RT Embedded http://www.kontronn.com

基于 UEFI 的可信 BIOS 研究与实现

周振柳[1,2]，李　铭[3]，翟伟斌[1]，许榕生[1]

(1. 中国科学院高能物理研究所计算中心，北京 100049；2. 沈阳航空工业学院计算机学院，沈阳 110034；
3. 中国电子科技集团信息化工程总体研究中心，北京 100083)

摘　要：分析固件基本输入输出系统(BIOS)的安全需求，定义了可信 BIOS 概念。基于 UEFI 规范和可信计算机制设计 UTBIOS 体系结构。UTBIOS 的实现以新一代符合 UEFI 规范的 BIOS 产品为基础，使用可信测量根核对 BIOS 运行和系统引导过程中各部件进行可信测量，构建操作系统运行前的可信链，讨论可信测量对 BIOS 引导过程的性能影响。

关键词：可信计算；可信测量；基本输入输出系统

Research and Implementation of Trusted BIOS Based on UEFI

ZHOU Zhen-liu[1,2], LI Ming[3], ZHAI Wei-bin[1], XU Rong-sheng[1]

(1. Computing Center, Institute of High Energy Physics, Chinese Academy Scences, Beijing 100049;
2. School of Computer Science, Shenyang Institute of Aeronautical Engineering, Shenyang 110034;
3. Center of Information System Architecture Research, China Electron Technology Group Corporation, Beijing 100083)

【Abstract】 This paper analyzes security threats of firmware BIOS, and defines the concept of trusted BIOS. The architecture of UTBIOS, which is based on UEFI specification and trusted computing mechanism, is developed. To construct Pre-OS chain of trust, CRTM embedded in UTBIOS is used to measure the trustworthiness of entities in different phases of bootstrap. Implementation of UTBIOS based on UEFI BIOS product is described and the performance of trusted measurement is analyzed.

【Key words】 trusted computing; trusted measurement; Basic Input Output System(BIOS)

1　概述

建立在操作系统级别上的传统信息安全机制已不能满足信息安全技术发展需求，计算机系统安全需要进一步延伸到固件层甚至硬件层。固件 BIOS 系统作为计算机处理器最早执行的软件，其安全可信直接影响整个计算机系统的安全度。BIOS 安全问题，已成为信息安全领域关注和研究的新热点。

传统 BIOS 的设计没有考虑安全问题，存在较多的安全隐患。William A.Arbaugh 在 1997 年提出一种计算机安全引导架构 AEGIS[1]。AEGIS 基于 IBM PC 的传统 BIOS，采用认证的方法保障固件 BIOS 代码的完整性，增强 BIOS 引导过程中代码的安全保护。Dexter Kozen 在 1998 年提出一种基于语言证明的方法[2]，在编译过程中检查代码控制流安全性、内存访问安全性、堆栈操作安全性以增强代码安全，并将该方法用于对固件 BIOS Open Firmware 中的恶意代码检测[3]。这些方法只是对已有传统 BIOS 架构和产品的局部安全修补。

2003 年 Intel 向外界公布了其开发制订的新一代 BIOS 规范：EFI(Extensible Firmware Interface)，并推出可用的 EFI BIOS 产品。为推动 EFI 规范的发展，Intel 联合业界采用开源方式，共同制定推出 UEFI 规范[4]和符合 UEFI 规范的开源 BIOS 框架基础代码。我国信息产业部通过电子信息产业发展基金项目促进和支持国内 BIOS 研发单位与 Intel 合作，研发国产化安全 BIOS。国产化安全 BIOS 的研制，对于构造完整的信息产业链和保障计算机系统底层安全具有重大意义。

2　BIOS 系统安全威胁

计算技术的发展和应用需求导致目前存储 BIOS 系统固件的芯片普遍采用 Flash 芯片。Flash 芯片的使用为 BIOS 系统带来 2 个突出变化：(1)存储 BIOS 系统的芯片容量快速增加；(2)在操作系统环境下能够以纯软件方式读写 BIOS 固件系统。这 2 个变化导致针对 BIOS 系统的安全威胁增加，攻击 BIOS 系统的技术逐步成熟，使第三方恶意者向 BIOS 中植入恶意代码、病毒、木马等成为可能。

