μCOS-II在ARM上的移植

刘广路 郝红旗
(洛阳师范学院计算机系 河南 洛阳 471001)

摘要：本文介绍了μCOS-II的概况和ARM体系结构中与移植工作相关的一些概念，并在此基础上分析了基于ARM的移植工作。

关键词：μCOS-II；嵌入式实时操作系统；ARM；移植
中图分类号：TP311.54 文献标识码：A 文章编号：1009-3044(2007)05-11316-02

μCOS-II ARM7 transplant

LIU Guang-lu,HAO Hong-qi
(Luoyang Teachers College,Luoyang 471001,China)

Abstract:This paper presents a profile of μCOS-II ARM architecture and some of the ideas associated with transplants. and on the basis of this analysis based on the ARM transplants.

Key words: μCOS-II RTOS ;ARM; transplant

1 引言

随着微电子技术、通信技术的发展，嵌入式系统设计及其应用在后PC时代对人类生活产生了巨大的影响。随着网络技术的发展，嵌入式技术必将继续改变人们未来的生活方式。在嵌入式研发中，一个重要的基础工作就是实现嵌入式实时操作系统在相关处理器平台上的移植。本文基于目前国内应用非常广泛的ARM处理器体系结构，对μCOS-II嵌入式实时操作系统内核的移植作出分析和介绍，并在LPC2210处理器上得到了实现。

2 μCOS-II概述

1992年，美国人Jean J. Labrosse编写了实时多任务内核μC/OS，μCOS-II是基于μC/OS的二者的内核一样，只是μCOS-II提供了更多的功能。经过10多年的发展，μCOS已经被移植到包括x86 ARM PowerPC MIPS等40多种处理器架构上，移植上的处理器还在不断增加。自1992年以来已经有数百个商业应用。另外，2000年7月，μCOS-II在一个航空项目中得到了美国联邦航空管理局的认证，该认证表明μCOS-II具有足够的安全性和稳定性，能够用于与人性命攸关的、安全性条件极为苛刻的系统。

3 ARM体系结构编程

ARM是精简指令集计算机（RSC），其设计实现了外形非常小但是性能高的结构。ARM处理器结构简单使ARM的内核非常小，这样使器件的功耗也非常低。下面主要介绍ARM中与μCOS-II迁移有关的知识。

3.1 处理器状态

当ARM微处理器执行32位的ARM指令集时，工作在ARM状态；当ARM微处理器执行16位的Thumb指令集时，工作在Thumb状态。在程序的执行过程中，微处理器可以随时在两种工作状态之间切换，并且，处理器工作状态的转变不影响处理器的工作模式和相应寄存器中的内容。为了最大限度地支持芯片的特性，任务应当可以使用任意一个指令集并可自由切换，而且不同的任何应当可以使用不同的指令集，移植的代码已经实现了这一点。

3.2 处理器模式

ARM体系结构支持7种处理器模式：用户模式、快速中断模式、外部中断模式、管理模式、数据访问终止模式、系统模式和未定义指令终止模式。除用户模式外，其余的所有7种模式称之为非用户模式，或特权模式。ARM内部寄存器和一些片内外设在硬件设计上只允许，或可选为，特权模式下访问。此外，特权模式可以自由地切换处理器模式，而用户模式不能直接切换到别的模式。

除了用户模式和系统模式外的5种处理器模式称为异常模式，它们是快速中断模式、外部中断模式、管理模式、数据访问终止模式和未定义指令终止模式。它们除了可以通过程序切换进入外，也可以由特定的异常进入。当特定的异常出现时，处理器进入相应的模式。每种模式都有某些附加的寄存器，以避免异常退出时，用户模式的状态不可靠。

至于系统模式除了它是特权模式外，其他与用户模式一样，因而可选的给μCOS-II任务使用的模式只有用户模式和系统模式。为了尽量减少任务代码错误对整个系统的影响，缺省的任务模式定义未用户模式，可选为系统模式，同时提供接口，使任务可以在这两种模式之间切换。

