文章编号:1001 - 9081(2003)12 - 0108 - 04

一个嵌入式 Linux 应用系统在 Lubbock 开发板上的实现

郭胜超,吕　强,杨季文,钱培德
(苏州大学 计算机工程系,江苏 苏州 215006)

摘　要:系统地介绍了一个嵌入式 Linux 应用系统在 Intel DBPXA250 ARM 开发板 Lubbock 上的构建过程,描述了系统结构和功能,重点介绍了如何利用开放源码项目 ARMBoot、ARMLinux 和 Qt-Embedded 构建该应用系统,给出了整合系统的全部过程。

关键词:ARMBoot;ARMLinux;Qt-Embedded;整合
中图分类号: TP368.2;TP316.89　　**文献标识码**:A

Constructing an Embedded Linux Application System on Lubbock Development Board

GUO Sheng-chao, LüQiang, YANG Ji-wen, QIAN Pei-de
(Department of Computer Engineering, Soochow University, Suzhou Jiangsu 215006, China)

Abstract: The article introduces an embedded Linux application system based on Intel DBPXA250 ARM development board called Lubbock. Beginning with the description of the system function and structure, it focuses on how to construct the application system with ARMBoot, ARMLinux and Qt-Embedded from open source projects. The authors illustrate the whole procedure of integrating all these components.

Key words: ARMBoot; ARMLinux; Qt-Embedded; integration

信息技术迅猛发展,个人数字助理、掌上电脑、机顶盒等基于嵌入式系统的数字产品成为市场热点。随着集成电路制造工艺的不断提高,以 ARM[1]系列为代表的嵌入式微处理器架构,为这些嵌入式系统的构建提供了高性能、低功耗、稳定可靠的硬件平台,一批优秀的嵌入式操作系统,如 Vxwork、Neculeus、Palm OS 和 Windows CE 等[2]也应运而生。但这些商业化的专用操作系统,其高昂的价格令许多生产低端产品的小公司望而却步。同时,Linux[3]以其开放源码、容易定制和扩展、多硬件平台支持和内置网络功能等优良秉性,逐渐成为嵌入式系统的研究热点和广泛使用的系统平台,以 Sharp Zaurus[4]系列为代表的嵌入式 Linux 手持设备受到了广大用户的青睐。

作者所在项目组应 PDA 硬件制造商合作要求,在 GPL 许可协议下,整合开放源码组织提供的项目源码,构建了一个基于 ARM 处理器的掌上信息处理终端,集个人数字助理应用、网络应用、多媒体应用于一身,并成功运行在 Intel DBPXA250 Lubbock 实验开发板上。在没有任何软件花费的情况下,为最终的合作产品提供了一个完整的应用系统原型,并为开发目标系统作好了设计思路和实现技术的充足准备。

1　应用系统概述

应用系统原型的构建选用了 Intel 的 Lubbock 实验开发板作为硬件平台,它使用新一代为无线手持式应用产品开发的 XScale 微处理器架构,相对其前一代产品 StrongARM[5],XScale[6]具有更好的性能、更低的功耗和更高的可靠性,其处理器芯片 PXA250 的最高主频可达 400MHz,高档的核心部件配置和完备的外围设备支持,使其成为价格不菲却又物有所值的 ARM 实验开发板。

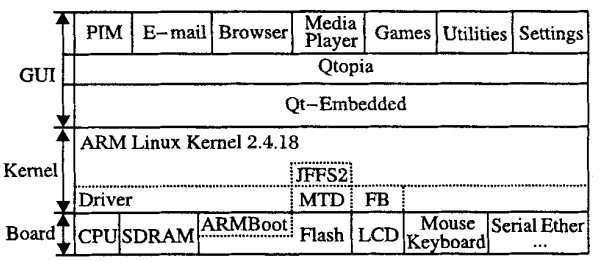

图 1　系统结构图

项目组的工作目标就是在上述硬件平台上,由下至上构建所有的软件系统,从而得到目标系统的完整原型。整个系统由三个主要部分组成:1)引导装载程序(BootLoader),是一小段驻留在开发板上的代码,它在实验板加电后首先被执行,对 CPU、内存等主要资源进行初始化后,完成内核映象文件的装载和引导。本系统使用小巧灵活的 ARMBoot[7]作为引导装载程序。2)Linux 内核,由 Russell King 主持的开放源码项目 ARMLinux[8],将 Linux 移植到 ARM 平台上,为所有基于 ARM 体系构建的系统提供了一个可运行的内核,其中还包括了开发板外围设备的驱动和嵌入式文件系统的建立。3)图形用户界面(GUI),系统采用基于 Qt-Embedded[9]的

