基于 UEFI 的国产计算机平台 BIOS 研究

周　洁，谢智勇，余　涵，吴荣泉

(中国电子科技集团公司第三十二研究所，上海 200233)

摘　要：传统 BIOS 的局限性给计算机技术的发展造成严重的阻碍。为此，通过对比 PMON 和 UEFI BIOS 的特点，阐述龙芯国产计算机平台上应用 UEFI BIOS 的优点，描述在国产计算机平台上基于 EDKII 框架实现 UEFI BIOS 的原理和方法，该实例体现出 UEFI 标准化、模块化、方便移植、易于扩展等特点，为国产计算机平台的国产 BIOS 研究提供新的方法。

关键词：龙芯处理器；国产计算机平台；PMON 固件；统一可扩展固件接口；EDKII 框架；DXE 驱动

Research on Domestic Computer BIOS Based on UEFI

ZHOU Jie, XIE Zhi-yong, YU Han, WU Rong-quan

(No.32 Research Institute of China Electronics Technology Group Corporation, Shanghai 200233, China)

【Abstract】The limitation of the legacy BIOS is a serious setback for the development of Computer technology. This paper compares the characteristics of the PMON with the UEFI BIOS, proposes the advantage of the UEFI BIOS based on domestic computers. The theories and method of implementing this UEFI BIOS with the EDKII are described later. This example of UEFI BIOS reflects that UEFI is standard, modularized, and easy to be transplanted and extended. It provides new method for researching the BIOS of domestic computers.

【Key words】Loongson CPU; domestic computer platform; PMON firmware; Unified Extensible Firmware Interface(UEFI); EDKII framework; DXE drivers

1　概述

传统 BIOS 有着诸多的弊端，使得系统维护代价大，阻碍了国内计算机技术的发展。UEFI 提供了标准化的 BIOS 规范，为计算机系统的扩展和升级提供了方便，同时也使得系统引导更为迅速、对操作系统引导器以及操作系统本身的限制大大减少。

随着龙芯等国产高性能处理器芯片的研发成功及稳定性增强，国内陆续展开了许多基于国产 CPU 的计算机系统研究开发工作。计算机系统中 BIOS 是连接硬件和软件的关键组件，也是系统安全性验证的重要环节，然而长期以来，BIOS 技术基本被国外几家厂商掌控，给计算机系统的安全埋下了安全隐患。UEFI 已经在国外诸多知名计算机厂商中得到广泛的认可和使用，它的推广和应用也使得国产 BIOS 的研究和开发成为可能，国产计算机平台上的 UEFI BIOS 研究是国产 BIOS 发展的要求，也是提高计算机系统安全性的需要，基于国产计算机平台的 UEFI BIOS 研究具有十分重要的意义。基于此，本文研究基于 UEFI 的国产计算机平台 BIOS。

2　龙芯国产计算机平台简介

本文的硬件平台是一款自主研发的龙芯 2F 处理器计算机平台，其逻辑结构如图 1 所示。龙芯处理器自带内存控制器，支持最高容量 2 GB 的 DDR2 内存，处理器通过 PCI 总线与南桥、以太网控制器、显卡相连，向外提供最多 3 个 PCI 设备扩展。通过南桥向外提供 2 路 RS232 接口、4 路 USB 接口、1 路数字音频接口、1 路 IDE 接口和 1 路 LPC 接口；通过以太网控制器向外提供两路千兆以太网口；通过显卡提供 LVDS 和 VGA 2 种显示接口。Flash 通过 Local BUS 与 CPU 连接。该平台原先使用 PMON 作为 BIOS。PMON 是 MIPS 处理器最常使用的 BIOS，最新版本为 PMON2000，支持 MIPS、ARM、PPC 和 X86 体系结构，可以从 Flash、IDE、网络以及 USB 启动操作系统，包含调试系统，支持多种调试命令，使用串口作为输出。

