独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：张颖　　日期：2009.12.30

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。

非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：张颖　　日期：2010.3.12

导师签名：[illegible]　　日期：2010.3.12

嵌入式 SWF 文件解码器的高分辨率应用设计与实现

摘 要

近年来，Flash 技术越来越受到了人们的关注和喜爱，应用领域也在不断扩展，它超越了计算机的界限，广泛地应用在各种设备中，各种支持 Flash 文件播放的嵌入式产品也应运而生。同时，随着三网融合的进一步推进，高清数字电视和机顶盒的迅速普及，Flash 播放功能也将逐渐成为高清数字电视和机顶盒的标准配置，促进多媒体应用的多样化发展，丰富人们的文化生活。

SWF 是 Flash 矢量图形文件格式的一个推荐版本，是一种支持矢量和点阵图形的动画文件格式。其独特的二维网页多媒体技术，将矢量动画、音频压缩编码和动作脚本等多种要素结合在一起，从而创造出了一种有声有色、精彩互动的新的多媒体形式，成为真正意义上的新一代网络动画标准。

本文主要研究设计了一种嵌入式 SWF 文件解码器。该系统主要应用于高清数字电视或机顶盒，是一种高分辨率的应用解决方案。本文在对 SWF 文件的技术标准进行深入研究的基础上，完成了以下工作：分析了具体的解码流程，同时对矢量图形的渲染工作原理进行了分析。研究设计了 Linux 嵌入系统下的 SWF 解码器的整体构架，并实现了 SWF 文件的解码功能模块。为达到良好的 Flash 播放效果，本文结合了硬件开发环境的性能情况，对解码器进行优化研究，主要采用了矩阵运算、帧间复用和位图缓存优化方法。在按照上述设计方法并实现 SWF 文件解码播放的前提下，完成对其性能的测试，在图形渲染效果、功能实现和播放速率等方面进行评估。该解决方案有效地解决了 SWF 文件播放运行时占用系统资源较多、播放不流畅等问题，可顺利播放网络上大部分动画。

关键词： 嵌入式 高分辨率 SWF 解码器 矢量图形

Design and Implementation of an Embedded SWF Decoder with High-Resolution

ABSTRACT

In recent years, Flash technology is winning more and more attention as well as favor from people, and its application field is also expanding. It has exceeded the limitation of computer and is extensively used in various kinds of equipments. As a result, the embedded products, which support Flash files, appear. At the same time, as further propelling of the Merger of the Three Networks and rapid prevailing of high definition digital TV and set top box, the play function of Flash will gradually become a standard configure for the high definition digital TV and set top box, which accelerates the diversity development of multi media application and enriches people's cultural life.

SWF is a recommendation version of Flash vector graphics file format, which is a kind of animation file format supporting vector and point lattice graphics. Its unique two-dimension webpage multi media technology combines various elements such as vector animation, voice compressed encoding and action script etc. all together, to create a new multi media format full of sound and color with spectacular interaction, which actually become a new generation of cyber animation standard.

This article mainly focuses on the research and design of an embedded SWF file decoder. The embedded SWF file decoder system is mainly applied in high definition digital TV or set top box, and it is an application solution with high-resolution. Initially, this article analyzes the technological standard of SWF file, and elaborates on the general structure design of SWF file decoder and the realization of specific function module; then it puts forward the optimization method for the key part of decoder; finally, based on above mentioned design method and realization of SWF file decoding playing, with

accomplished function test, it evaluates the graphic embellishing effect, function realization and playing velocity etc.. This solution can solve problems effectively, including the problem of occupying too much system resources while SWF file is running, the problem of disfluency, etc. and ensure that most of the animation files can be played on the internet.

KEY WORDS: embedded high-resolution SWF decoder vector graphics

目录

第一章 绪论

1.1 研究背景

近几年，国内电视行业正在处在从模拟电视转变到数字电视，从标准清晰度电视（SDTV）转变到高清晰度电视（HDTV）的重要阶段[1]。高清数字电视是相对目前国内电视机普遍使用的“标清”来定义的。高清是一种视频标准，物理分辨率在1280×720 以上，全高清物理分辨率高达 1920×1080 像素，即 1080i 和 1080p，是高清的顶级规格。与标准清晰度电视相比，高清晰度电视具有身临其境的逼真性和感染力，极大的满足了家庭用户欣赏水平日益提高的需求。此外，高清数字电视具有广泛的外延性，一方面现有的影视传播网络将可以开展游戏、数字增值服务等新业务，另一方面影视内容可以通过移动宽频等新手段进行传输。因此，除了在传统广播电视领域外，高清数字电视在电子出版、广告宣传、视频影院、教育、医疗等等需要高清晰度视频的场合也起到越来越重要的作用[2]。

随着计算机技术的飞速发展，三网融合的进一步推进，有线广播电视网正在向数字化、网络化、产业化方向发展,最终建成宽带综合信息网。依托有线广播电视网提供综合信息业务的关键设备之一是用户终端设备——数字机顶盒，数字电视机顶盒可以支持几乎所有的广播和交互式多媒体应用[3]。高清数字电视机可以分为“一体机”和“分体机”两种类型。一体机就是在电视显示器内置高清机顶盒（信源编码、信道解码、条件接受）的完整功能，可以标识为 HDTV receiver。分体机就是不内置高清机顶盒功能的数字电视显示器，以 HDTV ready 来标识，表示可以接收并正确显示出高清机顶盒输出的 HDTV 图像格式信号[4]。未来的机顶盒将能提供更多的业务功能：视频点播、上网浏览、电子邮件、互动游戏及 IP 电话、可视电话等。作为越来越多功能化的终端产品，因此它也必须支持更多的多媒体格式，现在市面上的机顶盒能够支持 H.264、MPEG-2、MPEG-4 和 RMVB 等格式，但因为技术上的一些瓶颈，能够支持 Flash 动画文件播放的机顶盒还比较难见到[5]。

Flash 是一种交互式矢量多媒体技术，集各类媒体元素、动态效果、用户交互于一体，通过内部对象及其属性特征表达丰富的语义信息，是 Internet 上高效传递矢量图形、文本、视频和声音的媒体格式[6]。它的前身是 Future Splash, 为早期网上流行的矢量动画插件, 被 Macromedia 公司收购了后, 便将其改造为 Flash 2。现在，Flash 已经被 Adobe 公司购买，最新的版本是 FlashCS4，也就是 flash10。Flash 动画也是

利用人的视觉暂留特性，快速播放一系列连续运动变化的图形图像，包括画面的缩放、旋转、变换、淡入淡出等等特殊效果[7]。由于 Flash 动画的实用性、互动性和娱乐性很强，它已经逐渐融入人们的工作和生活[8]。随着嵌入式系统及软硬件平台的发展，Flash 应用领域也在不断扩展，它超越了计算机的界限，广泛地应用在各种设备中。然而，在某些嵌入式平台上,由于种种条件的限制,按照通常思路设计出的解码器,在嵌入式平台上运行时速度和效果远达不到实用要求。例如应用于高清数字电视或机顶盒等设备时，存在兼容性差、对硬件环境要求高、占用系统资源较多、播放不流畅等缺陷。

SWF 是 Macromedia 公司的动画设计软件 Flash 的专用格式，是基于 Shockwave 技术的流式动画格式，源文件为.fla。由于其具有体积小，功能强，交互能力好，支持多个层和时间线程等特点，故越来越多地应用在网络动画中[9]。SWF 文件采用曲线方程描述其内容，不是由点阵组成内容，因此这种格式的动画在缩放时不会失真，非常适合描述由几何图形组成的动画[10]。

在这样的背景下，本课题通过分析 SWF 文件规范和所使用的嵌入式平台的软硬件资源情况，以及深入研究 SWF 解码器在高分辨率应用（比如高清数字电视）下的优化方法，设计实现了一款 SWF 解码器，目前已经应用到 PowerLayer 公司高清数字多媒体 SOC 芯片，取得出众的使用效果和用户反馈。