收稿日期:2003 - 06 - 23

作者简介:郭胜超(1978 -),男,江苏镇江人,硕士研究生,主要研究方向:计算机中文信息处理;　吕强(1965 -),男,江苏苏州人,教授,主要研究方向:计算机操作系统、分布式计算、计算语言学;　杨季文(1963 -),男,江苏无锡人,教授,主要研究方向:计算机中文信息处理;钱培德(1948 -),男,江苏无锡人,教授,博士生导师,主要研究方向:计算机中文信息处理.

Qtopia[10]作为桌面环境,这一专为手持设备定制的 GUI 系统,为掌上信息处理提供了强大的网络、媒体工具和简洁明了的集成环境。系统结构如图 1 所示。

系统使用 Linux 内核的 MTD (Memory Technology Device)驱动接口,在 Flash 设备上建立 JFFS2 物理文件系统,作为信息处理和存储的载体。系统的显示输出则是通过 Linux 内核的 FB(Frame Buffer)机制实现,它为 Qt-Embedded 的输出部分提供了一个抽象的图形设备,系统只要在 FB 定义好的结构框架下实现具体显示控制器的驱动函数,GUI 系统就可以不用关心具体硬件的驱动,而直接使用 FB 框架定义的接口完成显示输出功能。

2　应用系统构建

整个系统各部分代码都是遵照 GPL 许可协议从开放源码组织免费下载,在对各部分代码分别进行交叉编链、配置修改和运行调试后,再将每个部分有机整合起来,得到最终的应用系统。整个系统的构建依赖于一个强大的宿主机开发环境,包括一个针对目标平台的交叉编译器、一个功能强大的调试器、宿主机的开发资源及其与实验开发板的通信能力。项目使用一台 Redhat 7.3 和 Windows 2000 双系统 PC 作为开发主机,可以使用以太网口、串口、并口与 Lubbock 开发板通信。虽然系统各部分的构建会需要不同的工具,但对 ARM 平台的交叉编译器贯穿整个系统构建的全过程,手工整合各编链工具建立交叉编译环境是一件比较繁琐的事情,本项目在 Redhat 7.3 中安装了一个整合好的 2.95.3 版 ARM 交叉编链环境,其中包括了 GNU 的 C、C++ 编译器、链接器等工具,只需将下载的工具包解压至/usr/local/arm,并据此将交叉编译器所在路径加入系统环境变量 PATH 中即可。Windows 2000 系统则用来运行必要的 Windows 平台工具,如 Lubbock 实验板上用来在裸机状态下烧写 Flash 的工具 JFlash。

2.1　引导装载程序 ——ARMBoot

2.1.1　ARMBoot 概述

ARMBoot 是一个针对 ARM 平台设计和实现的系统引导装载程序,该项目源于嵌入式 PowerPC 平台的 PPCBoot,秉承了 PPCBoot 小巧灵活、容易定制、功能丰富的特点,加之良好的可移植性和强大的网络支持,ARMBoot 已成为 ARM 平台上理想的引导工具。

ARMBoot 对于各种基于 ARM 体系结构的主流平台都给出了相应编链配置,从而可以将 ARMBoot 分别编链到这些硬件平台运行,基于 XScale 的 Intel Lubbock、采用 ARM720T 处理器核的 EP7312 等多种 ARM 开发平台,均可以使用 ARMBoot 作为 Monitor/BootLoader。对于具体功能的配置选项,ARMBoot 对应每个具体的硬件平台都有一个.h 文件以宏定义的方式给出,一系列常量和以 CFG-、CONFIG-开头的宏定义决定了 ARMBoot 的程序功能和参数,给使用者最大的选择余地和配置灵活度。

2.1.2　ARMBoot 的编链与使用

下载 ARMBoot 稳定版本 1.1.0 的源码包,解压得到 armboot-1.1.0 目录,其中的 board 和 cpu 目录分别包含了与具体处理器和开发板相关的文件,include/configs 中存放着 ARMBoot 具体功能的配置文件。结合项目所用 Lubbock 开发板,分别对应于 board/lubbock、cpu/xscale 目录中的文件和 include/configs/config-lubbock.h。项目要求 ARMBoot 在实验板加电以后,自动将安排在 Flash 中的内核映像加载到内存执行,据此,需要对 config-lubbock.h 中的下列几个关键选项进行相应配置。

```
# define CONFIG-COMMANDS
   (CONFIG-CMD-DFL & ~CFG-CMD-NET)
# define CONFIG-BOOTDELAY 3
# define CONFIG-BOOTCOMMAND "bootm 0x00040000"
# define CFG-LOAD-ADDR 0xa0200000
```

