文章编号：1005-8451（2011）10-0035-03

嵌入式车载流媒体播放终端设计与实现

段录平，李 锐，李 伦
（株洲南车时代电气股份有限公司 技术中心，株洲 412001）

摘 要：本文设计并实现了基于Au1250处理器的嵌入式流媒体播放终端。播放系统由流传输设施和解码终端组成。流传输设施主要实现媒体数据的传输，解码终端负责流媒体的音频、视频解码及播出。流传输层利用JRTPLIB开发库实现，解码终端采用嵌入式技术，通过MAI Engine实现媒体数据的解码播放。对于文本数据，直接交由上层应用程序的GUI显示。播放终端与媒体服务器之间通过以太网相连，采用RTP协议进行数据通信。

关键词：流媒体；Au1250；实时传输协议；JRTPLib
中图分类号：U285 **文献标识码**：A

Design and implementation of embedded on-board player terminal for streaming media

DUAN Lu-ping, LI Rui, LI Lun
(Technology Center, ZhuZhou CSR Times Electric Co., Ltd., Zhuzhou 412001, China)

Abstract: In order to implement playing multimedia with a streaming mode on vehicles, an embedded terminal for streaming media based on Au1250 processor was designed. The architecture of the player consisted of stream transport infrastructure and decoder terminal. The former was responsible for transporting media, and the later for decoding and playing. A development library named JRTPLib was used for developing the transporting layer. With the support of MAI engine, decoder terminal adopted embedded technology to implement decoding and playing. For text data, it was delivered directly to upper application 's GUI to display. The player terminal was connected to media server by Ethernet. RTP was adopted for data communication.

Key words: streaming media; Au1250; RTP; JRTPLib

车载多媒体应用非常广泛，例如车载导航系统、车载闭路电视等，都涉及到音频、视频的处理。虽然应用较多，但是基于流媒体技术的车载应用案例却不多见，原因在于实现流媒体技术的难度相对较大。但基于流媒体技术的播放系统有很多优点，如实时、可交互、易扩展等。为此，本文设计了一种适合车载的嵌入式流媒体播放解决方案。

1 流传输层设计与实现

1.1 RTP协议

1996年，由民间自发成立的互联网工程任务组（IETF）在RFC1889文档中制定并公布了一个网络实时传输协议RTP（Real-time Transport Pro-tocol）。它是专门为交互式音频、视频、仿真数据等实时媒体应用而设计的协议，已广泛应用于各种多媒体流传输系统中[1]。

收稿日期：2010-12-17
作者简介：段录平，工程师；李 锐，高级工程师。

RTP协议是进行实时流媒体传输的标准协议和关键技术。它位于UDP之上，是应用层的网络传输协议，适合通过组播和点播传送实时数据。但RTP本身不能提供任何传输可靠性的保证和流量的拥塞控制机制，这些服务需要依靠RTCP（RTP Control Protocol）来提供。

1.2 RTP流传输模型

实时流传输模型由流媒体服务器和解码终端构成，如图1。服务器端负责媒体文件的RTP打包、发送，实现媒体数据的流传输。解码终端负责RTP数据包接收和处理，并分析RTCP数据包得到当前网络状态，并向服务器端反馈接收质量。

1.3 RTP流传输实现

实现流传输可以利用一些开源免费的RTP库来实现，如JRTPLIB、LIBRTP、LiveMedia等。JRTPLIB是一个用C++语言实现的开源免费的RTP开发库，可运行在多个操作系统上。使用该库编程时可以不用关心RTCP数据包发送和接收的细节，这些都由JRTPLIB自动完成，因此可简

