UEFI 计算机系统快速调试方法的实现

高云岭[1]，杨国栋[1]，郭元兴[2]
([1]海军 704 厂，山东 青岛 266109；
[2]中国电子科技集团公司第三十研究所，四川 成都 610041)

[摘　要] 针对目前 UEFI BIOS 计算机系统 Flash 更新困难、调试速度慢的弊端，文中提出了一种在 UEFI 下免烧 Flash 快速调试的方法。该方法在不改变 BIOS 的写保护跳线的前提下，利用 UEFI Recovery 思想，使得系统从 U 盘启动，而 U 盘中的 FVMAIN.FV 可以由开发人员根据自己的需求通过编译相应的驱动程序来产生，从而模拟 Flash 更新的效果，免除了反复烧写 Flash 的操作。同时在 Intel 945G Express Chipset 平台架构上对该方法的正确性进行了检测验证。

[关键词] UEFI；免烧 Flash；Recovery；Flash 更新
[中图分类号] TP216　　[文献标识码] A　　[文章编号] 1009-8054(2012)07-097-04

Implementation of Quick Debug Method in UEFI Computer System

GAO Yun-ling[1]，YANG Guo-dong[1]，GUO Yuan-xing[2]
([1]No.704 Factory of Navy，Qingdao Shandong 266109，China；
[2]No.30 Institute of CETC，Chengdu Sichuan 610041，China)

[Abstract] With purpose to eliminate the reprogram and restart the system，a new quick debug method based on Recovery idea is proposed to debug complex computer system without Flash update. Without any change of BIOS jumper，the system boot is started from U disk，while the FVMAIN. FV in the U-disk may be produced through compiling relevant driver in accordance with the developer's requirement，thus to model the effect of the Flash update. And this method is verified successfully on Intel 945G Express Chipset platform.

[Keywords] UEFI；without Flash update；Recovery；Flash update

0　引言

BIOS 是一组固化到计算机主板上一个 ROM 芯片中的程序，它保存着计算机最重要的基本输入输出的程序、系统设计信息、开机后自检程序和系统自启动程序。人们经常需要重新对 Flash 芯片进行编程以便能够升级 BIOS，并通过升级 BIOS 获得新的功能。

Flash 芯片以页为单位对数据进行读写操作，多个相邻的页组成一个擦除单元。在对数据页进行更新时，需要先对整个页所属的擦除单元进行擦除操作，才能在该页中写入相应的数据。每个擦除单元在达到擦除次数上限后，芯片将无法正常工作。当更新操作频繁时，大量的擦除单元将影响整个系统的整体性能。同时，Flash 的烧写是一个非常复杂并且容易出错的过程，一旦烧写了一个错误的 Flash 进入平台，往往就会毁了这个平台[1]。

基于这样的背景，文中根据当前的 UEFI 技术，在 UEFI BIOS 的 PEI 阶段，不改动 BIOS 的写保护跳线的前提下，实现类似于 BIOS Recovery 的工作，使得系统从 U 盘启动，从而可以让开发人员加载自己所需的驱动程序，免除了重新烧写 Flash 重新启动的过程。

1　UEFI 简介

2000 年，Intel 向业界展示了 BIOS 的新一代接口程序 EFI (Extensible Firmware Interface)，并将此技术应用于其安腾服务器平台上。EFI 是由 Intel 推出的一种在未来的电脑系统中用来替代 BIOS 的升级方案。2005 年，在工业界达成共识的基础上，Intel 将 EFI 规范交给了一个由微软、AMD、惠普等公司共同参与的工业联盟进行管理，并将实现该规范的核心代码开源于网站上。与此同时，EFI 也正式更名为 UEFI(Unified Extensible Firmware Interface)。UEFI 联盟将负责开发、管理和推广 UEFI 规范。

UEFI 定义了操作系统与系统硬件平台固件之间的开放接口[2]。该规范定义的接口包括平台相关信息、启动服务例程以及操作系统运行时的服务例程。操作系统

收稿日期：2012-07-04
作者简介：高云岭，1977 年生，男，硕士，工程师，研究方向：舰船计算机通信；杨国栋，1979 年生，男，工程师，研究方向：舰船信息技术；郭元兴，1980 年生，男，工程师，研究方向：计算机科学与技术。

