Design of High-Fidelity Lossless Music Player Based on ARM Embedded System

*CHEN Zilong, ZHOU Shujie, TANG Yongming**

(School of Electronic Science & Engineering, Southeast University, Nanjing 210096, China)

Abstract The design of High-Fidelity lossless music player which is based on ARM 11 embedded platform with embedded Linux kernel is presented. It supports 3 kinds of lossless compression format including WAV, FLAC and APE. Functions such as browsing music directories, music playback or pause and adjusting playback progress have been realized by tapping the touch screen. External mass storage devices such as mobile hard disk, SD card and USB flash disk are also supported. High-Fidelity lossless music (WAV, FLAC and APE) playback has already been realized on the development platform with music sampling bits 16 or 24 and sampling rate lower than 48 kHz.

Key words embedded linux; ARM11; multimedia player; lossless compression format

EEACC 6130 **doi** 10.3969/j.issn.1005-9490.2012.06.016

基于 ARM 嵌入式系统的高保真无损音乐播放器设计

陈自龙，周书杰，汤勇明*

（东南大学电子科学与工程学院，南京 210096**）**

摘　要 介绍了 1 种基于 ARM11 嵌入式平台和嵌入式 Linux 内核，界面美观、操作便捷的高保真无损音乐播放器的设计。它支持 WAV、FLAC 和 APE 3 种无损压缩格式音乐，还可通过触摸屏实现浏览音乐目录，控制音乐的播放、暂停和调节播放进度等控制功能，它支持外接移动硬盘、SD 卡和优盘等大容量存储设备。在开发平台上已实现采样位数 16 或 24，采样率上限 48 kHz 的 WAV、FLAC 和 APE 高保真音乐播放。

关键词 嵌入式 Linux；ARM11；多媒体播放器；无损压缩格式

中图分类号 TN972.231　　**文献标识码** A　　**文章编号** 1005-9490(2012)06-0692-07

嵌入式系统播放 MP3 等有损压缩格式音乐的技术已经比较成熟[1~3]，但是对于播放无损压缩音乐却鲜有报道。

本次设计选择基于 ARM11 处理器核心的 TINY6410 开发板进行，其良好的计算能力可满足对高压缩率音乐软解码的需求。针对开发板硬件资源和软件设计要求，本设计裁剪 Linux 内核并对大容量存储器做了支持工作，研究了 Linux 内核的 ALSA 音频架构，移植 FLAC 解码库和 APE 解码库，设计了音乐播放器的用户界面，使用 Qt/Embedded 开发工具完成了播放器的软件开发工作。

1　高保真音乐

1.1　WAV 格式

WAV 是微软公司开发的一种音频文件格式，用于保存 WINDOWS 平台的音频信息资源。该格式支持 MSADPCM、CCITT A LAW 等多种采样压缩算法，支持多种音频位数、采样频率和声道，标准格式的 WAV 文件和 CD 格式一样，也是 44.1 kHz 的采样频率，速率 88 kbyte/s，16 bit 量化位数。

WAV 格式的优点是：编/解码简单，几乎直接存储来自模/数转换器（ADC）的信号，多系统支持以及无损压缩。

其主要缺点是需要较大的音频存储空间。

1.2　FLAC 格式

FLAC 是 Free Lossless Audio Codec 的缩写，即无损音频压缩编码，是一套著名的自由音频压缩编码，其特点是无损压缩，即音频数据以 FLAC 编码压缩后不会丢失任何信息。

FLAC 解码只需整数运算，相对于大多数音频编码方式而言，对计算速度要求不高。

FLAC 编码有很多优点：可以定位、便于对 CD

进行备份、抗损伤、富于弹性的 Metadata 等等。而且，FLAC 是开源项目，其文件格式对公众完全开放，其文件格式和编/解码的实现方式都不受任何已知专利的限制。FLAC 解码库所有的源代码都可在开放源代码的授权方式下得到。

1.3 APE 格式

APE 格式的音乐是流行的数字音乐格式之一，由 Monkey's Audio 推出的一套无损音乐压缩算法压缩而成，该压缩算法针对音频数据进行了专门的优化，因而压缩率相对比较高，而且解压之后的音频数据没有任何损失。按照 Monkey's Audio 官方的说法，同样一首歌曲的 APE 格式仅是 WAV 格式文件大小的一半左右，比 FLAC 格式文件也要小。由于 APE 格式音乐的解码涉及到浮点运算，而 FLAC 格式只需要整数运算，所以通常 FLAC 的解码速度比 APE 快 30%。而且，APE 格式使用了对称算法[4]，在解压缩时还需要进行一些编码工作，这也消耗了部分 CPU 和内存资源，相比之下，FLAC 的格式则没有这个问题。