1.2 国内外研究现状

所谓嵌入就是将计算机的硬件或软件嵌入其他电器设备中，构成了一种新的系统，即嵌入式系统。如今，随着信息技术的飞速发展，嵌入式技术和设备的应用已经广泛应用到工业控制、仿真系统、医疗器材、信息家电、通信设备等众多领域。目前，围绕嵌入式系统展开的研究和开发，已经成为计算机软硬件技术发展最活跃的方向之一[11]。而被称为是“最为灵活的前台”的 Flash，由于其小巧、可交互、跨平台的特点，越来越多的支持 Flash 播放的嵌入式产品出现在人们的工作生活中，并逐渐成为各种智能终端的标准配置。但是各类嵌入式硬件环境的 CPU 计算能力和内存的大小是一定且各不相同的，故各大生产厂商推出的支持 Flash 播放功能的多媒体终端，对各自成熟平台依赖性较强，移植较为困难。并且各厂商对 Flash 播放功能的支持也参差不齐，很多用户只能受限对硬件资源要求不高的文件，比如 Flash 中的游戏开发已经进行了多年的尝试。但至今为止仍然停留在中、小型游戏的开发上。游戏开发的很大一部份都受限于它的 CPU 能力和大量代码的管理。

Macromedia 在 2003 年推出了手机播放 Flash 的解决方案，但是由于兼容性差，

大多数的手机和机顶盒生产厂商无法受惠该技术[12]。直到2009年初，Adobe公司宣布其最新的版本Flash多媒体平台，将在根本上把Flash技术带到连接Internet的电视机、机顶盒、蓝光播放器以及其他数字家庭设备。将Flash引入这些设备的主要目的，是使得用户可以在他们的电视机上观看高清视频、运行互动程序以及获得新的用户界面。目前，Adobe已经寻求到多家公司的支持，计划在他们的设备上使用这项技术，包括Intel, Comcast, Disney Interactive, Netflix, Atlantic Records, 以及New York Times Company。Adobe同时也将Flash技术移植到了智能手机平台上，移动版本的Flash使得用户可以观看采用Flash技术的视频。而现在Adobe则将注意力转移到了起居室中高清电视的大屏幕上面，这意味着人们可以在他们的电视上直接获取内容丰富的网络视频资源，而不是一小部分针对电视特别编码的资源。尽管何种电子消费产品将使用新版本的Flash还未公开，但是这项技术已经对设备制造着与应用开发者开放，预计支持Flash的电视机与机顶盒将在年内上市。

近几年,国内也出现了一批嵌入式 Flash 播放器产品，但由于硬件资源的限制，往往仅限于播放运营商提供下载的Flash动画和游戏。使用户无法利用互联网上丰富廉价的SWF资源，也限制了产品对网页浏览的支持。同时各厂商对SWF技术的支持强烈依赖各自的平台，可移植性差，加大的开发成本[12]。

1.3 本文主要工作及安排

本文主要研究目前流行的Flash动画技术在高清数字电视或机顶盒中的应用，基于嵌入式Linux设计并实现了嵌入式SWF解码器。具体工作如下：

1. 在理解SWF文件标准技术基础上分析了具体的解码流程，同时对矢量图形的渲染工作原理进行了分析；

2. 研究设计Linux嵌入系统下的SWF解码器的整体构架，并实现了SWF文件的解码功能模块；

3. 为达到良好的Flash播放效果，结合了硬件开发环境的性能情况，对解码器进行优化研究。主要采用了矩阵运算、帧间复用和位图缓存优化方法；

4. 在按照上述设计方法并实现SWF文件解码播放的基础上，完成对其性能的测试，在图形渲染效果、功能实现和播放速率等方面进行评估。

论文结构安排如下：

第一章阐述了嵌入式SWF解码器应用研究背景及意义，介绍了国内外Flash解码技术的发展及研究情况，然后归纳了本论文的研究工作和主要内容；

第二章介绍了SWF文件的技术标准，首先引入了矢量图形的原理及特点，然后

详细分析介绍了 SWF 文件的文件头结构、标签结构、元素字典结构、显示列表结构和基本数据类型；

第三章介绍了 SWF 解码器的总体框架设计与具体功能模块的实现，以及解码器的软硬件开发环境；

第四章介绍了三种优化解决方法，并详细说明了方案的可行性；

第五章介绍了系统功能和性能测试结果；

第六章进行总结，提出需要进一步改进的方面。

第二章 SWF 技术分析

2.1 矢量图形简介

计算机中所使用的图像类型一般可以分为矢量图和位图两种。所谓位图，就是由点阵所组成的图像，一般用于照片品质的图像处理。位图可以逼真的反映外界事物，但放大时会引起图像失真，并且文件占用空间大。而矢量图是以矢量方式来记录图像内容的，由轮廓和填充组成，其构造原理与位图完全不同。例如，一段圆弧的数据，只记录圆心和圆弧两个端点的坐标，以及线条的粗细和色彩等[13]。矢量图使用一系列计算机指令来表示一副图像，如画点、直线、曲线、圆形、矩形和多边形等等，这种方法实际上是通过数学公式计算获得的，与分辨率无关，这些公式中包括矢量图形所在的坐标位置、大小、轮廓色和填充色等信息，当放大或缩小图形时，只需要在相应数字上乘以放大的倍数或除以缩小的倍数即可[14]。因此，矢量图所占空间非常小，同时将一张矢量图形任意放大或缩小时，其边缘都很平滑，也不会产生颜色块，画质不会随图形的放大或缩小而改变。它克服了位图所固有的缺陷，具有无级缩放、不失真的优点，并可以方便的进行编辑修改。矢量图具有位图所不可比拟的优势，也正因为如此，矢量图在信息资源数字化过程中发挥着重要作用[15]。在计算机显示矢量图时，也往往能看到画图的过程。

矢量图形虽然文件体积小，画面表达准确，但是由于其本身的特性决定了它只适合用来表达一些几何形状的信息。对于照片等层次丰富、细节复杂的画面，矢量图除了增加文件大小，降低图像效果外，丝毫不能体现出其优势，而这正是位图图像所擅长的[16]。

2.2 SWF 文件简介

SWF[17]（shock wave flash）是 Macromedia 公司（现已被 Adobe 公司收购）的动画设计软件 Flash 的专用格式，是一种支持矢量和点阵图形的动画文件格式。SWF 使用矢量来显示大量的动画内容，而矢量图的笔触和填充都是进行科学计算的，因此所制作的动画具有高度的精确性与灵活性。在图像传输方面，SWF 不必等到文件全部下载才能观看，因此特别适合网络传输。即使是在网络传输速率不佳的情况下，也能取得较好的效果。其独特的二维网页多媒体技术[18]，将矢量动画、音频压缩编码和动作脚本等多种要素结合在一起，从而创造出了一种有声有色、精彩互动的新的多媒体形

式, 成为事实上的新一代网络动画标准。

其特点主要是[19]:

(1) 屏幕显示：这种格式主要是为了在屏幕上显示的需求,它支持任何颜色格式、动画和交互按钮的位图的快速显示;

(2) 可扩展性：SWF 文件是采用以标签为基本单元的文件结构,所以它能在实现新的功能特性的同时兼容旧的版本;

(3) 网络传输：这种格式可以在有限的带宽和不可预测的网络状态下传输。而且文件被压缩得非常小,并且支持流式播放。SWF 文件是一种二进制的文件，它并不像 HTML 那样是可阅读的。SWF 文件还使用了一些诸如比特压缩、复用素材等方法来减小文件的尺寸;

(4) 简单性：这种格式简单,使 Flash 播放器小巧易移植。而且 Flash 对具体操作系统特性的依赖很小;

(5) 文件独立性：文件的渲染独立,不依赖诸如字体之类的外部资源;

(6) 灵活性：这种格式能在较差的硬件环境下工作，而且能够尽可能的利用这些硬件资源。而这一点非常重要，因为不同的设备有不同的分辨率和色深;

(7) 速度：SWF 文件能够实现高速和高质量的渲染;

(8) 支持脚本：这种格式包含了约定格式的标签，标签规定了堆栈式机器解释字节码的顺序。字节码支持 ActionScript（动作脚本），能够控制渲染以及和服务器交互。