装载器与操作系统可通过接口调用这些服务例程。UEFI 规范是一个公开的纯接口定义，它不依赖于某个特定的 BIOS 制造商或某个特定的 BIOS 的实现，仅仅定义了平台固件必须实现的接口，以及操作系统可能使用的一系列接口与数据结构，其实现的方式与细节均取决于该规范的实现者。

UEFI 规范还定义了固件驱动程序模型，使得所有遵循此模型开发的固件驱动程序能够互相协作。

不同于传统的 BIOS 实现，EFI/Tiano 基于现代软件体系设计的思想，对 UEFI Framework 采用模块化设计，并根据其执行流程主要划分为：SEC、PEI、DXE、BDS、TSL、RT 和 AL 这 7 个阶段，其运行机理如图 1 所示。

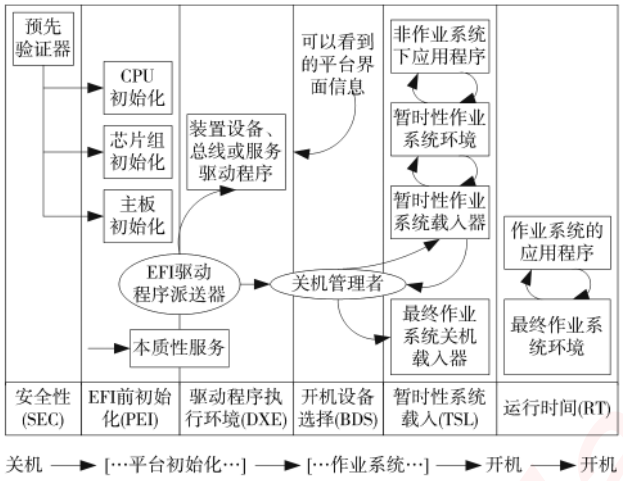

图 1 UEFI 执行流程

SEC(Security) 是平台上电后最先执行的步骤。这个阶段的主要目的是对平台固件进行验证，确保选择的平台固件映像没有被破坏。主要工作是初始化临时内存区并对平台早期初始化代码进行验证 [3]。

PEI(Pre-EFI Initialization) 阶段有两个主要任务：确定重新启动的来源和做尽可能少的工作以便寻找和初始化内存，为 DXE 阶段提供少量的固定内存 [4]。

DXE(Driver Execution Environment) 被设计来处理与外围设备的通信，它通过加载驱动的方式 (轮询检测) 为操作系统的启动管理构建环境。

BDS(Boot Device Selection) 是 UEFI 拥有平台控制权的最后一个阶段。BDS 与 DXE 阶段一起工作，为启动操作系统建立控制台。

TSL(Transient System Load)，即操作系统启动管理器尝试引导操作系统的阶段。

RT(Run Time) 是操作系统启动运行后，UEFI 提供的一组运行时服务。

AL(After Life) 阶段，即最后一个阶段，提供一种机制来保证用户在有意或者无意的情况下终止操作系统后，让 UEFI 重新获得系统控制权。

2 基于 UEFI 的 Flash 更新方法的开发

2.1 Recovery 原理

Recovery 的定义就是要保留足够的系统固件以便系统可以启动，它可以做到以下两点 [3]：

1) 从被选择的外围设备中读取丢失数据的一份备份。

2) 重新用这些数据编写固件卷。

在 PEI 调度器运行的过程中，如果它遇到已经损坏的 PEIM(例如，该 PEIM 接收到一个错误的 hash 值)，它自己必须改变启动的模式去进行恢复。一旦系统被设定成了 Recovery 模式，其他的 PEIMs 必须不能够改变系统到其他的状态。在 PEI 调度器发现系统处于 Recovery 模式时，它会重启，只调度那些需要恢复的 PEIMs。也就是说，一个 PEIM 在检测到错误条件时，会产生一个强制恢复的事件去通知 PEI 调度器进行一个恢复调度，一个 PEIM 可以通过现在的启动模式进行或者操作 BOOT_IN_RECOVERY_MODE_MASK 位来警告 PEI Foundation 模块去开始进行恢复操作。此后，PEI Foundation 模块重启使得系统的启动模式进入 BOOT_IN_RECOVERY_MODE，然后调度器以 BOOT_IN_RECOVERY_MODE 为当前模式的开始阶段。

