摘要

伴随着信息对整个世界的影响越来越大，人类进入了信息时代。集成电路制造工艺的发展，摩尔定律的发现，计算机性能的提高与价格的减少，都揭示着信息技术的进步。现如今，各式各样的电子通知，电子广告已充满了大街小巷。从最起先的烽火和信件，到现在的各种电子屏，信息量的大小，传播和处理的速度以及应用信息的程度都在飞速的增长着。数字电视的普及，多种多样的增值业务也发展起来。然而对于这个新兴的行业，增值业务已不局限在电视领域了，地铁，街头各种各样的广告屏，网络上的广告随处可见。目前，数字电视的广告技术简单原始，对于处理各种各样的需求，出现了很多问题，格式互不兼容，技术上传输层应用层混杂，难以满足客户的灵活多样的需求。

本文提出了一种数字电视多媒体广告系统，能够兼容数字电视标准，将传输层和应用层分离，可以推广到其他各种平台（IPTV 等），可以为了满足客户的需求添加参数和其他设置。本设计包含从前端的发送，到传输，到后端的接收处理。主要的工作在前端读取需要发送的文件，生成文件信息表，并打包成一定的格式的文件，选择对应地址和端口,发送文件。搭建了测试环境，采用了 IPQAM 调制器，实现发送与接收文件，并根据文件进行相关处理。

关键词：数字电视，多媒体广告，IPQAM，IPTV

作者： 王耀

指导老师：张骥

Abstract

With the development of the information technology, we have entered the information age.We have achieved the advancement of information technology due to the development of the IC manufacturing process, the discovery of Moore's Law, the improvement of computer performance and reduction in price. A wide range of e-notification and e-advertising are very popular today.From the beacon fire and letters in the ancient times to the variety of electric screen in the modern times,we can see the development of the dissemination and processing speed of the information as well as the extent of the application of information in rapid speed. With the popularity of digital television, a wide range of additional services have been developed. However, for this emerging industry, additional services are not confined to the field of television. Advertising screen can be seen everywhere. Currently, advertising technology based on digital TV is too simple to deal with a variety of needs such as the incompatibleness of the format and the confusion of the transport layer and the application layer.

This paper presents a digital TV multimedia advertising system, which is compatible with the DVB-C standard and separates the transport layer and application layer , can be transplanted to a variety of other platforms (IPTV...). Its parameters can be added to meet customers' demand. The main process is as follows: get the information of the files to be transmitted, generate file information table and package them into a certain format file, select destination IP and port,send data.I set up a test environment, used IPQAM modulator for sending and receiving files and carried out file processing.

Keywords: digital television, multimedia advertising, IPQAM, IPTV

Written by Yao Wang

Supervised by Ji Zhang

目录

第一章 绪论

1.1 引言

数字电视网络电话等其他电子科技的推广与普及，使得我们的生活进入了一个全新的时代。从古代，人们使用烽火飞鸽传书传递信息，到后来的电报电话，到现在的网络等等各种高科技的通讯手段传递信息。传播和处理信息的方式在不断变化，并且信息量的大小，传播和处理的速度以及应用信息的程度都在飞速的增长。现在地铁街头路边随处可见的信息屏，以及电视网络都在不停的向我们传递信息，我们的生活离不开信息的传递。

数字电视多媒体广告也成为了传播各种信息的有效途径，游走字幕，开机广告，挂角广告等各种各样的广告都在给我们传递着信息，给我们的生活带来方便。本设计为数字电视多媒体广告系统，包含有字幕，图片，动态图以及视频等多种样式的广告，并可以根据用户要求自行添加其他种类的广告，也可以移植到其他平台使用，比较灵活。并且有以下优点：

1.各种参数完全可配置，传递文件时，针对有线电视单向网络出现包丢失情况采用重复发送机制，以实现文件接收。

2.支持多种文件格式，并可以根据实际要求添加其他文件格式，也可以完全定制自己的文件格式，本设计只对常用的格式进行研究。

3.本设计基于数字电视，也可以移植到其他平台（IPTV）上面，通过不同的传输方式将文件以及文件信息表传递到终端，再进行处理显示。

4.统一管理这种需要传输的文件信息，比较方便，避免了很多格式不支持，部分或所有广告文件丢失的问题。

5.成本低，无需其他硬件设备支持，只需要在 PC 端运行软件，读取广告文件，配置显示方式速度重复次数等相关信息，即可按照设定的模式运行。

6.修改方便，前端的改动，几乎可以瞬间在终端体现出来，在紧急情况时，可以将紧急通知快速发送至终端。

1.2 课题研究背景和意义

随着人们的生活水平的提高，人们已经不满足于小康的标准，都在不停的提高自己的生活质量，购置高档的生活用品。各种厂家商家也在积极制造改善人们生活水平的设备用品，不止是为了提高生活水平的用品，其他还有很多比如各种科研仪器等设备。然而对于如此大的世界，如何将商家的信息及时的传递给人们，并且需要很大一部分人接收到商家发出的信息。一个有效的传输途径就是通过数字电视。

基于数字电视的多媒体广告，可以有效的将多种格式的广告文件传递到用户家里，比如游走字幕，可以在电视屏幕顶端或低端循环显示一条或多条文本信息，并可以控制在一定的时段显示；另外的如 EPG 广告，可以在用户开启机顶盒加载机顶盒软件的时候显示一些广告商提供的图片广告，有效的利用了用户等待机顶盒软件启动的时间；另外如音量条右侧或者挂角的广告，都在不停的将广告传递给客户。

现有广告系统都基于互不兼容的私有协议，传输层与应用层混杂，在实际应用当中造成大量的问题，例如兼容性差、调试困难。本文提出了一种简单有效的将各种广告统一管理传输的思想，在前端发送界面读取相关文件信息，生成文件信息表，然后根据传输方式将文件以及文件信息表打包成需要的格式，然后发送，终端接收到，解出相关的文件，以及配置信息。根据配置信息在不同的位置，不同的时间，将广告商提供的信息传递到用户。

1.3 本课题目标和主要内容

1.3.1 课题研究目标

本课题设计一套数字电视多媒体广告系统，统一的管理各种多媒体广告，可以处理任何格式的广告文件，并且可以移植到其他平台上面使用。可以处理多个广告文件，按照指定的显示方式在终端显示出来。

1.3.2 课题主要完成的工作

本课题由 PC 机上位机读取文件信息，生成文件信息表，再将文件打包成需要的

格式，通过 IPQAM 调制信号，使用机顶盒接收信息，解出文件，再按照指定的显示方式显示。

课题工作主要分为软件和测试环境的搭建两个部分。

软件部分：

1.使用 Labwindows 编写上位机软件，实现从上位机读取文件，生成文件信息表。

2.对广告文件和文件信息表进行打包，打包成指定格式的 TS 文件，满足 DVB-C 标准。本设计采用的是包长为 188 字节，包 PID 为 0x1CC0。

3.混流程序的编写。将生成的 TS 流文件与原始视频流文件混合，替换原始视频流里面的部分空包。

4.发流程序的编写。向 IPQAM 的数据端口发送数据包，支持最多 8 路信号传输，并且按照指定的码率进行传输。

环境搭建部分：

1.建立环境

2.配置 IPQAM

1.4 论文结构安排

第一章为本论文的绪论部分，介绍了背景和意义以及本论文需要完成的工作；第二章介绍了本设计的总体方案；第三章详细描述了硬件环境的搭建和仪器配置；第四章主要介绍软件部分的设计；第五章为测试结果；最后在第六章做了总结。

第二章 总体方案

本系统由 Labwindows 编写的上位机，PC 端文件打包软件，TS 文件处理软件，IPQAM 发包程序等其他程序组成软件部分，由 Linux，Windows 机器，路由器，交换机，IPQAM，机顶盒和电视机等组成的硬件环境。

硬件环境部分主要的功能是将电脑 IPQAM 和机顶盒连接起来，所有设备都必须连入局域网，为了模拟实际情况，将某些设备放在较远的位置，并使用较长的线进行连接。本设计用到两个交换机，每个交换机负责一个实验室的设备，两个交换机之间使用千兆网线互联。另外使用路由器自动给连入局域网的设备分配 IP。由于 IPQAM 不支持自动获取 IP,所以要在 IPQAM 的硬件配置界面对 IPQAM 进行配置。除了网线的连接，机顶盒与 IPQAM 之间需要有线电视 Cable 线进行连接，并且可以使用功分器将有线电视信号分支出来，接入多个机顶盒构成有线电视网络。

软件的主要功能为从编写的软件界面获取广告文件，分析文件数据，生成文件信息表，再将文件信息表文件和广告文件按照用户指定的格式装入数据包，在不违背 MPEG-2 标准的前提下，所有参数可以根据用户的需求设定。同时本设计提供了混流程序，将生成的广告流文件附在原始节目流里面，使用替换原始节目流里面的空包的方法混入广告流，该方法不会对原始节目流里面的业务信息表以及音视频数据带来任何影响。

第三章 硬件环境的搭建和仪器配置

3.1 硬件环境的主要组成部分及连线

硬件主要由 Linux，Windows 机器，路由器，交换机，IPQAM，机顶盒和电视机等组成。

其中 Windows 机器用于编写上位机软件，处理文件信息，发送文件。Linux 机器比较容易对时间进行精确的控制，用于给 IPQAM 发送多路信号，也可以登录机顶盒，查看机顶盒状态，调试机顶盒，出现问题可以辅助查找问题所在。路由器交换机用来搭建千兆网络环境，分配 IP 等功能。IPQAM 接收网络端发来的数据包，将数据包调至在一定的频点上，经过有线电视线传播。机顶盒接收到有线电视传来的信号，解调信号，提取出广告文件，并将广告内容在电视机上面做出显示。硬件连接图如下图所示：

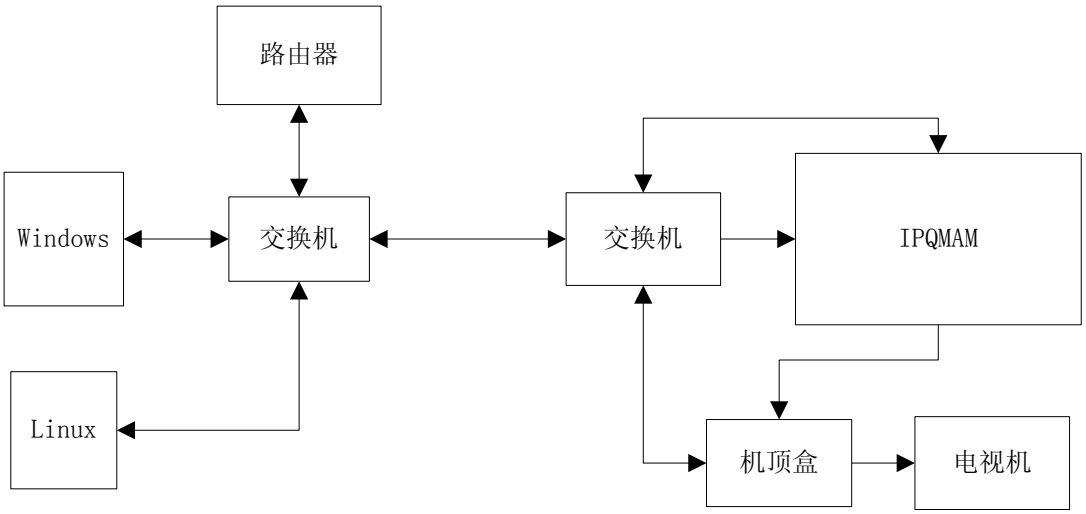

图 3-1 硬件连接图

其中 IPQAM 与机顶盒之间的使用有线电视先进行连接，可以通过分配器功分器分出多路信号，连接到多个机顶盒；机顶盒与电视间的连线可以根据实际的机顶盒与电视支持的接口进行连接，本设计使用 AV 线连接机顶盒与电视机。交换机选择千兆交换机，所有的网线全部使用千兆的网线。路由器用于给电脑以及机顶盒分配 IP。交换机和 IPQAM 之间使用了两根网线连接，主要是因为 IPQAM 有数据端口和控制

端口，这两个端口在硬件上面是独立的。除了交换机与机顶盒之间的信号和机顶盒与电视机之间的信号是单向的，其他信号的传递均为双向。

3.2 上位机软件运行平台的选择

根据设计目标，需要编写上位机软件，对文件进行相关处理，并发送文件。本设计选择使用 Windows 平台下的 Labwindows 软件。Windows 操作系统简单易用，可以根据该平台的简单易用的性质实现很多功能，比如 Windows 有很多优秀的软件开发平台和各种辅助开发软件。

另外本设计也选用了一台 Linux 机器，主要是因为本设计使用的机顶盒是基于 Linux 系统开发，通过网线连接，登录至机顶盒，可以查看机顶盒状态，辅助调试软件。

