异的路由协议可以减少网络传输时延。此外，为了实现智能家居的普及，还

收稿日期： 2011－12－01
基金项目： 国家自然科学基金资助项目(70971111)
作者简介： 熊光彩(1970—)，男，博士后，讲师，主要研究方向为 ERP、PLM 及企业信息化管理、自动化及物联网技术。

需充分考虑到成本因素。笔者重点针对家居物联网络终端的实时性要求,基于 eCos 系统和 OLSR 路由协议,设计并实现了一套经济适用的嵌入式家居物联网实时终端。首先移植 eCos 作为操作系统平台,接着在 eCos 基础上实现了 OLSR 路由协议的移植。该系统以 eCos 内核的实时功能设计,保证了系统任务调度的实时性;同时利用 OLSR 协议的先验式特点,减少了网络数据包转发时延,进而也为实时性能提供支持。实验表明,移植后的 OLSR 协议可以很好地配合 eCos 系统运行,为实时家居物联网络应用提供良好支撑。

1 总体设计及关键技术简析

1.1 终端总体功能结构模型分析设计

在构建实时的家居物联终端的过程中,稳定的硬件设施是系统的基础,需要考虑到硬件本身的性能和软件移植的兼容性和难度。在 QoS 提升上主要考虑智能家居系统的任务调度和网络传输的实时性能,使得系统对事务处理更加快速、敏捷。而实时性能取决于家居终端节点操作系统本身的实时性和网络传输环节的实时性。在操作系统层面,采用实时操作系统是提高系统实时性最有效的办法;对于后者主要取决于网络环境、网络协议栈和路由协议性能。针对智能家居终端系统的需求,在分析硬件和软件需求基础上设计了系统的总体功能模块。硬件上采取成熟的 FL2440 平台,软件实现上主要涉及到 eCos 系统移植和 OLSR 路由协议移植两方面。而 OLSR 路由协议是以 eCos 系统移植为基础的,因而首先要实现 eCos 系统的移植,之后根据系统网络协议栈特点完成对 OLSR 的修改移植。

操作系统采用 eCos 具有很多优点,最显著的就是其所具备的细粒度的可配置性,能最大限度地实现软硬件的裁剪,提升物联系统的性价比。家居物联网一般采用有线或无线的形式接入广域网,OLSR 作为一种混合路由协议,很适宜应用在网络拓扑稳定的家居网络环境中。综合系统软硬件等多方面的分析,终端的总体功能模型设计如图 1 所示。

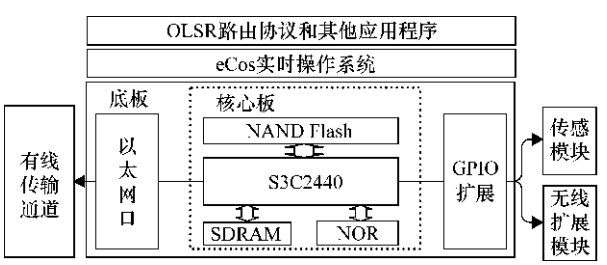

图 1 智能家居终端总体功能模型设计

1.2 硬件选型

终端设计采用的硬件平台为 FL2440,一方面该平台能够满足基本常用硬件需求,同时也具备了丰富的外设扩展接口。在设计上,该平台采用了底板 + 核心板的模式。核心板集成了 CPU、SDRAM 和 Flash 芯片,Flash 包括了 NAND Flash 和 NOR Flash。CPU 采用了 S3C2440,具有高达 400 MHz 的主频,保证了终端的处理能力;采用了型号为 H57V2562GTR 的大容量 SDRAM,提供了 64 MB 的物理地址空间;核心板上两种 Flash 并存为启动方式提供了选择,提高了系统启动运行的灵活性。2 MB 容量 NOR Flash 提供了 ROM 启动支持,大容量的 NAND Flash K9F1G08U0C 主要用于大量数据的存储,比如文件系统映像。在某种程度上讲,核心板构成了硬件平台的一个最小系统。底板上承载了绝大部分相关外部设备,对应了具体应用需求。底板包括的外部设备有多种串口、并口、A/D 转换器、以太网接口和 USB 接口等。FL2440 的优势就在于其还提供了许多可供扩展的 GPIO 接口,通过这些预留接口可以扩展外设,如增添外部传感器、添加多个网络通信接口等,故终端入网可以通过以太网口,必要时也可以加入无线扩展,通过无线连接入网。FL2440 硬件基础平台具备了接口丰富扩展性强的特性,同时价格合理并且获得了广泛的开发和使用,在硬件层面保证了稳定性和经济性。