3.3 寄存器组织

ARM微处理器共有37个32位的寄存器，其中31个为通用寄存器，6个为状态寄存器。但是这些寄存器不能被同时访问，具体哪些寄存器是可编程访问的，取决于微处理器的工作状态及具体的运行模式。但在任何时候，通用寄存器R14~R15，程序计数器PC，一个或两个状态寄存器都是可访问的。表1所示为ARM状态下，各模式可访问的寄存器。

表1 ARM状态各模式下的寄存器

寄存器类别	寄存器在汇编中的名称	各模式实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快速中断
通用寄存器和程序计数器	R0	R0						
	R1	R1						
	R2	R2						
	R3	R3						
	R4	R4						
	R5	R5						
	R6	R6						
	R7	R7						
	R8	R8						R8_fiq
	R9	R9						R9_fiq
	R10	R10						R10_fiq
	R11	R11						R11_fiq
	R12	R12						R12_fiq
	R13	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

4 μCOS-II的移植

μCOS-II中要移植的部分见表1所示。根据μCOS-II的要求，移植μCOS-II到一个新的体系机构上需要提供2个或3个文件：OS_CPU.H（语言头文件）、OS_CPU_C.C（程序源文件）及OS_CPU_A.ASM（汇编程序源文件）。其中OS_CPU_A.ASM在某些情况下不需要，及其罕见。不需要OS_CPU_A.ASM的必须满足以下苛刻条件：

(可以直接使用 语言开关中断 ;
(可以直接使用 语言编写中断服务程序 ;
(可以直接使用 语言操作堆栈指针 ;
(可以直接使用 语言保存 CPU的所有寄存器。

同时支持以上一点的 语言编译器几乎不存在,即使存在,移植代码往往也会使用部分汇编语言来提高移植代码的效率。由表可以看出,移植μCOS-需要在 OS_CPU.包含几个类型的定义和几个常数的定义,在 OS_CPU_C.和 OS_CPU_A.ASM中包含几个函数的定义和时钟节拍中断服务程序的代码。实际上还有一个 includes.文件需要关注,因为每一个应用都包含独特的 includes.文件。

表 2μCOS-需要移植的代码

移植内容	类型	所属文件	说明
BOOLEAN、INT8U、INT8S、INT16U、INT16S、INT32U、INT32S、FP32、FP64	数据类型	OS_CPU.H	与编译器无关的数据类型
OS_STK	数据类型	OS_CPU.H	堆栈的数据类型
OS_ENTER_CRITICAL() 和 OS_EXIT_CIRTICAL()	宏	OS_CPU.H	开关中断的代码
OS_STK_GROWTH	常量	OS_CPU.H	定义堆栈的增长方向
OS_TASK_SW	函数	OS_CPU.H	任务切换时执行的代码
OSTaskStrInit()	函数	OS_CUP_C.C	任务堆栈初始化函数
OSInitHookBegin()、OSInitHookEnd()、OSTaskCreateHook() 、OSTaskDelHook()、OSTaskSwHook()、OSTaskStatHook()、OSTCBInitHook()、OSTimeTickHook()、OSTaskIdleHook()	函数	OS_CUP_C.C	μCOS-II 在执行某些操作时调用的用户函数,一般为空
OSStartHighRdy()	函数	OS_CPU.A.ASM	进入多任务环境时运行优先级最高的任务
OSIntCtxSw()	函数	OS_CPU.A.ASM	中断退出时的任务切换函数
OSTickISR()	中断服务程序	OS_CPU.A.ASM	时钟节拍中断服务程序

根据 Jean J. Labrosse对μCOS-移植的要求,本移植也包括 OS_CUP.h OS_CPU_C和 OS_CPU_A三个文件。将 OS_CUP_A.as重名为 OS_CPU_A是遵照 ADS编译器的惯例。实际上还有一个文件很重要,就是 IRQ.in它,定义了一个汇编宏,是μCOS-为 ARM通用的中断服务程序的汇编与函数接口代码。时钟节拍中断服务程序也没有移植,因为其与芯片和应用都强烈相关,需要用户自己编写,不过可以通过 IRQ简化用户代码的编写。