2 硬件系统

TINY6410 开发板的硬件系统由主控制器 ARM11 S3C6410、触摸显示屏、大容量存储设备（移动硬盘、SD 卡和优盘）、高品质音频解码芯片 WM9714 和功放输出组成。其中除大容量存储设备和触摸显示屏为外接设备外，其余都集成在开发板上。

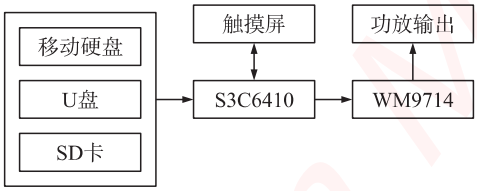


图 1 硬件系统框图

S3C6410 处理器采用 ARM11 架构，主频 533 MHz，最高 667 MHz。

S3C6410 的高主频特性保证了对各音频格式软解码时的要求，特别是压缩率很高的 APE 音乐。

丰富的外接存储接口满足了挂载多种存储器的需求。

开发板自带的 4.3 吋 TFT 真彩色触摸显示屏可满足用户的交互控制需求。

本嵌入式高保真音乐播放软件系统需要消耗可观的内存，主要原因包括：

1）程序运行需要加载大量且必要的动态链接库到内存；

2）播放器程序的界面设计中使用了很多 PNG 格式的图片以达到美观的目的，但是加载图片需要消耗内存；

3）挂载外接大容量存储设备（特别是大容量移动硬盘）也需要消耗内存。

TINY6410 自带的 128M DDR RAM 可以满足上述所有的内存消耗。

3 软件系统

本项目软件开发部分工作从底层到上层可以依次为：操作系统内核裁剪、程序库移植和编程应用、应用程序开发。图 2 给出了该软件系统的框架结构。

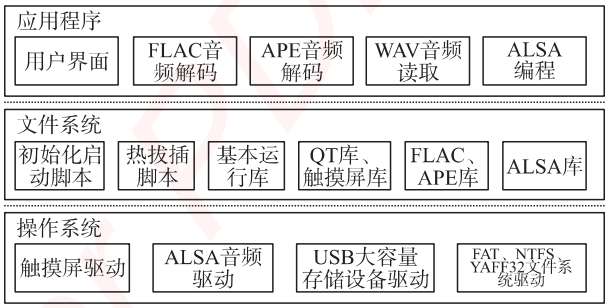


图 2 软件系统架构

3.1 Linux 内核裁剪

为了支持多种处理器和设备，Linux 内核庞大而复杂。就嵌入式高保真音乐播放器来说，内核的很多功能均未使用，将这些不必要的功能模块从内核裁减掉，有助于提高内核的运行效率，降低内存使用率[5]。主要被裁剪的模块有：

1）网络设备支持（以太网控制器驱动和无线网卡驱动）

2）多媒体支持（Video for Linux 图像采集框架以及相应的视频采集卡驱动）

3）输入设备支持（鼠标和键盘）

4）各种 USB 设备支持（除了 USB 大容量存储器支持）

5）不必要的文件系统支持（Network File Systems、Ext3 Journaling File System 等）

6）TINY6410 开发板自带的外接设备驱动（LED、蜂鸣器、按键驱动等）

经过裁剪之后的内核镜像大小为 2 519 036 byte，相比未裁剪之前的 3 689 004 byte，体积减小了约三分之一。

内核除了必要的裁剪，还需要添加对外接大容量存储器挂载支持[6]。

无损音乐文件体积大，开发板自带的存储空间不能满足需求，需要外接大容量存储设备。

不管是优盘、SD 卡还是移动硬盘，连接到开发

板后都会有相应的内核信息产生，利用 Linux 内核的热插拔机制（本系统利用 Mdev 机制，Linux 内核还有其他热插拔机制）运行相应的存储分区挂载命令就可以挂载外部存储设备到开发板上。不过，NTFS 格式的移动硬盘挂载还需要在编译 Linux 内核的时候添加 NTFS 文件系统支持选项。

3.2 程序库移植和编程应用

3.2.1 ALSA 库移植和应用

对 Linux 内核进行音频编程的本质是要使用音频驱动程序提供的编程接口，完成对声卡的各种操作。目前 Linux 内核中声卡驱动程序主要是 ALSA 架构。

ALSA 是 Advanced Linux Sound Architecture 的缩写，即高级 Linux 声音架构，它为 Linux 内核提供了音频驱动和应用程序音频编程接口。其主要特性包括：