1.3 ECOS 系统概述

当前嵌入式系统有很广泛的应用[5-6],常见的嵌入式操作系统有 Linux、RTLinux、μcLinux、μcos、VWorks 和 eCos 等。从性能上来讲,各系统间存在内核实时性的区别;从使用授权方式来讲,又可分为商用操作系统和开源操作系统。而开源的操作系统以其经济适用的特点,迎来了业界的广泛关注和大力支持,已成为当前嵌入式开发的一大潮流。通过文献[7]中的分析可见,eCos 作为一款年轻的实时嵌入式可配置操作系统,无论从经济成本还是性能需求上都显示了很大的优势。

eCos 是嵌入式可配置操作系统的简称,相对于 Linux,eCos 具有更好的实时性能,更多地应用于工控场合。另外 eCos 还具有深度可配置特性,且配置粒度可以延伸到具体的应用中,这是其他操作系统无法做到的,因此更适合于特定应用的嵌入式系统的构建。也就是说,eCos 可以根据应用需求去除不需要的特性,实现最小配置。而其他操作系统的应用程序建立在内核上,对于一些不需要的特性无法做到细粒度的配置。再者 eCos 采用静态链接的方式实现应用程序的编译和链接,较之于动态链接库的方式更为简洁。eCos 能适用于多种不同的硬件体系结构,移植性强。同时也表明 eCos 要运行在本设计的目标平台 FL2440 上时,还需要做相应的移植工作。从 eCos 的组成架构

来看，移植工作主要集中在 HAL(硬件抽象层) 和外设驱动部分。eCos 的组成架构[8] 如图 2 所示。

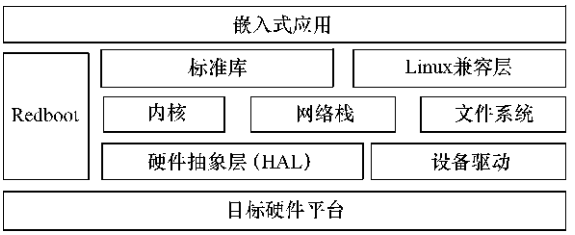

图 2　eCos 组件结构

1.4　OLSR 路由协议简析

为了从网络层面上保证家居网络的鲁棒性和数据传输实时性，需要有性能良好的网络结构和路由机制。路由协议按生成时间可分为先验式路由协议和反应式路由协议，按分组的转发机制又可分为源路由和逐跳路由。先验式的路由延迟相对较小，但随着网络拓扑变化速率的增大会急剧地增加网络的控制开销，所以先验式的路由协议一般适用于拓扑变化慢的网络环境。而反应式路由延迟相对较大，但随着网络拓扑变化速率的增大，其控制开销的增加不像先验式路由那样明显，所以反应式的路由协议更适合拓扑变化快的网络环境。依据家居传感网络的特性，宜采用先验式路由协议。

OLSR 路由协议是先验路由协议的一种。采用多种控制消息在节点形成网络拓扑信息，提高了路由判断的准确性; 并在拓扑信息的基础上，依据多种网络信息(跳数、链路带宽、队列长度) 以最短路径算法计算路由。OLSR 在控制消息的转发上做了优化，可以适当减小一般先验式路由的控制开销。主要策略是采用了 MPR 转发(Multipoint Relays) 方式，对所包含的局部拓扑信息进行了有效的压缩，并通过多点中继集转发广播消息来代替全网泛洪。OLSR 属于混合路由协议[9]，可应用在有线或无线网络中，同时实验也证实在网络拓扑变化较慢的情况下，OLSR 的路由性能十分优异。但是 OLSR 的发布版本(可参见 http: //www.olsr.com) 并没有对 eCos 提供支持，因此需要将 OLSR 在 eCos 系统中做相应的移植。