化编程。

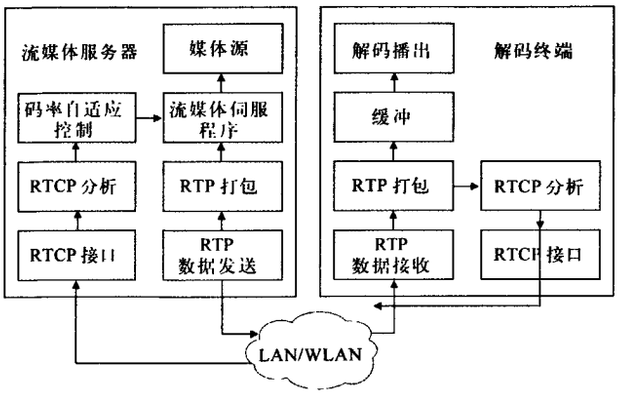

图1 RTP实时流传输模型

1.3.1 发送端

初始化会话：使用RTPSession类的构造函数创建一个会话对象实例，而后调用该类的Create()方法来该对象实例进行初始化。Create()方法有2个参数：(1) RTPSessionParams类型的参数，用于设置时间戳单元，其为必须设置的参数，不然创建会话对象失败；(2) 设定传输规则的参数，默认情况下为UDPIPV4传输协议。

设置会话参数：RTP会话被成功创建后，首先需要设置数据发送的目标地址和目标端口。通过调用RTPSession类的AddDestination()方法为该会话加入一个目标地址和端口，允许同一会话存在多个目标参与者。调用DeleteDestination()和ClearDestination()方法来删除会话参与者。对于同一个RTP会话，负载类型、标识和时间戳增量通常都是相同的，可调用SetDefaultPayloadType()、SetDefaultMark()和SetDefaultTimeStamp-Increment()方法来设置这些默认参数。

发送数据：设置完会话参数后，调用RTPSession类的SendPacket()方法发送数据。若SendPacket()方法被成功调用，RTCP数据报将自动被发送。

结束会话：数据传输结束后，必须调用BYEDestroy()函数结束RTP会话过程。

1.3.2 接收端

初始化：初始化过程与发送端相似，不同的是不需要设定目标地址和目标端口。

遍历数据源：首先需要调用RTPSession类的PollData()方法接收所有源发送过来的RTP/RTCP数据报，再调用RTPSession类的GotoFirst-SourceWithData()和GotoNextSource-WithData()方法来遍历那些带有数据的源。

抽取数据：检测出有效的数据源之后，调用RTPSession类的GetNextPacket()方法返回一个RTPPacket类型的指针。通过这个指针调用Get-PayloadData()方法获取实际需要的数据。还有其他相关函数可从数据报中提取出SSRC、时间戳、CSRC以及负载类型等信息。数据提取完成之后必须调用DeletePacket()来释放RTPPacket内存。

2 播放终端硬件设计

2.1 硬件平台方案

Au1250有以下几个显著的优点：

(1) 处理器工作频率较高。最高可达700 MHz。其自带的DDR SDRAM控制器最高支持DDR400和DDR2 533，为高速数据处理提供支持。(2) 功耗低。提供2种休眠模式，降低了系统功耗，减少发热量。(3) 集成多媒体加速引擎(MAE)。不需要外部DSP电路就能实现视频的硬件解码播放，可减少外部视频处理的复杂度。硬件上支持MPEG2、MPEG4、H.263、WMV9等多种视频解码。(4) 集成的静态总线控制器，支持NorFlash、NandFlash、IDE、CF/PCMCIA等多种存储外设。只需要很少的外部器件即可方便扩展系统存储。(5)外围接口丰富。包括GPIO、USB、UART、VGA、LCD、Audio、SD以及摄像头等接口。

2.2 硬件设计

Au1250的主电路由处理器、电源、存储器、外设接口等组成。

(1) 处理器：采用Alchemy Au1250，这是一款MIPS架构的低功耗、高性能处理器，支持Win CE，Linux等操作系统，主频可达700MHz功耗低于400 mw，集成媒体加速引擎MAE，支持多种媒体格式。

(2) 电源：解码板使用4种电压，5 V，3.3 V，1.8 V和1.2 V。5 V是解码板的输入电压，采用3颗TPS62040开关电源芯片将5V转为3.3 V，1.8 V和1.2 V，最大输出电流为1.2 A。