1）高效地支持从消费类入门级声卡到专业级音频设备所有类型的音频接口；

2）完全模块化的设计；

3）支持对称多处理（SMP）和线程安全；

4）对 OSS（Open Sound System，Linux 内核早期的音频子系统架构）的兼容；

5）提供了用户空间的 ALSA-Lib 库来简化应用程序的开发。

图 3 给出了 ALSA 的层次结构示意。

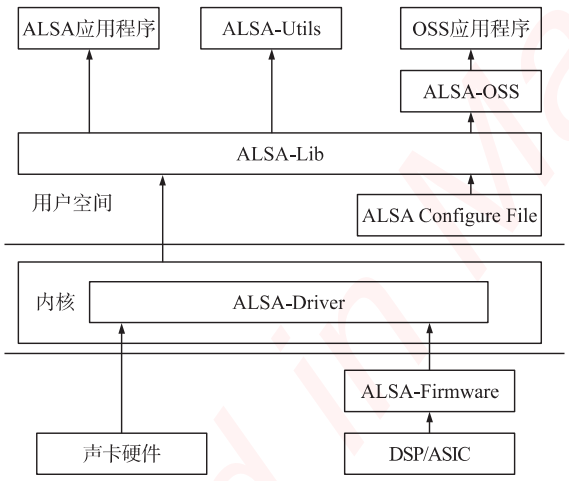


图 3 ALSA 层次结构图

可以看出 ALSA 架构包括驱动包（ALSA-Driver）开发包（ALSA-Libs）开发包插件（ALSA-Lib-Plugins）设置管理工具包（ALSA-Utils）其他声音相关处理小程序包（ALSA-Tools）特殊音频固件支持包（ALSA-Firmware）OSS 接口兼容模拟层工具（ALSA-OSS）共 7 个子项目，其中驱动包是必须的。

ALSA-Driver 指内核驱动程序，包括硬件相关的代码和一些公共代码，非常庞大。ALSA-Libs 指用户空间编程的函数库。ALSA-Utils 包含一些基于 ALSA 的用于控制声卡的应用程序。

在 ALSA 的官网上下载 ALSAlib 的源码包，使用如下命令编译 ALSA 库：

```
$tar-xzvf alsa-lib-1.0.22.tar.gz
$cd alsa-lib-1.0.22
$./configure--host=arm-linux
    \--prefix=/home/zsjproject/alsa_arm/   \
--enable-static   \
--enable-shared   \
--disable-python\
--with-configdir=/usr/share/   \
--with-plugindir=/usr/local/lib/alsa_lib
```

完成编译配置之后，进行编译：

```
$make && make install
```

从上面的编译配置可以看出，指定的交叉编译平台是 ARM，采用静态编译并且禁用了 Python 组件，指定了 ALSA Config Files 和 ALSA Plugin Files 路径。