1.5　家居物联终端应用模式分析

在当前物联终端市场上，也出现了比较成熟的物联终端产品。典型应用有物联视频监控系统，这类解决方案固然有很多用处，但是并不是所有的环境参数都可以通过图像来完成检测，如温度、湿度等。从这点来看，这些产品在信息采集上呈现应用单一; 同时终端呈现固定的集中式组网模式，终端间的通信需要靠专门的物联网关完成。如果要获得较为全面的参数检测和监控，在终端应用单一的条件下需要多种类型终端，配合物联网关完成。一方面网络的灵活性欠佳，另一方面成本会成为一个阻碍其实施的问题。特别是对于家居物联网应用来说，其要求应用多样性的同时应尽量降低成本。这时就需要终端具有多种参数采集的能力，本研究中的终端在扩展性上的设计解决了这个问题，一个终端可以扩展以同时接入多路参数采集信号，而且传感器的类型也可以根据需要进行替换和调整，使得一个终端可承载多种参数采集任务。在网络形式上，采用 OLSR 路由协议奠定了网络形式多样性的基础，终端自身可以起到物联网关的作用，可以为其他节点实现路由和转发，无需额外购置网关; 可以实现集中式的网络管理，也可以配置为点对点的对等网络形式。而点对点的对等网络形式在具有无线扩展的条件下会更显示出其优势。因此对于现有的物联实现方案，本研究实现的家居物联终端的应用适应性和组网灵活性都有所提升。

2　家居物联终端 eCos 系统移植

eCos 系统结构的所有上层组件都建立在 HAL 上。HAL 分为体系结构抽象层、变体抽象层和平台抽象层，它屏蔽了具体硬件之间的差异，使得上层组件可以实现无缝跨平台移植。eCos 系统移植的主要工作就是 HAL 层的移植，具体移植内容要根据所采用的硬件平台，参考相近的平台模板进行相应源码修改，以使 eCos 运行在新的平台上。

2.1　eCos 系统移植

eCos 提供了许多可用的系统移植模板，通过移植类似硬件平台的系统模板，可以大大降低移植难度。RedBoot 作为 eCos 提供的一个最小应用，是 eCos 系统移植的最好参考，也是 eCos 移植工作的第一步。本文选用系统提供的 SMDK2410 模板来实现，具体如下:

① 在源码目录对应的地方复制一份 SMDK2410 的源文件目录结构，并将该目录改为 FL2440 对应的源码目录结构。

② 修改关键的源文件以符合 FL2440 平台的硬件特性。这些文件位于 FL2440 源码目录下的 include/ 文件夹和 src/ 文件夹下面，这些文件的名称和用途如表 1 所示。

表 1　FL2440 移植关键文件

文件名称	文件用途
Include/plf_cache. h	预处理器相关的特殊缓存处理
Include/hal_platform_intrs. h	定义平台的中断向量和专用的平台中断处理
Include/hal_platform_setup. h	平台的启动代码宏定义
Src/FL2440_misc. c	平台初始化，中断初始化，中断的使能和禁止，时钟延时功能等
Src/hal_diag. c	包含了诊断输出的功能实现

③ 添加 HAL 包到 ecos. db 数据库文件中，使得 eCos 系统配置工具 configuretool 能够使用和配置新加入的包。同时还需在该数据库文件中为 FL2440 定义一个新的硬件目标平台。完成新平台的定义和添加后，就可在配置工具中找到新平台进行配置。如图 3 所示。