(3) 存储器：主要是内存储器。内存由Flash和DDR2共同组成，主要功能是存储并运行BootLoader、驱动程序以及应用程序等。

(4) 外设接口：采用的接口有RS232串口、10/100M、VGA，Audio以及USB口。串口主要用于底层程序调试和下载等。

3 播放应用软件设计

3.1 软件开发环境

整个软件开发是基于Linux操作系统。RTP传输层实现采用第三方开源库 JrtpLib v3.4.0，它完全遵循RFC标准设计，支持Linux、Windows等多个操作系统平台。播放应用软件开发需要 RMI官方提供的MAI Engine SDK支持，它为音频、视频软硬件解码提供接口支持。

3.2 图形用户界面设计

Au1250芯片集成了硬件解码单元MAE，视频数据经过MAE硬解码后被直接输送到VGA显示。车载应用需借助GUI来显示。嵌入式GUI设计有多种开发工具可选，如Qt、MiniGUI等。

3.3 软件模块设计

基于MAE的播放系统主要包括4个基础模块：File Reader模块、Demux模块、Decoder模块和Render模块。为实现RTP流播放，需要在原播放系统的基础上增加一个RTP接收处理模块。除RTP模块外，其它模块都以共享库的方式加载，装载后以线程的形式独立运行，而模块与模块之间共用一块缓冲区，通过PV操作模式进行数据流的传输，由MAI Engine负责协调。

(1) RTP模块

该模块是利用JRTPLib库开发，主要实现RTP流的接收和处理，并将处理后的数据写入文件缓冲区供Reader模块使用。

(2) Reader模块

该模块主要实现从文件缓冲区读取RTP模块处理过的数据，按照Demux模块的数据缓冲区大小要求写入该缓冲区，并对流媒体数据类型进行检测，依据检测结果调用相应的库文件进行解复用。提供的接口主要有MAICompSetStream()设定数据源，MAICompReadBuffer()读缓冲区和MAICompWriteBuffer()写缓冲区。调用MAIengine_AutoConnect()引擎接口可以自动完成Demux、Decoder和Render功能。

(3) Demux模块

该模块从缓存区读到数据后根据多媒体文件的类型动态加载相应的Demux库文件对数据进行解析复用，分解后的音频流送Audio decoder进行解码，视频流则送VideoDecoder进行解码。支持库包括Libmaimpeg2demux.so和Libmaimpeg4demux.so。Demux模块内部调用MAIDemux_GetBuffer()读缓冲区数据，并调用Demux-Thread()函数对数据进行解复用，最后调用MAIDemux_PutBuffer()写缓冲区。

(4) Decoder模块

该模块根据解码复用后的音频、视频格式装载相应的解码库文件。对于音频数据的解码由Audio-Decoder模块完成，音频解码后直接送往Audio-Render模块驱动硬件发出声音。视频数据则由Video-Decoder模块初步解码后送到MAE的前端进行处理，前端处理不支持的视频格式(如H.264)须经软件解码形成YUV数据，再将YUV数据输送到MAE的后端做处理。

(5) Render模块

该模块主要实现对音频、视频数据进行渲染并实现同步。音频数据渲染后送到音频输出设备，视频数据渲染由MAE的前端处理完成后，再送到MAE的后端进行解码播出。渲染操作由MAIEngine自动完成，它没有提供用于渲染的上层接口函数。支持渲染的库文件有Libmaiaudrend.so和Libmaividrend.so。

4 结束语

本文提出了一种基于Au1250处理器的嵌入式车载流媒体播放终端设计与实现方案，该方案在硬件方面具有功耗低、支持硬件解码的优点，不需要外围DSP电路即可实现视频数据的硬件解码，从而简化了电路设计；软件方面，利用外部免费开源的JRTPLIB库来实现流媒体传输和接收，在MAI Engine SDK支持的基础上实现对RTP流媒体数据的解码播放功能。

参考文献：

[1] 张占军，韩承德，杨学良. 多媒体实时传输协议RTP[J].计算机工程与应用，2001 (4)：9-11

责任编辑　陈　蓉