基于 MPC8247 嵌入式电力交换系统的设计与实现

姚启桂[1]，于　海[1,2]

（1. 中国电力科学研究院 信息与通信研究所，江苏 南京　210003；2. 东南大学 电气工程学院，江苏 南京　210003）

摘　要：首先介绍了基于 MPC8247 的嵌入式电力交换系统的硬件平台，着重阐述了如何在 MPC8247 处理器上开发嵌入式电力交换系统，主要包括开发环境的搭建、系统引导模块 U-Boot 修改、内核的裁剪和移植，交换软件 SDK 移植，应用软件 ZebOS 的编译等。通过电力交换设备（DPN8000）的现场应用，证实该系统具有体积小，功耗低，高带宽等优点，为电力通信网络和系统的开发做出了贡献。

关键词：嵌入式系统；MPC8247；电力交换系统；移植

中图分类号：TN919-34；TP391　　**文献标识码**：A　　**文章编号**：1004-373X(2012)20-0015-04

Design and implementation of embedded power exchanging system based on MPC8247

YAO Qi-gui[1]，YU Hai[1,2]

（1. Research Institute of Information Technology & Communication，EPRI，Nanjing 210003，China；
2. School of Electrical engineering，Southeast University，Nanjing 210003，China）

Abstract：The hardware platform of embedded power exchanging system based on MPC8247 is introduced. The method to develop the MPC8247 processor in embedded power exchanging system is elaborated emphatically，mainly including the establishment of the development environment，modification of the system boot module U-BOOT，shearing and transplantation of Linux kernel，transplantation of the exchanging software SDK and compiling of the application software Zebos. The field application of the power exchanging equipment（DPN8000），proved that the system has the advantages of small volume，low-power consumption and high bandwidth，etc. The design contributes to the development of the power communication networks and systems.

Keywords：embedded system；MPC8247；power exchanging system；transplantation

随着电力通信的发展，电力业务量的增加，对嵌入式电力设备的 I/O 处理能力有了更高的要求，尤其需要能提供大容量、高速率和高带宽的语音、数据和视频业务支撑的嵌入式接入设备，因此，性能高，功耗低的大容量嵌入式电力设备倍受青睐。

Freescale 公司生产的 PowerPC 处理器 MPC8247 具有集成度高，功耗低和能支持丰富的外围设备等优点[1-2]，再搭配 IP Infusion 的 ZebOS(R)软件，实现了快速通信与高吞吐率的数据处理，有效地提高了网络的性能、降低了网络基础设施的成本并增加了运营商和企业的收益，很好地解决了嵌入式电力通信设备遇到的问题。本文以开发大容量电力光交换设备 OLT（DPN8000）为背景，介绍了电力交换系统的硬件设计和嵌入式交换系统系统开发的搭建，重点针对如何实现嵌入式电力交换系统给出解决办法。

1　硬件平台的概述

MPC8247 板主要包括 CPU 子系统，其系统框图如图 1 所示。

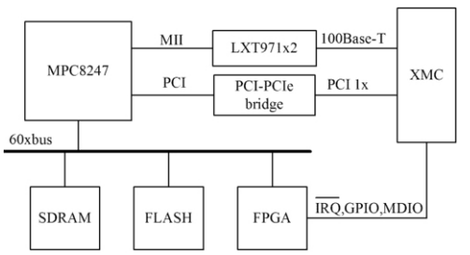

图 1　MPC8247 板的硬件系统框图

MPC8247 是 Freescale 公司 MPC82XX 系列微处理器的一种，主要由 G2_LE 内核（PowerPC 603e 内核的一种），系统接口单元 SIU 以及通信处理模块 CPM 组成。支持 60x 总线，其数据线宽为 64 位，地址线为 32 位；支持 PCI/LOCAL 总线，其数据线宽为 32 位，地址线为 32 位。PowerPC8247 内核工作时钟最高为 400 MHz，CPM 工作时钟最高为 200 MHz[3]。外设接口大致如下：

(1) SDRAM:64 位 256 MB 容量,由 4 片 64 MB 的 MT48LC32M16A2TG 组成,使用 Local Bus 的 D[0:63]和 A[16:28]。

(2) FLASH:32 MB 容量,由 2 片 16 MB 的 S29GL128N 组成,设计中该 FLASH 工作在 WORD 模式,使用 Local Bus 的 D[0:31]和 A[4:29]。

(3) PCI 总线:引出一个 32 位 66 MHz 的 PCI 总线,经过 PCI-PCIe 桥片 PI7C9X10 转换成为 PCIe 总线,经 XMC 连接器连接到底板的交换芯片 BCM56338,作为 Fabric 芯片的管理端口。