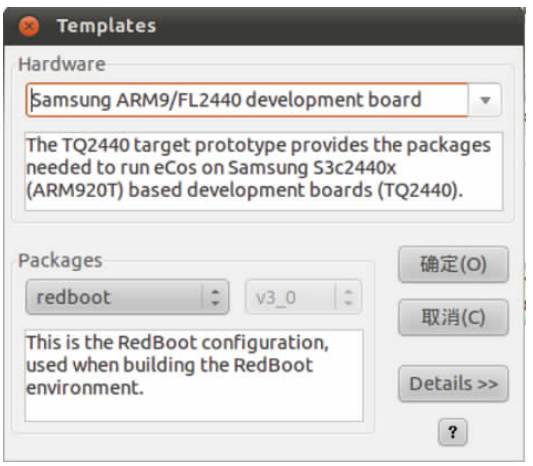

图 3 添加配置新的硬件平台

④ 配置 Redboot 启动方式为 ROM，设定全局编译链接标志、特定包相关的编译标志和相关硬件参数。配置过程中会出现各种冲突，依照配置工具提示可解决部分冲突，必要时还需修改源码。确定解决所有的冲突后，编译链接源文件可产生 redboot. bin 二进制文件。

生成的 redboot. bin 文件可以通过 H-JTAG 工具烧写到系统的 NOR Flash 中。将系统设置为 NOR 启动方式，即可从 NOR Flash 启动系统。当 redboot > 提示符出现在串口终端上时，表明 RedBoot 移植成功了。

2.2 eCos 关键驱动移植分析

Redboot 正常启动后，接着进行设备驱动移植以支持平台特定的外部设备。eCos 驱动开发要遵循一定的开发框架[10]，该框架为系统的上层 I/O 子系统提供接口。

本文硬件平台主要驱动有：串口驱动、以太网卡驱动、NAND Flash 驱动等。串口驱动相对较为简单，这里以 DM9000AE 网卡驱动为例详细讲述 eCos 驱动移植的开发及实现过程，NAND Flash 驱动的移植和网卡驱动类似。

2.2.1 网卡驱动框架分析

在 eCos 驱动开发中需要重点完成两部分工作。首先是定义和实现驱动函数 API(应用程序接口) ; 其次，将这些 API 通过一个设备驱动结构体链接起来并在操作系统中注册。网络驱动框架的数据结构及其相互关系如图 4 所示。

网络驱动移植从创建一个 cyg_netdevtab_entry_t 结构实例开始，将代表驱动设备的条目注册到系统的网络驱动列表中。系统启动初始化时，即可通过查找设备表访问到该条目，调用注册初始化函数 xxx_init( ) 进行相应的初始化。初始化完成后网卡工作运行时所需要的功能就要追溯到 cyg_netdevtab_entry_t 结构的 device_instance 成员，该成员指向一个 eth_drv_sc 结构。eth_drv_sc 结构将许多下层和硬件交互的接口集合为 eth_hwr_funs 结构，形成一个网络函数操作句柄，通过该句柄可以访问到相关联的下层 API 以实现硬件访问。驱动移植的大部分工作就是根据具体的硬件编写这些底层 API，然后遵循图 4 的结构填充驱动框架。