3.3 IPQAM 的选择与配置

建立本设计需要的工作环境，最主要的一部分就是调制器，本设计选择使用 IP 调制器 IPQ6800。

3.3.1 IPQ6800 简单介绍

IPQ6800 是一款高品质 IPQAM 边缘调制器，支持单播，广播，VOD 点播和数据等业务。该设备充分利用了千兆以太网络的带宽，支持最多 48 个 QAM 频道。该调制器集复用、加扰、调制、频率变换四个功能为一体，将来自数据端口输入的节目流重新复用在指定的传输流中，再使用 QAM 调制以及频率变化，最终输出射频信号。

3.3.2 IPQ6800 的配置

该设备支持多种工作方式，需要指定射频频率，管理口 IP 地址，数据口 IP 地址以及复用加扰等一些参数和选项需要设置。

在浏览器中打开 IPQAM 控制端口 IP，输入用户名和密码之后可以登录到 IPQAM 控制首页如下图所示：

图 3-2 IPQAM 配置主页面

如上图所示，可以看到设备名称设备编码标识号等相关信息，其中最为重要的是数据口 IP 地址和管理口 IP 地址（远程）。另外可以看到首页、输入、复用、加扰、输出、状态、系统等七个标签页。点击系统，数据网口设置，可以根据局域网的网段设置一定的数据端口 IP，点击管理网口设置可以设置管理网口 IP，其中管理网口 IP 分为远程管理网口和本地管理网口。由于该设备不支持自动获取 IP，所以 IP 一定要手动设置。

设置好数据端口和控制端口 IP 之后，需要设置 IPQAM 的数据端口号，该设备支持最多 8 路节目源，由于只有一个 IP 地址，只能通过端口号来区分 8 路节目源，点击输入选项卡，可以看到网络节目源设置页面，打开 8 路节目源开关，并设置端口号分别为 1201 到 1208，主输入源地址设置成刚刚设置的数据端口 IP，输入源类型选择 UDP 点对点，选择应用保存设置信息。设置成功后的页面如下图所示：

图 3-3 端口设置页面

接下来需要设置复用通道，选择复用选项卡，勾选该通道旁通设置，选择输入源，为了便于记忆和管理，复用通道一就选择数据端口号 1201 为输入源，复用通道二选择数据端口号 1202 作为输入源，依次类推至复用通道八。设置结束后的页面如下图所示：

图 3-4 复用通道设置页面

最后需要设置的频点的设置，点击输出选项卡，选择 QAM 输出，勾选所有八个频点的开关选项，QAM 模式选择 QAM64，符号率设置为 6875，电平设置为 85dbuv。最主要的一项为中心频率的设置，由于国家的规定，有线电视的频率需要设置在几个固定的频率点上，不可以设置为其他的频点。另外根据该 QAM 的设置，只能设置两个中心频率，其他每 3 个中心频率按照设置的中心频率依次加 8 兆赫。由于有线电视机顶盒选择 403 兆赫频点作为中心频点，所以我选择 403 兆赫频点作为第一个中心频率。另外 506 兆赫也是比较常用的一个频点，所以将第二个中心频点设置为 506 兆赫。QAM 输出设置完毕，点击提交按钮保存设置，最终的页面显示如下图所示：

图 3-5 QAM 设置页面

3.3.3 IPQAM 状态检查

使用 IPQAM 发包程序（后面会陈述）向 IPQAM 数据端口发送节目流，在状态选项卡里面可以看到每个频点的信息，如下图所示：

图 3-6 状态查看界面

可以看到每个频点的复用状态显示为正常，同步时钟也是显示为正常，使用发包程序的配置是每个频点的比特率都是 38Mbps，这里可以看到码率在 38M 上下浮动，由于本设计每次向调制器发送 7 个数据包，每包为 188 字节，所以会带来细微的差异，

不过发包程序每次发送数据都会重新计算时间，不会带来积累误差，瞬时码率会在38M 左右波动，不过平均码率一定很接近设定的码率。

另外如图 3-3 所示，可以点开节目信息按钮，可以看到每个输入源的 TS 流状态和节目信息，点开之后的界面如下图所示：

图 3-7 节目信息查看

由上图可以看到 TS 流的状态为 YES，并可以看到节目的信息，说明调制器工作正常。

3.4 局域网的搭建及设置

3.4.1 局域网拓扑结构

本设计需要使用两台台式机，IPQAM 和机顶盒，这些设备都需要连接入网络，由于内部的高速数据传输不可以收到外网的影响，又需要外网联入某些设备起到控制作用，根据需要，设计如下的网络拓扑图：

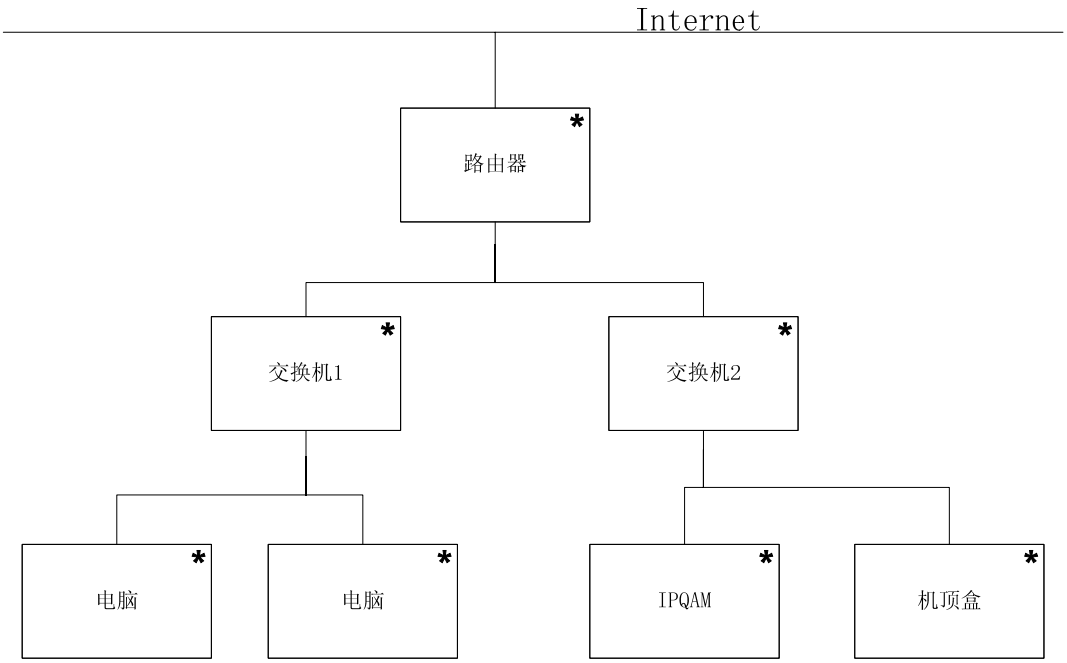

图 3-8 局域网的网络拓扑图

上图仅表示网络连接，不包含除网络连接其他的线路连接。路由器链接外网，可以通过外部控制端口对 IPQAM 进行控制，并可以通过外部端口对电脑以及机顶盒做相关设置。由于数据量巨大，要保证 IPQAM 正常工作需要提供至少 329M 的网络带宽（对于包长为 204 字节的 TS 包，码率为 41.2M，IPQAM 设备支持最大 8 路数据的传输），所以交换机选择千兆的以太网交换机，并且所有网线均使用千兆的标准。

3.4.2 网络 IP 的设置

由于 IPQAM 不支持自动获取 IP，所以 IPQAM 网络 IP 的设置需要与路由器的设置一致，根据习惯我将局域网的 IP 设置在 192.168.1 号段，由于本实验室外网的 IP 不在此号段，所以外网不会对内网造成影响；否则要避开外网的 IP 号段，重新设置局域网 IP。使用路由的另一个重要的原因就是路由可以自动给连接到路由器上面的设备分配 IP，本设计使用的两台电脑和机顶盒都可以自动从路由器获取到 IP。

设置完毕以后，既可以通过远程控制端口对局域网内部设备进行控制，由于控制信号占的网络带宽极小，所以也不会影响局域网内部的高速数据传输。另外如果需要使用多台电脑或使用多个机顶盒进行测试，也可以自动的分配到 IP 地址。

3.5 终端设备与显示设备

本设计的终端设备选择使用北京迈视的 MB-2000 机顶盒，该机顶盒基于 Linux 操作系统开发。显示设备选择电视机。将网络端口和有线电视端口接在机顶盒上面，再使用 AV 线将机顶盒与电视连接。

第四章 系统软件部分的设计

本设计主要部分为软件部分。软件部分主要包含上位机的编程和接收端的处理。由于本设计以数字电视多媒体广告系统为主，接收端的设备的设计不在此设计的范围之内，所以仅提供接收端的文件提取的相关程序。本设计使用的接收端为北京迈视的MB-2000 机顶盒。

4.1 软件部分使用的开发环境

使用 C-Free 编写打包程序，文件信息表生成工具，混流工具及其他流处理软件。使用 Labwindows 编写上位机软件，调用之前的一些工具，生成最终的文件和 IPQAM 配置文件。Linux 平台下提供了 IPQAM 发包程序，最多将 8 路数据信息传送给 IPQAM。

4.1.1 Labwindows 的介绍

上位机软件设计软件选择使用 Windows 平台下的 Labwindows 软件。该软件是美国国家仪器公司推出的交互式 C 语言开发平台，在 Labwindows 的下可以利用其提供的丰富的库函数来满足各种设计和验证的需要。

使用该软件的用户界面编辑器可以编辑图形界面，并可以在程序内部使用库函数对图形界面里面的控件属性进行修改或者添加删除控件。Labwindows 也提供了丰富的函数库，利用这些库不仅可以完成常规的软件设计，也可以完成一些复杂的数据采集和一起控制系统的开发。此外,为 GUI 的设计 Labwindows 设计了很多的如曲线图控件、表头、指示灯等专业控件。

4.1.2 C-Free 的介绍

C-Free 是一款支持多种编译器的集成开发环境，使用 C-Free，开发者可以轻松的编辑、编译、运行并调试程序。

主要有以下特征：

1.支持多种编译器，并且可以根据需要配置添加其他编译器。

2.提供了增强的语法加亮器和只能输入功能，可以添加工程类型，并且可以定值其他工程的向导。

3.完善的代码定位功能，函数或者变量的查找声明实现和引用都变得简单明了。

4.代码完成功能和和函数参数提示功能。

5.另外，最新的 C-Free 版本已经可以支持 c99 标准。

4.2 上位机软件设计

上位机软件的设计主要包含图形界面的设计和功能的编程。

4.2.1 图形界面的设计

根据设计的需求以及实际运行软件后的效果做出修改，最终 GUI 的设计如下图所示：

图 4-1 用户界面

如上图所示，有一个名称为文件列表的 listbox，点击添加文件按钮，会弹出文件选择的对话框，如图 4-2 所示，定位到相关目录，选择相关文件双击即可。选中列表里面某个文件，点击移除文件按钮，即可将此文件从文件列表内部删除。

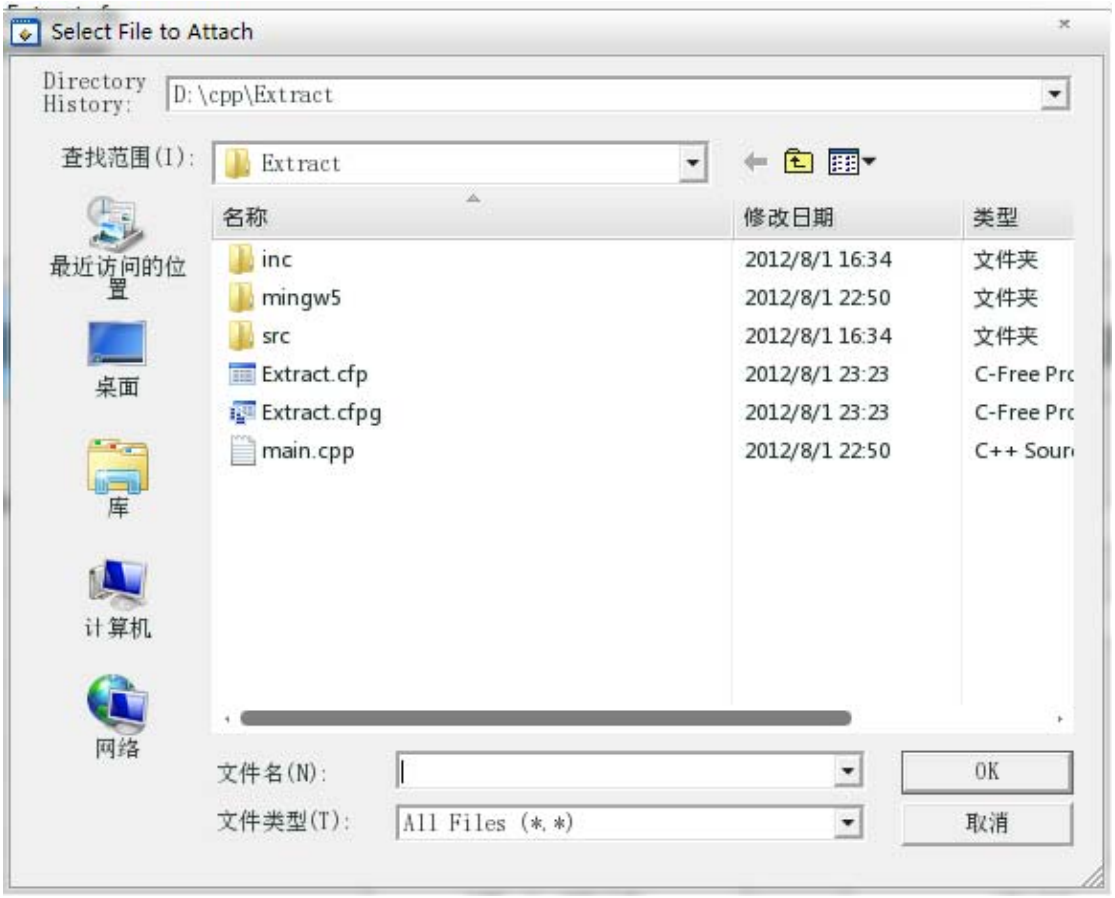

图 4-2 文件选择框

生成文件信息表按钮用于生成文件信息表，点击之后会读取列表内部的文件路径，打开对应文件，获取文件信息，并按照一定格式生成 XML 文件。文件的信息全部软件刚启动生成文件列表按钮是不可以操作的，在每次成功添加文件或者移除文件之后该按钮可以操作。