如今，曾经只是基于网站制作的 SWF，不仅与其他任何在线媒体制作软件相比是一个巨大的突破，而且推动了交互式数字领域的发展。它广泛的流行性和它显著的性能超越了在线领域，已经被集成为移动设备、广播媒体和控制台游戏的多媒体开发工具和平台。但是 SWF 标准的制定权在 Macromedia 公司,所以对 SWF 的第三方支持比较少。同时 SWF 作为最终的动画生成格式, 其创作过程封装在 SWF 文件中, 几乎无法再进行二次编辑。

2.3 SWF 文件结构分析

SWF 文件是由文件头和一组标签（Tag）数据块组成，最后以一个特殊的结束标签（End Tag）结束文件。图 2-1 表示了 SWF 文件的完整结构图。

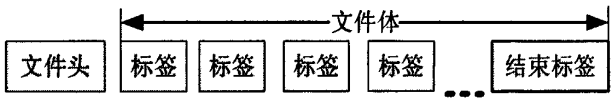

图 2-1 SWF 文件结构[17]

2.3.1 文件头结构

SWF 文件的文件头定义了 Flash 文件的格式、版本、文件长度、帧大小、帧播放速率和帧总数，其格式如表 2-1 所示：

表 2-1 SWF 文件头结构[17]

SWF File Header		
Field	**Type**	**Comment**
Signature	UI8	Signature byte: "F" indicates uncompressed "C" indicates compressed (SWF 6 and later only)
Signature	UI8	Signature byte always "W"
Signature	UI8	Signature byte always "S"
Version	UI8	Single byte file version (for example, 0x06 for SWF 6)
FileLength	UI32	Length of entire file in bytes
FrameSize	RECT	Frame size in twips
FrameRate	UI16	Frame delay in 8.8 fixed number of frames per second
FrameCount	UI16	Total number of frames in file

FLASH 文件头，一般以无符号 8 位整型数据 0x46、0x57、0x53（FWS 简称“F”）或 0x43、0x57、0x53（CWS 简称“C”）开始。如果标识符是“F”表示该文件未被压缩，如果是“C”则表示在整个文件前八个字节，即文件长度字段之后所有的内容，都是采用开放标准的 ZLIB 压缩方法进行压缩。因此，对于标识符是“C”的文件，在处理数据前必须先用 ZLIB 压缩包进行解压，取得原始数据后再进行数据解析[17]。

在标识符之后的一个字节是版本号. 这个版本号不是一个 ASCII 字符，而是一个 8 位的数字. 例如，SWF4 文件的版本号是 0x04，不是 ASCII 字符“4”(0x35)。

FileLength 字段代表包括文件头整个文件的总长度，是无符号 32 位整型数据。如果是一个未压缩的 SWF 文件（FWS 标识符），文件长度字段表示文件的精确大小；如果是一个压缩的 SWF 文件（CWS 标识），文件长度字段表示解压后文件的大小，这样一般就不是实际文件的大小了. 让未压缩（解压后）的大小可见，则可以使解压过程更加有效。

FrameSize 字段表示影片的宽度和高度. 它存在一个 RECT 结构中，表示它的大小可以根据坐标（四个点的坐标）数值的变化而变化. 文件大小 RECT 通常是这样的形式：X_{min} 和 Y_{min} 成员都为 0；X_{max} 和 Y_{max} 成员声明宽度和高度。

FrameRate 字段表示理想的每秒播放帧数. 如果 SWF 文件包含声音流数据，或者 Flash 播放器运行在一个慢的 CPU 上，这个速率是不能保证的。

FrameCount 字段表示 SWF 动画总帧数。

2.3.2 标签（Tag）结构

文件头之后是一些标签化的数据块，这些数据块的格式一致，当播放器播放时如果发现数据块无法解析，就可以跳过，这样就不会影响其它数据块的播放。在每个块中的数据可以指向这个块中的偏移量，但绝不能指向另外一个块的偏移量。这样，在用工具处理SWF文件的时候就任意可以删除、插入和修改(而SWF文件不会被破坏)。

标签是SWF文件的主要内容。每个标签是由标签类和标签长度两个字段组成的。标签类型决定了这个标签本身是短型（<62bytes）或者是长型（<4GB）的标签。用功能来分标签，一般可以分为两类：定义标签(Definition Tags)和控制标签(Control Tags)。定义标签用来定义Flash动画中的元素（Character），包括形状（Shapes）、文本（Text）、图（BitMaps）、声音等[27]。用定义标签是不会绘制任何图形的，不会产生任何动画的。flash 播放器则把这些元素放到一个存储空间里面，这个存储空间一般称之为元素字典。每一个定义标签都有唯一的一个 ID 来对应它自己的字典（Dictionary）[20]。控制标签实现各类组成元素的属性变化、动态效果和人机交互等。比如，DefineShape 和 DefineText 都是定义标签，分别用来定义形状和文本；PlaceObject 和 ShowFrame 都是控制标签，前者在舞台上放置一个对象，后者显示一帧的内容。标签遵循“先定义，后使用”的原则，采用相同的格式，每个标签都是一个独立的个体，任何一个标签与其它标签都没有数据上的关联。如果动画文件有 FileAttributes 标签，则 FileAttributes 标签必须是第一个标签。最后一个标签是结束标签 END[6]。一般定义标签可分为 6 种类型，控制标签分为 3 种类型：Display List、Control 和 Action，其中每一类型又可分为若干子类[27]，详见图 2-2：

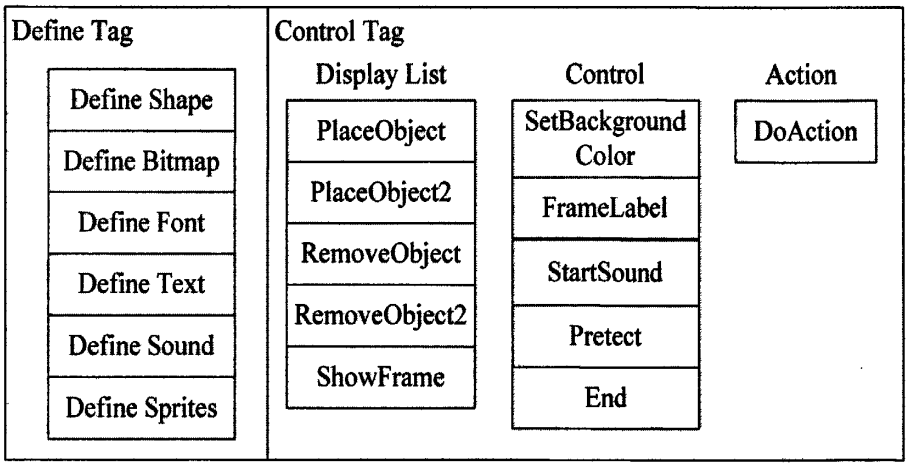

图 2-2 SWF 标签类型

总的来说，标签可以出现任何情况的排序方法，但也不是随便乱排， 它遵循一些规则：

1. 一个标签只能引用之前已经定义过的标签，不能引用在它之后的标签；

2. 一个定义了动画元素的定义标签必须在引用这个元素的控制标签之前；

3. 由于媒体文件被分隔成多个定义标签，流媒体标签必须有顺序，没有顺序的流媒体播放起来也是没有顺序的；

4. 结束标签(End Tag)应该在 Flash 文件的最后。

2.3.3 元素字典(Dictionary)结构

在对标签解析的过程中，会生成元素字典(Dictionary)，是 SWF 文件中使用的各种对象的仓库。所有被定义过的元素，在文件解析过程中，将被记录在元素字典中，供控制标签调用[27]。元素字典的创建和使用如下：