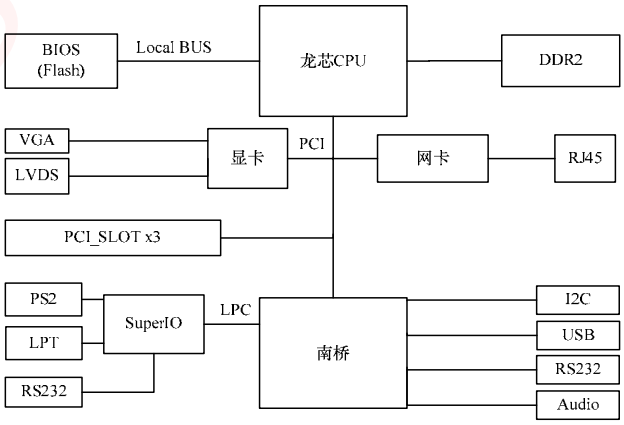

图 1　龙芯国产计算平台逻辑框图

3　UEFI BIOS 的优势

可扩展固件接口(Extensible Firmware Interface, EFI)首先由 Intel 为解除传统 BIOS 对安腾处理器体系结构性能的限制而提出，后交由国际标准化组织 UEFI 管理，改名为统一可扩展固件接口(Unified Extensible Firmware Interface, UEFI)。

UEFI BIOS 的执行流程如图 2 所示，它一般包括 SEC (security)、PEI(pre-EFI Initialization)、DXE(Driver Execution Environment)、BDS(Boot Device Select)4 个阶段，SEC 执行系统基本初始化，准备 C 语言执行环境；PEI 阶段进入 C 代码环境，描述系统资源和初始化信息，结束后传递给 DXE 阶段；DXE 阶段对计算机系统设备进行初始化和配置，构建

系统表，提供对资源的访问接口；BDS 阶段为 BIOS 引导的最后阶段，完成进入操作系统引导前的准备工作，最终加载 OS Loader，系统控制权交给 OS Loader，仅保留运行时服务可为系统使用[1-2]。至此，UEFI BIOS 对系统的控制结束。

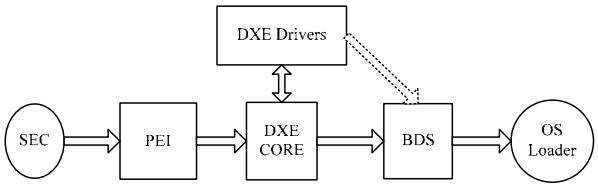

图 2　UEFI BIOS 代码执行流程

UEFI 具有模块化结构、C 语言风格、EFI 驱动模型硬件操作方式等优点。下面分别从可移植性、开机速度、可扩展性、操作界面 4 个方面对 PMON 和龙芯国产计算机平台的 UEFI 固件两者进行比较：

(1)开机速度

PMON 会对它发现的所有挂到主板上的设备进行驱动加载和执行，因此挂载设备的多少直接影响到系统的引导启动速度；UEFI 设备驱动遵循 UEFI 驱动模型，DXE 阶段不会对该类型的驱动进行更多的初始化工作，直到 BDS 中调用 EFI_DRIVER_BINDING_PROTOCOL 中的 start()将该驱动与设备控制器连接并初始化设备[3]，在 BDS 阶段设置界面中可对这些设备进行选择以及调整启动顺序，某些启动时不需要的设备可以推迟到后阶段进行初始化和配置，从而提高系统引导速度。

(2)可移植性

PMON 与 UEFI BIOS 一样由 C 语言编写，使用轮询方式发现设备驱动。PMON 中与平台硬件相关的部分分布不规则，移植工作量较大，且要求开发者对代码结构了解程度高；UEFI 的模块化结构以及标准的硬件操作接口使得驱动与核心控制代码之间耦合度低，移植时只需修改这些标准接口的实现，而不影响其上层或者其他平行模块，所以，UEFI 可移植性要远优于 PMON。UEFI 的标准硬件操作接口即一组 DXE Architectural Protocols[2]，它们屏蔽底层硬件细节，使得 UEFI BIOS 具有高可移植性。