广告 PID，指定了传输广告时采用的 PID，默认设置为 0x1CC0，如果与节目流文件内部 PID 有冲突可以修改次数值。

生成的文件信息表文件和广告文件信息全部存在同一个 PID 的数据包内部，为了从流中提取出各个不同的文件，设定了 XML 文件 TID 和文件起始 TID 两个参数，其中 XML 文件 TID 默认设置为 0，文件起始 TID 默认设置为 1。文件起始 TID 为第一个文件的 TID，第二个文件的 TID 按照每次增加 1 的规律递增。为了避免引起不必要

的冲突，如果设置文件起始 TID 比 XML 文件 TID 小时，会取消设置并弹出错误提示。

由于本设计是基于数字电视的，需要满足 DVB-C 标准，数据的传输就要符合标准的规定。文件在传输的过程中是分成一个或是多个 section，再由 section 分成一个或是多个 TS 包，根据标准的规定，TS 包长为 188 字节或是 204 字节两种。本设计默认每个 section 由 6 个包组成，每个包包长为 188 字节。

右侧为节目流文件、IP 地址、数据端口、频点等几个控件。点击节目流文件会弹出一个选择文件的对话框，选择文件之后，文件的路径名称会显示在后面的空白里面，如果第二次选择文件会覆盖第一次选择的文件，也就是说，这个节目流文件只有一个。

IP 地址和数据端口两个控件，可以选择数据发送 IP 和数据发送 IP 的端口，由于可能 IPQAM 的配置会有所变化，所以这里提供修改这些参数的功能，默认的 IP 为 192.168.1.210，该 IP 为本设计中设置的 IPQAM 的数据端 IP。数据端口为一个下拉列表，可以选择 1201 到 1208 当中的一个端口，这里端口的设置与 IPQAM 的设置一致，也就是说，可以选择在 IPQAM 的哪一路通道传送广告信息。

频点控件，显示根据默认的 IPQAM 配置，当前数据端口对应的频点信息。如果默认的 IPQAM 的数据端口与频点的对应关系改变了，此控件显示的数据无意义。

4.2.2 功能的编程

该上位机软件包含有添加删除文件功能，生成文件信息表功能，节目流文件的选择功能，参数的获取功能、TID 判断功能以及频点自动显示等多个功能模块。

4.2.2.1 添加删除文件功能

该功能主要实现将需要传输的广告文件添加到文件列表内部，并可以移除某个文件，为了防止意外的操作失误，每次仅可以对一个文件进行操作。

点击添加文件按钮，执行相关回调函数，弹出文件选择对话框，如图 4-2 所示，选择一个文件并点击 OK 按钮或者直接双击文件即可将文件添加到文件列表里面。实现该功能的部分代码如下：

```
int CVICALLBACK AddFileControl (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
```

```
{
    switch (event)
    {
        case EVENT_COMMIT:
        char path[MAX_PATHNAME_LEN];
        if (FileSelectPopup ("", "", "", "Select File to Attach",
                                    VAL_OK_BUTTON, 0, 0, 1, 0, path) <= 0)
            return 0;
        InsertListItem (panel, PANEL_LISTBOXFilelist, -1, path, path);

    SetCtrlAttribute(panel,PANEL_COMMANDBUTTONGenXMLFi,ATTR_DIMMED,0);
    break;
    }
    return 0;
}
```

其中，FileSelectPopup 函数用于点击之后弹出的文件选择框函数，前三个参数选择默认参数，表示默认的路径，显示全部文件类型，以及所有的文件。第四个参数为弹出的选择框的标题，第五个参数表示点击 OK 按钮即表示选择操作。最后一个参数为 path，表示将文件的路径以及文件名称保存在 path 变量内部。最长的文件路径以及文件名为 255 字节。

InsertListItem 函数表示将 path 存储为文件列表控件内部的一个 item，标题与值均为 path 内部存储的字符串，即获取到的文件路径以及文件名。

这样就实现了文件添加的功能。

SetCtrlAttribute 函数将生成文件信息表按钮的不可操作状态取消。在软件刚启动的状态和生成了文件信息表之后的状态，生成文件信息表按钮均为不可操作状态，在修改了文件列表控件里面的 item 时，该按钮可以操作。

实现删除文件功能的部分代码如下：

```
int CVICALLBACK RemoveFileControl (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
        int i;
        GetCtrlIndex (panel, PANEL_LISTBOXFilelist, &i);
        if (i >= 0)
            DeleteListItem (panel, PANEL_LISTBOXFilelist, i, 1);
```

```
            SetCtrlAttribute(panel,PANEL_COMMANDBUTTONGenXMLFi,ATTR_DIMMED,0);
            break;
        }
        return 0;
}
```

GetCtrlIndex 函数在这里的功能是从文件列表里面读出当前选中的文件的索引信息，并存在 i 变量内部。

DeleteListItem 函数将文件列表里面的索引号为 i 的条目删除。配合上面的函数，读取出当前选中的条目的索引号，再删除，实现了删除选中文件的功能。

4.2.2.2 生成文件信息表的功能

4.2.2.2.1 文件信息表格式的规定

本设计中的文件信息表使用标准的 xml 格式，使用 ProgramList 作为 xml 文件的一个元素，该元素可以拥有一个或多个 Program 子元素，Program 元素包含有一个或多个 AdsFile 子元素。每个 AdsFile 元素对应文件列表里面的一个文件，每个 AdsFile 元素含有 Name，Length，category 等子元素，并可以按照实际情况添加其他元素。该 xml 文件树形图如下图所示：

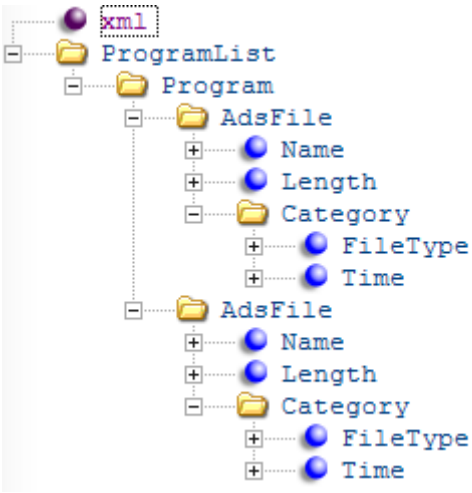

图 4-3 xml 文件的树形图示例

AdsFile 元素拥有的子元素和文件列表中的条目的属性一一对应，一个节目列表可以包含多个节目，每个节目可以包含多个文件，每个文件又有多个属性。这些都和具体的文件对应，把这些信息全部统一在一起，发送给接收端，这样接收端就可以根

据该文件获取前端的配置信息以便于在显示设备中显示。

4.2.2.3 文件信息表生成的实现

对文件列表修改之后，生成文件信息表按钮可以操作，点击该按钮，执行对应的回调函数，从文件列表控件里面读出每个条目，将属性填入文件结构体，并依次写入 xml 文件。这个操作会生成一个 FileList.xml 中间文件，这个文件就是最终生成的文件。实现该功能的软件流程图如下图所示：

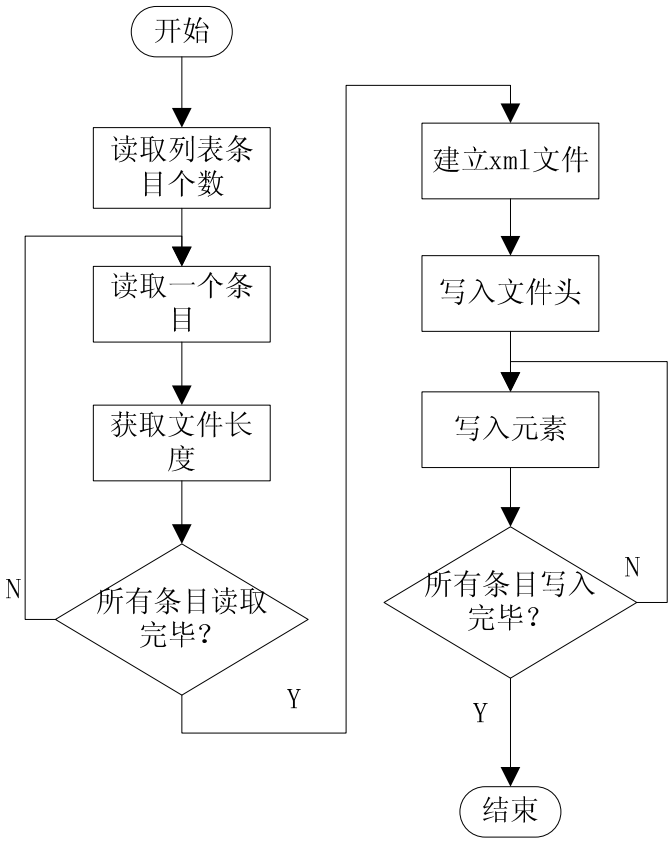

图 4-4 文件信息表生成流程图

实现的部分代码如下所示：

```
GetNumListItems (panel, PANEL_LISTBOXFilelist, &num);
for(i=0;i<num;i++)
{
    GetValueFromIndex (panel, PANEL_LISTBOXFilelist, i, AdsFile[i].FileName);
}
CreateXMLFile(XMLHandle);
for(i=0;i<num;i++)
```

```
MakeFileNode(XMLHandle,AdsFile[i].FileName,AdsFile[i].FileLength,AdsFile[i].FileType,AdsFile[i].DispTimes);
SetCtrlAttribute(panel,PANEL_COMMANDBUTTONGenXMLFi,ATTR_DIMMED,1);
```

如上所示，首先通过 GetNumListItems 函数获取到文件列表内部有几个条目，将次信息存在 num 变量里面，这个信息尤其重要，此变量的数值表示了需要传递的广告的文件的数目。

根据 num 变量依次从文件列表内部读取文件名，根据 ftell 获取文件的大小。

生成 xml 文件，按照指定的格式写入文件名、文件长度、文件类型、广告显示时间等参数。

其中 CreateXMLFile，InsNodeHead，InsNodeEnd 函数的功能分别为建立一个 xml 文件并写入文件版本信息编码格式，插入一个元素起始标识以及插入一个元素结束标识。

4.2.3 节目流文件选择的功能

此功能模块提供选择一个原始节目流的功能，第二次的选择的文件总是会覆盖第一次选择的文件。原始节目流必须为符合 DVB-C 标准的节目流，可以从有线电视线末端抓取。实现的代码如下：

```
int CVICALLBACK SelectTSFile (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
       case EVENT_LEFT_CLICK:
          char path[MAX_PATHNAME_LEN];
          if (FileSelectPopup ("", "", "", "Select File to Attach",
             VAL_OK_BUTTON, 0, 0, 1, 0, path) <= 0)
                return 0;
          SetCtrlVal(panelHandle,PANEL_STRINGTSFile,path);
          break;
    }
    return 0;
}
```

如上所示，左键点击节目流文件控件，弹出选择文件对话框，选择某个文件按确认或双击某个文件，即可选择文件。SetCtrlVal 函数将文件路径以及文件名储存为节

目流文件控件的值。

4.2.4 TID 数值的判断与恢复功能

该功能的主要实现获取 xml 文件的 TableID 数值和广告文件的起始 TableID 的数值，并加以判断。由于 xml 文件唯一，广告文件可能会有很多，所以默认的设置为 xml 文件 TableID 为 0，广告文件的起始 TableID 为 1，第二个广告文件对应的 TableID 为 2，依次增加 1。如果不使用默认的设置，为了避免广告文件的 TableID 和 xml 文件的 TableID 相同，带来文件识别不出的错误，所以本设计要求 xml 文件的 TableID 必须小于广告文件的起始 TableID。根据默认设置，xml 文件的 TableID 为 0，广告文件的起始 TableID 为 1，就不会带来文件识别不出的错误。本设计也对这个数值加以判断，如果不符合要求，那么会弹出一个错误的信息框，告之用户这两个参数设置错误，并提示用户如何设置。点击信息框确认以后会重新装填默认的参数值。实现的主要代码如下：

```
GetCtrlVal(panelHandle,PANEL_STRINGXMLTid,xmlTidTemp);
xmlTid=atoi(xmlTidTemp);

GetCtrlVal(panelHandle,PANEL_STRINGFileTid,adsStartTidTemp);
adsStartTid=atoi(adsStartTidTemp);
if(xmlTid>=adsStartTid)
{
  MessagePopup ("错误","XML 文件 TID 数值应小于文件起始 TID 的数值，请重新设置");
  SetCtrlVal(panelHandle, PANEL_STRINGXMLTid,"0");
SetCtrlVal(panelHandle,PANEL_STRINGFileTid,"1");
  xmlTid=0;
  adsStartTid=1;
}
```