2 嵌入式系统开发环境的搭建

2.1 准备开发环境

通常嵌入 Linux 的开发分为主机系统和目标系统,他们之间通过网络接口和串口互连,网口一般用以下载程序和内核映像文件,串口一般作为 console 控制台来使用,在主机通过超级终端与目标系统进行命令交互,就像 PC 的键盘和显示器。

嵌入 Linux 系统模型如图 2 所示。

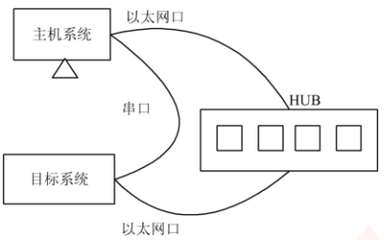

图 2 嵌入 linux 系统开发模型

2.1.1 安装开发编译环境

通常主机与目标板的 CPU 都不相同,需要进行交叉编译。在本项目中使用了 GNU GCC 工具链。GNU GCC 的 PowerPC 版本需要交叉编译,所有源代码可以从 FSF 的 FTP 站点[4]。

整个建立过程为:下载源文件、补丁和建立编译的目录;建立内核头文件;建立二进制工具(binutils);建立初始编译器(bootstrap gcc);建立 C 库(glibc);建立全套编译器(full gcc)。

安装好之后可以用"which ppc_8xx-gcc"来查找一下产生的目标执行文件在哪里。

2.1.2 配置 DHCP

主机系统需要设置 DHCP 服务器,用于目标系统在采用 BOOTP 启动时获取 IP 地址。设置之前首先要知道目标系统的 MAC 地址,同时要保证主机系统安装了 DHCPD 服务器。修改虚拟机中/etc/dhcpd.conf 配置文件,然后用如下命令重新启动 dhcpd 服务:

```
# /etc/init.d/dhcpd restart
```

同时启动 dhcpd 服务,使主机每次启动时自动启动该服务。

2.1.3 配置 TFTP 服务器

TFTP(简单文件传输协议)服务器用于目标系统从主机系统上获取可执行的代码或内核映象文件。

以超级用户登录主机系统,编辑下面文件:

```
# vim /etc/xinetd.conf
```

去掉下面一行的注释:

```
# tftp dgram udp wait root /usr/sbin/tcdp in.tftpd
```

根目录下创建 tftpboot 目录:

```
# mkdir tftpboot
```

将每次编译好的目标代码拷贝到该目录下,在系统服务中启动 TFTPD 服务(运行 # setup ,在"System services"中选择。)然后执行:# service xinetd restart 即可。

2.1.4 配置 NFS

NFS 主要是给目标系统提供一个根文件系统(Root Filesystem)。

首先,编辑配置文件/etc/exports:

```
# vi exports
```

如果原来不存在,添加一行:

```
/home/com_target_master*(rw,no_root_squash)
```

"/home/com_target_master"是根文件系统所在的目录,"*"允许主机所在网段所有机器访问。和配置 TFTP 一样,在系统服务中启动 NFS 服务,重新启动系统或运行:

```
# service nfs restart
```

2.2 目标板 FLASH 资源分配

对于系统平台来说,建立良好的 FLASH 分区是系统成功运行的基本保证。具体 FLASH 资源分配如表 1所示。

表 1 开发板 FLASH 资源分配

地址范围	内容
0xFE000000～0xFE100000	Linux kernel 1 MB(内核)
0xFE100000～0xFE900000	ramdisk=8 MB(文件系统)
0xFE900000～0xFFF00000	user_JFFS2=22 MB(应用 APP)
0xFFF00000～ 0xFFFFFFFF	U-Boot=1 MB(U-Boot 引导)

3 嵌入式系统开发的实现过程

本文的软件平台是基于电力交换设备,DPN8000 的开发环境为模型。搭建软件层次模型如下:

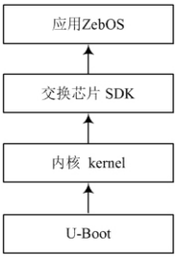

图 3　软件层次模型

3.1　Linux 系统引导模块 uboot

U-Boot 是德国 DENX 软件中心依照 GPL（General Public License）发布的系统引导模块，支持多种处理器，如 ARM 系列、PowerPC 系列、MIPS 系列、X86 系列等。根据自己的目标系统修改编辑 U-Boot 包下的源文件，进行移植，最后编译 U-Boot 后生成二进制文件，用于引导系统[5-7]。