文献[5]描述了一种在 BIOS 中实现 Rootkit 恶意代码的方法，互联网上也在局部范围内出现向某些版本 BIOS 系统内植入木马的 BIOS 系统组合攻击工具包。攻击者还可以破坏 BIOS 代码的完整性，通过修改 BIOS 系统少量字节代码使得计算机系统不能正常引导和启动，达到拒绝服务攻击的目的。这类攻击的典型是 CIH 病毒及其系列变种病毒。由于 BIOS 厂商通常需要提供对 BIOS 产品的在线升级和更新，因此对存储 BIOS 的 Flash 芯片进行完全的物理写保护，禁止计算机处于运行状态下对 BIOS FLASH 芯片进行读写操作不是解决这种攻击的好办法。

由于 BIOS 系统处于计算机系统底层，因此 BIOS 系统遭受到上述威胁攻击时，会造成计算机系统的彻底崩溃，或导致计算机系统被恶意者从底层控制，从而给发现和清除这些威胁攻击带来更大的困难。安全 BIOS 系统必须能够防范上述的恶意攻击，阻止攻击者植入 BIOS 芯片中的恶意代码的执行，

基金项目：信息产业部电子信息产业发展基金资助项目；北京市教委科技发展计划基金资助项目(KM200610772006)

作者简介：周振柳(1971－)，男，博士研究生，主研方向：网络安全，可信计算；李　铭，研究员；翟伟斌，博士研究生；许榕生，研究员、博士生导师

收稿日期：2007-04-30　**E-mail：**zhouzl@ihep.ac.cn

Created with PDF Editor

RT Embedded http://www.kontronn.com

保障 BIOS 系统自身代码和数据的完整性，保证 BIOS 系统的执行代码只来自可信任的 BIOS 厂商、硬件驱动厂商等。

由于不可预知的故障或攻击导致 BIOS 系统部分完整性遭到破坏时，因此 BIOS 系统必须提供安全有效的失败自恢复机制。实施该机制的 BIOS 代码必须受到硬件保护，保证失败自恢复机制自身不会遭到破坏。

3 可信 BIOS 概念

为解决第 2 节 BIOS 系统面临的安全威胁，提出可信 BIOS 概念：如果组成 BIOS 系统的各部件在计算机引导过程中是可测量并被证明是可信的，且 BIOS 系统能够对其装载的下一环节的部件进行可信测量，则该 BIOS 系统是可信的。这里可信的含义符合 TCG 对可信的定义。

在现行计算机体系架构下，CPU 上电最早指向并执行的指令是包含在 BIOS 固件中而非 TPM 中。因此，在不改变现行计算机体系架构的情况下可信计算定义的核心测量根(Core Root of Trust for Measurement, CRTM)只能是包含在 BIOS 中的最早执行的代码部分，TPM 安全芯片只能作为可信计算的存储根核(Root of Trust for Storage, RTS)和报告根核 RTR(Root of Trust for Reporting)。因此，可信 BIOS 的结构由 2 部分构成：CRTM 部分和非 CRTM 部分。

可以将计算机引导运行过程划分为 2 个阶段：Pre-OS 和 Post-OS 阶段。Pre-OS 阶段包括 BIOS 运行、MBR 载入、OS Loader 载入、OS 内核载入过程；Post-OS 阶段则指 OS 运行后的系统服务和应用载入运行过程。在这 2 阶段运行过程中可信链的建立从 BIOS 中的 CRTM 开始，由上一级对下一级过程进行可信测量，逐级建立可信链。由于 MBR 通常只有 512 B，其自身不可能对后续的 OS Loader 部分进行可信测量，因此可把 MBR 和 OS Loader 合并为 IPL(Initial Program Loader)，由 BIOS 负责对 IPL 进行可信测量。即可信 BIOS 的功能除完成传统 BIOS 所有功能外，还要负责对可信 BIOS 的非 CRT 部分和 IPL 进行可信测量。

可信 BIOS 保障了 BIOS 自身以及 IPL 的完整性和真实性，不仅能够有效解决第 2 节中分析的 BIOS 系统面临的安全威胁，而且能够建立起 Pre-OS 阶段的信任链，使 Post-OS 阶段的信任链的建立有了信任保障。