如上面代码所示，修改 xml 文件 TID 和文件起始 TID 其中一个之后，就会回调这个函数，由于这两个控件返回的数据是字符串类型数据，而本设计需要使用的数据为整数型数据，所以每个变量都有两种类型存在与本设计之中，一个是整形的一个是字符串类型的数据。

GetCtrlVal 函数和 atoi 函数配合使用，将数据先读到一个字符串内部，在转换成整形数据，一共操作两次，分别对应着 xml 文件 TID 和广告起始 TID 的获取。

获取了 TID 之后需要保证参数符合本设计的规定，所以，后面对这两个参数加以判断，如果后者的数值小于或等于前者，就会弹出错误的提示框，提示用户如何设定这两个参数才符合设计的规定。

如果参数不符合设计规定，会重新装填这两个控件的值，并恢复软件内部变量为默认值。SetCtrlVal 函数用于重新装填控件的值，最后两句用于恢复软件内部的变量值。

4.2.5 频点自动显示功能

在选择了数据 IP 之后，需要选择数据端口号，根据本设计 IPQAM 的设定，有效的数据端口号为 1201 到 1208，分别对应 403，411，419，427，506，514，522，530 兆赫兹八个频率，为了方便使用，一般使用频率做区分，所以设计了这个功能，在选择端口号的同时，在频率控件内部装填端口号对应的频点，方便使用。实现该功能的代码如下：

```
int CVICALLBACK Freq (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{
      switch (event)
      {
      int i; char FreqTemp[4];
      case EVENT_COMMIT:
      GetCtrlIndex(panelHandle,PANEL_RING_3,&i);
      if(i<4)
         sprintf(FreqTemp,"%d",403+8*i);
      else
   sprintf(FreqTemp,"%d",506+8*(i-4));
SetCtrlVal(panelHandle,PANEL_STRING_5,FreqTemp);
      break;
      }
      return 0;
}
```

如上所示，设定两个局部变量，获取数据端口当前选中的条目的索引值，根据索引值确定需要在频点控件里面显示的数据。由于本设计使用的 IPQAM 规定，每 4 个频点的频率必须间隔为 8 兆赫并保持相连。根据 IPQAM 的规定，本设计使用了 403 和 506 两个频点作为起始频点，其他每 3 个频点的频率按照这两个频点的频率依次增

加 8 兆赫。

If 条件语句内部确定计算需要显示的数值，并使用 sprintf 语句装填入字符串。最后通过 SetCtrlVal 将字符串里面的内容写入频点控件的值里面。

4.2.6 参数获取功能

点击启动按钮之后，会对生成的文件信息表文件和广告文件按照一定的格式装成 TS 包，TS 包符合 DVB-C 标准，本设计涉及的所有文件均使用同一个 PID，默认的 PID 数值为 0x1CC0，根据实际情况并符合 DVB-C 标准的前提下可以任意修改。Xml 文件的 TID 和文件起始 TID 默认值设定为 0 和 1，在符合本设计的要求的情况下，支持修改这两个参数。

装包过程中需要将文件分成一个或多个段，每个段再分成一个或多个数据包，根据 MPEG-2 协议，数字电视信号的数据包为 188 或者 204 字节两种情况，本设计选择使用的比较多的 188 字节的数据包作为默认的参数。也可以根据需要改成 204 字节包长，但是不可以改成其他数值，如果修改成了其他数值，会跳出提示错误的信息框，选择确定之后，恢复默认的数值 188。

这些参数均可以修改，所以可能不是默认的设置，所以，在点击启动按钮之后，会对所有的参数重新获取，并更新程序内部对应的变量。

4.3 文件打包程序

这个程序实现将广告文件及文件信息表文件按照一定的格式生成一个 TS 文件，需要满足 MPEG-2 标准，作为用户私有数据发送给接收端。这个程序使用 Windows 平台下的 C-Free 编写，并在 Labwindows 下面调用。

4.3.1 文件打包程序步骤

首先从上位机软件的界面获取文件信息，以及生成的文件信息表，和其他相关的参数。然后将文件分割成一个或多个 section，再将 section 分解成多个 188 字节数据包，最后写入文件。这个程序会产生是 ts 格式的中间文件，用于后面混入原始节目

流使用。

实现这个程序的流程图如下：

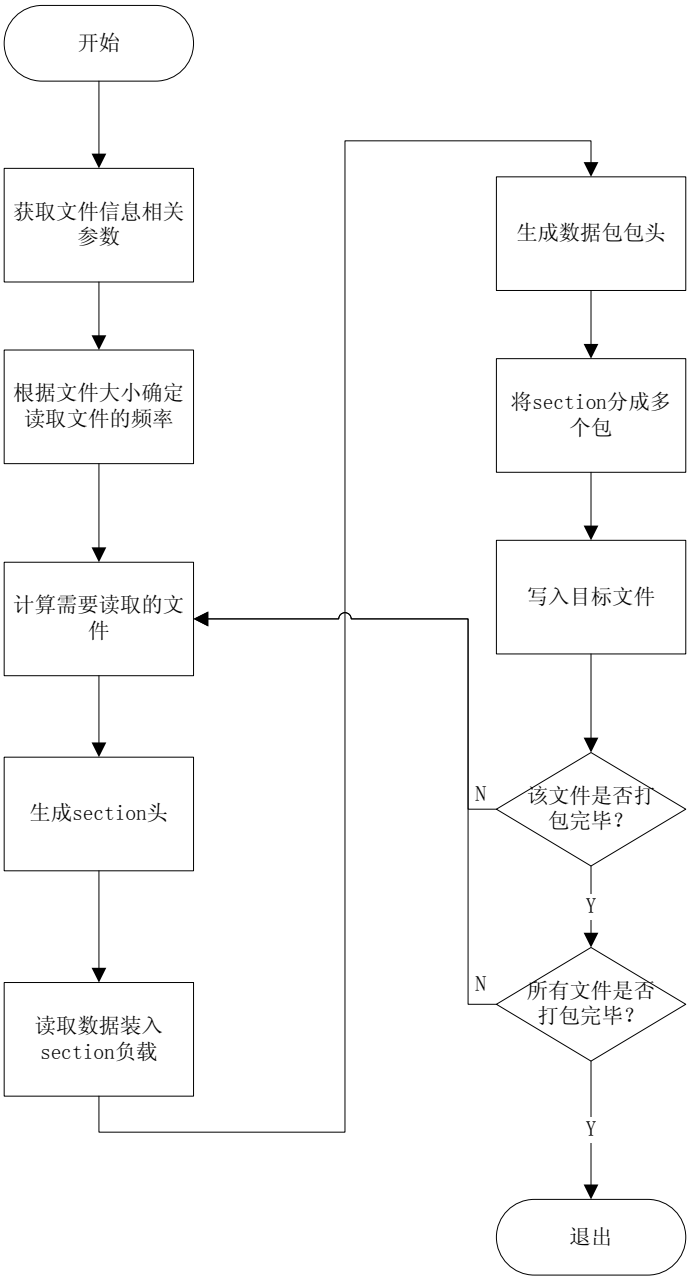

图 4-5 打包文件程序流程图

4.3.2 数据类型与结构体的定义

这个程序定了一些常用的数据类型，以及程序本身需要使用的结构体。下面为为了方便使用而定义的一些常用的类型定义：

```
typedef unsigned int u32;
```

```
typedef unsigned short u16;
typedef unsigned char u8;
```

分别为无符号的 8 位，16 位和 32 位变量。

定义了 FileInfo 结构体，用于存储从上位机界面读取文件信息，结构体的定义如下所示：

```
typedef struct
{
    u8 FileName[255];
    u32 FileSize;
    u32 SectionNumber;
    u32 SectionCounter;
    FILE* Handle;
    u32 ReplayTimes;
    u32 TsSectionNumber;
    u32 TsSectionCounter;
    u32 TsPacketCounter;
    double TempFreq;
    double Pos;
}FileInfo;
```

这个结构体内部包含有 11 个变量，其中 FileName 用于存储文件名，由于 Labwindows 软件固定文件名不得长于 255 个字节，所以这里选择预留 255 字节的空间给文件名使用。FileSize 用于存放文件大小信息，可以看到这个变量的位宽为 32 位，可以最大支持 4G 的文件传输，足够使用。

SectionNumber 用于存放文件被分成多少个 Section 的相关信息，SectionCounter 表示当前已经处理了多少个 section。每次处理结束一个 section 都会对这两个变量的数值进行更新，并比较做出判断，当这两个数值相等的时候，这一次的文件处理完毕。一次文件处理完毕之后清空计数器。

File*为打开文件之后的返回句柄。

ReplayTimes 默认设置为 3，对于数字电视的有线信号传输会出现丢包的现象，考虑到需要将文件完整的传送过去，那么需要将文件重复传输几次，这样文件残缺的概率大大降低了。如果由于硬件问题，丢包率比较大，可以试着增大此参数的数值，可能会改善文件接收不完全的情况，但是如果由于线路的问题导致除了本设计涉及的文件之外其他的数据也不能正确传输，不在本设计讨论范围之内。

TsSectionNumber，TsSectionCounter 这两个变量用于存放该文件在整个目标文件

内部所有的 Section 的个数和计数，其中 TsSectionNumber 数值由 SectionNumber 和 ReplayTimes 相乘得到。每次处理完一个 Section 之后都会更新计数器的值，在这两个数值相等的情况下，表示这个文件处理完毕，之后的软件无需再读取次文件的数据。

TsPacketCounter 表示使用该文件在目标文件内部占有的包的数目。可以查看该变量的值，判断软件是否正常工作。

另外定义了全局变量 AllTsSectionNumber，表示所有的文件的 Section 个数，由每个文件的 TsSectionNumber 累加得到。相对应的定义了全局变量 AllTsSectionCounter 表示程序处理过的所有文件的 Section 个数，两个变量相等时，表示所有文件处理结束，可以退出该程序。

TempFreq，Pos 这两个属性存放计算出的读取文件的频率和当前发送的包所在的时间信息。根据这两个属性，可以按照顺序读取当前最需要处理的文件的 Section，达到每个文件的 Section 在目标文件中均匀分布。TempFreq 的数值由 AllTsSectionNumber 与 TsSectionNumber 相除得到，文件的 Section 个数在所有的 Section 个数的比重越大，那么这个数值就越小。每次处理完一个 Section 都会对 pos 属性做更新，具体操作为将 pos 的数值增大 TempFreq。下一次就读取 pos 数值，哪一个文件的 pos 属性最小，说明这个文件最需要被处理。

定义了 SectionProperty 结构体，主要用于生成 section 头。结构体的定义如下：

```
typedef struct
{
    u8 TableID;
    u8 VersionNumber;
    u16 SectionSize;
    u32 AllSectionNumber;
    u32 AllSectionCounter;
    u32 SectionNumber;
    u32 SectionCounter;
    u32 FileLength;
    u8 SectionData[SECTIONDATASIZE];
    u32 CRC;
}SectionProperty;
```

TableID 指明了这个 section 属于哪一个文件。对于本设计的默认参数，此数值为 0 时，这个 section 为 xml 文件所有，也就是文件信息表文件所有。此数值为 1 或其他数值时，对应的属于第一个或其他的广告文件。

VersionNumber 表示了这个文件的版本号，默认为 1，修改之后这个参数会加一，版本号的意义仅在版本变化时做判断使用，其他时刻这个属性可以忽略。由于一般的 TS 文件内部的 section 都会保留这个字节，所以为了软件的通用性，同样保留了这个字节。MPEG-2 标准内部，这个变量是 4 位的，本设计取这个变量的低 4 位作为有效数据。

SectionSize 表示了这个 Section 的大小，MEPG-2 标准中，这个变量为 12 位的，可以存放的最大值是 4096，去除每个包的包头 4 个字节，以及 section 头一包的一个代表有效负载偏移量的字节，也就是一个 section 最多可以由 22 个包组成。对于 204 字节的数据包，由于该格式的数据包只比 188 字节的数据包多了 16 个字节的校验位，对有效数据并没有影响，所以对于任意包长的包，一个 section 也只能最多由 22 个数据包组成。本设计中，默认设定每个 section 由 6 个数据包组成，计算得到这里的 SectionSize 为 1100。如果在上位机的界面上改动段长参数，那么会在改动后判断此数值是否大于 22，如果大于 22，会出现提示错误的提示框，确认之后恢复默认值 6。