2.2 设计原理分析

在正常情况下，UEFI 的 PEI 阶段通过 HOB 将信息 (主要是 Firmware Volume 的存储位置) 传输给 DXE 阶段，从而使得系统可以正常地加载 DXE Core 中的驱动程序进行启动。文中利用 BIOS 的 Recovery 思想，将系统的外围设备 U 盘中的 FVMAIN.FV 作为唯一拥有 Firmware Volume 的位置，记录在了 HOB 中。此后，在系统启动的过程中，系统便不会再去加载默认的驱动程序，而是加载 U 盘中的驱动程序，进而达到更新 Flash 的目的。

当然，在整个过程中，一直不改变 BIOS 的写保护跳线。为此，需要做的就是在 PEI 阶段插入一个函数 DynamicReplaceToSwitchFv()，使得系统在非 Recovery 模式下，当 PEI Dispatcher 调用 DXE Initial Program Load(IPL) PEIM 之后，使得 DXE IPL PEIM 去调用一个特殊的 PPI，即 Recovery Module PPI(EFI_PEI_RECOVERY_MODULE_PPI)，从而可以做到首先加载 U 盘中的驱动程序，实现通过软件的方法来达到改变 BIOS 写保护跳线的效果 [4-5]。

2.3 UEFI 的技术支持

UEFI 提供了 2 个重要的函数支持，分别是 PeiServicesInstallPpi() 和 InitializeRecovery()。

(1) PeiServicesInstallPpi() 函数

PeiServicesInstallPpi() 函数将所给的一个 PEIM 注

册到 PEI Foundation 中，若能够成功 install，则返回 EFI_SUCCESS，但如果在 install 的过程中没有额外的 PPI database，则返回 EFI_OUT_OF_RESOURCES，如果 PpiList 的指针为空或者在列表中的 EFI_PEI_PPI_DESCRIPTOR_PPI 没有设置 bit 位，则返回一个 EFI_INVALID_PARAMETER。其定义如下：

```
typedef
EFI_STATUS
(EFIAPI *PeiServicesInstallPpi) (
  IN EFI_PEI_PPI_DESCRIPTOR    *PpiList
);
```

(2) InitializeRecovery() 函数

InitializeRecovery() 函数主要是通过 install 一个 Recovery 的 PPI 来初始化 Recovery 的功能，而输入的参数 PeiServices 是将一些通用的 PEI 服务提供给每一个 PEIM，如果这个能够成功 install，将会返回一个 EFI_SUCCESS。其定义如下：

```
typedef
EFI_STATUS
(EFIAPI *InitializeRecovery) (
  IN EFI_PEI_SERVICES    **PeiServices
);
```

2.4 实现方法

在 UEFI 下，通过对 EDK II 源代码的编译过程，会产生一个 Full Firmware Image(.fd) 和一个 Firmware Volume Image(.FV)。其中，Full Firmware Image 就是烧写到系统中使得系统启动的 Flash。而 Firmware Volume Image 则是在系统做 Recovery 操作时用到的，当系统需要执行 Recovery 时，系统就会从其外围设备 (如 U 盘，软驱等) 中寻找 Firmware Volume Image，然后利用该 FV 中的数据完成 Recovery 操作。

因此，利用此思想，在需要编译的 Firmware 的 Flash Description File(fdf) 文件的 PEI 阶段加入一个 DynamicReplaceToSwitchFv() 函数来完成驱动更新的目的。对于该函数，需要提供 3 个参数，分别是 FileHandle、PeiServices 和 BootMode。其中，FileHandle 表示文件的句柄；PeiServices 则表示 PEI 阶段所能够使用的一些通用服务，这里主要用到的是 PeiServices 中的 InstallPpi 功能；BootMode 则表示系统启动模式。

DynamicReplaceToSwitchFv() 函数的主要功能是：首先判断系统的启动模式，如果系统处于非 Recovery 模式，同时系统存在外围设备 U 盘，则系统会从 U 盘中加载 DXE Core，而该 DXE core 便是 U 盘中的 FV 文件 (该 FV 文件中存放的是开发人员希望此次系统加载的驱动程序，可以通过 EDK II 的编译过程得到)，然后继续执行启动的后续过程。如果系统处于 Recovery 模式，则会直接返回一个 EFI_ABORTED，使得系统继续它的 Recovery 过程。