1. 定义标签(Definition Tag)定义一些元素，例如形状、声音、字体和位图等；

2. 这些元素将会分配唯一的 ID 号，这是由解码器流程提供的，一般顺序编号，供控制标签(Control Tag)调用；

3. 以 ID 号为顺序在字典中保存被定义过的元素；

4. 控制标签(Control Tag)通过查找 ID 号获取相应的元素，并据此执行一些动作，如显示图形或播放音频。

每个控制标签都只能又为一个每个控制标签都只能有唯一一个 ID，不能有重复的。比如，第一个元素 ID 是 1，第二个元素 ID 是是 2，依次类推。元素 ID 是为 0 的是一个特殊的标识，被看作是空元素。控制标签并不是唯一指向元素字典的标签。定义标签也可以指向多个元素来定义一些更复杂的元素。例如，定义按钮(Define Button)和定义精灵(Define Sprite)标签都是根据其它元素来定义它们的内容的。定义文字(Define Text)标签可以指向字体元素来为文字选择不同的字体。

图 2-3 展示了元素字典、定义标签、控制标签之间的关系：

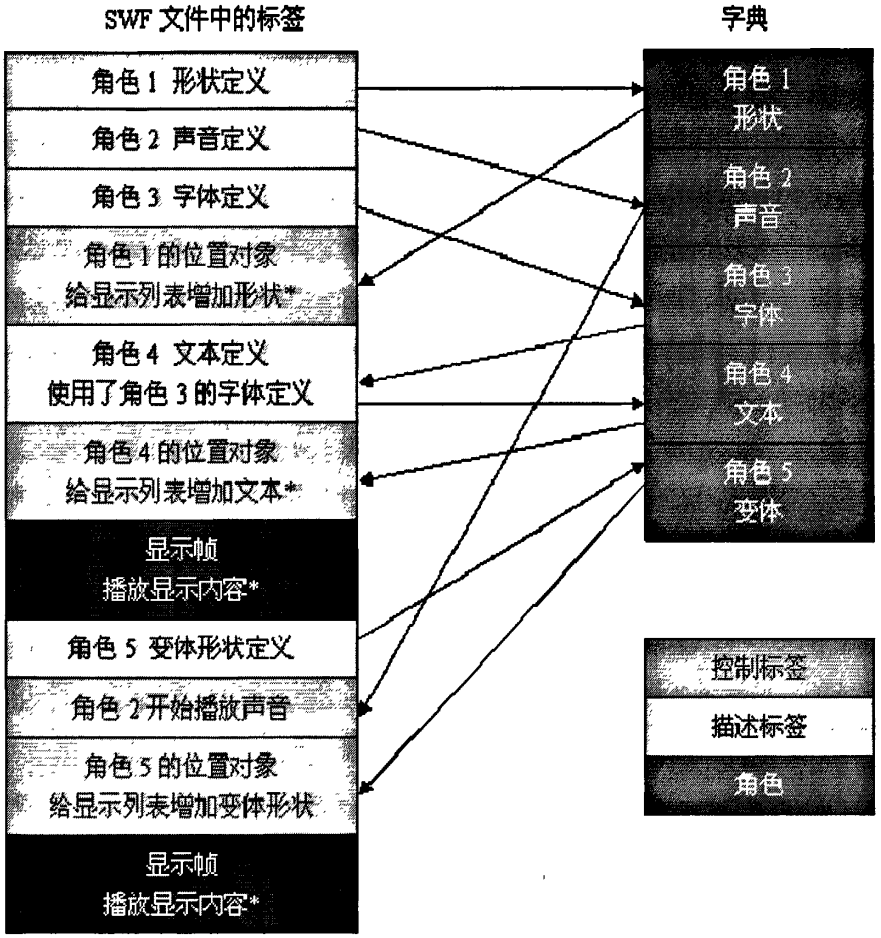

图 2-3 元素字典、定义标签和控制标签关系图[17]

2.3.4 显示列表(Display List)结构

Flash 是一种基于关键帧的动画，通过在两个关键帧之间使用一定的插补算法插入一些中间帧形成，时间域上的基本单位是帧，空间域上的基本单位是对象，帧是对象的载体，场景是对象载体的集合，多个帧顺序播放构成场景，多个场景构成 SWF 文件[21]。

SWF 文件的每一帧的显示都需要经历三个步骤[27]：

1. 定义标签定义对象（定义标签包括 DefineShape,，DefineSprite 等）。每个对象都有一个唯一的 ID，该 ID 被叫做元素（Character），被存储在被叫作字典（Dictionary）的库中。

2. 根据控制标签，将该帧中要显示的元素从字典中复制到显示列表（Display List），该列表内含有下一帧将要显示的所有元素。

3. 完成以上两个步骤后，显示列表中的内容就会通过 ShowFrame 标签呈递给屏幕。显示列表中的所有元素都含有一个深度值（depth），深度决定了元素的排列层次。低深度的元素在高深度的元素之下。如果深度为 1 则元素显示于最底层。显示列表中可以有多个相同的元素，但它们必须处于不同的深度，一个深度只能有一个元素。

图 2-4 说明了 Flash 的显示过程，首先定义三个对象，形状（shape），文本对象（text object）和子图形（sprite）。这些对象被赋予元素 ID 并储存在字典中。元素 1（形状渐变）放在深度 1，当这一帧显示时上层元素会覆盖相同位置的这个元素。元素 2（文本对象）被放置了两次，一次在深度 2，一次在深度 4。元素 3（子图形）置于深度 3。

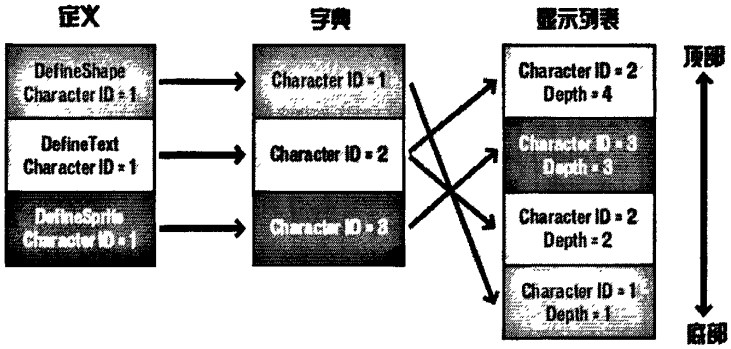

图 2-4 定义标签、元素字典和显示列表关系图[17]

元素显示列表是由控制标签生成的，SWF 文件中主要有五种控制标签用于对元素显示列表进行操作：

- PlaceObject 在显示列表中添加一个元素
- PlaceObject2 在指定的深度添加或修改元素
- RemoveObject 从显示列表移除指定的元素
- RemoveObject2 从指定的深度移除元素
- ShowFrame 显示该帧

2.3.5 矢量图形结构

SWF 文件的矢量图形的结构设计特点是小巧、灵活和渲染高效。它的绘制过程和大多数矢量图形类似：一系列封闭或开放的曲线组成一个路径（Path），然后确定该路径（Path）的线条样式（Line Style）与填充样式（Fill Style）；再由许多这样的路径（Path）组成一个图形形状（Shape）[27]。

1. 填充样式（Fill Style）

填充样式（Fill Style）即封闭的路径形成的区域，可以填充 SWF 文件支持的颜色、渐变色或位图图像。Flash 支持三种填充方式：Solid Fill、Gradient Fill 和 Bitmap Fill。Solid Fill 只填充一种简单的 RGB 或 ARGB 的单一颜色；Gradient Fill 填充以线性或放射性变化的渐变颜色；Bitmap Fill 用提供的位图数据进行填充，有两种方式：原始大小填充和平铺填充[27]。

2. 线条样式（Line Style）

线条样式（Line Style）用于指定渲染路径线条的颜色和宽度。用 RGB 或 ARGB 表示一种颜色，并用 twips 为单位定义一个线条宽度。同填充样式一样，可以指定多种样式，在形状描述中可以以索引号来使用不同的样式[27]。

3. 形状（Shape）

介绍了填充样式（Fill Style）与线条样式（Line Style），就能引出 SWF 文件中最重要的元素——形状（Shape）了[27]，其结构如图 2-5 所示：