CONFIG-COMMANDS 是 ARMBoot 配置功能选项的一个重要宏,上例中的(CONFIG-CMD-DFL & ~CFG-CMD-NET)排除了 ARMBoot 对网络功能的支持,去掉了主要在开发调试阶段使用的 bootp、tftp 等网络支持,最终 ARMBoot 目标文件只有 60 KB,而同样是运行在 Lubbock 上的另一个 BootLoader,RedBoot 的典型大小是 130 KB。

CONFIG-BOOTDELAY 定义了 ARMBoot 自动加载运行指定程序的方式,在上例中,ARMBoot 运行后会等待 3 秒钟,如果用户不在主机的串口终端向目标板发送按键事件,它就会自动运行 CONFIG-BOOTCOMMAND 中定义的命令,否则 ARMBoot 将给出命令提示符,让用户通过串口在主机与其交互;若 CONFIG-BOOTDELAY 定义为 0,则 ARMBoot 启动后直接去运行 CONFIG-BOOTCOMMAND 指定命令;若 CONFIG-BOOTDELAY 的值小于 0(常用 - 1),则 ARMBoot 会关闭自动运行给定指令的功能,启动后给出提示符,等待用户的交互命令。

有了上述自动运行给定命令的机制,系统只要将 Linux 内核安排在开发板 Flash 的 0x00040000 地址处,定义 CONFIG-BOOTCOMMAND 为 bootm 0x00040000,并定义 CFG-LOAD-ADDR 为 0xa0200000,ARMBoot 启动后便会自动将 Flash 地址 0x00040000 处的内核加载到内存地址 0xa0200000 处,然后让系统从该地址执行,至此,ARMBoot 便完成了其引导装载的使命。

安装并设置好交叉编译环境,便可以使用 make lubbock-config; make all 对 ARMBoot 进行编链,从而得到 ARM 平台 ELF 格式的目标文件 armboot,以及 armboot.bin 和 armboot.srec 两种目标文件,使用 JFlash 工具将 armboot.bin 下载至 Lubbock Boot ROM 的 Block 0(0x00000000),因为 ARM 体系处理器加电后,最初的 PC 寄存器的值就是 0x00000000,所以 ARMBoot 首先获得系统控制权,对 CPU、存储器和串口等必要硬件初始化后,就可以正常工作了。

值得注意的是,上述内核加载过程中的内核映像在下载至 Flash 中之前,要用 armboot-1.1.0/tools/mkimage 工具进行一次封装,给内核映像 zImage 加上大小、类型、CRC 校验等信息,在装载时再由 armboot 根据封装的信息进行拆封,以确保正确无误地进行装载引导。本系统的做法是在内核文件的 arch/arm/Makefile 中加入:

```
aImage: zImage
   ($ARMBOOT)/tools/mkimage-A arm-O linux-T kernel-C none-a
   0xa0008000-e 0xa0008000-n "Linux"-d arch/arm/boot/zImage
   aImage
```

在编链内核时,使用 make aImage 得到 ARMBoot 要求的内核映像文件 aImage,最终下载至 Flash 地址 0x00040000 处的并不是 zImage 而是 aImage。

2.2　系统内核 ——ARMLinux

2.2.1　内核的配置与编链

系统采用 2.4.18 版的 ARMLinux 内核,该内核是在 Linux

官方的 2.4.18 内核的基础上改写了与 CPU 体系相关的代码,使之可以很好地运行在 ARM 体系的处理器上,并加入相应平台的硬件驱动,从而得到完整的应用系统内核。就 Lubbock 开发板来说,首先从 Linux Kernel 官方站点下载 linux-2.4.18.tar.gz,然后从 ARMLinux 的 FTP 或其镜像站点得到 patch-2.4.18-rmk7 和 diff-2.4.18-rmk7-pxa3。其中,前者就是将 Linux 移植到 ARM 平台的代码补丁;后者是本项目所用 Lubbock 开发板的外围设备的驱动补丁。解压内核文件包并依次打上两个补丁,就得到了供配置编链的内核文件目录树。