AllSectionNumber，AllSectionCounter，SectionNumber，SectionCounter 这四个变量直接从文件的属性中获取，作为 section 的头存入 section 内部，接收端使用这 4 个变量的信息拼接接收到的 section，最后组成一个文件。

FileLength 从文件的属性中获取，接收端使用这个变量获得文件的长度信息，以便动态申请内存，作为文件缓冲区。

SectionData 用于存放从文件读取的数据，最大为 SECTIONDATASIZE 个字节，SECTIONDATASIZE 可以根据需要变化。这个字符数组存放的就是有效的文件数据，接收到 section 之后，需要去除 section 的头和 CRC 校验字节，将这一部分的数据存入缓冲区。

MPEG-2 标准中，section 的结尾需要加入 CRC32 校验字节。为了符合 MPEG-2 标准，结构体最后的一个参数为 CRC32 的计算结果，装完一个 section 之后，会对这个 section 的除了这四个字节之外的其他数据做 CRC32 计算，并入 section 末端。MPEG-2 标准中使用的 CRC32 表达式为 0x04c11db7。

4.3.3 软件的实现

这部分的软件主要由文件信息获取和处理、计算下一个需要被处理的文件和读数据写数据三个部分组成，其中读数据和写数据部分中包含 CRC 的计算。

文件信息获取和处理，主要是从上位机界面获得参数传递过来，计算一共生成多少个 section，以及每个文件需要处理的次数和频率；计算下一个需要被处理的文件主要是比较每个文件中的 TempFreq 属性，哪一个最小，就表示这个文件最需要被处理。其中，在某个文件被处理结束，那么该文件的 TempFreq 属性将不会再变化，这时根据文件属性中的 TsSectionNumber 和 TsSectionCounter 属性来判断该文件是否处理完毕，如果结束了，那么查看下一个文件是否处理结束。读数据写数据模块，主要的工作是根据上面一个模块的输出，读取相应的文件，由于以文件流的方式读入数据，会保留文件指针，切换文件句柄也不会对指针造成影响，这样就可以在需要的时候切换文件句柄，打到在目标文件内部每个文件的 section 都是均匀分布。

4.3.3.1 文件信息获取和处理

这里的文件信息获取和上位机文件信息的获取不一样。上位机获取文件信息之后生成了文件信息表，存在 xml 文件内部。这里需要将 xml 文件和广告文件的信息一起获取，并加以处理，计算每个文件的各个属性，填入文件结构体。具体流程图如下图所示：

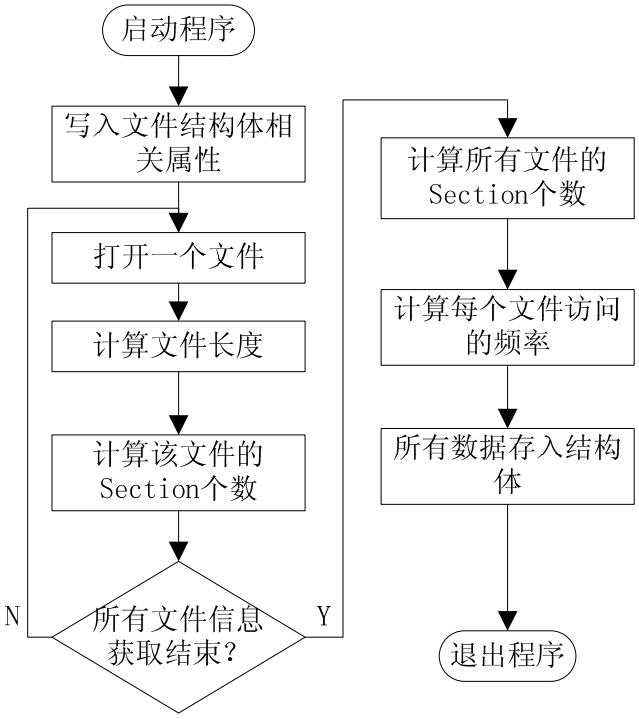

图 4-6 文件信息获取处理流程图

主要实现的代码如下所示：

```
int GetFileInfo(void)
{
    int i;
    for(i=0;i<FileToConformNumber;i++)
    {
     FileToConform[i].Pos = 0;
     FileToConform[i].ReplayTimes=REPLAYTIMES;
     FileToConform[i].Handle = fopen(FileToConform[i].FileName,"rb");
     if(FileToConform[i].Handle == 0)
     {
      printf("Failed to open file %s.\n",FileToConform[i].FileName);
      exit(0);
     }
     FileToConform[i].FileSize = ftell(FileToConform[i].Handle);
     fseek(FileToConform[i].Handle,0,SEEK_SET);
     FileToConform[i].SectionNumber = (FileToConform[i].FileSize+SECTIONSIZE-1)/
                                                          SECTIONSIZE;
     FileToConform[i].TsSectionNumber = FileToConform[i].SectionNumber*
                                                          FileToConform[i].ReplayTimes;
     FileToConform[i].TsSectionCounter = 0;
     AllTsSectionNumber += FileToConform[i].TsSectionNumber;
     fseek(FileToConform[i].Handle,0,SEEK_SET);
```

```
    }
  FileToConform[i].TempFreq = AllTsSectionNumber/
                                        (FileToConform[i].TsSectionNumber*1.0);
    return 0;
}
```

在该模块被调用的时候，会将所有的文件 Pos 属性清零，这个属性用于计算需要被处理的文件。并将 ReplayTimes 赋给外部输入的一个值，默认值为 3。并使用 fopen 函数打开文件，获取返回的句柄存入结构体，并对其数值加以判断，如果为空，说明文件打开失败，此时会在标准输出窗口中打印一行提示信息，提示文件打开失败。使用 fseek 函数将文件指针指向文件末尾，获取当前的指针相对文件头的偏移量得到文件的长度信息，并存入结构体。信息获取结束。

接着要计算每个文件的 SectionNumber 和 TsSectionNumber 属性，根据这两个属性计算全局变量 AllTsSectionNumber 的数据，并根据上面三个变量计算每个文件的 TempFreq 属性。SectionNumber 的计算公式如下所示：

$$SectionNumber = \frac{FileSize + SECTIONSIZE - 1}{SECTIONSIZE} \tag{1}$$

由于都是整形数据，计算出的结果也是整形，并会去除余数，所以这里给 FileSize 做了修正，保证 section 的数目足够装下文件，并且不多出 section。TsSectionNumber 表示了这个文件重复处理之后所有的 section 个数，计算公式如下所示：

$$TsSectionNumber = SectionNumber \times Re\ playTimes \tag{2}$$

AllTsSectionNumber 由所有的 TsSectionNumber 相加得到。

在针对每个文件的信息获取完之后，AllTsSectionNumber 也计算结束了，这时需要根据每个文件的 TsSectionNumber 占 AllTsSectionNumber 的比例计算出处理这个文件的频率，第二次 for 循环，主要就是计算频率表并写入结构体。计算的公式如下：

$$TempFreq = \frac{AllTsSectionNumber}{TsSectionNumber \times 1.0} \tag{3}$$

程序里面 AllTsSectionNumber 和 TsSectionNumber 的变量都是整型数据，做除法运算之后依然是整型数据，然后对于要求较高的均匀度，需要比较精确的计算，这里给分母乘以了 1.0，看似没有实际意义，不过将整型数据的计算转换成了浮点数据的计算，除法计算的结果准确性大大的提高了。

4.3.3.2 下一个文件的计算

在程序启动的开始，每个文件的 pos 属性均为 0，此时没有最小值，需要被处理的程度相同，此时处理哪一个文件都是一样的，本设计根据文件的先后，优先处理文件列表靠前的文件。在两个文件大小相仿，或者刚好两个文件的 pos 的属性相同，均优先处理在文件列表位置靠前的文件。实现的部分代码如下：

```
int Min(void)
{
    int i,m;
    for(i=0;i<FileToConformNumber;i++)
    {
      if(FileToConform[i].TsSectionCounter!=FileToConform[i].TsSectionNumber)
      {
        m=i;
        break;
      }
    }
    for(i=m;i<FileToConformNumber;i++)
    {
      if(FileToConform[m].Pos>FileToConform[i].Pos)
if(FileToConform[i].TsSectionCounter!=FileToConform[i].TsSectionNumber)
          m=i;
    }
    return m;
}
```

使用两个 for 循环语句来实现找到需要被处理的文件序号，第一个 for 循环为找到一个需要被处理的文件，不分是否紧急，第二个 for 循环，从找到的需要被处理的文件的序号开始，找到比第一个更需要被处理的文件，直到最后一个文件。最后返回最需要被处理的文件的文件序号。其中，如果某个文件的 TsSectionCounter 属性和 TsSectionNumber 属性数值相等的话，说明这个文件已经处理结束，所以对此文件不再做任何处理，直接跳过。可以看到在最后一个 if 语句内部使用了!=判断。

4.3.3.3 读数据写数据模块

上一个模块的返回值表示了急需要处理的文件的文件序号，这个模块获取到上一

个模块的返回值之后，获取相应的文件结构体中的相关属性填入 section 结构体，并从相应的文件内部读取数据，填入 section 结构体中的数据域，在属性和数据全部读取结束之后，需要将 section 结构体中的属性包括数据全部写入缓冲区，计算缓冲区里面的数据的 CRC 值，并附加在缓冲区结尾。最后将缓冲区里面的数据分成一个或多个数据包，写入文件。本设计默认将缓冲区里面的数据分成 6 个数据包。总体模块流程图如下图所示：

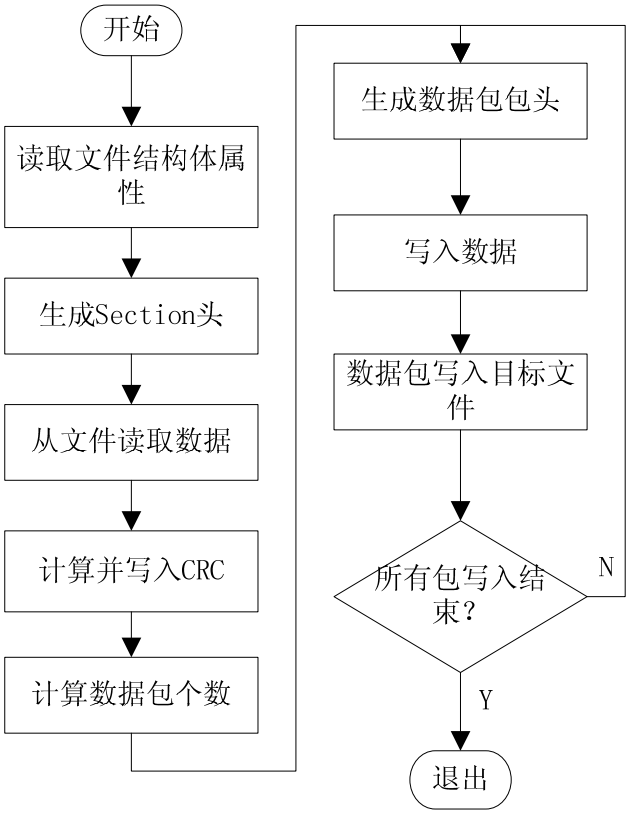

图 4-7 读数据写数据流程图

由于本设计是基于数字电视的，所以生成的目标文件需要满足 MPEG-2 标准，生成的数据包带有 4 个字节的包头信息。根据 MPEG-2 标准，第一个字节必须为 0x47，为同步字节，传输误码指示符，有效负载单元起始指示符，传输优先标志位分别为 1 个比特，存放于第二个字节的高位，PID 为 13 个比特，存放与第二个字节的地位和第三个字节内，传输加扰控制位和自适应控制位分别为 2 个字节存放于第四个字节高四位，另外连续计数器为 4 位，存放在第四个字节低四位。具体的存储结构如下图所示：

1 B	1 bit	1 bit	1 bit	13 bit	2 bit	2 bit	4 bit
同步字节	传输误码指示符	有效荷载单元起始指示符	传输优先	PID	传输加扰控制	自适应控制	连续计数器

图 4-8 传输流的包头结构

读取文件属性的代码如下所示：

```
Section.TableID = No;
Section.VersionNumber = 0x0B;
Section.SectionSize = SECTIONSIZE;
Section.AllSectionNumber = AllTsSectionNumber;
Section.AllSectionCounter = AllTsSectionCounter;
Section.SectionNumber = FileToConform[No].SectionNumber;
Section.SectionCounter = FileToConform[No].SectionCounter;
Section.FileLength = FileToConform[No].FileSize;
```

其中 No 为上一个模块返回的整形数据，表示最需要被处理的文件的序号。读取文件数据的部分代码如下：

```
ReadCount = fread(Section.SectionData,1,SECTIONDATASIZE,
                  FileToConform[No].Handle);
if(ReadCount!=SECTIONDATASIZE)
  memset(Section.SectionData+ReadCount,0x88,SECTIONDATASIZE-ReadCount);
memcpy(&SECTIONBUF[24],Section.SectionData,SECTIONDATASIZE);
```