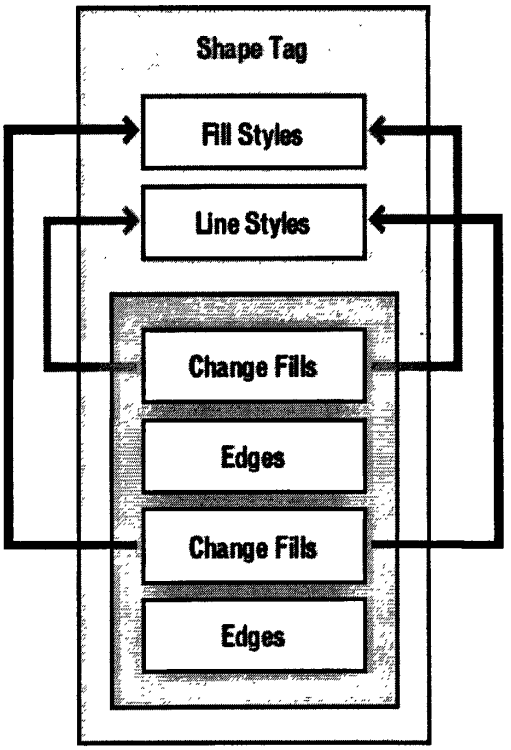

图 2-5 Shape 结构图[17]

由图可知，形状（Shape）结构首先定义了填充样式（Fill Style）列表与线条样式（Line Style）列表，它们只会被定义一次，使用时通过索引号获取[27]。

图中蓝色区域是形状记录（Shape Record）列表。其中，第一个形状记录从两个列表中分别选定了填充样式（Fill Style）与线条样式（Line Style），并将画笔移动至

形状（Shape）的起始点。紧跟着的记录（Record），就是一系列的贝塞尔曲线或直线（Edge）描绘出一条路径。然后，下一个记录（Record）重新选择填充样式（Fill Style）和线条样式（Line Style）；再下一个记录又是一系列的 Edge。如此反复，最终形成了一个复杂的形状[27]。

2.3.6 基本数据类型

SWF 文件里面有很多复杂的数据结构，但是这些复杂的数据结构都是由一些基本的数据类型组成的。

1. 坐标和缇（twip）

SWF 文件格式用整数来存储坐标，它的默认单位是缇（twip），20 twip = 1 pixel。在没有缩放的情况下，即 100%播放时，SWF 中的一个逻辑像素和屏幕上的一个像素是对应的。使用缇的好处就是能够获得比使用像素更高的精度。比如，一个 800twips 宽，400twips 高的矩形，就会被解释成 40×20 像素的大小，这时矩形的边缘是没有锯齿的。如果是 790×390twips（39.5× 19.5 像素）的话，它的就会有轻微的模糊边缘。

SWF 的坐标系和传统的图像坐标一样，x 轴代表水平方向，从左到右数值递增；y 轴代表垂直方向，数值从下到上是增加的。

2. 整数类型和字节顺序

SWF 文件使用是的 8 位、16 位、32 位、64 位有符号和无符号的整数。这些整数在 SWF 文件中是以 little-endian 的顺序来存储的，低字节在前，高字节在后，而每个字节的每一位则是按 big-endian 的顺序来排列的，即高位在先，低位在后。在 SWF 文件中，所有整数类型都必须是字节对齐的，也就是说，一个整数值的第一位必须存储在一个字节的第一位。

3. 定点数

SWF 文件格式支持两种类型的定点数：32 位定点数和 16 位定点数。32 位定点数的格式是 16.16，也就是小数点前面和小数点后面的每一部分各占 2 个字节。并且也采用了 little-endian 的形式。比如:7.5 = 0x0007.8000 在 SWF 文件中就是以 00 80 07 00 的形式存储的。

4. 浮点数

swf8 及其后续的版本支持和 IEEE Standard 754 兼容的浮点数类型。一共有三种类型，分别是：Half-precision (16-bit) floating-point number 半精度、Single-precision(32-bit)单精度、Double-precision (64-bit)双精度。除了半精度的这种浮点数据类型以外，其它两种都符合 IEEE Standard 754 的标准。半精度的也是和 IEEE

Standard 754 类似的，只是改变了标准中分配给尾数和指数的位数。在半精度型的浮点数中，一位作为符号位；5 位是分配给指数部分，实际的指数是 5 位数表示的数和 16 的差值；剩余的 10 位用来表示尾数。

5. 位值

位值是用多少位来表示一个数值是不确定的，也就是说，表示值的时候，位数是可变的。它可以表示三种类型的数值：无符号整数、有符号整数和有符号的 16.16 格式的定点数。位值不是位齐的，而其它一些数据类型，比如前面提到过的无符号整数等都是必须位齐的。如果一个位齐的数据类型后面跟着一个位值，那么最后几位如果不能填满的话，应该用 0 补齐。并且以上这些数如果需要扩展的话，是按符号扩展来进行扩展的。

6. 字符串类型

字符串类型就是指以空字符结尾的字符的序列。字符值的格式就是一些字节的序列，这个序列以空字符字节结束。

2.4 本章小结

本章首先简单介绍了矢量图形的原理及特点，其次详细分析了 SWF 文件格式标准。其中主要讨论了 SWF 文件的文件头结构、标签结构、元素字典结构和显示列表结构，对 SWF 矢量图形的绘制过程做了简单描述。最后介绍了 SWF 文件的基本数据类型，为之后的解码器设计提供了理论基础和参考依据。

第三章 SWF 解码器的设计与实现

3.1 嵌入式系统运行环境

嵌入式系统一般指非 PC 系统，有计算机功能但又不称之为计算机的设备。其定义为：以应用为中心，以计算机技术为基础，软硬件可裁剪，从而能够适应实际应用中对功能、可靠性、成本、体积、功耗等严格要求的专用计算机系统。简单地说，嵌入式系统就是系统的应用软件与系统硬件一体化。具有软件代码小，高度自动化，响应速度快等特点。特别适合于要求实时的和多任务的体系。嵌入式系统主要由嵌入式处理器、相关支撑硬件、嵌入式操作系统及应用软件系统等组成。它们之间的关系如图 3-1 所示。

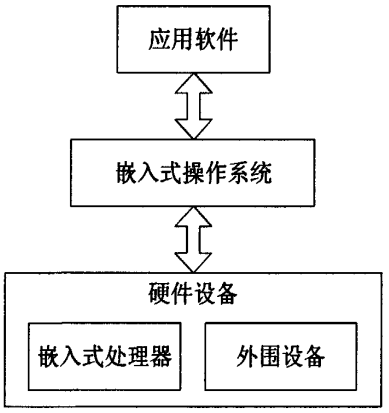

图 3-1 嵌入式体系结构

嵌入式系统几乎包括了人们周围的所有电器设备：掌上 PDA，电视机顶盒，数字电视，汽车，微波炉，数字相机，家庭自动化系统，自动售货机，蜂窝式电话，消费电子设备，工业自动化，医疗仪器等等[22]。

3.1.1 硬件环境

本方案使用的硬件平台是 PowerLayer 公司的 SOC 芯片，支持全高清 MPEG-2 视频解码，多种流行的音视频格式的解码，包括 MPEG-4，H.264/AVC，DIVX，Real-8/9/10，FLV，Dolby AC-3，MPEG，AAC，Real audio 等。该芯片内部具有两颗 MIPS24k 核，一个 DSP 核，2D Graphics Engine，以及图像增强等功能单元，并带有丰富的外围接口，包括 BT.656/601，10-bit HD YUV DAC，30-bit 数据高清 YUV 输入，I2S and S/PDIF audio，smart card，2 个异步通用串口（UART），2 个 USB 2.0 可