在/usr/src/arm/linux/arch/arm/def-configs 中存放着很多特定硬件平台的内核配置文件,其中的 lubbock 文件就存放了本系统的内核配置选项。依次运行 make lubbock-config; make oldconfig; make dep; make aImage 对内核进行配置和编链。其中,前两条 make 命令会根据 lubbock 文件的选项对内核文件作相应配置,然后用 make dep 检查依赖关系,正常编链 ARMLinux 应使用 make zImage,但据 2.1.2 的叙述,此处要用 make aImage 编链得到 ARMBoot 所需格式的内核映像。

2.2.2 文件系统的挂载

嵌入式 Linux 文件系统可以通过 JFFS2、CRAMFS、RAMFS 等形式实现,其中,在 Flash 设备上通过内核 MTD 机制实现的 JFFS2 文件系统,以其强大稳定的存储管理功能,成为手持信息处理终端的首选物理文件系统。

1) ARMLinux 内核的 JFFS2 支持

需要结合应用系统的具体情况对内核文件作相应的配置和调整。首先,在 2.2.1 节提到的内核配置文件 lubbock 中打开 JFFS2 的相关功能支持,包括 CONFIG-MTD、CONFIG-MTD-LUBBOCK 和 CONFIG-JFFS2-FS 在内的一系列配置选项,初始的 lubbock 配置文件已经对这些选项作了完备的配置;其次,在/usr/src/arm/linux/drivers/mtd/maps/lubbock.c 中配置 Flash 设备的分区信息,本系统建立了如图 2 所示的 Flash 分区表。

```
#define WINDOW_ADDR 0x00000000
#define WINDOW_SIZE 32*1024*1024
static struct mtd_partition lubbock_partitions[] = {
  {
      name:          "Bootloader",
      size:          0x00040000,
      offset:        0,
      mask_flags:  MTD_WRITEABLE
  },{
      name:        "Kernel",
      size:        0x00100000,
      offset:      0x00040000,
  },{
      name:        "Filesystem",
      size:        0x01e00000,
      offset:      0x00140000
  }
};
```

系统使用 0x00000000 处的 32M Boot ROM Flash 作为存储设备

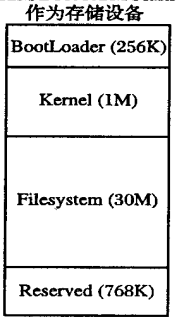

图 2 Flash 分区结构图

Lubbock 使用 Intel StrataFlash 存储器,该 NOR 型 Flash 的每个 Block 大小为 256K,而 Block 又是其最小擦写单位,故上述各分区的大小都必须是 256K 的整数倍。

2) JFFS2 根文件映像

有了上述分区结构以后,在内核参数中加上 root=/dev/mtdblock2,则内核在启动时会去 Boot ROM Flash 的 0x00140000 偏移处挂载 JFFS2 作为系统的根文件,故系统要在该偏移处安排 JFFS2 的映像文件。在/mnt/jffs2 下建立具体的根文件目录结构,将必要的配置文件、系统程序、应用程序和相关库文件分别复制到相关目录中,从而得到根文件的具体内容。通常的做法是,从相关嵌入式 Linux 站点下载 ARMLinux 根文件系统的 RAMDISK 文件,用 mount 命令将该文件解至临时目录,然后复制临时目录中的文件目录至/mnt/jffs2(dev 目录不能复制,要用 mknod 命令手工建立需要的设备文件),就得到了基于 BusyBox[12] 实现的 ARMLinux 系统根文件,再加入应用程序(Qt-Embedded)的相关文件,便得到了完整的文件系统内容。最后,用 JFFS2 工具 mkfs.jffs2 将/mnt/jffs2 中的内容打包为一个目标文件系统的映像文件,并将该映像下载至 Flash 的 0x00140000 处,从而为内核的启动做好了根文件挂载的准备。

2.3 图形用户界面——Qt-Embedded

2.3.1 Qt-Embedded 概述

Qt-Embedded 是著名的 Qt 库开发商 Trolltech,针对嵌入式 Linux 系统推出的基于 Qt 库构建 GUI 系统和应用的 C++开发包。Qt-Embedded 的移植性较好,支持 ARM、PowerPC、MIPS、Dragonball 等多种 CPU 体系,只要有相应平台的 C++编译器,几乎所有可以运行 Linux 的设备都可以使用 Qt-Embedded 构建 GUI 系统;由于 X11-Qt 与 Qt-Embedded 系统核心 API 大致相同,故 Linux 桌面系统中基于 X11-Qt 的应用程序可以很方便地移植到 Qt-Embedded 上;C++语言和基于 Signal/Slot 的对象间通信机制,使面向对象的程序设计方法在 Qt-Embedded 系统开发中得到了很好的应用;QMake、QVFB、Qt-Designer 等众多强大开发工具的支持,大大提高了 Qt-Embedded 系统开发的效率;如今的 Qt-Embedded 也开放了源代码,使得开发人员可以在 GPL 许可协议下自由地使用 Qt-Embedded 进行嵌入式 Linux 应用系统的开发。

