RT Embedded http://www.kontronn.com

基于 PowerPC 双核处理器嵌入式系统 U-Boot 移植

王长清[1] ,林桂竹[2]

(1. 河南师范大学 物理与信息工程学院 河南 新乡 453007 ;2. 新乡职业技术学院 数控技术系 ,河南 新乡 453002)

摘　要:借助于 ELDK 开发工具 ,针对我们自己开发的基于 PowerPC 双核处理器 MPC8641D 的 ATCA 架构信号处理与存储硬件平台 ,进行了 U-Boot 移植. 介绍了 U-Boot 的启动过程 ,着重阐述了 U-Boot 移植方法和步骤 ,并对交叉开发环境建立和启动过程中双核的处理进行了简要的说明.

关键词:U-Boot ;移植 ;嵌入式系统; MPC8641D ;双核处理器

中图分类号:TP316　　**文献标志码**:A

Bootloader(引导加载程序) 是操作系统运行前执行的第一段程序 ,其作用是初始化硬件设备 ,建立内存空间映射表 ,为最终调用系统内核建立适当的系统软硬件环境. 嵌入式系统一般没有通用的 Bootloader ,不同的系统硬件组成结构对应不同的代码 ,因而 ,针对特定的硬件平台 ,需要对 Bootloader 进行移植. 现在 ,已经进入了多核时代 ,在一些高性能的嵌入式系统中 ,多核处理器也开始得到应用. 由于多核处理器的复杂性 ,对基于多核处理器的系统的 Bootloader 移植难度将大大增加.

在嵌入式系统中 ,常见的引导加载程序有 U-Boot ,Red Boot 和 BLOB 等. 其中 U-Boot 功能最多 ,灵活性最强 ,能够支持 PowerPC ,ARM ,MPIS ,X86 体系架构的上百种开发板 ,引导加载 Linux ,VXWorks ,QNX 等多种操作系统 ,有丰富的开发调试文档资料与强大的网络技术支持. 针对我们自己开发的基于 PowerPC 双核处理器 MPC8641D 的信号处理与存储硬件平台 ,借助于 ELDK 开发套件 ,选用 U-Boot-1.2.0 版本源码 ,修改源程序 ,进行了 Bootloader 开发移植.

1　硬件平台简介

基于 PowerPC 双核处理器 MPC8641D 的信号处理与存储系统的组成如图 1 示.

系统采用飞思卡尔公司高性能 PowerPC 双核处理器 MPC8641D. 该处理器是在 e600 Power Architecture 内核和 PowerQUICC™片上系统(SoC) 基础上开发的,采用两个运行频率最高 1.5 GHz 的 e600 核,每个核都有自己受 ECC 保护的 1MB 后端 L2 缓存,提供 AltiVecR 128 位矢量处理引擎,集成了 IIC ,UART ,Local Bus , DDR2 ,千兆以太网 ,Serial Rapid IO 和 PCI Express 接口[1].

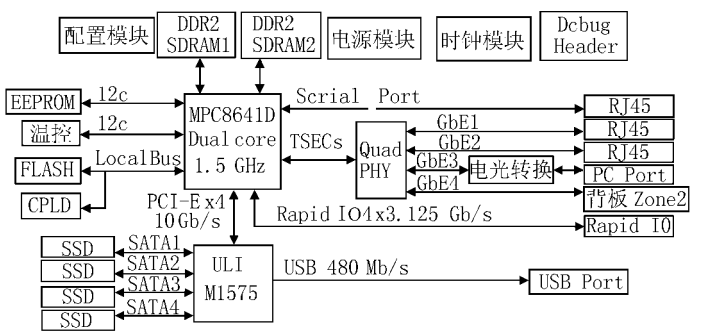

图 1　信号处理与存储硬件平台原理框图

系统主要实现信号大规模高速数据处理与存储功能 ,采用有 ATCA 板卡形式 ,提供了实施数据处理与存储管理的设备和多种高速数据接口. FLASH 用于存放内核镜像文件、文件系统、应用程序和备份数据 ;4

RT Embedded http://www.kontronn.com

个 SATA2 接口的固态电子盘组成 RAID0 阵用于数据的大容量高速存储;SDRAM 用作内存;x4 Serial Rapid IO、USB2.0 、3 个集成千兆以太网口和 1 个光纤接口用于数据的输入和输出;此外还有 RS232 串口、I2C 以及 JTAG 支持等.