当然，若系统处于非 Recovery 模式，而系统并不存在外界设备 U 盘，则该系统会继续默认的启动过程。

函数原型如下：

```
typedef
EFI_STATUS
  (EFIAPI *DynamicReplaceToSwitchFv) (
    IN EFI_PEI_FILE_HANDLE  FileHandle,
    IN CONST EFI_PEI_SERVICES    **PeiServices,
    IN EFI_BOOT_MODE       BootMode
  );
```

3 Flash 更新的测试检验

以 Intel 945G Express Chipset 为测试平台，检验过程中首先在 Flash Description File(fdf) 文件中选择一个处于 DXE 阶段的 Driver(此处选择了 ScsiBus Driver)，然后利用 CpuDeadLoop() 函数在该 Driver 中设置一个断点 (CpuDeadLoop() 函数的功能是使得 CPU 处于死循环)，为了说明断点是否设置成功，加入了一个 DEBUG 语句，输出“ Test for UEFI Flash update ”。然后利用 EDK II 的编译过程，得到该测试平台的 simple.fd 和 FVMAIN.FV。最后将 simple.fd 烧写到 Flash 中 [6]。当平台上没有外接的 U 盘或者在 U 盘中没有 FVMAIN.FV 文件时，系统在启动的过程中会停在断点设置的地方，同时输出“ Test for UEFI Flash update ”，如图 2 所示。

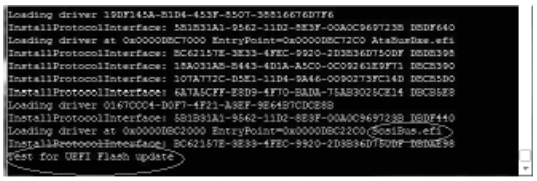

图 2 ScsiBus Driver 停在断点设置处

而如果在平台上外接一个 U 盘，同时在 U 盘中放有正常的 FVMAIN.FV 文件时，则系统能够正常启动，也就说明了 Flash 的更新方法正常工作了。如图 3 和图 4 所示，图 3 说明了此时系统利用了 FVMAIN.FV 中的数据进行恢复 ScsiBus Driver，从而使得该驱动能够正常地被加载。图 4 说明了系统最后成功启动到 Shell 环境。

图 3 利用 FVMAIN.FV 中的数据 ScsiBus Driver 正常启动

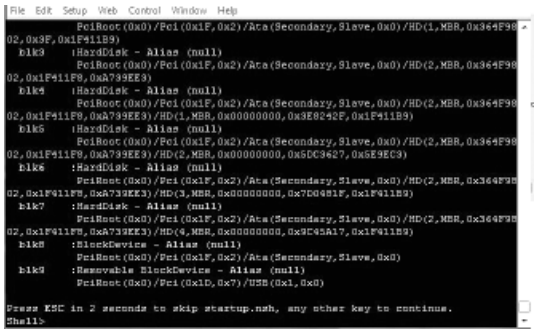

图 4 系统成功启动到 Shell

4 结语

文中主要利用 UEFI BIOS Recovery 思想，在不改动 BIOS 的写保护跳线的前提下，在 UEFI 的 PEI 阶段利用软件的方法，使得系统从 U 盘启动，从而做到免烧写 Flash 快速调试的目的，减少了重新烧写 Flash 导致系统损坏的风险。在以后的过程中，一方面希望将此方法推广到 UEFI 的 PEI 阶段，另一方面，可以将此方法推广到驱动程序的远程调试方向。

该方法的实现，对于快速提高基于 UEFI BIOS 的复杂嵌入式设备的开发和调试，具有非常现实的积极意义。

参考文献

[1] 吴松青，王典洪 . 基于 UFEI 的 Application 和 Driver 的分析与开发 [J]. 计算机应用与软件，2007，24(2)：98-100.

[2] 倪光南 . UEFI BIOS 是软件业的蓝海 [EB/OL]. (2007-06-13) [2012-03-27]. http：//soft.chinabyte.com/90/3382590_1.shtml.

[3] UEFI Forum. UEFI Specification. Version 2.3[EB/OL]. （2010）[2012-03-27]. http：//www.uefi.org/specs/.

[4] 霍卫华 . 基于 UEFI 的安全模块设计分析 [J]. 信息安全与保密通信，2008(7)：91-93.