4编写 OS_CUP.h

4.1数据类型定义

数据类型的修改与所使用的编译器相关,不同的编译器使用不同的字节长度表示同一数据类型,比如 int同样在 x86平台,GNU的 gcc编译器为 4byte的,而 MS VC则为 2byte的。根据 ADS编译器的要求,这些代码定义如下 :

```
typedef unsigned char BOOLEAN;
typedef unsigned char INT8U;
typedef signed char INT8S;
typedef unsigned short INT16U;
typedef signed short INT16S;
typedef unsigned int INT32U;
typedef signed int INT32S;
typedef float FP32;
typedef double FP64;
typedef INT32U OS_STK;
```

4.1使用软件中断 SWI作底层接口

为了使底层接口函数与处理器状态无关,同时在任务调用相应的函数不需要知道函数的位置,本移植使用软件中断指令 SWI作为底层接口,使用不同的功能号区分不同的函数。软件中断功能号如表 所示 :

表 3软件中断功能

功能号	接口函数	简介
0x00	void OS_TASK_SW(void)	任务级任务切换
0x01	_OSStartHighRdy(void)	运行优先级最高的任务,由 OSStartHighRdy 产生
0x02	void OS_ENTER_CRITICAL(void)	关中断
0x03	void OS_EXIT_CRITICAL(void)	开中断
0x80	Void ChangeToSYSMode(void)	任务切换到系统模式
0x81	Void ChangeToUSRMode(void)	任务切换到用户模式
0x82	Void TaskIsARM(INT8U prio)	任务代码是 ARM 代码
0x83	Void TaskIsTHUMB(INT8U prio)	任务代码是 THUMB 代码

用软件中断作为操作系统的底层接口就需要在 语言中使用 SWI指令。在 ADS中,有一个关键字 __swi用,它声明一个不存在的函数,则调用这个函数就在调用这个函数的地方插入一条 SWI指令,并且可以制定功能号。同时,这个函数也可以有参数和返回值,其传递规则与一般函数一样,其代码如下 :

```
- - swi(0x00) void OS_TASK_SW(void);
- - swi(0x01) void _OSStartHighRdy(void);
- - swi(0x02) void OS_ENTER_CRITICAL(void);
- - swi(0x03) void OS_EXIT_CRITICAL(void);
- - swi(0x80) void ChangeToSYSMode(void);
- - swi(0x81) void ChangeToUSRMode(void);
- - swi(0x82) void TaskIsARM(INT8U prio);
- - swi(0x83) void TaskIsTHUMB(INT8U prio);
```

4.1定义堆栈增长方向

虽然 ARM处理器可以支持堆栈向上增长,也可以支持堆栈向下增长,但是 ADS的 语言编译器仅支持一种方式,即从上往下增长。所以定义如下 :

```
#define OS_STK_GROWTH 1
```

4编写 OS_CPU_C文件

本文件要求编写 1个简单的 函数 :

```
OSTaskStkInit();
OSTaskCreateHook();
OSTaskDelHook();
OSTaskSwHook()
OSTaskIdleHook();
OSTaskStatHook();
OSTimeTickHook();
OSInitHookBegin();
OSInitHookEnd();
OSTCBInitHoo();
```

唯一必要的函数是 OSTaskStkInit (),其他 个函数必须声明,但是并不一定要包含任何代码。

4.2.1 OSTaskStkInit()

该函数初始化任务的堆栈,而任务的堆栈结构跟 CPU的体系机构、编译器有密切关系。本移植的堆栈结构如图 所示。根据图示,很容易写出 OSTaskStkInit()的代码,如程序清单 1。2

程序清单 1-函数 OSTaskStkInit()代码

```
OS_STK *OSTaskStkInit (void (*task)(void *pd), void *pdata,
OS_STK *ptos, INT16U opt)
{OS_STK *stk;
opt = opt;
/* ' opt'没有使用。作用是避免编译器警告 */
stk =ptos; /获取堆栈指针 */
/建立任务环境 , ADS1使用满递减堆栈 */
*stk = (OS_STK) task;/* pc */
*- - stk = (OS_STK) task; /* lr */
*- - stk = 0; /* r12 */
*- - stk = 0; /* r11 */
*- - stk = 0; /* r10 */
*- - stk = 0; /* r9 */
*- - stk = 0; /* r8 */
*- - stk = 0; /* r7 */
*- - stk = 0; /* r6 */
*- - stk = 0; /* r5 */
*- - stk = 0; /* r4 */
*- - stk = 0; /* r3 */
*- - stk = 0; /* r2 */
*- - stk = 0; /* r1 */
*- - stk = (unsigned int) pdata;
/*第一个参数使用 R0传递 */
*- - stk = (USER_USING_MODE|0x00);
/* spsr允许 IRQ, FIQ中断 */
*- - stk = 0; /*关中断计数器 OsEnterSum; */
return (stk);}
```