2 U-Boot 相关源码分析

移植 U-Boot 工作就是添加修改与新硬件相关的代码和配置选项,然后配置编译,所以,在移植 U-Boot 过程中,除了要熟悉硬件电路外,还要熟悉 U-Boot 相关源码. 下面的 U-Boot 相关源代码分析针对 U-Boot-1.2.0 版本支持的 MPC8641HPCN 开发板.

2.1 U-Boot 相关源码结构

U-Boot 是遵循 GPL 版权协议的自由软件,具有和 Linux 源文件相似的目录结构. 在相关目录下分别存放不同的源码,这些源码是通过 GCC 和 Makefile 组织编译的,可以很方便地在不同的硬件平台上进行移植. 顶层目录下的 Makefile 设置开发板的定义,递归地调用各级子目录下的 Makefile,编译相关的程序链接成 U-Boot 映像. 源码结构可查看顶层目录下的 read me 文件,其中与移植相关的主要文件夹如表 1 所示.

表 1 移植相关的 U-Boot 源码顶层目录说明

目录	特性	说明
board	平台依赖	目录下的每个文件夹对应支持的目标板
cpu	平台依赖	目录下的每个文件夹对应支持的 CPU 架构
lib_xxx	平台依赖	存放 xxx 体系架构通用的文件,对本项目相关的为 lib_ppc
include	通用	头文件和开发板的配置文件
common	通用	通用多功能函数实现
lib_generic	通用	通用库函数的实现
net	通用	网络程序
drivers	通用	通用设备驱动程序

2.2 U-Boot 代码启动过程

当 MPC8641D 上电或者施加复位信号时,CPU 通过读取硬复位配置信号,启动 Local bus 存储控制器 CS0 # (对应于 Flash 的片选信号) 有效,选中 Flash,CPU 地址线上输出硬件复位中断向量对应的地址 0xFFF00100,开始读第 1 条指令,从/board/mpc8641hpcn/U-Boot.lds 连接脚本文件可以看到,这条指令对应于/cpu/mpc86XX/start.S 中的_start:标号处. 下面简要介绍启动过程中的几个关键函数和文件.

(1) 运行 start.S(/cpu/mpc86XX/start.S)

系统入口从_start:标号处执行,完成 CPU 本身基本的初始化. 主要是初始化 CPU 内部寄存器的一些状态,清 HID0、MSR,无效 BATS 和 L2CACHE,根据配置文件中 BAT 信息设置指令和数据 BAT,设置 CCSRBAR 地址,Local Access Windows,时钟比率,时基寄存器,(L2,数据和指令) Cache,初始化堆栈,初始化 GOT 表等.

(2) CPU 的底层初始化(/cpu/mpc86XX/cpu_init.C)

从 start.S 中跳转到函数 cpu_init_f() 处,进行 CPU 的底层初始化,首先对堆栈清零,主要设置了 GPCM 存储控制器.

(3) 板卡的第一次初始化(/lib_ppc/board.c)

从 start.S 中跳转到函数 board_init_f() 处,该函数实现板上的第一次初始化和一些硬件测试. 尤其是 RAM 初始化,并分配内存空间,保存板子的信息,准备好 RAM 中重定向代码.

(4) 搬运代码到内存中(/cpu/mpc86XX/star.S)

从函数 board_init_f 跳到/cpu/mpc86XX/start.S 中的 relocate_code() 函数处,然后将代码搬至 SDRAM 工作,调整 GOT 表,做一些重定位后开始在 RAM 中运行代码.

(5) 板上的第二次初始化(/lib_ppc/board.C)

在 relocate_code() 函数后将跳转到 board_init_r() 函数处执行第二次初始化,主要完成一些数据结构、

RT Embedded http://www.kontronn.com

高端模块及系统设备的相关初始化.

(6) 命令的解析与执行(/commom/main.C)

U-Boot 会执行函数 board_init_r()中的 main_loop()函数,即监控程序,该函数在/Commom/main.c 中.函数 main_loop()会根据用户从控制台的输入,调用 commom/main.C 中的 run_command()函数完成命令的解析,并转去执行相应的处理函数.