4 UTBIOS 体系结构

根据可信 BIOS 的概念和安全需求，以新一代符合 UEFI 规范的 BIOS 产品为基础，本文设计实现了 UTBIOS，该结构模型如图 1 所示。

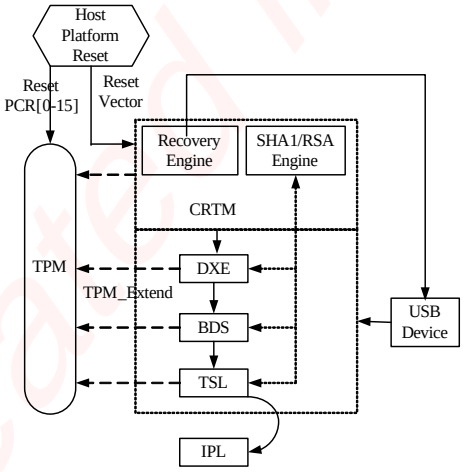

图 1 UTBIOS 体系结构

在 UTBIOS 体系结构中，采用了硬件安全芯片 TPM 作为可信测量的存储根核。TPM 中存储可信测量需要用到的各种密钥，是 UTBIOS 可信的硬件基础。

UTBIOS 固件部分由 2 部分组成：CRTM 部分和非 CRTM 部分。CRTM 是可信链建立的起始点，CRTM 是被无条件信任的，包含在 CRTM 中的功能是能够满足系统后续运行和可信测量的最小运行功能集。CRTM 提供 3 种主要功能：(1)完成 CPU、芯片组、主板、内存的初始化工作，初始化可用的堆栈，建立 C 语言程序的运行环境；(2)为后续的可信测量提供一个消息摘要和非对称加密引擎；(3)为可信测量失败或系统引导失败提供一个失败自恢复引擎。将失败自恢复引擎和用于可信测量的加密引擎设计在 CRTM 中，保证了后续的用于自恢复和可信测量过程的代码是可信且不能被恶意者破坏或修改。

UTBIOS 的非 CRTM 部分完成 BIOS 固件的其他功能。这些包括 UEFI BIOS 的 DXE、BDS、TSL 等阶段的代码和数据。UEFI BIOS 这些阶段的功能在文献[6]中有规范性描述。

系统开机上电后，首先执行 UTBIOS 的 CRTM 代码，CRTM 执行后会装载 DXE 代码并对其进行可信测量。若测量成功则继续执行，测量失败说明 DXE 阶段代码不可信(遭到修改或破坏、或来源不可信)，则 CRTM 会启动失败自恢复引擎要求用户对 BIOS 系统进行可信恢复。在后续的 DXE 装载执行 BDS、BDS 装载执行 TSL、TSL 装载执行硬盘上的 IPL 的过程中，都要调用 CRTM 提供的加密引擎对装载执行的代码进行可信测量。测量成功执行下一个环节，否则就调用 CRTM 的失败自恢复引擎做 UTBIOS 的可信恢复。

UTBIOS 体系结构中，采用 USB 块存储设备保存用于可信恢复的 UTBIOS 的可信备份。

5 UTBIOS 实现

UTBIOS 是基于新一代符合 UEFI 规范的 BIOS 产品实现的可信 BIOS。在本文研究中，中国电子科技集团信息化工程总体研究中心提供了用于研究实现的 UEFI BIOS 产品和开发环境：硬件主板采用了 Intel D945G 芯片组的 FOXCOM 主板，TPM 安全芯片采用 SINOSUN 的 SSX35B 产品。UEFI BIOS 产品几乎全部采用 C 代码实现，方便了 UTBIOS 的可信测量中消息摘要和非对称加密算法的实现。

5.1 TPM 的设置与管理

TPM 是 UTBIOS 可信测量的硬件信任基础，用于把可信测量的密钥密封存储在 TPM 中。在 UTBIOS 实现中，同时使用 TPM 的 PCR 累积存储事件日志条目的 Hash 值，对事件日志提供完整性保护。TPM 通过 LPC 接口同主板的南桥芯片组连接。UTBIOS 在 CRTM 的最初阶段对 LPC 作初始化，读取 TPM 芯片的 VID(Vendor ID)和 DID(Device ID)，确定主板上是否存在 TPM，以决定运行过程中是否使用 TPM 进行可信测量。若检测到 TPM 存在，则向 TPM 发出 TPM_Startup(ST_CLEAR)和 TPM_ContinueSelfTest 命令使 TPM 进入完全操作状态，以使 TPM 可用于后续的可信测量过程。