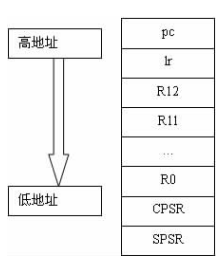

图 1任务堆栈结构

4.2软件中断异样 SWI服务程序 语言部分

分替代或增强。这就要求我们在进行设计开发时要注意开发的原则 遵循必要性原则、可行性原则、可控性原则、直观性原则和经济性原则。注意真实实验和虚拟实验的“真”“虚”互补 注意实验内容的“虚”“实”交替 达到虚中有实 实借助虚 使虚拟实验和真实实验在实现实验教学方面相得益彰 优势互补。

4做到有组织、有规划 融合科研力量 加强交流与合作

虚拟实验从诞生起 因其具有资源共享 信息服务等功能显示出其公益性和社会化的属性。虚拟实验室应该是全社会的 应该由政府部门统一组织、协调、规划。从中国的国情出发 遵循开发原则 选择社会迫切需要的内容 以避免无序开发 重复建设。同时 利用现有网络设施 如教科网 汇集全国的科研力量 搭建虚拟实验的开发 科研 技术交流的平台。

4加强技术培训 建立开放的虚拟实验资源库

为了使虚拟实验技术更好的与教育教学实践接轨。正在教育一线的教师或者是即将走向教学岗位的师范生是不可忽视的重要力量。他们具备着或即将具备教育教学的经验 掌握着教育学、心理学的相关知识 熟悉学生的认知心理和现状 进行着教育教学实践 并能够直接的获得学生的反馈。他们是虚拟实验的实践者 是用户 是推广和普及的传播者 理所当然的也应该是开发虚拟实验的直接参与者。对他们 应该进行相关专业知识的学习和培训 并建立开放的虚拟实验资源库 以利于教师获取知识、利用虚拟实验资源并获取他们在教育教学实践中的反馈。

4鼓励多技术层面的虚拟实验的开发 创造良好的开发使用氛围

从理想的角度出发 虚拟实验应尽可能的模拟真实实验的环境 让学生自主的进行实验 尽可能的让学生获得真实实验的氛围和感受。要达到这样的要求 从技术层面上说 对于普通的教师 即便是普通计算机专业教师来说 都是有相当难度的。但从教学应用的实际出发 我们应该鼓励教师开发较低技术层次的虚拟实验。比如基于 Flash, Authorware或 java技术的虚拟实验的开发 这对于中学所需的大部分实验都是可以胜任的。这样可以弥补中等教育中虚拟实验应用的空白 扩大虚拟实验开发应用的范围 毕竟我们现在的学生是虚拟实验的使用者 也是今后的开发者和推广者。尽早的接触虚拟实验 有助于营造良好的开发使用氛围。

结束语

虚拟实验是信息时代的产物 它在教育、科研等方面有着广阔的发展前景 是远程教育的重要组成部分 是教育教学的重要资源。对于实现发达地区和欠发达地区的教育资源的共享都有着重要作用。应尽快的建设一批符合教学大纲 学科门类齐全 实用性高的虚拟实验室 更好地为当前的教育服务。

参考文献：

[1]陈代武 多媒体教学课件制作中虚拟实验的设计 [J].湖南：计算机时代，2003(7).

[2]陈秀清 关于虚拟实验室的建设 [J].福建 福建广播电视大学学报，2003(1).

[3]刘敏 基于Flash的虚拟实验室的开发 [J].四川 长沙大学学报，2004(2):67-70.

[4]刘亦红 论物理虚拟实验与真实实验的互补作用 [J].陕西：中国电化教育，2004(7):42-44.