2.3 U-Boot 代码启动过程中双核的处理

在 bootloader 阶段,大多数的多核系统的启动过程都由一个处理器完成,其它处理器处于待命状态.在 Powerpc 平台规范中,负责启动的处理器称作主处理器,其余的称为从处理器.在 SMP(对称多处理系统)系统中,MPC8641D 的核 0 作为主处理器,完成操作系统的初始化后激活核 1,并做一些基本初始化后,跳转到 Linux 复位向量 0x100 处[2].

3 构建交叉开发环境

嵌入式系统及应用软件的开发一般采用主机——目标机模式,通常主机和目标机的体系架构和操作系统不同,需要构建交叉开发环境.交叉开发环境的构建主要有两方面的工作:交叉开发工具链的建立和配置主从开发调试环境[3].

我们的开发主机是 X86 架构的 Linux 操作系统(windows 系统下,用 VMware 软件安装 Linux Fedora 6.0 虚拟机),目标平台是 PowerPC 架构的 Linux 操作系统.交叉开发环境中,主机和目标机接口连接如图 2 所示:

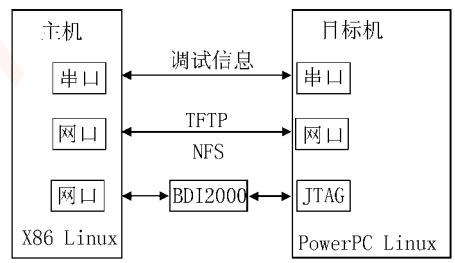

图 2 主机和目标机接口连接图

3.1 交叉开发工具链的建立

在主机上针对目标机编写的程序,需要进行交叉编译后才能在目标机上运行.自己编译建立嵌入 linux 的交叉开发工具链有点烦琐,这里我们利用德国 DENX 软件中心提供的 ELDK 开发工具,在主机上,配置并安装支持 mpc8641d 的嵌入式交叉开发工具链(Gcc3.4.3, Glibc2.3.3, Binutils 2.15)主要步骤如下:

```
$./install -d /opt/ELDK ppc_74xx      //安装 GNU e600 交叉开发工具
$ export CROSS_COMPILE=ppc_74xx-
$ export PATH=/opt/ELDK/usr/bin:/opt/ELDK/bin:$PATH //配置主机交叉开发工具链
```

3.2 主从开发调试环境配置

在主机开发环境下生成的目标码可以通过串口、以太网或 JTAG 下载到目标机,系统及应用程序在目标机运行,用户可以使用 GDB 或 BDI2000 调试运行在目标机上的程序,调试信息通过串口或以太网进行交互.在主机,需要配置串口通信程序 SecureCRT,主机和目标机 IP 地址(注意目标板的 IP 地址应和 TFTP 服务器的 IP 地址要在同一个网段内),设置 TFTP 服务器和 NFS 服务器.

4 U-Boot 的移植

我们开发的基于 PowerPC 双核处理器 MPC8641D 的信号处理与存储硬件平台和 MPC8641HPCN 开发板硬件结构相近,并且 MPC8641HPCN 开发板已经获得 U-Boot-1.2.0 版本支持,所以移植的过程主要参考 U-Boot-1.2.0 中 MPC8641HPCN 开发板相关代码,降低移植难度和缩短移植时间.

移植 U-Boot 到新的开发板上仅需要修改与硬件相关的部分即可.主要包括两个层面的移植,第一层是针对 CPU 的移植,第二层是针对 Board 的移植.由于 U-Boot-1.2.0 里面已经包含 MPC8641D 的支持,所以对新的开发板的移植主要是针对 Board 的移植.U-Boot 的移植的步骤说明如下:

(1) 在顶层 Makefile 中为开发板添加新的配置选项

//ppc 对应 PowerPC 架构,mpc86xx 对应源码/cpu/mpc86xx 目录,atca_8641d 对应/board/atca_

RT Embedded http://www.kontronn.com

8641d 目录(我们的新开发板文件).

ATCA_8641D_config:unconfig

@$(MKCONFIG) $(@:_config=) ppc mpc86xx atca_8641d

(2)在 U-Boot-1.2.0 开发包目录 board 下创建新开发板文件目录 atca_8641d,并拷贝 mpc8641hpcn 目录下相应文件,并修改其中文件名.参考 mpc8641hpcn 板文件,增加相应其他目录下相关文件.