(3)可扩展性

PMON 中添加驱动或者功能时需在初始化流程中添加对相应设备或者功能的调用，编程者需要十分熟悉整个代码的结构和调用位置；UEFI BIOS 中设备驱动符合 EFI Driver Model，BIOS 核心代码会发现并加载驱动，不需要编程者关注核心代码。同时 UEFI 对 PCI 总线体系结构的支持也体现了其良好的可扩展特性。UEFI PCI 总线驱动采用分层结构，从下至上包括 PCI Root Bridge、PCI BUS 和 PCI 设备驱动三层[1]，关于总线的大部分初始化和配置工作在硬件相关的 HostBridge 驱动中实现，PCI 设备驱动程序的实现与平台 PCI 体系结构无关，总线结构发生改变时，也仅需对 PciHostBridge 进行改变。

(4)操作界面

PMON 通过 shell 命令行提供大量的命令和功能使系统与用户进行交互，包括网络通信、读写内存、在线烧写 Flash 等存储设备等等；UEFI BIOS 在 BDS 阶段可进入 shell 界面，与 PMON 不同的是，用户能自定义 UEFI 应用程序，它们不影响固件核心程序，在 shell 中加载运行，图形界面中还可选择加载 UEFI 应用程序对系统进行检测或者修复工作，或者管理系统信息、配置启动选项，UEFI shell 是一种应用程序。

国产计算机平台的 UEFI BIOS 是国产化 BIOS 发展的需要，针对国产计算机平台的不同结构和应用场合，其 BIOS 的实现和维护也更加方便。固件运行阶段可使用外部设备，并拥有图形化设置界面，支持从除 Flash 之外的位置加载固件模块，同时操作系统的启动位置和方式的选择也更加灵活。还可以开发功能丰富的 UEFI 应用程序，操作系统中某些功能或者机制的实现可以在固件中实现，例如固件层电源管理功能[4]、固件层数据备份和恢复[5]等，解决了这些关键机制过分依赖于操作系统的问题，使得国产计算机系统更为可靠。

4　固件设计与实现

本文 BIOS 的实现以 EDKII 框架为基础，EDKII 是遵循 UEFI 规范的开源框架，它包含了大量开发示例和基本的底层库函数，这些代码中一部分与平台体系结构无关，另一部分则与硬件结构关系密切。实现 BIOS 首先分析出必须实现的模块，属于 EDKII 通用部分的模块则直接使用，平台相关部分的则必须针对硬件细节进行修改，最后添加 EDKII 中没有提供参考实现或者实例的功能模块，组成一个能在平台上运行并且使平台能正常工作的代码集合。

遵循 UEFI 模块化的特点，除 EDKII 中直接被使用的通用代码外，其余修改或者自定义编写模块包含在一个独立的包(package)中，它与 ArmPkg、BeaglePkg 等目录平行，并按照执行流程进行划分。BIOS 的实现内容如图 3 所示，可以大致划分为 3 大部分，如图中虚线框所示，在包中体现为 3 个子目录。