[5] 周伟东，池亚平，方勇，吴丽军 . 一种基于信任根加强 EFI BIOS 自身安全的方案 [J]. 信息安全与保密通信，2007(7)：87-91.

[6] 吴小艨，李正宇，周进松 . 通过 RS232 串口实现 DSP 并行 FLASH 程序升级 [J]. 通信技术，2012，45(1)：77-80.

(上接第 96 页)

此对于这类网络在该方案中通常采用静态路由。

同时，为解决完全依靠人工方式生成和配置静态路由耗时长、易出错的问题，该方案设计并实现了一种静态路由自动生成算法，算法采用最短路径优先算法并针对机动通信网络特点进行了适应性改造，通过该算法可以自动生成全部超短波电台网和短波网的静态路由，很好地满足了快速和准确生成静态路由的需求，最后由相应的设备管理软件通过专用的管理接口完成路由设备的静态路由配置。

2.4 IP 地址资源预留方案

为减少机动通信网络重组时设备的重配置操作，缩短网络重新开通的时间，该方案对机动通信网络中的子网和通信节点的 IP 地址资源都按照用户设定的策略进行了一定的预留，在 IP 地址资源足够的前提下，一旦网络拓扑发生变化，只需要对发生变化的设备或节点的 IP 地址进行重新分配，并通过相应的设备管理软件更新该 IP 地址。通过 IP 地址资源预留的方式可以在一定程度上避免网络重组时需要重新分配和配置全部设备和节点的问题，大大减少了网络重新开通的难度，提高了网络运行效率。

3 结语

文中对通信网络中两种常见的 IP 地址分配方案进行了简单的分析，并根据机动通信网络的特点及 IP 地址分配的要求，提出了一种适合机动通信网络使用的 IP 地址分配方案，经过实际使用，证明该 IP 地址分配方案很好地满足了机动通信网络的使用需求，全网 IP 地址分配时间从原来的数小时减少到几秒钟，大大缩短了网络规划的时间，为机动通信网络的高效运行和可靠管理打下了良好的基础。

参考文献

[1] 刘月阳 . 无线分布式网络中基于能量的路由算法和 MAC 算法研究 [D]. 北京：北京邮电大学，2006.

[2] 谭玉珊，汪福荣 . 论网络管理的职能 [J]. 信息安全与通信保密，2007(1)：82-86.

[3] 王晓凯，侯朝桢 . 战术互联网的无线通信网络模型及网络管理策略 [J]. 计算机工程，2003，29(15)：75-77.

[4] 韩军浩 . IP 地址管理系统的研究和实现 [D]. 北京：北京邮电大学，2007.

[5] 曾任杰，余敬东 . Ad Hoc 网中 IP 地址分配技术综述 [J]. 通信技术，2007，40(12)：130-133.

[6] 谢水珍 . 子网划分方法研究 [J]. 信息安全与通信保密，2011(9)：60-62.

[7] 陈宜冬，赵丽萍 . 基于 CIDR 的网络划分方案 [J]. 信息技术，2004，28(6)：62-63.

[8] 魏晓，胡金初，魏仕民 . 基于数据包嗅探的主机网络地址自动配置方法 [J]. 计算机工程与设计，2007，28(4)：3357-3360.

[9] ABDELMALEK A，FEHAM M，TALEB-AHMED A. On Recent Security Enhancements to Autoconfiguration Protocols for MANETs Real Threats and Requirements[J]. International Journal of Computer Science and Network Security，2009，9(4)：401-407.

[10] 戴晖，王莹，高铁，等 . 战术异构网络互联端到端 QoS 研究 [J]. 信息安全与通信保密，2011(7)：51-53.

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
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度
61. RS485 和 PROFINET 网关设计
62. 基于 IPv6 的 Socket 通信的实现
63. MVB 网络重复器的设计
64. 一种新型 MVB 通信板的探究
65. 具有 MVB 接口的输入输出设备的分析
66. 基于 STM32 的 GSM 模块综合应用
67. 基于 ARM7 的 MVB CAN 网关设计
68. 机车车辆的 MVB CAN 总线网关设计
69. 智能变电站冗余网络中 IEEE1588 协议的应用
70. CAN 总线的浅析 CANopen 协议
71. 基于 CANopen 协议实现多电机系统实时控制
72. 以太网时钟同步协议的研究
73. 基于 CANopen 的列车通信网络实现研究
74. 基于 SJA1000 的 CAN 总线智能控制系统设计
75. 基于 CANopen 的运动控制单元的设计
76. 基于 STM32F107VC 的 IEEE 1588 精密时钟同步分析与实现