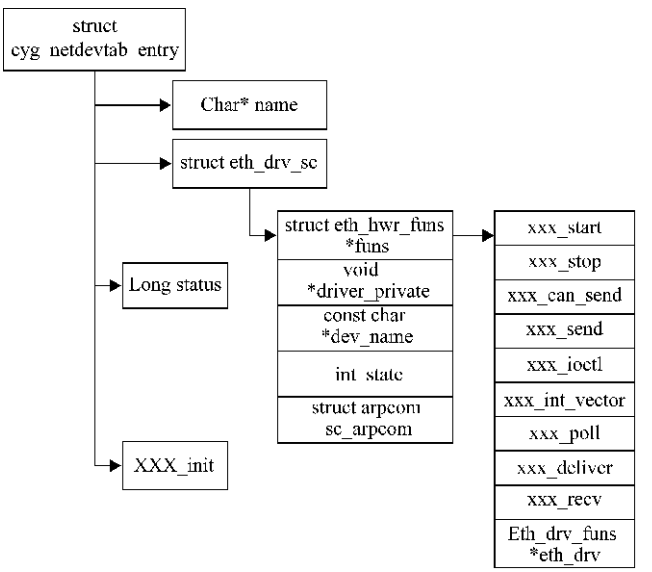

图 4 网络驱动数据结构关系

2.2.2 网卡驱动中断处理

eCos 驱动有轮询和中断两种工作模式。中断模式下一般会涉及到 ISR(中断服务函数) 和 DSR(延时服务函数) 两类函数，二者均是通过 cyg_drv_interrupt_create 函数注册给系统的。ISR 仅处理耗时较少的中断细节，若还需要耗时的后续事务处理，则递交给 DSR 进一步完成。这取决于 ISR，即如果需要 DSR 被调度，则 ISR 的返回值为 CYG_ISR_CALL_DSR，否则返回 CYG_ISR_HANDLED 表示中断处理完成。网卡驱动采用“ISR + DSR”的中断处理模式。在 ISR 处理完成后即调用 DSR 函数，DSR 函数结束前一般要调用系统的 eth_drv_dsr 函数唤醒“Network Delivery Thread”线程，该线程调用 xxx_deliver 驱动函数去处理耗时的数据搬移工作。在中断模式下，框架中的 xxx_poll 函数不需使用，故这里不实现该函数。

2.2.3 驱动添加与测试

所有的驱动源码开发完成后，都要添加到 ecos. db

数据库中。FL2440 目标模板也要做相应的修改以加入硬件驱动包支持。使用配置工具 Configuretool 对新的模板做出合理配置并编译之后，就产生了新的 eCos 系统库文件。图 5 为新的 eCos 配置产生的 Redboot 运行测试。分别通过了 FIS(文件映像系统) 和网络命令测试，表明驱动移植的正确性和可用性。

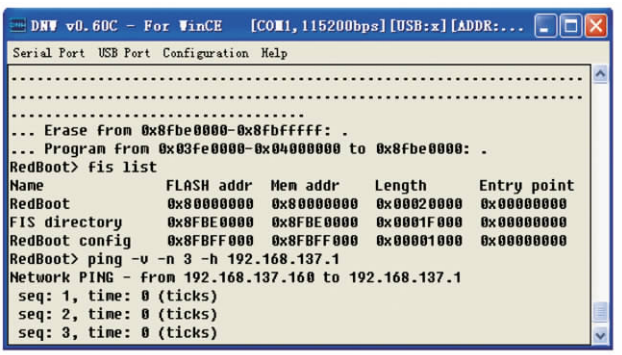

图 5　FIS 和网络测试

3　家居物联终端 OLSR 协议移植

eCos 系统移植完成之后，接下来要进行 OLSR 路由协议的移植工作。OLSR 协议会根据不同的操作系统使用不同的源文件，从而生成适应特定系统的可执行文件。因此 OLSR 移植工作需要结合特定的操作系统，对协议中与操作系统相关的属性进行分析，找出其目录结构中与操作系统对应的部分作为移植的重点。OLSR 协议的实现源码 OLSRD 可以从 OLSR 官方网站下载。

3.1　OLSR 协议的源码结构分析

要进行 OLSR 移植工作，必须分析其源码结构。而了解源码结构最直观的方法，就是分析源码的 Makefile 文件结构，获得源码编译链接的详细过程信息。不同操作系统的 Makefile 文件组成不同，因此需在分析源码结构中 Makefile 文件体系的基础上，再进一步具体分析 Makefile 文件内容以掌握协议的组件构成。主 Makefile 文件包含 Makefile. inc 文件，Makefile. inc 文件又包含 make/Makefile. xxx 文件(xxx 表示对应目标操作系统) 。在 Makefile. xxx 中实现对应操作系统的源码文件和编译链接标志的添加。源码中 src 文件夹是主要的源文件目录，是移植工作研究的重点，而其他目录则与协议扩展或 GUI 功能有关，这里不予讨论。不同操作系统的相应源码放在 src 目录的相关文件夹下，主要有 bsd/、linux/、unix/、win32/，分别代表了所支持的 BSD、Linux、Unix 和 Win32 操作系统。

3.2　基于 eCos OLSR 路由协议移植

eCos 系统硬件抽象层和驱动的移植，给 eCos 应用程序开发和移植提供了良好的基础环境。在 eCos 中路由协议的移植和其他应用移植一样，都被当成系统的一个特定应用。eCos 应用程序实现的一般步骤如图 6 所示。