使用 fread 函数从文件里面读取 SECTIONSIZE 个字节的数据，并存入 section 的数据域，该函数的返回值为实际读取的数据的长度，如果读到了文件末尾，或者并没有读取到有效的数据，那么函数返回值应该比 SECTIONSIZE 小，这样 section 的数据域后面的没有被有效数据填充的部分的数据就变得没有意义，本设计使用 0x88 填充。最后使用 memcpy 函数将 section 数据域里面的数据填充至缓冲区相应地址。由于 section 带有 section 索引信息，所以数据存放在缓冲区起始地址之后的一个地址。本设计中，section 索引信息包含 TableID，版本号，section 长度，这些 MPEG-2 标准规定的信息，另外也包含有设计自定义的 AllSectionNumber，AllSectionCounter，SectionNumber，SectionCounter 以及 FileLength 等五个 section 的相关信息。其中前三个信息一共占了 3 个字节的地址空间，自定义的每个数据为 4 个字节，加上第一个表示偏移量的字节，一共是 24 个字节，存放数据的位置就是起始位置向后偏移 24 个字节，所以这里 memcpy 函数里面的参数选择&SECTIONBUF[24]。自定义的五个数据

按顺序以大端数的格式存放在缓冲区偏移地址为 4 到 23 的空间里面。

在写入 section 头并装填完数据之后，需要对 section 的数据进行 CRC 计算并填入缓冲区结尾。MPEG-2 标准里面使用的 CRC32 的表达式为 0x04c11db7。CRC 计算的部分代码如下所示：

```
u32 Byte_CRC(u8 IByte, u32 ICrc)
{
    int     i;
    unsigned int Crc=ICrc;
    Crc=Crc^((unsigned int)IByte<<24);
    for(i=0;i<8;i++)
     if((Crc&0x80000000)!=0)
       Crc=(Crc<<1)^0x04c11db7;
     else
       Crc=Crc<<1;
    return Crc;
}
u32 GetCRC(u8* buf, u32 len)
{
    unsigned int i;
    unsigned int Crc=0xffffffff;
    for(i=0;i<len;i++)
         Crc=Byte_CRC(buf[i],Crc);
    return Crc;
}
```

如上面所示，这部分代码包含有两个函数，单个字节的 CRC 计算和整个块的 CRC 计算，其中第一个函数主要提供给第二个函数调用，第二个函数可以在任何程序，任何平台下面运行。GetCRC 函数，根据输入的参数控制循环的次数，每次计算一个字节的 CRC 数据，并将结果保存，计算下一个字节的时候作为输入。等待循环执行完毕，CRC 计算结束，函数返回 CRC 的计算结果。最后使用下面的语句将 CRC 的计算结果填入 section 缓冲区的最后：

```
SECTIONBUF[SECTIONSIZE]   = (Section.CRC>>24)&0xFF;
SECTIONBUF[SECTIONSIZE+1] = (Section.CRC>>16)&0xFF;
SECTIONBUF[SECTIONSIZE+2] = (Section.CRC>>8)&0xFF;
SECTIONBUF[SECTIONSIZE+3] = (Section.CRC)&0xFF;
```

由于 MPEG-2 标准规定 CRC 数据以小端数存储，所以这里也为了满足标准，采用小端数存储 CRC 的值。

数据全部填入缓冲区之后，需要将缓冲去里面 section 的内容装入 TS 包，本设计默认每个 section 由 6 个 TS 包组成，也支持其他数目的包组成，这里装包之前对包数目重新进行了计算，计算公式如下：

$$PacketsNumber = \frac{SectionSize + SECTIONHEADSIZE + Offset + PacketSize - 1}{PacketSize} \quad (4)$$

其中 SectionSize 默认为 1100，根据段长参数的变化重新计算。SECTIONHEADSIZE 根据 MPEG-2 标准，由 TableID、版本号以及段长 3 个部分组成，共 3 个字节。offset 为 1 个字节的偏移量，加上偏移量的数值，一般偏移量的数值为 0，所以这里的 offset 取值为 1。后面的 PacketSize-1 为修正项，由于这里是整型数据的计算，会去除余数，加上这个修正项，达到进一算法，保证有足够的空间存放所有 section 的数据。装包的部分代码如下所示：

```
packets = (SECTIONSIZE+4+183)/184;
for(packetscnt=0;packetscnt<packets;packetscnt++)
{
  memset(PacketBuf,0xFF,188);
  PacketBuf[0]= 0x47;
  if(packetscnt ==0) PacketBuf[1] = 0x40|((ADSPID>>8)&0x1f);
  else PacketBuf[1] = 0x00|((ADSPID>>8)&0x1f);
  PacketBuf[2] = ADSPID & 0xFF;
  PacketBuf[3] = 0x10|(CC&0x0F);
  CC++;if(CC>15)CC = 0;
  fwrite(PacketBuf,1,188,AdsTempHandle);
}
FileToConform[No].Pos += FileToConform[No].TempFreq;
```

可以看到，第一行对 packet 的数目使用上面提到的计算公式进行了重新计算，作为循环的控制条件起着很大的作用。循环内部，先将包缓冲区数据清楚，默认全部设置为 0xFF。

每个 section 的第一数据包需要将有效负载起始标志位置高，这个在 MPEG-2 标准中有明确说明。这里在 if 条件语句中实验这个要求。并将广告的 PID 存放在第二个字节低位和第三个字节内部。

连续计数器存放一个计数值，4 个比特，可以从 0 记到 F，然后再回到 0。这个计数器保证了数据包的连续性。如果连续计数错误，表示发生了丢包现象，那么相对应的 PID 的 section 的数据需要被丢弃。这个在 MPEG-2 标准里面有明确规定，本设计

也满足这个规定。

包头 4 个字节处理结束后，从 section 缓冲区读取 184 个字节的数据存入包缓冲区。包数据组织结束之后，再将包写入文件。

循环结束后，一个 section 的数据被处理结束，数据全部被写入文件了。这个时候要对文件的重要属性进行更新，每个文件的 pos 属性需要增加 TempFreq，另外各个相关的计数器也要同时加上 1。每个广告文件被写入目标文件的次数为 ReplayTimes，如果 ReplayTimes 不为 1，比如本设计的默认设置为 3，这种情况下，文件指针指向了文件末尾，需要将文件指针移至文件头，进行第二遍处理，并将相关计数器复位。

4.3.3.4 函数的调用以及文件末端的处理

上面的三个函数分别实现了相应的功能，在需要将文件打包的时候调用这三个函数即可。本设计编写了一个打包函数调用这三个函数。并对文件的末尾加以处理，最后一个数据包的连续计数器不为 0xF 时，需要给文件末尾加上没有实际数据的数据包。实现的代码如下：

```
GetFileInfo();
while(1)
{
 Conform(Min());
 if(AllTsSectionCounter==AllTsSectionNumber)
 {
  if(CC!=0)
   for(i=CC;i<=15;i++)
   {
    PacketBuf[0] = 0x47;
    PacketBuf[1]  = 0x00|((ADSPID>>8)&0x1f);
    PacketBuf[2]  = ADSPID&0xff;
    PacketBuf[3]  = 0x10|(i&0xf);
    memset(&PacketBuf[4],0x88,184);
    fwrite(PacketBuf,1,188,AdsTempHandle);
   }
  fclose(AdsTempHandle);
  break;
 }
}
```

其中 GetFileInfo 为获取文件信息函数，对于每次打包只调用一次该函数，获取文件信息之后，文件信息不会再做变动。Min 函数为计算当前最需要被处理的文件，返回值为这个文件的文件序号。Conform 为读数据写数据函数，接收 Min 函数的返回值之后做一系列操作将文件的数据以 TS 包的格式写入目标文件。

这个函数的退出条件是，所有文件的 Section 数目与所有文件的 Section 计数器数值相等，其中所有文件的 Section 数目由文件长度，重复次数，段长度等相关信息计算而来。在满足这两个变量相等之后，对连续计数器的数值进行了判断，如果连续计数器不为 0，说明文件末尾的数据包的计数器不为 0xF，这样就要对文件末尾填充数据包，直到连续计数器的数值符合 MPEG-2 标准。

最后释放文件句柄，退出循环，到这里打包文件执行结束，所有的广告文件以及广告信息表文件都已经被写入目标文件内部。

4.4 混流程序

由于本设计基于数字电视，本设计的广告流文件需要混入原始节目流里面，这样就可以在电视节目播出的时候，可以接收到发送端发出的广告信息。为了防止接收端长时间接收不到广告数据包误以为信号丢失的情况，混流的时候根据混流码率的设定，保证均匀的混入原始节目流。

混流程序不改变原始视频流码率，原始流里面的各种表，也不改变原始节目流里面相关的音视频数据，仅使用广告数据替换原始视频流内部的空包。广告数据混入原始视频流的码率为 2 兆。混流程序支持将多个文件流同时均匀的混入原始节目流，对于本设计，只需将一个文件混入原始节目流。

混流程序由信息获取程序，寻找同步字节程序，定位空包程序，获取广告数据包程序等四个子程序组成。其中信息获取程序主要获取广告流文件相关信息，打开广告流文件，将返回的句柄存入结构体。寻找同步字节程序的主要功能是找到原始节目流的同步字节，本设计使用替换空包的方式混入广告流，所以需要寻找到同步字节 0x47。空包定位程序的主要功能是将文件指针定位到空包的位置，便于后面插入数据包。获取广告数据包程序，主要用于获取一个广告数据包，对于广告流结束，原始节目流却没有结束的情况，该程序也将文件指针定位到文件头，使得再次获取广告数据

包时从广告流第一个数据包开始获取。

4.4.1 广告流信息获取程序

这里的广告流为上面打包文件生成的目标文件，混流程序需要将这个广告流混入到原始节目流，需要获取广告流的相关信息及数据。广告流的信息从外部文件获取，实现的部分代码如下：

```
InfoHandle = fopen(InfoName,"rb");
if(InfoHandle == NULL)
{
  printf("%s open failed!\n",InfoName);
  return -1;
}
printf("%s open sucess!\n",InfoName);
int i,j;
for(i=0;i<100;i++)
{
  if(fscanf(InfoHandle,"%s",Temp)==EOF)
    break;
  sprintf((char*)MergeFile[i].FileName,"%s",Temp);
  MergeHandle[i] = fopen(MergeFile[i].FileName,"rb");
  if(MergeHandle[i]==NULL)
    exit(0);
  fscanf(InfoHandle,"%x",&MergeFile[i].PID);
  fscanf(InfoHandle,"%d",&MergeFile[i].Loop);
}
return i;
```

先打开存有广告流文件名和重复次数的外部文件，如果文件不存在会提示错误。接着从文件读取数据，按顺序一个一个存入结构体，本设计支持最大混入文件的数目为 100 个。100 个以内软件自动识别个数，超过 100 个，后面的文件将会被忽略。

4.4.2 寻找同步字节程序

由于混流程序最小以一个数据包为单位，原始节目流可以由于抓取工具，软件平台的不同，流文件起始未必就是一个完整的数据包，另外对于抓包过程中的干扰也会造成数据包的不完整，考虑到这两种情况以及其他未知的情况导致流文件数据包不完

整的现象，本设计提供了寻找同步字节程序。

在读取文件的第一个数据包，或者文件处理过程中出现同步字节丢失的情况，需要调用此子程序。具体实现的部分代码如下：

```
int FindTSHead(FILE* TSHandle)
{
 u8 BUF[900];
 int i=0;
 if(fread(BUF,1,900,TSHandle)!=900)
  return 0;
 for(;i<190;i++)
  if(BUF[i]==0x47&&BUF[i+188]==0x47&&BUF[i+188*2]==0x47)
   break;
 if(i<188)
 {
  fseek(TSHandle,-900+i,SEEK_CUR);
  return 1;
 }
 else
  return 0;
}
```

实现此功能的主要方法是，先从文件里面读取 900 个字节的数据，如果从文件指针的位置到文件结束，所有的数据不满 900 个，会直接返回，这种情况下找到的文件头没有任何意义，对于这种情况，本设计返回值为 0，表示寻找数据头失败。检测到同步字节之后，依次判断偏移地址为 188 和 376 的两个字节是否为同步字节，如果满足，那么表示找寻到同步字节，否则寻找下个同步字节。另外需要保证的是必须在前 188 个字节内找到同步字节，本设计默认的包长为 188 字节，对于丢失部分数据的情况，188 个字节内一定有同步字节，如果满足，就重新定位文件指针，返回成功，否则返回失败。

4.4.3 空包定位程序

本设计的混流方式是将广告流数据包替换节目流中的空包，这样就需要将节目流的数据包读出来，并加以解析，解析的结果存在结构体中，然后读取结构体的参数，再根据是否为空包进行替换。MPEG-2 标准规定空包的 PID 为 0x1FFF，所以只需要解析出数据包的 PID，判断是不是 0x1FFF 即可。