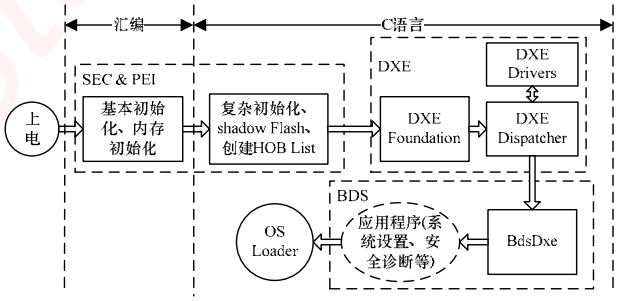

图 3　BIOS 实现内容

4.1　EDKII 简介

EDKII 中各基本模块的依赖关系如图 4 所示，其主要模块的功能如下：

(1)BaseTools：包含代码编译所需的二进制编译工具集和编译环境配置文件。

(2)MdePkg：包含各个平台通用的基本的底层库函数、协议和工业标准，各种平台架构的 UEFI BIOS 都可以在模块中引用这些库函数，有效减少了开发工作量。

(3)MdeModulePkg：包含一系列各平台通用的模块,其中包括 MdePkg 中公共库的应用模块示例。

(4)Conf：保存编译环境信息、编译目标路径以及编译器参数，工具将在该路径下产生 3 个配置文件。

(5)EdkShellPkg、ShellPkg：提供一个平台通用的 UEFI Shell 应用程序开发环境。

(6)EdkFatBinPkg：包含针对不同 CPU 架构的原始 FAT 驱动。

(7)Nt32Pkg：一个在 Windows 操作系统下可加载 32 位模拟器，提供 UEFI 运行环境的平台。UnixPkg 是 Linux 操作系统下的模拟环境。

(8)ArmPkg、ArmPlatformPkg：针对 ARM 平台的实现，与具体平台硬件相关。相应的 ArmRealViewPkg、EmbeddedPkg、Omap35xxPkg 分别是在 Arm RealView、嵌入式、Omap35xx 平台上的参考实现。

(9)NetWorkPkg、UefiCpuPkg：网络、CPU 驱动参考实现。

(10)DuetPkg：提供基于传统 BIOS 运行环境的支持库。

(11)OptionRomPkg：提供针对不同 CPU 架构编译 PCI 兼容映像的示例。

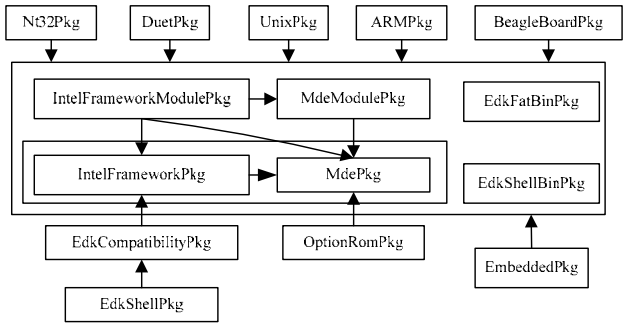

图 4　EDKII 主要模块关系依赖图

4.2　固件实现

固件以模块为单位，每个模块实现若干功能的集合。各模块之间的依赖关系在模块的配置文件(.inf)中进行定义，并通过包中的配置文件 DEC(package declaration file)、DSC(build description file)和 FDF(Flash Description File)选择需要编译到 BIOS 中的代码、对 PCD 和变量赋值描述内存空间等资源的分配以及描述二进制代码在 Flash 上的布局。本文使用 0x80000000~0x90000000 的一段空间作为系统内存，其分配情况如图 5 所示。代码编译时，编译器解析配置文件将指定的代码按照指定的结构生成二进制可执行文件。

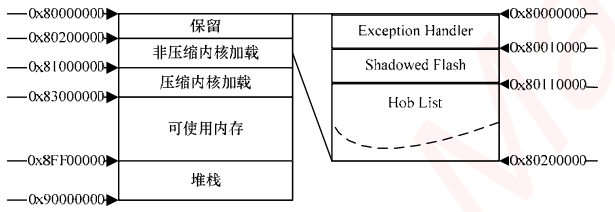

图 5　系统内存分配

4.2.1　SEC 和 PEI

SEC 和 PEI 是 UEFI BIOS 中最先执行的 2 个阶段，UEFI 中关于 PEI 的设计类似于一个微型 DXE，它执行的操作应该尽量的少。本文中将 SEC 和 PEI 的功能作为一个模块实现，不采用规范中的 PEI 的(加载器-PEI 模块)结构，该模块分为 2 个部分。

(1)系统上电后的执行起点，负责对系统硬件进行检测以及系统内存、cache、堆栈空间的基本初始化，为之后 C 语言代码准备执行环境。调试时通常要使用串口输出信息，龙芯国产计算机平台通过南桥向外提供串口，因此该阶段还需初始化北桥、南桥、串口。这部分采用 MIPS 汇编语言实现，完成基本的初始化之后调用第 2 阶段的 C 语言部分入口函数。