77. 分布式控制系统精确时钟同步技术
78. 基于 IEEE 1588 的时钟同步技术在分布式系统中应用
79. 基于 SJA1000 的 CAN 总线通讯模块的实现
80. 嵌入式设备的精确时钟同步技术的研究与实现
81. 基于 SJA1000 的 CAN 网桥设计
82. 基于 CAN 总线分布式温室监控系统的设计与实现
83. 基于 DSP 的 CANopen 通讯协议的实现
84. 基于 PCI9656 控制芯片的高速网卡 DMA 设计
85. 基于以太网及串口的数据采集模块设计
86. MVB1 类设备控制器的 FPGA 设计
87. MVB 接口彩色液晶显示诊断单元的显示应用软件设计
88. IPv6 新型套接字的网络编程剖析
89. 基于规则的 IPv4 源程序到 IPv6 源程序的移植方法
90. MVB 网络接口单元的 SOC 解决方案
91. 基于 IPSec 协议的 IPv6 安全研究
92. 具有 VME 总线的车载安全计算机 MVB 通信板卡
93. SD 卡的传输协议和读写程序
94. 基于 SCTP 的 TLS 应用
95. 基于 IPv6 的静态路由实验设计
96. 基于 MVB 的地铁列车司机显示系统研究
97. 基于参数优化批处理的 TLS 协议
98. SSD 数据结构与算法综述
99. 大容量 NAND Flash 文件系统中的地址映射算法研究
100. 基于 MVB 总线的动车组门控系统的设计与仿真研究

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
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计
42. 基于 VXWORKS 的 RS485MVB 网关的设计与实现
43. 实时操作系统 VxWorks 在微机保护中的应用
44. 基于 VxWorks 的多任务程序设计及通信管理
45. 基于 Tilcon 的 VxWorks 图形界面开发技术
46. 嵌入式图形系统 Tilcon 及应用研究
47. 基于 VxWorks 的数据采集与重演软件的图形界面的设计与实现
48. 基于嵌入式的 Tilcon 用户图形界面设计与开发
49. 基于 Tilcon 的交互式多页面的设计
50. 基于 Tilcon 的嵌入式系统人机界面开发技术
51. 基于 Tilcon 的指控系统多任务人机交互软件设计
52. 基于 Tilcon 航海标绘台界面设计
53. 基于 Tornado 和 Tilcon 的嵌入式 GIS 图形编辑软件的开发

54. VxWorks 环境下内存文件系统的应用
55. VxWorks 下的多重定时器设计
56. Freescale 的 MPC8641D 的 VxWorks BSP
57. VxWorks 实验五[时间片轮转调度]
58. 解决 VmWare 下下载大型工程.out 出现 WTX Error 0x100de 的问题
59. 基于 VxWorks 系统的 MiniGUI 图形界面开发
60. VxWorks BSP 开发中的 PCI 配置方法
61. VxWorks 在 S3C2410 上的 BSP 设计
62. VxWorks 操作系统中 PCI 总线驱动程序的设计与实现
63. VxWorks 概述
64. 基于 AT91RM9200 的 VxWorks END 网络驱动开发
65. 基于 EBD9200 的 VxWorks BSP 设计和实现
66. 基于 VxWorks 的 BSP 技术分析
67. ARM LPC2210 的 VxWorks BSP 源码
68. 基于 LPC2210 的 VxWorks BSP 移植
69. 基于 VxWorks 平台的 SCTP 协议软件设计实现
70. VxWorks 快速启动的实现方法[上电到应用程序 1 秒]

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
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发
48. 基于 Android 的视频软硬解码及渲染的对比研究与实现
49. 基于 Android 移动设备的加速度传感器技术研究
50. 基于 Android 系统振动测试仪研究
51. 基于缓存竞争优化的 Linux 进程调度策略
52. Linux 基于 W83697 和 W83977 的 UART 串口驱动开发文档
53. 基于 AT91RM9200 的嵌入式 Linux 系统的移植与实现
54. 路由信息协议在 Linux 平台上的实现
55. Linux 下 IPv6 高级路由器的实现
56. 基于 Android 平台的嵌入式视频监控系统设计