4.4.3.1 包头结构体的定义

根据 MPEG-2 标准，每个数据包的包头 4 个字节存放了包括传输错误标志位、有效负载起始标志位、传输优先标志位、包识别号 PID、传输加扰控制、自适应控制以及连续计数器等 7 个标志位信息。其中最重要的就是 PID，通过 PID 的值，我们可以确定这个数据包里面的数据归属。在解析包头的 4 个字节数据之后，将信息存入结构体，供后面使用。包头结构体的定义如下：

```
typedef struct
{
    u8 TSPacketValid;
    u8 TransportErrorIndicator;
    u8 PayloadUnitStartIndicator;
    u8 TransportPriority;
    u16 PID;
    u8 TransportScramblingControl;
    u8 AdaptionFieldControl;
    u8 ContinuityCounter;
    u8 Offset;
    u8 PacketData[184];
}TSPacketHeader;
```

结构体的第一个参数声明数据包是否有效，如果数据包第一个字节并不是同步字节，那么表示同步字丢失，这一包的数据也没有意义，每次读取数据包属性的时候，都要判断一下这个参数是否为 1，如果为 1 表示包数据有效，否则包数据无效。在同步字丢失之后必须调用寻找同步字字节程序重新寻找同步字。其他属性为参考 MPEG-2 标准，将包头带有的信息依次读取并存入结构体的对应属性里面。结构体的另一个属性是 PacketData，这里存放了所有除了包头之外的其他 184 个字节数据。这个结构体主要组成部分为包头的信息，也包含包的数据，在解析包头结束时，直接将数据包的数据存入 PacketData 属性内部，这样，需要获取包的数据时无需再对数据包或者文件进行读取，直接从 PacketData 中获取即可。

4.4.3.2 包头信息的获取与空包的定位

每读取到一个数据包，就使用包头信息获取程序，对包头的四个字节数据进行读

取，将信息全部存入结构体。判断结构体中的 PID 属性，如果该属性值为 0x1FFF，表示该包为空包，这样就定位到了空包。包头信息的获取实现代码如下：

```
TSPacketHeader GetPacketProperty(TSPacket Packet)
{
 if(Packet.data[0] == 0x47)
  PacketTemp.TSPacketValid=1;
 else
 {
  PacketTemp.TSPacketValid=0;
  return PacketTemp;
 }
 PacketTemp.PayloadUnitStartIndicator = (Packet.data[1]>>6)&0x01;
 PacketTemp.PID = ((((u16)(Packet.data[1]&0x1F))<<8)+
                          (u16)Packet.data[2])&0x1FFF;
 PacketTemp.ContinuityCounter = Packet.data[3]&0x0F;
 if(PacketTemp.PayloadUnitStartIndicator == 1)
 {
  PacketTemp.Offset = Packet.data[4];
  for(i=0;i<184-PacketTemp.Offset;i++)
  PacketTemp.PacketData[i] = Packet.data[5+i+PacketTemp.Offset];
 }
 else
 for(i=0;i<184;i++)
 PacketTemp.PacketData[i] = Packet.data[4+i];
 return PacketTemp;
}
```

函数的参数为一个 TS 数据包，函数内部定义了一个包头信息的结构体，用于存放分析的结果。if 条件语句用来判断数据包的第一个字节是否为同步字节，如果是同步字节，那么设置 TSPacketValid 属性为 1，否则设置为 0，并且直接返回。下面根据该属性值判断同步字节是否丢失。接下来根据 MEPG-2 标准，读取其他相关的信息并存入结构体，其中 PayloadUnitStartIndicator 为 1 时，说明这个数据包后面的负载数据是一个 Section 的起始数据包，包含有 Section 头，并且这个包的包头 4 个字节之后的第一个字节并非有效数据，为有效负载的偏移量，取出该属性之后，再将剩下的数据存入结构体的数据区域。其中 Offset 属性依赖于 PayloadUnitStartIndicator 属性，PayloadUnitStartIndicator 为 1 时 offset 属性值有意义，否则 offset 的属性值没有意义。最后返回此结构体。

数据包的属性获得以后，可以根据数据包的 PID 属性判断该数据包是否为空包。

4.4.4 获取广告数据包程序

混流程序为一个通用的混流程序，可以将多个不同类型的流混入原始节目流，本设计混入广告流的只用了混流程序的一个通道。获取广告数据包的同时，也会对广告数据包的有效性进行判断，如果无效会重新寻找同步字节。实现的部分代码如下：

```
if(fread(MergePkt.data,1,188,MergeHandle[i]) != 188)
      MergeFile[i].LoopCnt++;
else
{
  MergePktProperty = GetPacketProperty(MergePkt);
  if(MergePktProperty.TSPacketValid==0)
         while(FindTSHead(MergeHandle[i])==0);
  if(MergePktProperty.PID==0x1CC0)
    return;
  else continue;
}
```

先从广告文件读取 188 个字节的数据，如果实际读取到的数据不足 188 个字节，说明文件达到末尾，此时将文件指针重新定位到文件头。读取了数据包之后，使用包头信息分析程序 GetPacketProperty 分析该数据包，如果包无效，使用 FindTSHead 重新寻找同步字节，知道寻找到同步字节。之后对包的 PID 进行判断，如果为广告流文件的 PID，表示广告数据包获得成功，否则继续寻找，直到找到正确的广告数据包。

4.4.5 子程序的调用

编写顶层模块调用这几个子程序，首先需要使用 FindTSHead 程序找到节目流文件的包头同步字节，读取一个节目流数据包，分析数据包包头，对比 PID，如果为空包，那么执行替换操作，如果不是空包，就继续寻找空包。替换操作需要从广告流文件中读取一个完整的数据包，如果出现数据包无效的情况，也需要使用 FindTSHead 程序重新寻找同步字节，并读取一个完整的数据包。具体流程图如下图所示：

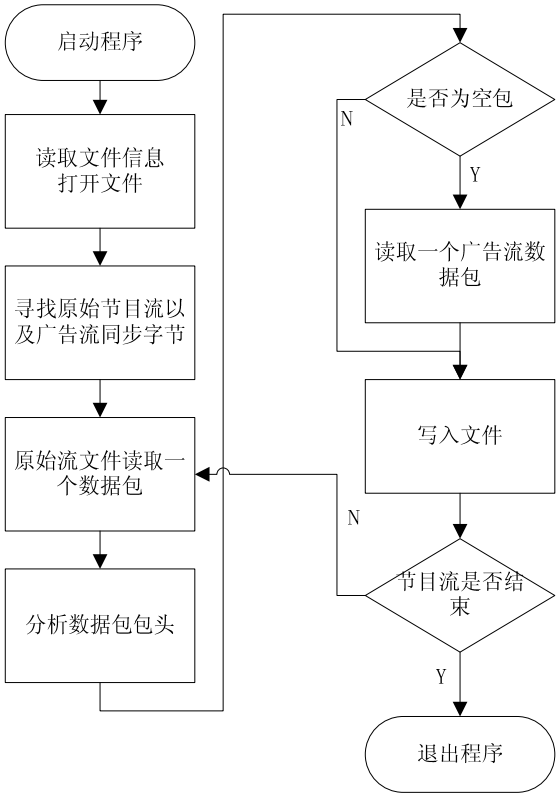

图 4-9 混流程序流程图

程序执行结束之后，会生成一个目标文件，该流文件包含有原始节目流的所有信息，同时也包含有广告流的所有数据。将此文件在相关设备中播出即可。

4.5 IPQAM 发包程序

本设计使用 IPQAM 做为调制设备，将生成的流文件数据发送到 IPQAM 的数据端口，通过 IPQAM 调制到一个高频信号上面，通过有线电视 Cable 线发送至接收端。本设计使用的 IPQAM 具体型号为 IPQ6800-A，支持最多 8 路信号传输。在 IPQAM 的配置界面配置完 IPQAM，需要使用此程序向 IPQAM 发送数据包。本设计的发包程序支持最多 8 路数据的传输，与 IPQAM 一致。

发包程序由获取通道信息、获取单个文件信息、唤醒时间计算、时间差值计算、发送数据、计算下一个通道等六个子程序组成。其中获取流文件信息从一个外部文件获取文件路径文件名、IP 地址、IP 端口号以及码率等四个信息，并且返回需要使用几个通道，使用几个通道为软件自动识别，如果需要改变这些信息，修改配置文件即可，无需对软件进行修改。获取单个文件信息，主要是计算当前文件的数据包包长，

以及找到同步字节，便于数据包的读取操作。

由于每个 UDP 包需要均匀的传输，本设计使用时间控制发包的速度，根据从配置文件内部获取的码率信息，计算发包的时间间隔。在发送完一个 UDP 包之后，会调用计算下一个通道子程序，找到下一个需要发送数据的通道，根据系统当前时间和下一个通道发送数据的时间的时间差，将程序休眠，一定时间后唤醒程序，并发送 UDP 包。主要实现的流程图如下：

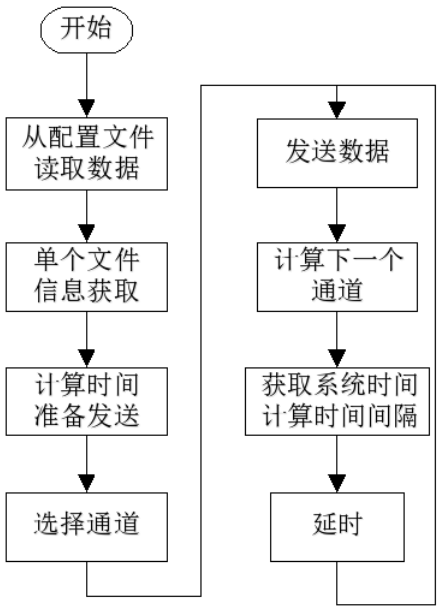

图 4-10 IPQAM 发包程序流程图

这个程序没有退出，除非软件错误或者用户手动的结束该程序，否则程序一直循环。由于抓取的码流不会很长，而 IPQAM 又要长时间工作，所以本设计在文件发送结束时重新将文件指针定位到文件头，在网络断开或者用户手动停止软件运行的时候，程序终止。

4.5.1 获取通道信息子程序

此程序功能为从外部配置文件获取通道的相关信息，主要包括文件路径文件名、IP 地址、IP 端口号、以及码率四个信息。配置文件的需要按照上面四个参数的顺序依次排列，使用空格或者回车符分开，配置文件内部可以写入一个或者多个通道的相关信息，不过只有前八个通道的相关信息有效。

```
for(i=0;i<8;i++)
{
```

```
  if(fscanf(info,"%s",Temp)==EOF)
    break;
  sprintf(FileName[i],"%s",Temp);
  fscanf(info,"%s",Temp);
  sprintf(IP[i],"%s",Temp);
  fscanf(info,"%d",&Port[i]);
  fscanf(info,"%d",&BaudRate[i]);
  TimeCell[i] = PACKETLENGTH*1.*7*8*1000000000/BaudRate[i];
}
```

由于本设计使用的 IPQAM 最多支持 8 个通道的数据传输，所以通道信息获取程序最多支持 8 个通道，以前八个通道为主。使用 fscanf 函数从文件中读取通道信息，可以存储为字符串格式和整形数据格式。最后根据码率计算最小时间单位。最小的时间单位用于计算下一次处理该通道的时间。

4.5.2 获取单个文件信息子程序

此 IPQAM 发包程序支持 188 和 204 的两种包长格式的流文件，并支持自动识别，在这里定义了一个结构体，结构体有包长和同步字节位置两个属性，在寻找同步字节的过程中获取包长信息。结构体的定义如下：

```
struct TsFilePro
{
    int HeadPosition;
    int PacketLength;
};
```

每个通道的流文件都会对应这样的一个结构体，确定了包长信息和同步字节位置之后，使用这两个数值确定每次操作读取多少字节的文件。由于支持两种格式的数据包，所以在寻找同步字节的时候就需要使用两种算法同时查找，并且这两种算法一定不会同时满足的，找到同步字节的同时包长信息也会确定下来。将这两个信息存入结构体，返回结构体。

4.5.3 唤醒时间计算子程序和时间差值计算子程序

这两个子程序需要用到 time.h 中的 timespec 结构体。结构体的包含有 tv_sec 和 tv_nsec 两个参数。

唤醒时间计算子程序的参数为一个时间和一个延迟时间，因为时间结构体包含有两个属性，所以不是简单的加法，需要对 tv_nsec 超过十亿的情况进行判断，决定是否要将 tv_sec 加一。此函数的延迟时间单位为纳秒，并且这个参数的值不可以大于一秒。计算结束之后将时间结构体返回。实现此功能的部分代码如下所示：

```
Temp.tv_nsec = tim.tv_nsec + nanodelay;
Temp.tv_sec = tim.tv_sec;
if(Temp.tv_nsec>=1000000000)
{
 Temp.tv_nsec -= 1000000000;
 Temp.tv_sec += 1;
}
return Temp;
```

其中 Temp 为定义的一个局部时间结构体变量，用于存放计算的结果，在计算结束之后返回。tim 和 nanodelay 分别为这个函数的两个参数。

时间差值计算子程序返回的时间差值以纳秒为单位，计算第二个时间参数的相对与第一个时间参数的时间差值。这个差值的计算主要是用于计算唤醒时间。实现的方式就是计算 tv_sec 的差值转换成纳秒再加上 tv_nsec 的差值。由于以第一个时间参数为基准，一般来说第二个时间参数比第一个时间靠后，返回的是一个正数。对于特殊情况，也可以返回负数或者 0。