在 U-Boot 代码中，主要有关的代码设置在 MYM（uboot）\cpu\mpc8260.c 的代码中实现，但是该地址的定义在其平台说明头文件中。故增加一个 mpc8247 平台—— mpc8260ads，在 $（u-boot）\include\config 目录下增加一个相应平台配置头文件说明-mpc8260ads.h，将与硬件有关的配置放在此文件中。对应 SDRAM，FLASH 等定义如下：

（1）SDRAM 的地址设定

```
#define CFG_OR1    0xF0002900
#define CFG_PSDMR    0xc4322423
```

以上定义的为 SDRAM 的地址设定，由 MPC8247 片选 CS1 相关的寄存器设定。

（2）FLASH 的地址设定

```
#define CFG_FLASH_BASE    0xfe000000
#define CFG_BR0_PRELIM
CFG_FLASH_BASE | 0x00001801
#define CFG_OR0_PRELIM
CFG_FLASH_BASE | 0x000008C2
```

以上定义的为 FLASH 的地址设定，由 MPC8247 片选 CS0 相关的寄存器设定。

（3）FPGA 地址设定

```
#define CFG_FPGA_BASE 0xF4500000
#define CFG__BR3__PRELIM    CFG__FPGA__BASE
| 0x00001801
#define CFG_OR3_PRELIM    0xFFFF8010
```

以上定义的为 FPGA 的地址设定，由 MPC8247 片选 CS0 相关的寄存器设定。

修改完成后，通过 make 得到 u-boot.bin 文件。

3.2　Linux 内核的移植[8-9]

由于嵌入式系统存储资源有限，精简内核就显得尤为重要。建立系统的新内核 ，最主要的工作就是配置内核参数配置内核文件，进入内核释放的目录下执行命令：#make menuconfig 会生成新的.config 文件。然后执行 #make uImage，生成新的 uImage，将生成的内核拷贝到 tftp 目录下即可。

使用 tftp 将内核镜像文件下载到目标板上，重新引导后，目标板即可成功启动。

3.3　文件系统的制作

将得到 Busybox 版本进行解压，配置和编译就可以得到想要的文件系统。具体命令如下：

```
下载一个 busybox-1.1.3.tar.gz
  #tar xvzf busybox-1.1.3.tar.gz
  #cd busybox-1.1.3
  #make defconfig
  #make menuconfig
```

Busybox 编译出一个单个的独立执行程序，就称为 Busybox，它集成了非常多的命令工具，如果要使用某一命令，需要做一个软链接就可以了[10]。在这个项目中，采用 busybox 来配置所需各类文件，具体方法参见 busybox 的手册。

3.4　SDK 的移植过程

将拿到的 sdk-xgs-robo-5.6.2.tar.gz 的源码文件解压到/home/com_work/目录下：

```
#cd /home/com_work
#tar -zxvf sdk-xgs-robo-5.6.2.tar.gz
```

3.4.1　设置环境变量

设置环境变量如下：

```
#export SDK=/home/com_work/sdk-xgs-robo-5.6.2
```

这样 SDK 的环境路径已经设好，进入 make 目录下，修改对应的 Makefile.linux-bmw 和 Make.local 文件，使这2个文件对应的内核，编译器路径和交换芯片类型是所需要的类型。

3.4.2　编译 SDK

如果想重编译 SDK，只需进入：/home/com_work/sdk-xgs-robo-5.6.2/systems/linux/kernel/bmw 目录下，执行 #make clean；make 将生成的模块拷贝到对应文件系统的对应的目录下即可。

生成的模块如表 2 所示。

然后将用到的模块加载。执行过程如下：

```
#insmod  linux-kernel-bde.o -f
#insmod  linux-uk-proxy.o -f
#insmod linux-bcm-diag-full.o -f
#./bcm.user.proxy
```

3.5　ZebOS 的移植过程

将拿到的 ZebOS771.tar.gz 的源码文件解压到/home/com_work/目录下：

```
# cd /home/com_work
# tar -zxvf  ZebOS771.tar.gz
```

表 2　sdk 编译生成的模块及其功能

输出文件名称	描述
Linux-kernel-bde.o	Broadcom 设备枚举模块
Linux-uk-proxy.o	Kernel/usr 空间控制台代理
linux-bcm-diag-full.o	完整的 Broadcom 诊断 shell.
bcm.user.proxy	User space console interface to the kernel diag shell