2.3.2 GUI 系统的具体构建

为了加快和方便嵌入式 Linux 应用系统的开发,Trolltech 根据掌上设备典型的应用需求,在 Qt-Embedded 的基础上开发了 Qtopia,这个专为嵌入式 Linux 系统设计的桌面系统集成了 PIM、Internet、Game 在内的应用套件,为系统用户提供了良好的使用和交互环境。

1) 程序的交叉编链

整个 GUI 系统的构建需要对 Qt-Embedded、Qtopia 和 Konqueror-Embedded[11] 三者依次分别编链,然后有机整合在一起,从而共同构成目标系统的 GUI 部分。Qt-Embedded 是 Qtopia 的底层支持,GUI 系统的图形库和窗口组件都由 Qt-Embedded 实现;Qtopia 则是由各类在 Qt-Embedded 基础上开发的应用程序所组成的桌面集成环境,它就是最终系统的用户交互界面;对于典型的 Internet 应用——Browser,Konqueror-Embedded 以其开放、小巧的特点成为首选方案,它是 KDE 桌面系统浏览器 Konqueror 在嵌入式 Linux 系统中的版本,主流的 Web 特征在 Konqueror-Embedded 都有相应支持,但因为这个程序比较独立,而且相对其它 Qtopia 应用比较大,所以并没有被包含在 Qtopia 源码包中,需要单独编链后集成到 Qtopia 中。

下载 qt-Embedded-2.3.4、qtopia-free-1.6.0 和 konqueror-embedded-snapshot-20021229 的源码压缩包,可分别参考其中的文档,根据需求对各程序配置和交叉编链。为了加速程序界面的设计,Qt 提供了 Qt Designer 工具帮助开发人员高效地生成程序的图形界面。Qt Designer 用可视化设计的方法让开发人员方便地生成程序地用户界面文件(*.ui)。在程序

编链的时候,首先要调用 uic(User Interface Compiler)对该文件进行预编译,然后才能为 C++编译器正常编链。另外,由于 Qt 程序采用 Signal/ Slot 的特殊代码实现对象间通信和消息响应机制,所以在编译这些代码的时候,需要调用 Qt 工具包中的 moc(Meta Object Compiler)对程序进行预编译,将 Signal/ Slot 代码转成标准 C++代码,再用 C++编译器进行真正的编链。所以在对上述文件包进行编链之前,要确认这两个 Compiler 在 $QTEDIR/ bin 中。此处使用的 qt-Embedded-2.3.4中就没有带 uic,在 Qt-X11 的文件包中可以很容易地找到这个工具。

在上述交叉编链的过程中,引起编链错误最多的原因就是找不到相关的库文件,如果能下载到现成的 ARM 平台链接库的二进制文件,可以很容易解决这类问题,否则就要下载相应库文件的源代码,然后交叉编译得到所需的库文件。

2) 目标文件的组织

在成功完成上述交叉编链过程后,就要把编链得到的应用程序、动态链接库和相关配置文件、资源文件按一定的目录结构组织在/ mnt/ jffs2 中,为得到上述系统根文件做好 GUI 系统部分的准备。