(2)使用 C 语言实现，其任务包括将 BIOS 拷贝到内存以加快执行速度(shadow Flash)，硬件的复杂初始化，PCI 地址空间映射，构建 HOB List 并将其交给 DXE。本文不实现标准 PEI 结构，而是使用 EDKII 中 MdePkg/Library/BasePeCoffLib 以及 EmbeddedPkg/Library/PrePiLib 中提供的关于二进制映像、HOB List、内存的操作来完成 PEI 的主要功能。最后，通过过渡函数查找并调用 DXE 入口函数。

4.2.2　DXE

DXE 阶段负责系统引导中主要初始化工作，分为核心和驱动两部分。核心创建各种服务和接口供其他模块使用，驱动则提供对硬件的操作、初始化设备以及实现对各种机制的支持。

(1)核心

DXE 核心建立在一系列对硬件的抽象接口之上，不依赖于 CPU、芯片组等特定平台相关细节，其实现没有平台差异。EDKII/MdeModulePkg/Core/Dxe 是 DXE 核心模块，包括 DXE Dispatcher 和 DXE Foundation，分别负责按顺序加载驱动和创建各种服务。在保证 DXE 核心所必须的功能完整的情况下，根据需求对其中某些部分进行增、删、改，如添加 debug 代码，即可作为固件的 DXE 核心部分。

(2)驱动

DXE 驱动是固件实现的重要部分，它可以是平台硬件和外部设备的驱动程序，或者支持特定功能和机制的模块。它分为两种类型：早期 DXE 驱动和 UEFI 驱动模型驱动[2]。对于不同的平台，驱动的类型、数量均有差异，可以由程序员重新编写，也可以通过修改 EDKII 中包含的大量驱动的参考实来获得需要的设备驱动，从而节省编码时间，规范代码风格使其便于阅读。

首先分析平台体系结构，实现对 CPU、芯片组、计时器等板级设备以及总线等平台底层硬件操作的驱动，添加固件内部实现机制支持的驱动，这些驱动将创建 DXE 架构协议，提供抽象接口。这部分驱动属于早期 DXE 驱动，它们最终产生出所有的 DXE 服务、启动服务、运行时服务，后续执行的程序使用这些服务进行平台底层操作，不直接与硬件交互。其次根据需要连接的外部设备确定要添加的设备驱动程序，这些模块遵循 UEFI 驱动模型。键盘、鼠标、显卡、网卡以及其他 PCI 设备驱动都通过这种方式加载。

本文硬件平台总线包括 PCI、USB、IDE。遵循 UEFI 总线驱动的风格，它们的驱动都分为 3 个层次，包括控制器驱动、BUS 驱动以及设备驱动，当总线体系结构发生改变时，只需对控制器驱动层进行适配。PCI 驱动主要实现 PCIHost Bridge、PCI BUS 驱动。EDKII 的 PcatChipsetPkg/PciHostBridgeDxe 下是 PC-AT 结构的 PciHostBridge 驱动的参考实现，不同的 PCI 架构扫描 PCI 总线时是不一样的，需针对平台具体 PCI 架构进行适当修改。MdeModulePkg/Bus/PciBusDxe 是 PCI BUS 驱动的实现，通常来说该驱动程序所做的工作与平台无关，不需要修改，除非是要对设备的配置空间做一些修正。USB 驱动包括控制器和 USB BUS，IDE 驱动包括 IDE 控制器和 IDE BUS 驱动，控制器与硬件相关，USB 控制器需针对平台细节编写，PcAtChipsetPkg/Bus/Pci/IdeControllerDxe 是 IDE 控制器的驱动实现，MdeModulePkg/Bus/Usb 是关于 USB BUS 的驱动实现，IntelFramework/Bus/Pci/IdeBusDxe 是 IDE BUS 的驱动实现。