4.5.4 发送 UDP 包子程序

使用发送 UDP 包子程序需要先配置通道相关参数，并测试网络是否可以使用。从结构体中取出 IP 地址和 IP 端口之后存入 si_other 结构体，并使用 socket 和 inet_aton 函数测试网络数据端口是否可以使用。

配置完每个通道的参数之后，直接使用 sendto 函数就可以将 UDP 包发送到 IPQAM。此发包程序设定每次从文件内部读取 NPACKET 个数据包，然后一次性将所有的数据全部发送至 IPQAM。

4.5.5 计算下一个通道子程序

此程序支持 8 个通道同时工作，并且要求每个通道的数据发送都满足码率的设置

均匀发送数据包。所以没发完一次数据包之后都要判断下一个发送数据时间最靠近当前系统时间的通道。实现此功能的部分代码如下所示：

```
for(i=1;i<TsNumber;i++)
{
 if(sendtime[Temp].tv_sec>sendtime[i].tv_sec)
   Temp = i;
 else if(sendtime[Temp].tv_sec == sendtime[i].tv_sec)
   if(sendtime[Temp].tv_nsec>sendtime[i].tv_nsec)
    Temp = i;
}
return Temp;
```

从第一个通道开始判断，依次根据秒属性值和纳秒属性值找到所有通道中最小的时间的通道。其中每次传输完数据之后都会将通的 sendtime 更新，具体的操作是将 sendtime 加上 timecell 数值，timecell 就是上面计算出的最小的时间单位。

4.5.6 发送 UDP 包主函数以及子程序的调用

在读取了相关的信息之后，进入函数的主题部分，就是不停的重复读取数据包、更新发送时间、确定下一个发送数据的通道和延迟这四个操作。实现此功能的主要部分代码如下所示：

```
FrameSend(channel);
sendtime[channel] = GetWakeTime(sendtime[channel],TimeCell[channel]);
channel = NextChannel();
clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME,&sendtime[channel],NULL);
```

其中 FrameSend 函数就是发送数据包函数，此函数包括了读取数据和发送数据两部分。第二个语句为获取通道发送时间，并存入数组内部。NextChannel 函数确定了下一个发送数据的通道，并将返回值存入变量 channel 内部，根据 channel 的数值，每次对不同的通道操作。

clock_nanosleep 函数为 Linux 系统自带的延迟函数，精度为纳秒级别，这里使用系统时间作为基准，sleep 的数值为绝对时间。考虑到可能由于网络或者其他干扰会对数据的传输带来一定的延迟，使用绝对时间可以大大的降低这种延迟，并且不会带来积累误差。

第五章 测试结果

5.1 上位机软件的测试

5.1.1 使用规则

上位机界面里面的参数全部可以设置，不过有一定限制。对于不满足软件自身限制的参数设置都会提示错误信息，并复位设置。具体的限制为：

1.广告 PID 必须大于或等于 0x1000，由于 MPEG-2 标准规定了某些业务表的 PID，为了尽可能的避免与业务表的 PID 冲突，本设计规定了 PID 必须满足上面的要求。默认值为 0x1CC0。

2.XML 文件 TID 必须小于文件起始 TID。本程序可以处理最多 100 个广告文件，广告文件的 TID 是按照起始 TID 累加的，XML 文件就一个，为了避免 XML 文件和广告文件的 TID 重复，设计规定 XML 文件 TID 必须小于文件起始 TID。XML 文件 TID 的默认值为 0，文件起始 TID 默认值为 1。

3.段长最大为 22 包。由于 MPEG-2 标准中 SectionSize 为 12 位，最大可以包含 4096 个字节的数据，每个数据包去除包头之外可以存储 184 个字节的数据，相除得到每个 Section 最多可以由 22 个包组成。所以大于 22 的设置全部会被重置。默认值为 6。

4.包长必须为 188 或者 204 字节，之外的设置全部被重置。根据 MPEG-2 标准，包长必须为这两个数值中的一个。默认值为 188。

5.1.2 测试结果

将广告 PID 设置为 0x345，不满足使用规则 1，弹出如下错误提示框：

图 5-1 PID 设置错误提示

将 XML 文件 TID 设置为 20，文件起始 TID 设置为 10，不满足使用规则 2，弹出如下错误提示框：

图 5-2 TID 设置错误提示

将段长设置为 23，不满足设计规则 3，弹出如下错误提示框：

图 5-3 段长设置错误提示

将包长设置为 200，不满足设计规则 4，弹出如下错误提示框：

图 5-4 包长食指错误提示

将上位机文件参数设置为下图所示设置：

图 5-5 正确的参数设置

点击生成文件信息表按钮，使用 XML NotePad 2007 打开，可以看到该软件生成的 XML 文件是可以被解析的，软件截图如下图所示：

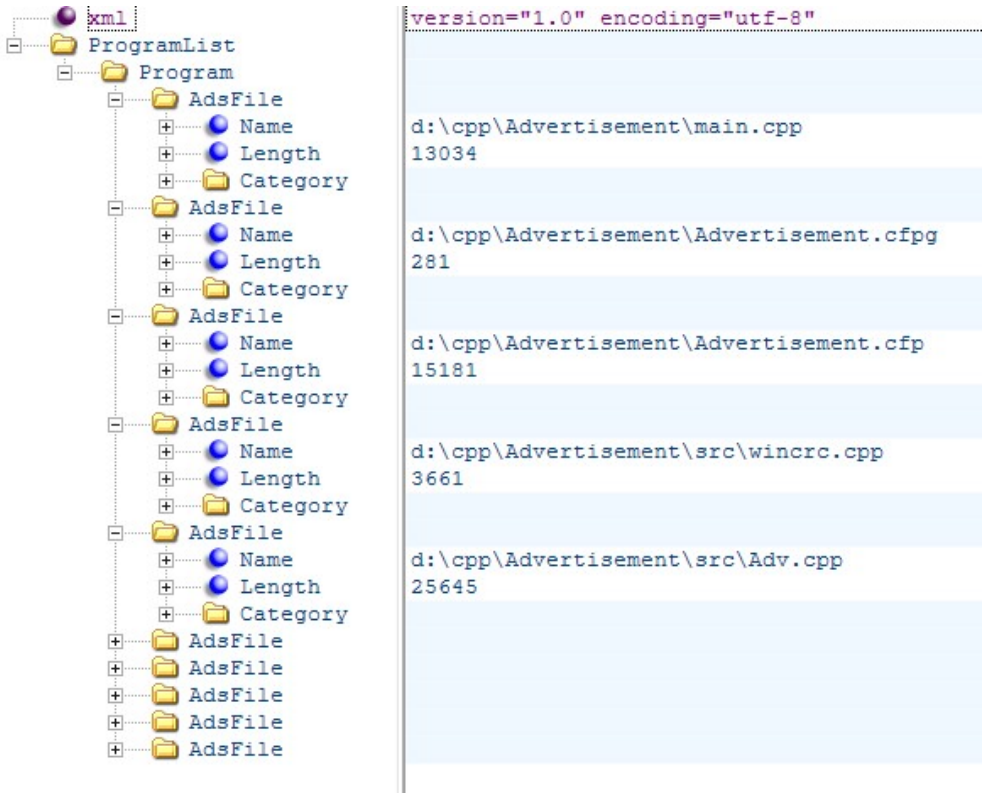

图 5-6 生成的 xml 文件

点击启动按钮，短暂的等待之后可以看到成功的提示框，如下图所示：

图 5-7 配置文件和广告流文件安生成成功提示

使用二进制查看软件打开生成的文件，可以看到数据被正确的装入了文件，如下图所示：

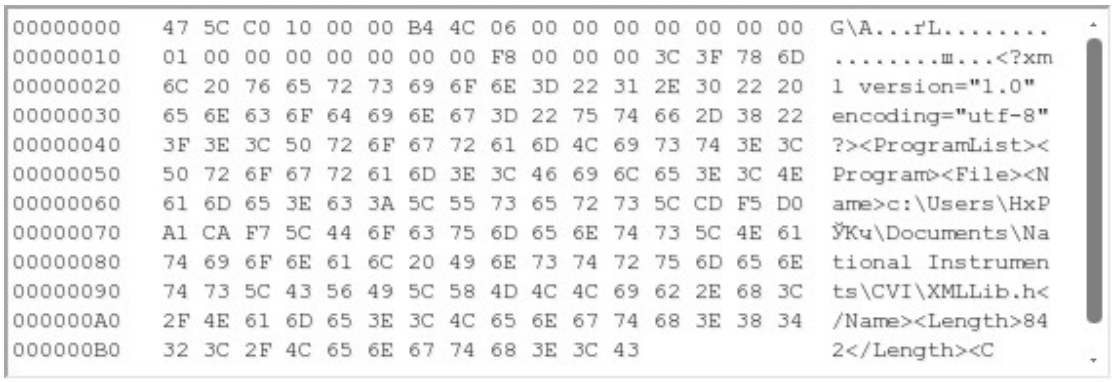

图 5-8 目标文件的二进制数据

第六章 总结

持续了四个多月的毕业设计终于结束了，从一开始拟定题目，到最后功能的实现，还是有很大差距的，现象中问题都是那么简单，实现起来却是相当复杂。调试的过程中也出现了难以发现的错误，多半是因为经验不足导致的，原本简单的逻辑，只是因为少了一个类型转换，少了一个变量没有初始化，使得实验结果与预想中的差别很大。

回顾我的毕设，总体上来说，在导师的指引下我独立的建立了一个数字电视多媒体系统，主要实现了将用户需要发送的多媒体广告发送到数字电视终端，其中包含很多个子程序。另外也涉及到 IPQAM 仪器的配置和发包程序的设计，所以设计的难度又加大了，也变的更复杂了。

我参与了承德中国有线的双向互动高清数字电视数千用户的增值业务（主要是广告业务）的现场测试及调试，遇到了很多现有技术局限造成的实际问题，例如：广告是相对技术简单的业务，但是实际运营中，简单的业务确对各个机顶盒厂商造成非常大的问题，各种 bug 长期困扰着机顶盒开发者，因而提出了解决问题的技术方法。本设计实现了一个数字电视广告系统的传输层，为客户的各种广告业务提供了一个兼容性平台，同时也可以通过此平台增加各种其他增值业务。 在论文结束的时候，我发现想对与刚入学的我，我有了很明显的进步，主要的表现为几点。首先，我不再惧怕大的项目，大的工程，再大的工程也是可以分成若干个小的工程的，认真的分析问题，将问题分解，编写代码的时候尽量模块化，在没做完一个小模块之后都做好测试，就不会出现那么多的问题。曾经我审视过自己，到底适合不适合做一个软件工程师，现在我不再怀疑自己。第二，我变的更有耐心去把一件事情做好，在项目的过程中，我的导师一再提醒我，要保证成功率，不是一次成功就代表每次都成功，必须要做上百次甚至上千次的测试，保证 99.9%以上的成功率，测试才算通过。起初我对这个想法很是质疑，没必要把事情做的那么复杂，后来在一次次的失败，我才知道每个模块都必须经过严格的测试，否则多个模块结合在一起，出了问题，根本无从下手。第三，在项目的过程中，我知道了，做好一个项目需要考虑到的问题，比如怎样可以让软件的使用着方便的使用并且尽可能的降低出错的概率，怎样提高软件程序的通用性，怎

样搭建一个局域网可以协调多个仪器设备之间高速数据和低速数据的传输。第四，在学习 MPEG-2 协议的同时，提高了自己严谨的态度缜密的逻辑和对资源充分利用的意识，协议里面提及的的每一个比特都有他的意义，里面的逻辑更是无懈可击。深刻体会到，如果也想把自己做的东西作为一个标准推广出去，必须具备这些素质。

学无止境，在今后的工作过程中，我会将学习到的领悟到的知识熟练的运用到项目中去，并在不停的项目中提升自己，只有不停的挑战自己，超越自己，才会在事业上面有所成就。

参考文献

1.ISO13818 标准

2.XML 示例程序导学(第 2 版) Benoit Marchal 清华大学出版社

3.XML 模式权威教程 Priscilla Walmsley 清华大学出版社

4.Visual C++程序设计 张岳新 苏州大学出版社

5.数字电视技术(第 2 版) 赵坚勇 西安电子科技大学出版社

6.数字电视前端系统 数字电视国家工程实验室（北京） 科学出版社

7. LabWindows/CVI 虚拟仪器测试技术及工程应用 王建新、隋美丽 化学工业出版社

8.基于 Lab Windows/CVI 的虚拟仪器设计与应用(第 2 版) 孙晓云 电子工业出版社

9.数字电视测试原理与方法 数字电视国家工程实验室（北京） 科学出版社

10.数字电视系统测量与监测 苏志武 林定祥 章文辉 电子工业出版社

11.数字电视业务及其编码 方涛 国防工业出版社