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
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计
27. 基于 XPEmbedded 的警务区 SMS 指挥平台的设计与实现
28. 基于 XPE 的数字残币兑换工具开发
29. Windows CENET 下 ADC 驱动开发设计
30. Windows CE 下 USB 设备流驱动开发与设计
31. Windows 驱动程序设计
32. 基于 Windows CE 的 GPS 应用
33. 基于 Windows CE 下大像素图像分块显示算法的研究
34. 基于 Windows CE 的数控软件开发与实现
35. NAND FLASH 在 WINCENET 系统中的应用设计

PowerPC：

WeChat ID: kontronn

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
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计
25. 基于 PowerPC603e 通用处理模块的设计与实现
26. 嵌入式微机 MPC555 驻留片内监控器的开发与实现
27. 基于 PowerPC 和 DSP 的电能质量在线监测装置的研制
28. 基于 PowerPC 架构多核处理器嵌入式系统硬件设计
29. 基于 PowerPC 的多屏系统设计
30. 基于 PowerPC 的嵌入式 SMP 系统设计
31. 基于 MPC850 的多功能通信管理器
32. 基于 MPC8640D 处理系统的技术研究
33. 基于双核 MPC8641D 处理器的计算机模块设计
34. 基于 MPC8641D 处理器的对称多处理技术研究

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
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现
33. 基于 ARM11 的电源完整性分析
34. 基于 ARM S3C6410 的 uboot 分析与移植
35. 基于 S5PC100 移动视频监控终端的设计与实现
36. UBoot 在 AT91RM9200 上的移植简析
37. 基于工控级 AT91RM9200 开发板的 UBoot 移植分析
38. 基于 ARM11 和 Zigbee 的人员定位防丢器

Hardware：

WeChat ID: kontronn

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
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法
31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化
33. 基于 GPU 的 FDTD 算法
34. 基于 GPU 的瑕疵检测
35. 基于 GPU 通用计算的分析与研究
36. 面向 OpenCL 架构的 GPGPU 量化性能模型
37. 基于 OpenCL 的图像积分图算法优化研究
38. 基于 OpenCL 的均值平移算法在多个众核平台的性能优化研究
39. 基于 OpenCL 的异构系统并行编程
40. 嵌入式系统中热备份双机切换技术研究

41. EFI-Tiano 环境下的 AES 算法应用模型
42. EFI 及其安全性研究
43. 基于 UEFI Shell 的 PreOS Application 的开发与研究
44. UEFI Bootkit 模型与分析

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
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案
10. 基于小波变换与偏微分方程的图像分解及边缘检测
11. 基于图像能量的布匹瑕疵检测方法
12. 基于 OpenCL 的拉普拉斯图像增强算法优化研究
13. 异构平台上基于 OpenCL 的 FFT 实现与优化
14. 数据结构考题 - 第 4 章 串
15. 数据结构考题 - 第 4 章 串答案
16. 用 IPv6 编程接口实现有连接通信的方法

FPGA / CPLD:

1. 一种基于并行处理器的快速车道线检测系统及 FPGA 实现
2. 基于 FPGA 和 DSP 的 DBF 实现
3. 高速浮点运算单元的 FPGA 实现
4. DLMS 算法的脉动阵结构设计及 FPGA 实现
5. 一种基于 FPGA 的 3DES 加密算法实现
6. 可编程 FIR 滤波器的 FPGA 实现
7. 基于 FPGA 的 AES 加密算法的高速实现
8. 基于 FPGA 的精确时钟同步方法
9. 应用分布式算法在 FPGA 平台实现 FIR 低通滤波器
10. 流水线技术在用 FPGA 实现高速 DSP 运算中的应用

11. 基于 FPGA 的 CAN 总线通信节点设计
12. 基于 FPGA 的高速时钟数据恢复电路的实现
13. 基于 FPGA 的高阶高速 FIR 滤波器设计与实现
14. 基于 FPGA 高效实现 FIR 滤波器的研究
15. FPGA 的 VHDL 设计策略
16. 用 FPGA 实现串口通信的设计
17. GPIB 接口的 FPGA 实现
18. 一种基于 FPGA 的 FFT 阵列处理器