作为机或设备的接口，以太网接口和红外接口。图 3-2 表示了该芯片内部功能框图。

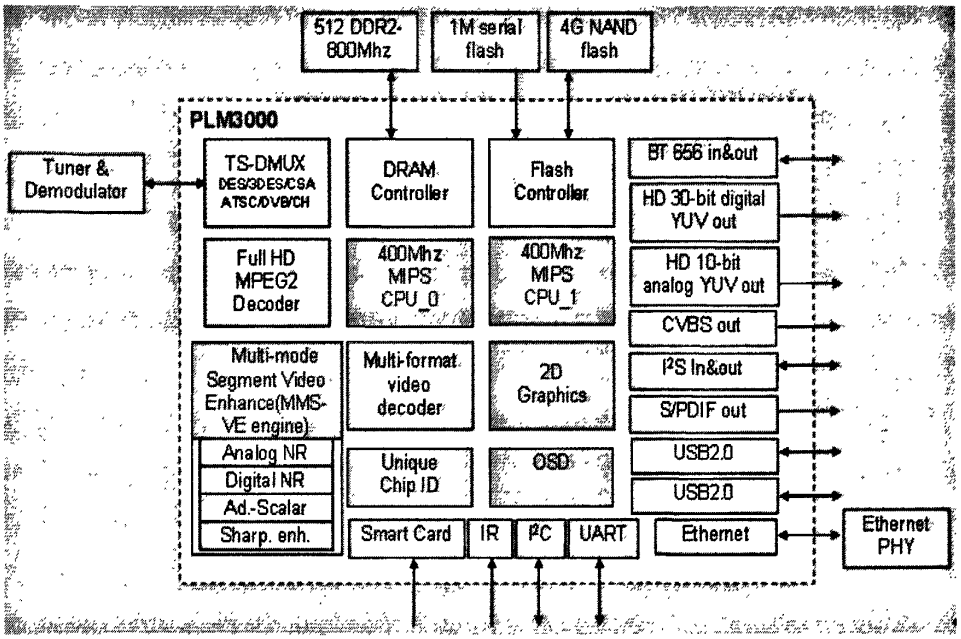

图 3-2　采用 CPU 功能结构框图

其硬件资源十分丰富，对于 Flash 应用主要的优势在于，多核协同支持，简单的 2D 图形加速，以及使用高性能图像增强技术来弥补在 Scale 低分辨率 Flash 过程中的画面损失。

3.1.2 软件环境

该方案开发平台软件系统采用嵌入式 Linux 操作系统[23]。该系统是一个完全符合 GNU/GPL 公约的操作系统，完全开放代码。Linux 作为一个典型的现代网络型操作系统，其中所涉及到的技术实现涵盖了操作系统技术的最新成果。它是一个多用户多任务操作系统，支持分时处理和软实时处理，并带有微内核特征（如模块加载/卸载机制），具有很好的定制特性。

嵌入式 Linux（Embedded Linux）是指对标准 Linux 经过小型化裁剪处理之后，能够固化在容量只有几 KB 或者几 MB 字节的存储器芯片或者单片机中，是适合于特定嵌入式应用场合的专用 Linux 操作系统。在目前已经开发成功的嵌入式系统中，大约一半使用的是 Linux。这与它自身的优良特性是分不开的。

嵌入式 Linux 同 Linux 一样，具有低成本、多种硬件平台支持、优异的性能和良好的网络支持等优点。另外，为了更好地适应嵌入式领域的开发，嵌入式 Linux 还在 Linux 基础上做了如下改进[24]：

1．改善的内核结构

Linux 内核采用的是整体式结构（Monolithic），整个内核是一个单独的、非常大的程序，这样虽然能够使系统的各个部分直接沟通，提高系统响应速度，但与嵌入式系统存储容量小、资源有限的特点不相符合。因此，在嵌入式系统经常采用的是另一种称为微内核（Microkernel）的体系结构，即内核本身只提供一些最基本的操作系统功能，如任务调试、内存管理、中断处理等，而类似于文件系统和网络协议等附加功能则运行在用户空间中，并且可以根据实际需要进行取舍。这样就大减小了内核的体积，便于维护和移植。

2．提高的系统实时性

由于现有的 Linux 是一个通用的操作系统，虽然它也采用了许多技术来加快系统的运行和响应速度，但从本质上来说并不是一个嵌入式实时操作系统。因此，利用 Linux 作为底层操作系统，在其上进行实时化改造，从而构建出一个具有实时处理能力的嵌入式系统，如 RT-Linux 已经成功地应用于航天飞机的空间数据采集、科学仪器测控和电影特技图像处理等各种领域。

3.1.3 嵌入式开发方法

嵌入式软件开发所采用的编译为交叉编译。所谓交叉编译就是在一个平台上生成可以在另一个平台上执行的代码。一般把进行交叉编译的主机称为宿主机，也就是普通的通用 PC，而把程序实际的运行环境称为目标机，也就是嵌入式系统环境。由于一般通用 PC 拥有非常丰富的系统资源、使用方便的集成开发环境和调试工具等，而嵌入式系统的系统资源非常紧缺，没有相关的编译工具，因此，嵌入式系统的开发借助宿主机来编译出目标机的可执行代码。本系统的软件就是在这种宿主机——目标机的开发环境下进行开发的。在宿主机与目标机之间的通信方式可以有多种：串口、并口、网络、JTAG 等[25]，如图 3-3 所示：

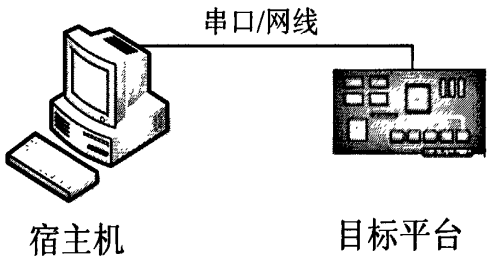

图 3-3 嵌入式系统开发方式

3.2 解码器总体设计

3.2.1 解码器整体构架

该解码器的程序设计采用模块化的设计思想，其系统的功能结构如图 3-4 所示。如果系统中提供了相应的硬件解码功能，则可以将部分功能由硬件实现。可见这种灵活的架构适合于嵌入式系统的广泛应用。

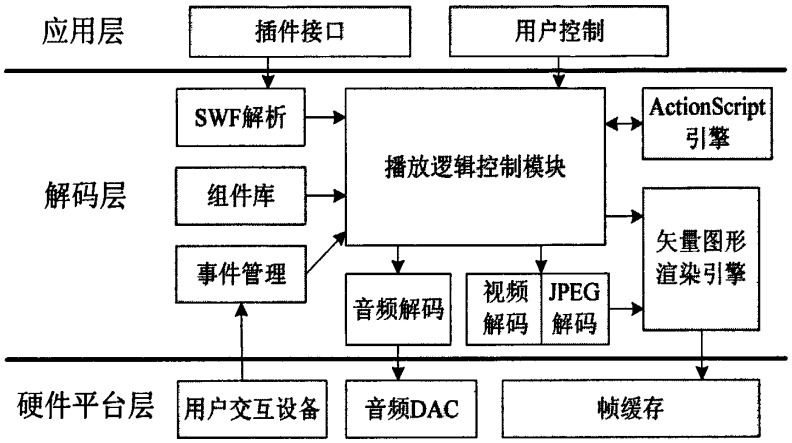

图 3-4 解码器系统结构

系统结构说明如下：

1．应用层提供了上层用户的调用接口，完成解码器的初始化和播放流程的用户控制。

2．系统的解码层则完成了 Flash 的解码与播放，由以下功能模块组成：

(1) SWF 解析：主要完成 Flash 文档的解析工作；

(2) 组件库：存储和管理所有定义标签定义的动画元素（Characters），即字典与控制列表的建立；

(3) 事件管理：用于接收和处理用户交互事件，如鼠标点击、遥控器按键等。并且在播放到相应帧时触发该事件；

(4) 播放逻辑控制模块：协调组件库、事件管理、各个解码模块，实现对 Flash 解码与播放的控制；

(5) 矢量图形渲染引擎：负责形状、字体等的渲染与填充。根据显示列表中的动画元素和相关显示数据，渲染出位图，并最终显示在屏幕上。该模块是 SWF 解码器核心的部分，也是最复杂的，因此也是优化工作的重点；