(3) 在 include/configs 目录下添加修改新板配置头文件 atca_8641d.h,直接拷贝 include/configs 目录下 mpc8641hpcn.h 文件,并根据新开发板硬件参数,修改 local bus,DDR2,FLASH,MAC,CCSRBAR,PHY 等参数.由于使用内存颗粒构建内存系统,需要根据内存颗粒构数据手册,仔细检查设置 DDR 控制参数否则系统将不能正常启动.系统地址范围参数的设置可参考表 2.

(4)修改 init.S 文件,设置 7 个 local access windows 参数,CCSR 不需要设置,具体参数设置参考表 2.

(5)修改 atca_8641d.c 文件的 fixed_sdram()函数,增加对第二个 DDR2 控制器的内存颗粒支持.部分关键代码如下,其中的 DDR 配置参数 CFG_DDR2_xxx 在 atca_8641d.h 中定义.

//定义第二个 DDR2 控制器的控制寄存器组指针

```
volatile ccsr_ddr_t *ddr2 = &immap->im_ddr2;
```

//设置第二个 DDR2 控制器的相关寄存器

```
ddr2->cs0_bnds = CFG_DDR2_CS0_BNDS;
ddr2->cs0_config = CFG_DDR2_CS0_CONFIG;
ddr2->ext_refrec = CFG_DDR2_EXT_REFRESH;
ddr2->timing_cfg_0 = CFG_DDR2_TIMING_0;
ddr2->timing_cfg_1 = CFG_DDR2_TIMING_1;
ddr2->timing_cfg_2 = CFG_DDR2_TIMING_2;
ddr2->sdram_mode_1 = CFG_DDR2_MODE_1;
ddr2->sdram_mode_2 = CFG_DDR2_MODE_2;
ddr2->sdram_interval = CFG_DDR2_INTERVAL;
ddr2->sdram_data_init = CFG_DDR2_DATA_INIT;
ddr2->sdram_clk_cntl = CFG_DDR2_CLK_CTRL;
```

(6) U-Boot 有启动加载和下载两种工作模式,其中参数的调整可以在 atca_8641d.h 中实现.启动加载模式也称为自主模式,即 U-Boot 从目标机上的某个固态存储设备上将操作系统自动加载到 RAM 中运行,这种模式是 U-Boot 的正常工作模式.下载模式就是在开发或生产过程中,U-Boot 通过网络连接等通信手段从主机下载操作系统内核和文件系统等到目标机的 RAM 中,然后再写到目标机上的 FLASH 类固态存储设备中.U-Boot 允许用户在这两种工作模式间进行切换,系统启动时会延时等待一段时间,如果这时用户没有按键,U-Boot 就默认进入启动加载模式.可以根据内核、跟文件系统和 DTB 文件的位置,以及跟文件系统种类,在 atca_8641d.h 中进行相关参数的调整设置.下面是启动有启动加载工作模式的部分关键参数设置:

表 2 系统内存映射分配表

起始地址	终止地址	设备类别	容量
0x0000_0000	0x1FFF_FFFF	DDR2	512M
0x2000_0000	0x3FFF_FFFF	DDR2	512M
0x8000_0000	0x9FFF_FFFF	PCI1/PEX1 MEM	512M
0xC000_0000	0xDFFF_FFFF	Serial Rapid IO	512M
0xF000_0000	0xF00F_FFFF	CCSR	1M
0xF010_0000	0xF01F_FFFF	cpld	1M
0xE200_0000	0xE2FF_FFFF	PCI1/PEX1 IO	16M
0xF800_0000	0xFFFF_FFFF	Flash	128M

//设置延迟后,自动执行 CONFIG_RAMBOOTCOMMAND 命令.

```
# define CONFIG_BOOTCOMMAND CONFIG_RAMBOOTCOMMAND
```

//设置根启动设备,控制终端参数,内核映像、根文件系统映像和 DTB 文件的地址.

```
# define CONFIG_RAMBOOTCOMMAND \
  "setenv bootargs root=/dev/ram rw " \
        "console=$consoledev,$baudrate $othbootargs;" \
        "bootm $kernel_addr $ramdisk_addr $dtb_addr"
```

(7)配置,编译,最终生成了 U-Boot.bin 映像文件.