上(接第131页)

软件中断的C语言处理函数代码比较简单 函数原型为：void SWI_Exception(int SWI_Num,int *reg);参数SWI_Num为功能号，而Reg为指向堆栈中保存寄存器的值的位置。程序通过一个switch语句把各个功能分割开 各个功能相对对立。

OS_ENTER_CRITICAL()和OS_EXIT_CRITICAL()

μC/OS-使用宏OS_ENTER_CRITICAL()和OS_EXIT_CRITICAL()分别关中断和开中断。在ARM处理器核中关中断和开中断时 通过改变程序状态寄存器CPSR中的相应位来实现。3. OSStartHighRdy

μC/OS-的启动多任务环境的函数是OSStart()，用户在调用OSStart()之前 必须已经建立了一个或更多任务。OSStart()最终调用函数OSStartHighRdy()运行多任务启动前优先级最高的任务，OSStartHighRdy()的代码如下：

程序清单1-3 OSStartHighRdy()的代码

```
void OSStartHighRdy(void){
 _OSStartHighRdy();}
```

4编写OS_CPU_A.S

4.3软件中断的汇编接口

该接口即实现了中断功能号、0同时也把C语言中断联系起来。代码如程序清单1-所示：

程序清单1-软件中断的汇编部分

```
SoftwareInterrupt
LDR     SP, StackSvc            重新设置堆栈指针
STMFD   SP!, {R0-R3, R12, LR}
MOV     R1, SP                  ;R1指向参数存储位置
MRS     R3, SPSR
TST     R3, #T_bit              中断前是否是Thumb状态
LDRNEH  R0, [LR,#-2]            是 取得Thumb状态SWI号
BICNE   R0, R0, #0xff00
LDREQ   R0, [LR,#-4]            否 取得arm状态SWI号
BICEQ   R0, R0, #0xFF000000
; r0 = SWI号, R1指向参数存储位置
CMP     R0, #1
LDRLO   PC, =OSIntCtxSw
LDREQ   PC, =__OSStartHighRdy
; SWI 0x0为第一次任务切换
BL      SWI_Exception
LDMFD   SP!, {R0-R3, R12, PC}^
```

4.3.2 OSCtxSw()

任务级的上下文切换 当任务被阻塞而主动请求CPU调度时被执行 由于此时的任务切换在非异常模式下进行 因此有别于中断级别的任务切换。它的工作是先将当前的任务的CPU现场保存到该任务堆栈中。然后获得最高优先级任务的堆栈指针 从该堆栈中恢复此任务的CPU现场 使之继续执行。这样就完成一次任务的切换。

4.3.3 OSIntCtxSw()

中断级的任务切换 在时钟中断ISR中发现有高优先级任务等待时钟信号的到来 则在中断退出后并不是返回被中断的任务 而是直接调度就绪的高优先级任务执行。从而能够尽快让高优先级的任务得到响应 保证系统的实时性能。其基本原理于任务的切换相同。但是由于进入中断时已经保存了被中断的CPU现场 因此不用做类似的操作 只需对堆栈指针做相应调整。

4.3时钟中断服务程序

其主要任务是处理时钟中断。示例代码如下：

```
void ISR(void)
{OS_ENTER_CRITICAL();
清除中断源 ;
通知中断控制器中断结束 ;
 OS_EXIT_CRITICAL();
用户处理代码 ;}
```

4结束

μC/OS-作为一个优秀的实时操作系统已经被移植到许多体系结构的处理器上 而ARM体系结构在嵌入式领域的应用越来越广泛。将μC/OS-移植到ARM平台 不仅可以加深对实时系统的理解 同时也是进行产品开发的重要基础。

参考文献：

[1]邵贝贝译 嵌入式实时操作系统μC/OS-II[M].

[2]周立功编.ARM微控制器基础与实践[M].

[3]ARM公司.ARM Developer Suite version1.2 Developer Guide.

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-OS II 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
39. 基于 NAND FLASH 的嵌入式系统启动速度的研究

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
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
45. UEFI 计算机系统快速调试方法的实现

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
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
19. 基于 FPGA 的 FFT 信号处理器的硬件实现