将生成的库文件复制到板载系统对应的路径中，尤其 Lib 和 Share 文件夹需要拷贝到/usr/local/lib 和/usr/share/路径下。

3.2.2 ALSA 音频播放编程

ALSA 架构能够实现音频的回放、录音和混音等绝大多数音频处理相关的功能，这里只讨论音频回放功能。

ALSA 架构播放音频的一般编程步骤是：

1）打开 ALSA 的音频播放接口；

2）设置硬件参数（访问模式、数据格式、声道数和采样率等）；

3）循环分次将所有 PCM 数据送入声卡设备进行播放；

4）播放完毕，关闭播放接口。

编写高保真音乐播放程序的思路是：先将 FLAC 和 APE 音乐解码（WAV 无需解码）然后将所得音频数据送入声卡设备即可实现播放，其流程如图 4 所示。

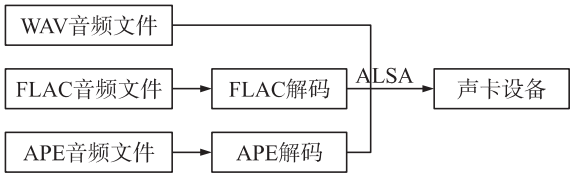


图 4 音频数据播放流程图

3.3 FLAC 格式解码支持

FLAC 的官方网站提供了一个开源的跨平台 FLAC 编解码库，使用这个库的 API 可以实现对 FLAC 格式音乐的解码工作。

3.3.1 FLAC 解码库的移植

在 FLAC 官网下载 FLAC 编解码库的最新版本 flac-1.2.1.tar.gz，解压后需要先修改 flac-1.2.1/examples/cpp/encode/file/main.cpp 文件，否则编译的时候会出错。打开 main.cpp，然后在开头添加一行#include <cstring>即可。接下来进行交叉编译的配置，执行命令：

```
$./configure \
--prefix=/opt/embedded/libflac \
--enable-cross-compile \
--build=i386-pc-linux-gnu \
--host=arm-linux \
--target=arm-linux \
--disable-ogg \
--disable-3dnow \
--disable-oggtest \
--disable-asm-optimizations \
--disable-xmms-plugin \
--disable-sse \
--disable-shared \
--disable-id3libtest \
CC=arm-linux-gcc \
CXX=arm-linux-g++\
AR=arm-linux-ar \
LD=arm-linux-ld \
RANLIB=arm-linux-ranlib \
STRIP=arm-linux-strip
```

配置结束后就可以编译和安装了：

```
$make-j3 && make install
```

这样 FLAC 库就被安装到宿主机的/opt/embedded/libflac 路径下，将此路径下编译好的 FLAC 运行库复制到开发板上的系统里，并设置 LD_LIBRARY_PATH 变量就可以让运行的程序调用这个库的 API。

3.3.2 FLAC 解码库的编程使用

FLAC 的解码库提供了 C 语言接口和 C++语言接口，本系统使用 C 语言接口。对一个 FLAC 格式音乐文件解码分为以下几个步骤：

1）申请解码对象；

2）初始化解码对象；

3）MD5 值检查（可选）；

4）FLAC 数据帧解码；

5）删除解码对象。

对应的函数分别是：

FLAC__stream_decoder_new（），

FLAC__stream_decoder_init_file（），

FLAC__stream_decoder_set_md5_checking（），

FLAC__stream_decoder_process_single（），

FLAC__stream_decoder_delete（.）

每解码一帧数据，都有一个回调函数执行，这个回调函数将解码后的 PCM 数据做适当调整后送入声卡设备播放。

至于音频参数信息的获取则需要 FLAC__Metadata_SimpleIterator 这个结构体获得。通过 FLAC__Metadata_SimpleIterator 从 FLAC 文件中得到 id 为 METADATA_BLOCK_STREAMINFO 的数据块，然后从这个数据块可以解析出具体的音频参数信息对声卡设备进行参数设置。

3.4 APE 格式解码支持

APE 格式的官方网站 Monkey's Audio 提供了一个 Monkey's Audio SDK 开发者工具包，使用其中的 API 编程可以实现对 APE 格式音乐的编解码工作。

3.4.1 Monkey's Audio SDK 的移植

Monkey's Audio 官方目前只是针对 Windows 平台做了支持，不过一些开源爱好者已经将 Monkey's Audio SDK 移植到 Linux 平台，我们要对这个移植版本进行编译和安装，使用下列命令：

```
$tar-xvf mac-3.99-u4.b5.tar.gz
$cd mac-3.99-u4-b5
$./configure \
--enable-assembly=no \
--host=arm-linux \
--prefix=/opt/embedded/libape
$make-j3 && make install
```

从上述配置可以看出，汇编被禁用，目标平台是 ARM，并且编译好的库被安装到宿主机的/opt/embedded/libape 路径下。将这些编译好的库复制到开发板上的系统里，并设置 LD_LIBRARY_PATH 变量就可以让应用程序使用 Monkey's Audio SDK 了。

3.4.2 Monkey's Audio SDK 的编程

由于只需要进行解码工作，所以这里只涉及到解码 API 的使用。Monkey's Audio SDK 使用 C++语言编写，要解码 APE 格式音乐，使用其解码类“IAPEDecompress”即可。APE 的解码接口只需调用“IAPEDecompress”类的“GetData（”函数、“Seek（”函数、“GetInfo（”函数即可，所有的数据同步和缓冲都由解码库自动完成。

3.5 播放器应用程序开发

本设计中的应用程序采用 C++开发语言在 Qt/Embedded 开发平台上实施。它是一个用于桌面系统和嵌入式开发的跨平台应用程序框架，包括一个直观的应用接口程序函数 API 和一个丰富的类库，以及用于 GUI 开发和国际化的集成工具。