RT Embedded http://www.kontronn.com

5 U-BOOT 的调试与部署

嵌入式系统的 bootloader 移植是嵌入式系统开发的一个难点,尤其对于象 MPC8641D 这样启动过程复杂的双核高端处理器,不利用象 BDI2000 这样的开发调试器,进行 U-BOOT 的调试,想成功非常困难. 在前面的交叉开发环境的建立中,已经介绍 BDI2000 的连接,要想 BDI2000 能够启动控制开发板,需要对 BDI2000 中的开发板启动参数进行正确的配置,具体参考硬件设计信息.

在 U-Boot 的启动过程串口初始化之前无法获得字符串提示信息,利用 BDI2000 进行 U-BOOT 的调试. 在串口的初始化之后,便可通过向串口打印信息来实时跟踪所启动的执行流程,以了解程序目前执行的具体部分或运行到哪一个阶段出现了问题.

利用 BDI2000 进行 U-BOOT 的调试时,需要注意的是调试代码的地址. 可以利用 powerpc-linux-objdump 命令得到反汇编代码中得到相关代码的地址. 调试 reloacate 之前的代码, 代码运行在 FLASH 中,可以从反汇编代码得到,reloacate 之后的代码地址,进行了线性搬移,可以利用 BDI 的单步跟踪调试, 或者设置断点跟踪调试.

U-Boot 在交叉开发环境下移植完成并成功测试后,通过 BDI2000 把 U-Boot 映像烧写到 FLASH 相应位置,FLASH 烧写地址必须与硬件所确定的硬件复位向量相吻合,对于 MPC8641D 而言,烧写地址是 0xfff00100.[4] U-Boot 经过部署后,目标机能够在本地引导内核,加载文件系统了.

6 结束语

U-Boot 是一个功能强大的 bootloader 开发软件,适用的 CPU 平台及支持的嵌入式操作系统很多. 针对我们开发的基于 PowerPC 双核处理器 MPC8641D 的 ATCA 架构信号处理与存储硬件平台,已经成功完成了 U-Boot 移植,并顺利实现了 linux-2.6.23 内核的引导和 Ramdisk 文件系统的挂载,硬件获得正确驱动并稳定运行. 本文不仅对于 PowerPC 双核高端处理器处理器平台的 U-Boot 移植和系统设计,有很好的借鉴作用. 由于 U-Boot 移植基本的方法和步骤是相同的,希望对不同的 CPU 和开发板的设计人员有所帮助.

参 考 文 献

[1] Freescale Semiconductor Corp, MPC8641D Integrated Host Processor Family Reference Manual[Z]. America: Freescale Semiconductor Corp,2008.

[2] 李相国,杨树元. 基于 PowerPC 处理器 SMP 系统的 U-BOOT 移植[J]. 微计算机应用,2008,29(9):95-99.

[3] 孙纪坤,张小全. 嵌入式 Linux 系统开发技术详解 —基于 ARM[M]. 北京:人民邮电出版社,2006:73-86.

[4] Freescale Semiconductor Corp. Programming Environments Manual for 32-Bit Implementations of the Power-PC Architecture[Z]. America: Freescale Semiconductor Corp,2005.

U-Boot Porting for Embedded System Based on Dual_core Processor of PowerPC

WANG Chang-qing[1], LIN Gui-zhu[2]

(1. College of Physics and Information Engineering, Henan Normal University, Xinxiang 453007, China;
2. Numerical Control Vocational & Technical College, Xinxiang Technology College, Xinxiang 453002, CHina)

Abstract: The paper introduces the U-Boot porting process for embedded system based on dual-core processor - MPC8641D. The main flow of U-Boot is introduced at first. How to porting U-Boot is described in detail, and a cross development and processing progress of Dual_cores Processor in U-Boot are discussed briefly.

Key words: U-Boot; porting; embedded system; mPC8641D; Dual_core processor

Created in Master PDF Editor

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
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用

Creator Master PDF Editor

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
22. 基于 VxBus 的高速数据采集卡驱动程序开发

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

Created in Master PDF Editor

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
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings

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
17. Windows CE 环境下无线网卡的自动安装

PowerPC：

Create by Master PDF Editor

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
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植

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
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现

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

Created with Master PDF Editor

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
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究

Created in Master PDF Editor