3.5.1　建立环境变量

进入 zebos 目录下：

```
# cd /home/com_work/ZebOS771
```

建立 set_export.sh 文件源码如下：

```
# export COMPILER_DIR =/home/mvl_pro/pro3_mpc82xx/montavista/pro/devkit/ppc/82xx/bin   # 编译器路径
# export COMPILER_PREFIX=ppc_82xx-    # 编译链
# export KERNEL_SOURCE =/home /linux-2.4.20_mvl31
# export LINUX_INCLUDE = MYMKERNEL_SOURCE/include
# export USE_LINUX_24=1
# export IPI_ZEBOS_PLATFORM=linux
# export BROADCOM_SDK=/Broadcom_SDK
# export SDK=MYMBROADCOM_SDK
# echo "OK！"
```

修改完 set_export.sh 文件，执行：# source set_export.sh 即可。

3.5.2　修改配置文件编译 ZebOS

进入 ZebOS 目录下，主要修改的文件是 config.sh，修改好自己要编译的模块，执行：#. config.sh 进入 platform/linux 目录下：# cd platform/linux

make clean; make all 就好了。将 platform/linux/bin 目录下生成的 bin 文件拷贝到文件系统对应目录下，这样，就完成了整个系统的开发搭建。

4　结　语

基于 MPC8247 嵌入式电力交换系统的设计，具有体积小、功耗低、性能高，丰富的应用接口等优点，随着大容量电力光交换设备 DPN8000 的现场应用，证实该系统充分发挥了对网络和配电等设备的强大支持功能。该平台系统在变电站工作环境比较恶劣的条件下有较好的适应性和强大的通信能力，提供了更高的转发速率，更大的吞吐率和更好的包处理能力，充分体现了其优异的性能。

参　考　文　献

[1] 杨佳松，张浩，牛志刚，等. 基于 mpc8248 的嵌入式 Linux 系统的研究与实现[J]. 测控技术，2008，28(1)：51-54.

[2] 方先康. 基于 PowerPC 处理器 MPC8541E 的嵌入式 Linux 系统开发[J]. 中国科技论文，2007(7)：12-15.

[3] 张娟，蒋瑜，蹇柯，等. 基于 PowerPC 8247 的嵌入式 Linux 系统开发[J]. 计算机系统应用，2009，18(12)：224-227.

[4] 楼杨，王广龙. 基于 MPC8548 的嵌入式设备的光通信接口设计与实现[J]. 设计与应用，2011，19(4)：911-914.

[5] 褚丈奎，樊晓光，黄培成. 基于 PowerPC 处理器 MPC8250 的嵌入式 Linux 系统开发[J]. 计算机工程与设计，2006，27(1)：179-182.

[6] YAGHMOUR Karim. Building embedded Linux systems [M]. USA：O'Reilly，2003.

[7] 罗滨. 一种嵌入式移动数据库事务处理模型的设计与实现[D]. 长沙：湖南大学，2008.

[8] 金中朝. 基于嵌入式 Linux 的数据采集系统设计与实现[D]. 太原：中北大学，2008.

[9] 毛德操，胡希明. Linux 内核源代码情景分析[M]. 杭州：浙江大学出版社，2001.

[10] 张晓林，崔迎炜. 嵌入式系统设计与实践[M]. 北京：北京航空航天大学出版社，2006.

作者简介：姚启桂　男，1983 年出生，安徽阜阳人，硕士，工程师。主要研究为电力通信系统，嵌入式产品底层驱动的开发和研究。

于　海　男，1980 年出生，江苏盐城人，博士生，工程师。主要研究方向为电力通信系统、电力系统自动化。

（上接第 14 页）

[3] Texas Instruments. TMS320 DSP_BIOS user's guide v3.3 [R]. USA：Texas Instruments，2004.

[4] 梁迅. 基于 NDK 的 DSP 网络编程[J]. 计算技术与自动化，2005(3)：79-81.

[5] Texas Instruments. NDK programmer's reference guide [R]. USA：Texas Instruments，2007.

[6] 黄敬礼，钱进. 基于 DM642 的实时多协议转换器设计[J]. 现代电子技术，2011，34(22)：147-149.

[7] 陶世芳. 基于 DSP 和 CPLD 的光纤陀螺信号采集系统设计[J]. 现代电子技术，2012，35(7)：133-135.

[8] 徐红伟. 高速数据采集记录装置研制[D]. 哈尔滨：哈尔滨工业大学，2010.

[9] 徐婷，屈姗姗. 基于 DSP/BIOS 的多线程并发执行的实现[J]. 测试测量技术，2011(9)：1-3.

[10] Texas Instruments. TMS320C6000 peripherals reference guide [R]. USA：Texas Instruments，2004.

作者简介：钱科威　男，1988 年出生，硕士研究生。研究方向为自动测试技术。

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统

7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析

Linux:

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究

PowerPC:

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置

ARM:

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计