3.5.1 程序功能描述

本项目软件旨在实现一个功能完善的高保真无损音乐播放系统。

当用户按下电源按钮启动系统后，系统第一个执行的用户程序就是高保真音乐播放器程序。此程序扫描所有外接存储器中的无损压缩音乐文件，并生成一个音乐列表，保存扫描结果。这样以后再启动程序时，程序只需要读取这个音乐列表得到所有的歌曲信息。如果外接存储器发生变更或改动后，用户可以启动播放器的更新音乐库线程来更新音乐列表。这种设计在保持音乐文件同步的同时，减少扫描等待时间，提升用户体验。

用户可以进行播放、暂停、停止、上一曲、下一曲、播放进度调节和音量调节的控制等操作。歌曲的选择通过一个简洁、美观的播放列表实现，同时，对某一首歌曲的详细信息 如专辑、歌手、音乐流派等 也可以进一步查看。

作为一个完整的系统设计，播放器主界面还提供了时间日期显示和设置、关机和更新本地音乐库等系统服务和功能。

3.5.2 界面设计

随着软件设计水平的提升，用户不仅关注软件的实用性，对软件界面设计的美观性和易用性也有着较高的要求。

本嵌入式高保真音乐播放器界面设计以简约、美观为主，配合以阴影、半透明和动画等效果，在实现系统完整功能的基础上，带给用户良好的人机交互体验。

主界面、播放列表界面、单曲信息查看界面等设计如图 5 ~ 7 所示，其它一些界面元素设计如图 8 所示。

3.5.3 应用程序设计

本应用程序涉及到 Qt 界面编程、多线程处理、Linux 内核音频编程以及无损压缩音频软解码等，其程序模块化设计如图 9 所示。

图 5 播放器主界面

图 6 音乐列表界面

图 7 单曲信息查看界面

图 8 播放器界面元素综合图

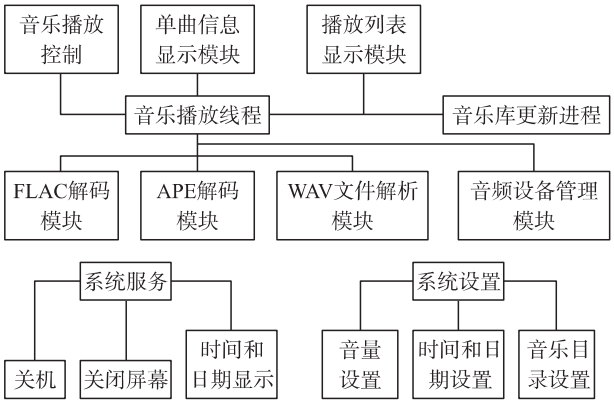


图 9 程序模块划分图

可以将上述模块粗略地划分为 ：

1) 界面显示模块 ；

2) 音乐播放控制模块 ；

3) 音频解码模块 ；

4) 系统设置和服务模块。

界面显示模块是播放器人机交互的接口。通过继承 Qt 的 QWidget 类,并实现相应的绘图函数和消息处理,可以实现各种界面元素和控件,有了这些最基本的界面元素和控件,同时利用 Qt 特有的“信号/槽”机制作为中间“通信”工具,不难构造出完整的程序界面[7]。

音乐播放控制模块是整个高保真音乐播放器的核心,利用 Qt 自带的多线程处理机制,可以实现音乐播放与界面显示的隔离,不会产生界面冻结和卡死问题。

通过继承 Qt 的 QThread 线程类,并按程序需要完善其“run”函数,可以比较容易的实现多线程处理,例如音乐播放模块的“run”函数代码如下:

```
void musicPlayCore::run()
{
if(WAVE==music_fmt){
  processWaveMusic();
}else if(FLAC==music_fmt){
  processFlacMusic();
}else if(APE==music_fmt){
  processApeMusic();
}
}
```

上述代码比较简单,就是按要播放的音乐格式,分别调用相应的处理函数,其中对于 FLAC 格式的音乐,其播放处理函数“processFlacMusic”代码如下:

```
void musicPlayCore::processFlacMusic()
{
flac_cur_samples=0;
while(flac_cur_samples<flac_total_samples){
  if(PLAYING==state){
    mutex.lock();
    //解码一帧 flac 音频数据,并更新 flac 帧计数器
    FLAC__stream_decoder_process_single(p_flac_
    decoder);flac_cur_samples+=cur_flac_block_size;
    mutex.unlock();
  }else if(PAUSED==state){
    if(paused_flag)
      paused_flag=true;
    msleep(10);//如果进入暂停状态,就让线程睡眠
  }else if(STOP==state){
    goto just_stop_flac;//如果进入停止状态,就复位
    flac 解码器对象
  }
}
state=STOP;
//发出歌曲播放完毕信号
emit signalSongPlayDone();
just_stop_flac:
FLAC__stream_decoder_reset(p_flac_decoder);//复位 flac
解码器对象
}
```