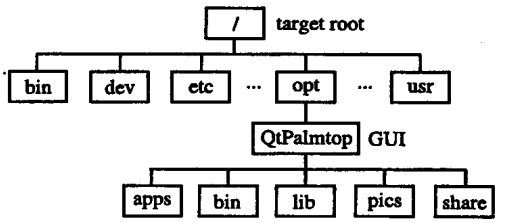

图 3　文件系统组织图

图 3 为下载到 Flash 中的 JFFS2 的文件系统结构。其中,根目录下除 opt 以外的文件目录都由前面所述 RAMDISK 获得,/ opt/ QtPalmtop 则用来安排 Qtopia 桌面系统的相关文件。要让包括 Konqueror-Embedded 在内的 Qtopia 运行在开发板上,在 QtPalmtop 下至少要建立图 3 所示的 5 个目录。apps、bin、pics 目录分别用来存放应用程序的桌面配置文件、可执行代码和图标资源文件,这三个目录可从 qtopia-free-1.6.0的编链目录中复制得到;lib 目录则用于存放 libqte.so.2.3.4、libqpe.so.1.5.2、libqtopia.so.1.6.0 等动态链接库和系统字体文件,该目录可从 qt-Embedded-2.3.4 目录中复制;对于 Konqueror-Embedded 的安装,可以在./ configure 时加入 -prefix = / mnt/ jffs2/ opt/ QtPalmtop 选项,然后在成功编链后使用 make install 来实现,该命令会将 Konqueror-Embedded 运行所需文件安装到 QtPalmtop 的相应目录中,其中 share 目录中安装了 Konqueror-Embedded 的一些配置和应用文件。这样,整个目标系统的根文件就在/ mnt/ jffs2 下组织完毕,再使用 mkfs.jffs2 创建文件系统映像,并将映像文件下载至 Flash 指定地址,从而就为应用系统的运行作好了文件系统的准备。

经过前面的准备工作,ARMBoot、Kernel、Filesystem 就已经按上述分区结构安排在 Boot ROM Flash 中了。此时,Lubbock 开发板就可以脱离开发主机独立地运行整个应用系统。开发板一加电,ARMBoot 首先获得系统控制权,对 CPU、内存等进行初始化后,将自身搬进内存执行,运行以后把 Flash 0x00040000 处的 aImage 载入内存,较验拆封后得到内核映像 zImage,并将之存储于内存的 0xa0200000 处,最后跳转至该地址将系统控制权交给内核;内核自解压后开始系统的启动,经过一系列的系统初始化,内核会在启动的最后阶段去 Flash 的 0x00140000 处挂载根文件,然后运行/ sbin/ init 完成用户空间的系统初始化,最后则是/ bin/ tinylogin 给出系统登录界面;进入系统以后,设置 QTDIR = / opt/ Qtpalmtop 等环境变量,然后就可以运行./ qpe 启动 Qtopia 桌面环境了,运行效果如图 4 所示。

(a) Qtopia桌面截屏图　　(b) Qtopia中Konqueror效果图

图 4

3　结束语

本文具体描述了一个嵌入式 Linux 手持信息处理终端在 Intel DBPXA250 Lubbock 实验板上的构建过程,为类似的系统开发提供了一种思路。虽然系统的各主要部分都免费使用开源组织的成果,但这决不意味系统的构建就是简单和缺乏价值的。根据应用需求修改和配置各部分开源代码,然后将其有机整合起来,共同构成一个完整的应用系统,也是一项具有挑战性的工作。此外,将开源成果有效地应用在实际系统的构建中,不但体现了各开源项目的工作意义,而且对 Linux 在嵌入式系统中的应用有着积极的促进作用。但在开发最终的目标系统时,应考虑系统性能和硬件成本之间的平衡,使产品具备较好的性价比。

参考文献

[1] 马忠梅. ARM 嵌入式处理器结构与应用基础[M]. 北京: 北京航空航天大学出版社, 2002.

[2] 唐寅. 实时操作系统应用开发指南[M]. 北京: 中国电力出版社, 2002.

[3] 邹思轶. 嵌入式 Linux 设计与应用[M]. 北京: 清华大学出版社, 2002.

[4] Sharp Zaurus[EB/ OL]. http://www.zauruszone.com/, 2003-01.

[5] 陈章龙. 嵌入式系统——Intel StrongARM 结构与开发[M]. 北京: 北京航空航天大学出版社, 2002.

[6] Intel XScale[EB/ OL]. http://www.intel.com/ design/ pca/ applicationsprocessors/, 2002-08.

[7] ARMBoot[EB/ OL]. http://sourceforge.net/ projects/ armboot/, 2003-03.

[8] ARM Linux[EB/ OL]. http://www.arm.linux.org.uk/, 2002-09.

[9] Qt-Embedded[EB/ OL]. http://www.trolltech.com/ products/ embedded/, 2003-01.

[10] Qtopia[EB/ OL]. http://www.trolltech.com/ products/ qtopia/ index.html, 2003-02.

[11] Konqueror-Embedded[EB/ OL]. http://konqueror.org/ embedded/, 2003-02.

[12] BusyBox[EB/ OL]. http://www.busybox.net/, 2002-09.