(6) 音视频、JPEG 解码：包括 MP3 解码库、H.263 解码库、JPEG 解码库，每个

解码单元都可以由硬件解码器替换；

(7) ActionScript 引擎：解析并执行 ActionScript 脚本。

3．硬件平台层则主要完成了与平台相关的操作。根据不同移植平台，设置用户输入方式，实现音频输出与显示缓存的管理。

3.2.2 软件工作流程

该方案采用了两个独立的线程分别完成 SWF 文件解析与影片播放。主线程完成播放控制与影片显示；而解析线程完成对 SWF 文件比特流解码。图 3.5 为解码器的程序总体流程图。

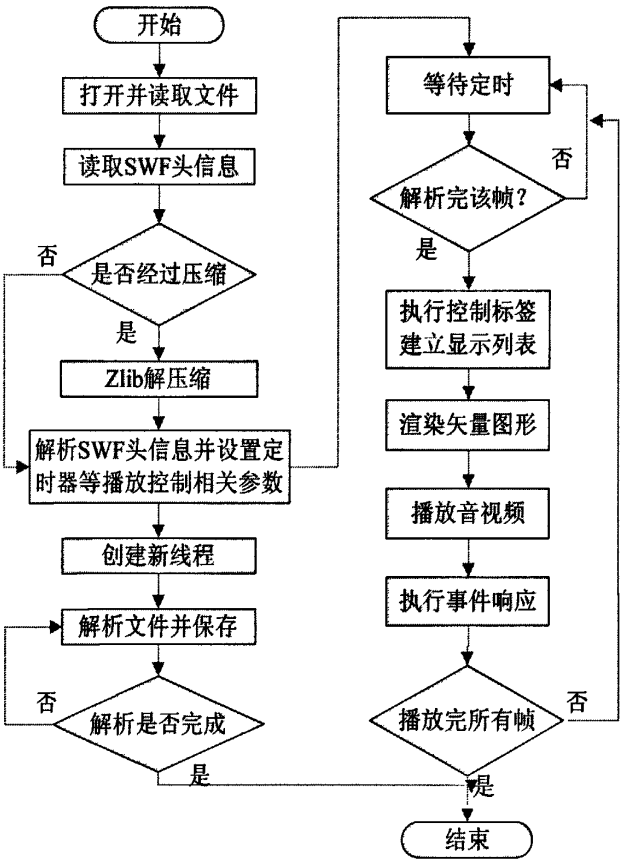

图 3-5 解码器程序流程

因此，对于 SWF 解码器的设计实现主要包括：

1．解析 SWF 文件头，根据文件头信息配置 Flash 的播放环境；

2．正确解析 SWF 文件体中的每个 Tag，并建立组件字典等主要数据结构；

3．根据控制标签建立显示列表，表明在每一帧中各层组件的显示逻辑结构；

4．渲染矢量图形的帧画面，执行动作(ActionScript)；

5．完整回收资源，结束播放。

3.3 标签解析模块的实现

SWF 文件是由不同的标签组成的，标签解析是 SWF 文件解析的基础，设计 SWF 解码器必须从解析标签入手，考虑对于不同标签的解析与应用。SWF 文件的组成结构参见 2.3 节中的介绍，依据 SWF 文件的多标签结构，确定标签解析模块的设计实现包括：

1．SWF 文件头解析，根据文件头信息配置 FLASH 动画的播放环境；

2．从 SWF 文件中解析出每个 Tag，并将定义标签放在建立的标签字典中；

3．根据从 SWF 文件中解析出的控制标签建立控制列表；

4．文件解析过程结束，得到控制列表与字典，释放中间资源。

3.1.1 文件头解析

SWF 文件头的结构在 2.3.1 节有介绍，依据其结构定义的特点，解析应按照如下方式进行[26]：

1．首先读取前 8 个字节，判断 SWF 文件标志是否正确，同时是否压缩；版本号保存下来配合 Tag 解析；文件长度保存用于结束时校验，若文件是压缩的，使用 Zlib 解压缩库解压出正常的文件，以便后续可以正确地进行解析工作；

2．继续读取信息直到 SWF 文件头结束，根据读到的文件头信息定制 Flash 播放环境：

(1) 根据 FileLength 确定文件缓存大小，根据屏幕显示区域的大小确定帧缓存的大小；

(2) 根据 FrameSize 和在屏幕上显示区域的大小，设定显示变换矩阵；

(3) 根据 FrameRate 帧速率设定解码器的帧率控制和时间控制，以实现原速动画重放，可设定跳帧以保证速度。

这样，根据 SWF 文件头定义的 FLASH 动画播放的基本环境就建立起来了，接下来就将进入解析 SWF 文件的主体部分，包括各种解析 Tag，生成相应的解码数据结构（控制列表、显示列表）。

3.1.2 标签解析

SWF 文件的主体是由一系列的标签组成的，在 2.3 节有详细说明。将所有的标签从文件中依次正确无误的提取出来，分析标签的类型，确定标签的长度，使标签解析独立方便，是标签解析模块的主要工作。解析过程如下[26]：

(1) 取得标签类型，判别是否合法的在 SWF 文件格式中规定的标签；如果是则执行(2)；否则报错，退出；

(2) 根据标签类型调用对应的处理函数，读取标签基本信息，包括长度，类型。如果是 Define Tag 则执行(3)；如果是 Control Tag 执行(4)；如果是 End Tag 执行(6)；

(3) 根据不同的 Define Tag 类型建立对应的组件数据结构，并从文件中读取相应的数据信息(包括唯一的 ID，显示矩形区域等)，初始化数据成员。最后将该标签数据结构以 ID 为标识存入字典；

(4) 根据不同的 Control Tag 类型建立对应的数据结构，当从文件中读取数据初始化该数据成员后，将其加入该帧的控制列表(Control List)中；

(5) 将读取指针指向下一个 Tag，执行(1)；

(6) 处理 End Tag 标签，将帧计数加一，同时更新控制列表。

SWF 文件解析程序是一个独立的线程在工作，其主要工作是文件中解析出不同的 Define Tag 和 Control Tag，并将它们加入到字典以及控制列表中。只要将一帧的所有信息解析完成就可以进行动画渲染和播放了。SWF 解码器中对于不同的 Flash 动画都会建立一个 Movie 类对象，只要该对象完成了 SWF 解析工作，就会有该动画的字典，以及每帧的控制列表。

3.4 播放逻辑控制模块

Flash 的播放与控制由播放逻辑控制模块来完成。该模块根据 Flash 文件中的控制标签（Control Tag），操作组件库中的动画元素（Characters），控制每一帧的显示内容，从而实现 Flash 的动画效果。

该模块中主要操作着两个重要的结构：控制列表（Control List）和显示列表（Display List）[27]。

3.2.1 控制列表的生成

标签解析模块根据 SWF 文件标准读取文件数据流，并将合法的信息存储在指定的数据结构中。如果遇到定义标签，定义标签定义对象的内容，就会为该组件元素指定一个唯一的 ID，存入字典中。如果遇到控制标签，就以 ShowFrame 控制标签为间隔，组成多个控制标签的列表，每个列表就是这一帧的所有动作内容。由控制标签

PlaceObject2 选择需要显示的组件，把这些组件从字典中取出按照层的属性放置到显示列表中去。图 3-6 表示了 SWF 解码过程中的每帧的控制列表是由一些 ControlTAG 组成的。