为方便计算机用户对 TPM 进行管理设置，UTBIOS 在 BIOS Setup 菜单中增加了对 TPM 的管理设置选项。BIOS Setup 菜单对 TPM 的设置选项包括：Enable/Disable, Activate/Deactivate, Owned/Disowned。

5.2 UTBIOS 的可信封装和可信测量

UTBIOS 使用数字签名和消息摘要技术实现对各种实体的可信测量。UTBIOS 在源代码编译生成烧写进 Flash 芯片固

RT Embedded http://www.kontronn.com

件影像的过程中，会对除 CRTM 外的其他模块进行可信封装，这个过程属于 UTBIOS 的生产过程。可信封装的主要作用是：将用于可信测量的包含消息摘要的数字签名同 UTBIOS 模块一起封装成一个固定格式的模块。可信测量则是根据封装模块中的数字签名，解签计算得到包含在签名中的消息摘要，并根据模块重新计算消息摘要，通过两者的比较，判定模块的完整性。由于公私钥具有对应唯一性，因此可同时判定模块是否来自合法的厂商。可信封装和可信测量的原理如图 2 所示。

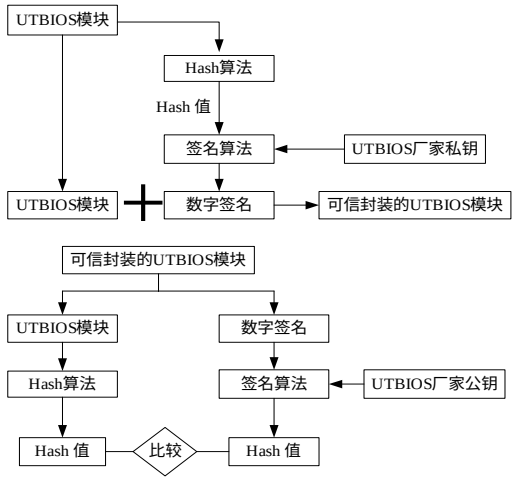

图 2　UTBIOS 可信封装和可信测量

在 UTBIOS 的实现中，目前采用的消息摘要算法为 SHA1 算法，生成 160 bit 的模块 Hash 值。签名和解签则采用 2 048 bit 的 RSA 算法。模块封装的格式采用文献[7]定义的 GUIDed Section 类型。GUIDed Section 除包含一般 Section 的头部数据外，另外包含一个自定义的 Section 头部，UTBIOS 使用这个自定义头部存放用于可信测量的数字签名信息。

如果攻击者使用自己的私钥对恶意模块进行签名，并使得 UTBIOS 在运行过程中能够使用对应的公钥进行解签校验，则会得到成功的可信测量结果，从而导致攻击者可向 BIOS 中植入恶意模块并成功运行恶意代码。为了防止此种攻击情况的出现，用于可信测量的解签公钥必须密封保存在 TPM 中进行保护。通常意义下非对称加密的公钥不需受到保护，UTBIOS 在这一点上同一般情况有所区别。

在 UTBIOS 的可信测量过程中会建立事件日志，并利用 TPM 的 PCR 累积保存事件日志条目的 Hash 值，通过 ACPI 表将事件日志传递给操作系统。用户在操作系统环境下可利用事件日志对 UTBIOS 引导过程进行重构和验证。

5.3　IPL 的处理

IPL(典型如 MBR、Grub 等)属于 UTBIOS 外部被测量的实体。对 IPL 的可信测量，其原理与 UTBIOS 内部模块的可信测量是相同的。这就要求改变现有的 IPL 产品，使其中包含用于校验的数字签名。IPL 的改造不属于本文的研究范围。