该硬件平台板级设备包括龙芯 CPU、芯片组、计时器、定时器、Flash 等，固件内部支持 PCD 机制、容错写机制、HII(Human Interface Infrastructure)以及 GPT 分区格式，DXE 驱动包含对以上内容支持以及产生架构协议的驱动。EDKII/MdeModulePkg/Universal 包含一组平台通用的驱动，包括 WatchdogTimerDxe、Metronome、Variable 等，同时

EmbeddedPkg 中包含对 RealTimeClock 和 Reset 驱动的实现，它们与硬件关系密切，针对平台硬件进行修改后即可加入该固件中。EDKII/MdeModulePkg/Universal/Disk 下的模块提供对 GPT 的支持，同时参考 FatBinPkg 的代码实现 Fat 文件系统驱动，或者实现其他文件系统驱动，如 NTFS[6]等。MdeModulePkg/Universal/PCD 实现 PCD 机制的功能和操作，MdeModulePkg/Universal/下 HiiDatabaseDxe 和 SetupBrowserDxe 对 Hii 进行支持。除此之外，其余驱动则需要根据主板硬件编写，如 Flash、南桥等，实现对这些硬件操作的抽象接口即协议。

外部设备驱动程序由平台上连接的外部设备数量和种类决定。控制台提供给用户基本的系统输入输出信息，MdeModulePkg/Universal/Console 中包含提供控制台服务的 UEFI 驱动。若目标平台要求支持网络引导或者要求图形显示，固件还应该分别实现该平台所使用网卡和显示芯片的驱动程序。如果平台使用了 PCI、USB 或者其它类型的设备，并且要在 BIOS 环境中使用这些设备,也必须实现相应的设备驱动程序。

4.2.3 BDS 架构

BDS 架构协议是引导进入 BDS 阶段的标准接口，由 BdsDxe 驱动模块产生。该阶段代码参考 EDKII/IntelFrameworkModulePkg/Universal/BdsDxe 实现，它在 DXE 阶段被发现并加载，DXE 结束时将调用其入口函数。

BDS 阶段可进入图形界面进行系统管理或者配置启动选项，用户还可以创建 UEFI 应用程序对系统进行安全检测、诊断系统问题或者提供其他安全保障方案。

5 调试及安装运行

EDKII 的 MdePkg/Libarary 中提供了 Debug 库用于 BIOS 开发调试，它们可以格式化打印输出代码的调试信息，便于对代码中的错误进行定位和分析。通过串口输出调试信息，还需根据平台串口配置编写相应的 Debug 库方法。

代码编译得到二进制 FD 文件之后，烧写到 IO 地址空间 0xBFC00000 所对应的 Flash 地址(在龙芯平台上，系统上电时从该地址寻找第一条执行指令)。系统上电后，BIOS 引导至所有初始化工作完毕，可选择直接进入 OS Loader 引导操作系统或者进入系统设置界面，在系统设置界面中，可以对系统进行诊断、启动设置、参数查看、日期设置等，或者进入 shell 界面，通过 shell 命令查看系统信息、设置参数、从指定路径手动安装新的设备驱动、应用程序或者从选择的启动设备引导操作系统。

6 结束语

本文分析对比了 PMON 与 UEFI BIOS，简单介绍 UEFI BIOS 的引导过程，以开源框架 EDKII 为基础，描述了龙芯国产计算平台的 UEFI BIOS 实现内容和方法。随着国产处理器技术和基于国产处理器的计算机研究及应用的发展，UEFI 的开放性、可扩展性、可移植性、标准接口编程以及快速开机等优点使其在国产计算机平台上具有广阔的应用前景。固件技术的发展将提高和促进计算机技术水平及其安全性，UEFI BIOS 在国产计算平台上的应用有着重大的意义。该 BIOS 实现了 UEFI 固件的基本功能和操作，运行良好，在之后的研究中其功能还有待进一步的完善。同时，UEFI 的特点也给 BIOS 带来了新的安全问题，这些工作将在今后的工作中得到改进，比如在其中加入可信支持[7]等。

参考文献

[1] Unified EFI Inc.. Unified Extensible Firmware Interface Specification2.3[Z]. 2011.