这段代码根据不同的播放状态进行处理的过程 在播放状态时不断解码 在暂停状态时不进行解码,并使线程睡眠 在停止状态时复位 flac 解码器对象并返回,结束播放线程。对于 WAV 格式和 APE 格式音乐的播放处理完全类似,不再赘述。

音频解码模块是音乐播放模块的基础,能对 FLAC 格式和 APE 格式音乐进行顺序解码以及随机位置解码,并将解码后的音频采样数据返回给音乐播放模块。顺序解码满足正常的音乐播放功能,随机位置解码可以满足播放进度调节功能。此外,解码模块可以获得音频文件的参数信息 采样率、采样位数、声道数等 和专辑信息 专辑名称、歌手、流派、发行时间等 。) 播放模块会根据参数信息对声卡设备进行设置,否则音乐无法正确播放,而专辑信息则可以以图文并茂的方式呈现给用户,提升用户体验。

系统设置和服务模块是为了系统的完整性而存在的。由于整个系统运行时只有高保真音乐播放器作为前台程序运行,必须提供必要的关机和时间日期设置功能。同时,为了使程序更加人性化,还提供了时间日期显示功能和关闭屏幕功能。作为嵌入式应用,功耗是重要的设计问题,就整个硬件系统而言,屏幕的功耗占了整个系统功耗的相当一部分,如果在用户不需要查看屏幕时,将屏幕关闭,可有效降低系统功耗。

4 系统验证

在完成系统设计的硬件和软件工作目标后,对该嵌入式高保真无损音乐播放器进行试听验证。对于 16 bit 或 24 bit 采样位数,采样频率上限为 48 kHz 的无损高保真音乐均可顺利播放。用户界面操作流畅,没有崩溃或卡死现象发生。播放器顺利加载 1TB 移动硬盘中的 1600 首无损高保真音乐。

5 结束语

基于 ARM11 开发板和嵌入式 Linux 内核,本文给出了一个界面美观,功能齐全的高保真音乐播放器设计,实现了 WAV、FLAC、APE 三种无损格式高保真音乐的播放,提升了嵌入式系统音乐播放的视

听感受。限于 S3C6410 芯片的 AC'97 音频控制模块最高只支持 48 kHz 采样频率的音频，所以造成了高保真无损音乐播放的采样率瓶颈。可以考虑使用 S3C6410 芯片的 IIS 音频接口 采样位数支持 8/16/24 bit，采样率支持从 8 kHz 到 192 kHz）外接高品质音频解码芯片[8]的方案，获得更好的高保真无损音乐播放体验。

参考文献：

[1] 郁峰. 基于嵌入式文件系统的 MP3 播放器的设计和实现[D]. 苏州 苏州大学，2009.

[2] 於少峰，严菊明，胡晨. 基于 AC97 标准的嵌入式音频系统设计与实现[J]. 电子器件，2004，27(4)：733-736.

[3] 焦正才，樊文侠. 基于 Qt/Embedded 的 MP3 音乐播放器的设计与实现[J]. 电子设计工程，2012，20(7)：148-150.

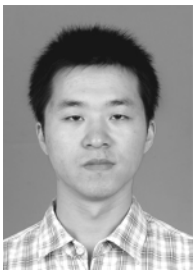
[4] Theory of Monkey's Audio[online]. http://www.monkeysaudio.com/theory.html.

[5] 李彬. 基于应用程序的嵌入式 Linux 内核自动裁剪[D]. 东南大学 计算机应用专业，2006.

[6] 查婧，刘波，曹剑中. Linux 内核在 S3C2440 上移植的方法[J]. 电子器件，2009，32(4)：844-845.

[7] Jasmine Blancehette，Mark Summerfield. C++ GUI QT4 programming(The second Edition)M. 2004：36-68.

[8] 章坚武，董平，马勇. 一种嵌入式多媒体播放器的硬件设计与实现[J]. 电子器件，2006，29(4)：1123-1125.



陈自龙（1988-）男，汉族，江苏南京人，东南大学本科（硕士在读）主要研究方向为嵌入式系统研究与应用，dkpure@126.com；

汤勇明（1973-）男，汉族，江苏省江都市人，东南大学研究员，主要研究方向为电子电路设计，tym@seu.edu.cn。

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

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

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

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

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
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

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

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

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