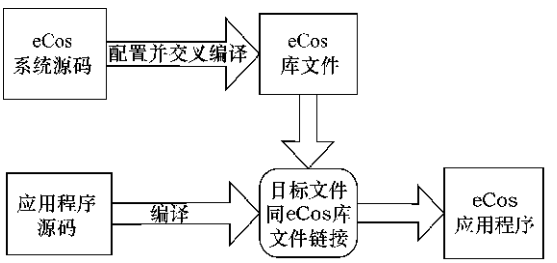

图 6　eCos 应用程序开发过程

总体上讲，eCos 首先需要通过合理配置编译产生静态库: 主要包括 libtarget. a、libextra. a 和链接脚本文件 target. ld。本文中用到的主要配置是 FL2440 硬件模板和 net 包套件，其他的应用需求可以通过添加相应的组件包到现有的配置中来完成。

OLSR 协议本来是为 Linux、Mac OS 和 BSD 系列操作系统而开发的。鉴于 eCos 中有一套 TCP/IP 协议栈是从 FreeBSD 移植而来，因此 OLSR 在 eCos 协议栈中的移植可参照 FreeBSD 的移植方式来实现。在分析了 OLSR 的源码结构和关键源文件后，通过以下关键步骤可完成该协议在 eCos 系统的移植工作。

① 修改相关 Makefile 文件，设置正确的交叉编译环境。包括设置正确的交叉编译器，正确的头文件包含路径和链接路径，并调整程序的编译和链接标志。主要设置如下:

```
CC = arm - eabi - gcc //设定交叉编译器
CFFLAGS += - I/ecos_installdir/include - L/ecos_installdir/lib //设定编译标志
LDFLAGS += - g - nostdlib - Wl, -- gc - sections - Wl, - static //设定链接标志
```

② 改变协议的运行模式。由于 eCos 是单进程操作系统，因此 OLSR 中的一些多进程调用在 eCos 下无法正常工作。解决方法是将多进程的实现部分替换为多线程实现模式。最典型的改动就是将 OLSR 的 main 函数修改为一个 eCos 的运行线程。

```
cyg_thread_create( OLSR_PRIORITY,olsr_main_function,( cyg_addrword_t) ( &entry_data) ," Thread Olsr",( void * ) stack,STACK_LENGTH,&olsr_thread,&thread) ; //创建线程
cyg_thread_resume( olsr_thread) ; //开启线程
```

③ 修改 OLSR 和 eCos 源码中的一些函数及变量的定义冲突。可通过编译 OLSR 源码来发现更多的错误并加以修正。完成编译链接过程后，olsrd 就会在 OLSR 源码文件夹下生成。

④ 运行并测试上一步生成的 olsrd，发现并改正运行时的错误，从逻辑上保证协议正确运行，从而使 OL-

SR 能流畅地运行在 eCos 系统中。

OLSR 的协议运行参数是通过配置文件来实现的,这样保证了协议配置上的灵活性。为了保持这种特性,基于 NAND Flash 驱动移植了 jffs2 文件系统[11],实现了配置文件的运行时访问。jffs2 文件系统在 eCos 中有相应的源码包支持,这里根据硬件特点做相应的源码修改,在配置中加入 jffs2 源码包,并添加相应的依赖包。添加包的相应配置如下:

```
cdl_configuration eCos {
    package CYGPKG_IO_FLASH current ;
    package CYGPKG_MEMALLOC current ;
    package CYGPKG_COMPRESS_ZLIB current ;
    package CYGPKG_IO_FILEIO current ;
    package CYGPKG_FS_JFFS2 current ;
    package CYGPKG_ERROR current ;
    package CYGPKG_LINUX_COMPAT current ;
    package CYGPKG_IO current ;
    package CYGPKG_CRC current ;
    package CYGPKG_LIBC_STRING current ;
};
cdl_option CYGPKG_IO_FILEIO_DEVFS_SUPPORT {
    user_value 1
};
cdl_component CYGPKG_IO_FLASH_BLOCK_DEVICE {
    user_value 1
};
```