[2] Unified EFI Inc.. Platform Initialization Specification1.2[Z]. 2010.

[3] 崔 莹, 辛晓晨, 沈钢纲. 基于 UEFI 的嵌入式驱动程序的开发研究[J]. 计算机工程与设计, 2010, 31(10): 2384-2387.

[4] 唐 笛. UEFI BIOS 电源管理研究和应用[D]. 上海: 上海交通大学, 2011.

[5] 王晓箴, 于 磊, 刘宝旭. 基于 EDK2 平台的数据备份与恢复技术[J]. 计算机工程, 2011, 37(15): 262-264.

[6] 彭 舰, 谢 纲. 可扩展固件接口的 NTFS 文件系统驱动[J]. 计算机工程, 2008, 34(9): 83-85.

[7] 周振柳, 李 铭, 翟伟斌, 等. 基于 UEFI 的可信 BIOS 研究与实现[J]. 计算机工程, 2008, 34(8): 174-176.

~~~~~~

(上接第 354 页)

由上述分析可知，虽然增加传感器数量能够提高精度，但是成本也随之提高。根据传感器数量与精度关系，确定了适合于森林防盗系统中安装的传感器数量。

5 结束语

本文提出森林防盗系统目标定位方法。以声学传感器网络的声源定位为实验基础，分析传感器数量对声源定位与跟踪性能的影响，利用极大似然法对声源位置进行估计。实验结果表明，增加传感器数量能够提高精度。由于传感器布局对声源的估计影响关系大[9-10]，因此今后需要研究传感器布局与声源定位精度之间的关系。另外，实验采用 200 m×200m 的方形区域，在区域面积和形状改变之后，其定位精度的稳定性也是需要研究的内容。

参考文献

[1] 吴成东. 基于禁忌搜索的无线传感器网络多源定位研究[J]. 东北大学学报: 自然科学版, 2010, 31(5): 609-612.

[2] 唐蒙娜, 熊伟丽, 徐保国, 等. 无线传感网对雷达干扰的优化配置研究[J]. 计算机工程, 2011, 37(7): 81-83.

[3] Li Xinrong. Collaborative Localization with Received-signal Strength in Wireless Sensor Networks[J]. IEEE Transactions on Vehicular Technology, 2007, 56(6): 3807-3817.

[4] Boukerche A, Oliveira H A. Localization Systems for Wireless Sensor Networks[J]. IEEE Wireless Communications, 2007, 14(6): 6-12.

[5] 陈积明. 基于声强的无线传感器网络目标跟踪方法研究[J]. 电子与信息学报, 2009, 31(11): 2791-2794.

[6] 匡兴红, 邵惠鹤. 一种新的无线传感器网络节点定位算法研究[J]. 传感技术学报. 2008, 21(1): 174-177.

[7] 贾子熙. 无线传感器网络中一种分布式声源定位方法[J]. 系统仿真学报, 2009, 21(20): 6552-6555.

[8] Sheng Xiaohong, Hu Yuhen. Maximum Likelihood Multiple-source Localization Using Acoustic Energy Measurements with Wireless Sensor Network[J]. IEEE Transactions on Signal Processing, 2005, 53(1): 44-53.

[9] 魏雅川, 梁 彦, 陈延军, 等. 无线传感器网络自适应声音目标定位算法[J]. 传感技术学报, 2010, 23(3): 418-422.

[10] 李石坚, 廖备水, 吴 健. 面向目标跟踪的传感器网络设计、实现和布局优化[J]. 传感技术学报, 2007, 20(12): 2622-2630.
~~~~~~

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
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究

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
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现

25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现

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
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
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现

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
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统

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
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
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植

13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计

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
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现

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
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
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法