针对现有不包含完整性验证数字签名的 MBR 的可信测量，本文设计的处理方法为：在系统第 1 次可信初始化时，由 UTBIOS 自动生成一对用于 MBR 验证的 RSA 密钥。UTBIOS 计算 MBR 的 Hash 值并使用 RSA 私钥对 MBR 的 Hash 值进行签名，签名后抛弃 RSA 私钥。利用 UTBIOS 的 FLASH 的 NVRAM 数据区保存 IPL 的数字签名，利用 TPM 保护这个 RSA 公钥。第 1 次可信初始化完成后，以后每一次启动 UTBIOS 就可以实现对 MBR 进行可信测量。

5.4　可信测量对 UTBIOS 的性能影响

在 UTBIOS 中，可信测量主要是做 SHA1 消息摘要的生成和密钥为 2 048 bit 位的 RSA 解密操作。RSA 加密操作(即签名过程)需要占用大量的时间，但加密过程只在 UTBIOS 生产过程中使用，在 UTBIOS 运行过程中则只有解密操作。根据文献[8]提供的数据，在 1 GHz Pentium 机上，使用 SHA1 算法和 2 048 bit 公钥的 RSA 算法对 1MB 数据进行一次数字签名的校验过程，所需时间为 143 ms。UTBIOS 固件影像的大小目前是 1 MB，即 UTBIOS 运行过程中可信测量的数据不超过 1 MB(MBR 只有 512 B)，因此整个可信测量只须消耗 143 ms 左右的时间。本文在采用 Pentium 3.0 GHz 的实验中实际测得可信测量的累计时间为 496 ms。这包括了在 Flash 影像中读取各模块映像数据的时间。不做可信测量的 BIOS 其启动时间一般在十几秒到几十秒之间(跟实际加载和测试的设备数量有关)，实际测得 UTBIOS 关闭可信测量的启动过程为 29 s。可见可信测量对 UTBIOS 的启动速度的影响几乎可以忽略不计。

6　结束语

BIOS 的可信是计算机系统可信链的起点。UTBIOS 结合硬件可信机制，通过完整性和真实性检查保障固件 BIOS 系统的可信，解决了计算机系统 Pre-OS 阶段的可信测量和可信链建立问题。下一步的研究工作主要有以下 2 点：(1)进一步利用数字证书加强 UTBIOS 对计算机系统的信任管理，在 UTBIOS 和操作系统之间实现对信任管理的平衡；(2)实现对 IPL 可信封装和 IPL 对 OS 的可信测量。

参考文献

[1] Arbaugh W A, Farber D J, Smith J M. A Secure and Reliable Bootstrap Architecture[C]//Proc. of IEEE Computer Society Conference on Security and Privacy. Philadelphia, PA, USA: [s. n.], 1997: 65-71.

[2] Kozen D. Efficient Code Certification[R]. Ithaca, NY: Computer Science Department, Cornell University, Technical Report: 98-1661, 1998-01.

[3] Adelstein F, Stillerman M, Kozen D. Malicious Code Detection for Open Firmware[C]//Proceedings of the 18th Annual Computer Security Applications Conference. [S. 1.]: IEEE Press, 2002: 403-412

[4] The Unified EFI Forum. Unified Extensible Firmware Interface Specification Version 2.0[Z]. (2006-01-31). http://www.uefi.org.

[5] Heasman J. Implementing and Detecting an ACPI BIOS Rootkit[EB/OL]. (2006-10-30). http://www.ngssoftware.com/jh_bhf2006.pdf

[6] Intel Corporation. Intel Platform Innovation Framework for EFI Architecture Specification Version 0.9[Z]. (2003-09-16). http://www.intel.com/technology/framework/.

[7] Intel Corporation. Intel Platform Innovation Framework for EFI Firmware File System Specification Version 0.9[Z]. (2003-09-16). http://www.intel.com/technology/framework/.

[8] Menasce D A. Security Performance[J]. IEEE Internet Computing, 2003, 7(3): 84-87.

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

Creat by Master PDF Editor

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

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

Linux：

Create Master PDF Editor

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

Created with Master PDF Editor

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

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

ARM：

Created by Master PDF Editor

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

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

Created with Master PDF Editor

16. 基于 UEFI 的 Application 和 Driver 的分析与开发

Programming:

1. 计算机软件基础数据结构 - 算法

Created in Master PDF Editor