新的配置编译完成后就可以实现配置文件的运行时访问了。首先将配置文件按照 OLSR 源码中指定的访问路径层次存放,然后使用 mkjffs 工具将该文件夹制作为文件系统镜像,利用 FIS 工具烧写文件系统到 NAND FLASH 中,这样 OLSR 协议在运行时就可以通过文件系统顺利访问到配置文件,在获取配置参数后便可进入正常运行状态。OLSR 移植运行效果如图 7 所示。最后编写网络数据传输线程,实现整个实时家居物联终端的设计。

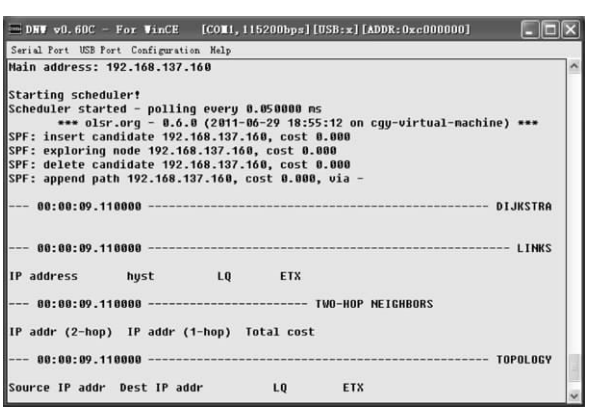

图 7 OLSR 路由协议运行

4 结束语

本文针对家居物联网在实时性和经济性方面的需求,实现了一套适用的家居物联终端设计。首先将具有实时功能设计的开源 eCos 系统移植到 FL2440 目标平台,并基于此进行了 eCos 系统驱动的开发工作,重点包括网卡和 NAND Flash 的驱动移植;接着选用 OLSR 进行了路由协议移植,实现了相对稳定的物联网络路由维护,同时利用 eCos 的 jffs2 文件系统存储 OLSR 协议配置文件,保留了路由协议便捷的可配置性能。通过 eCos 系统和 OLSR 协议这两方面的移植工作,构建了一个兼具经济性和实时性的家居物联网络终端系统。本文的研究,为满足家居物联网络越来越高的实时性和经济性需求奠定了基础,具有普遍的借鉴意义和实用价值。

参考文献:

[1] Yang Z H,Peng Y F,Yue Y Z,et al. Study and application on the architecture and key technologies for IOT [A]. 2011 International Conference on Multimedia Technology [C]. 2011:747 - 751.

[2] 徐卓农. 智能家居系统的现状与发展综述 [J]. 电气自动化,2004,26(3):3 - 4,8.

[3] Han M,Miao C Y. The design of intelligent household system based on wireless communications [A]. 2011 International Symposium on Computer Science and Society [C]. 2011:206 - 209.

[4] 纪晴,段培永,李连防,等. 基于 ZigBee 无线传感器网络的智能家居系统 [J]. 计算机工程与设计,2008,29(12):3064 - 3067.

[5] Chu H Y,Xie Z J,Shao Y H,et al. Design and implement of WSN based on Bluetooth and embedded system [A]. 2010 International Conference on Computer Application and System Modeling [C]. 2010:641 - 644.

[6] Helps C R,Armstrong J. Review of current embedded system hardware,OS,development systems and application domains for instructional design [A]. 114th Annual ASEE Conference and Exposition [C],2007.

[7] 石强. 基于方舟 CPU 的 eCos 嵌入式操作系统移植与裁剪的实验研究 [D]. 内蒙古:内蒙古工业大学,2009.

[8] Massa A J. Embedded software development with eCos [M]. Prentice Hall,2003.

[9] 钟辉,刘倩,赵德平. 应用于无线 Mesh 网中的 OLSR 路由协议研究 [J]. 沈阳建筑大学学报(自然科学版),2009,25(2):371 - 374.

[10] 秦怀峰,施笑安,周兴社,等. eCos 设备驱动程序设计分析 [J]. 微电子学与计算机,2003,20(7):8 - 13.

[11] Chen C H,Huang W T,Chen C T. et al. Improving the reliability of JFFS2 [A]. 2006 International Symposium on VLSI Design,Automation and Test [C]. 2007:1 - 4. □