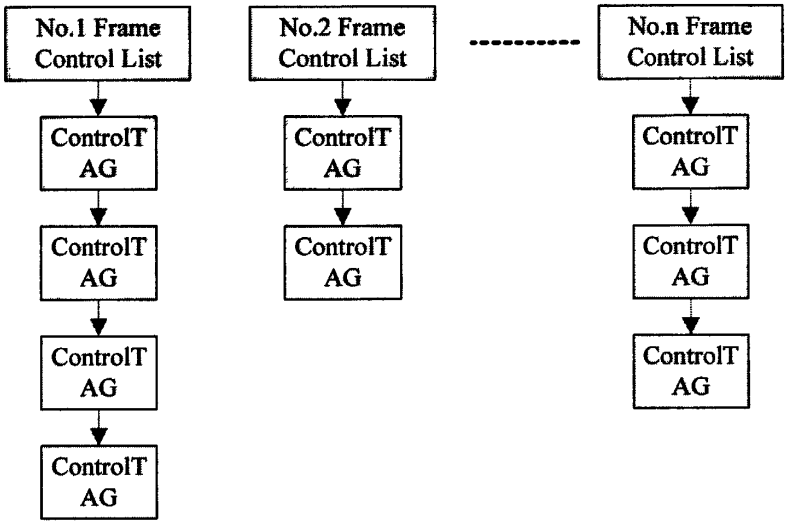

图 3-6 控制列表（Control List）结构示意图

增加到控制列表中的控制标签有三类：

1．与显示列表有关

这类的控制标签有 PlaceObject, PlaceObject2, PlaceObject3, RemoveObject, RemoveObject2, ShowFrame。通过执行 PlaceObject2 和 RemoveObject2 控制标签来产生动画要素。前者可以在指定层放置组件、修改组件、替换组件；后者是把某一层的组件移除。执行这类控制标签后，会生成一个显示列表（Display List）。当执行到 ShowFrame 标签时，显示列表中的内容将按照排列顺序进行渲染，最终显示到屏幕上。

2．与 SWF 播放属性有关

这类标签有很多，如 SetBackgroundColor、FrameLable、StartSound、StartSound2、End 等。这些标签定义了 SWF 播放所需要的一些属性。在执行这些标签时，即可改变播放属性。

3．DoAction 与动作脚本有关

ActionScript 是 Flash 文件的内置脚本语言，用它可以实现动画特效、良好的用户交互。对于 Flash 文件中的 ActionScript 被编译成一系列字节码，因此必须通过解释器得到动作语言。ActionScript 引擎就负责与播放逻辑控制模块交互信息，完成 ActionScript 的翻译。播放逻辑控制模块将由 SWF 解析产生的动作字节码和该字节码附属的动画对象交由 ActionScript 引擎处理[27]。

控制帧动作的 DoAction Tag 将在解析后加入控制列表。还有其它两类的动作类

型：

(1) 解析控制标签 PlaceObejct2 时如 PlaceFlagHasClipActions 标记为真，则可以得到影片剪辑动作类型的字节码。

(2) 在解析 DefineButton 的 Actions 属性时可以得到 ACTIONRECORD，解析该 Record 可以得到 Button 动作类型的字节码。

3.2.2 声音的控制处理

SWF 文件支持两种声音形式：事件声音和流声音。事件声音用来表现某些事件发生时的声音，如鼠标点击或播放到特定帧。事件声音必须先由 DefineSound 标签定义声音数据，并解析出来送给 Sound Engine 进行音频数据的解码，Sound Engine 把解码后的 PCM 音频数据调用底层的音频播放接口，进行声音的播放。事件声音播放由控制标签 StartSound 打开播放预先存储在字典里由声音定义标签指定的声音信息。流声音可以随播放时间轴进行同步下载和播放，因为声音数据包含在每帧的控制标签 SoundStreamBlock 里面。流声音会随着播放的速度来适时下载音频数据进行解码播放。

3.2.3 用户交互控制处理

SWF 文件中的按钮用于用户交互组件，它可以相应鼠标点击、拖拽等复杂事件，并在制作 Flash 文件中定义不同的按钮相应处理。

按钮由三个状态组成：UP、OVER、DOWN。从图 3-7 中可以清楚地表示按钮的状态。

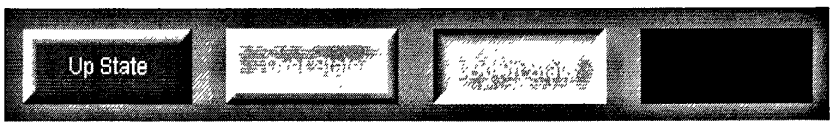

图 3-7 Button 的四种状态示意图[26]

Up State 是按钮的默认显示状态，即是 Flash 开始播放时的显示状态，也是鼠标在按钮以外区域的显示状态。Over State 是当鼠标移动到按钮区域内的显示状态。Down State 是鼠标在按钮区域内按下键时的显示状态。而图中的最后一个显示方式被称作 Hit State，它定义了属于按钮的区域。这个状态从来不会被显示，它是用来通知鼠标该区域是这个按钮的相应区域。

按钮的每个状态是由字典里定义好的元素来描述的，并用 Define Button 标签来表示。对于按钮的不同状态显示，会有不同的元素被加入到显示列表中，最终完成按

钮形状的渲染。

在 Flash 播放过程中，当用户交互的条件符合该按钮的触发条件时，则把该标签中的字节码提出，交给 ActionScript 引擎进行字节码的解码。

Button 解码分为以下几个步骤[26]：

(1) 解析一帧中出现的所有的 Button 对象，把这些 Button 放到 Button 数组中。

(2) 根据当前 Button 的状态，将对应的角色元素加入显示列表，并将 Button 的触发方式设定，并保存相应事件所需要执行的 Action 字节码。

(3) 通过鼠标的移动或按键发生产生事件，再逐一对该帧中等待事件处理的按钮进行遍历，判断事件的属性是否与其中的某个按钮一致，如果一致将该按钮的状态进行转换。

(4) Button IdelToOverUp 的状态转换将要以何种图形显示 Button，把这种图形对象加到显示列表中，并把以前显示的图形从显示列表中移除。如果 Button 没有图形显示，则要用设焦点的方法来标识该 Button，使其属性状态变为 Over。

(5) 按回车键或鼠标左键即 Press 事件触发 Button 的 OverUpToOverDown 转换，使选中 Button 的状态变为 Down。并判断是否有新的图形需要显示以表示该按钮的 Down 状态。当按键释放后，即引发 OverDownToOverUp 状态转换，此时触发 Button 事件，把该 Button 对象的 Action 字节码压入 Action 队列栈。

在执行下一帧控制列表时，就可以执行触发按钮的 Action 字节码，也就实现了按钮动作的作用。

3.5 矢量图形渲染引擎

该模块是 SWF 解码器中最核心的模块。其主要工作是根据 Shape 中的记录（Record）列表进行绘图操作，包括：移动画笔位置、绘制贝塞尔曲线、根据指定的填充样式对某封闭区域进行填充、根据指定的线条样式描线等[27]。该方案在设计过程中综合考虑软硬件环境最终选择 AGG 作为矢量库实现。

3.5.1 图形库 AGG

设计嵌入式 SWF 解码器，需要考虑到解码器在不同的硬件和软件环境下的可移植性，因此必须选择可跨平台的图形库。同时由于嵌入式平台的存储空间通常很有限，因此选择的图形库还应该尽可能的小，并具备裁剪功能[27]。经过反复比较，最后选择了 AGG 图形库，下面对其作简单介绍：

AGG 是一个开源、高效的跨平台 2D 图形库。AGG 的功能与 GDI+的功能非常

类似，但提供了比 GDI+更灵活的编程接口，其产生的图形的质量也非常高。

AGG 是一个开源的二维图形引擎。它提供一套结合了亚像素（subpixel accuracy）技术与反走样（anti-aliasing）技术的图形算法，实现高效率、高质量的二维图形处理功能。其采用 C++和标准的 C 运行时函数(Standard C Runtime Function)进行编写。这赋予了 AGG 良好的跨平台能力。

AGG 的另一个特点在于它极大的灵活性。其作者将它描述为“创建其它工具的工具”。AGG 提供一系列松耦合的算法，而且其所有类均采用模板（template）进行描述，开发者可以自由地组合、改写、替换其中部分或全部算法，以满足其具体的图形操作需求[28]。

AGG 同时也具有如下功能[29]：

(1) 支持 ALPHA、GAMMA 等变色处理，以及用户自定义的变色处理；
(2) 支持任意 2D 图形变换；
(3) 支持 SVG 和 PostScript 描述，适于网上图形生成；
(4) 支持高质量的图形处理，支持反走样插值等高级功能；
(5) 支持任意方式的渐变色处理；
(6) 支持所有颜色格式；
(7) 支持对位图的多种处理；
(8) 支持直线的多种处理，类似于 GDI+；
(9) 支持 GPC，即通用多边形裁剪方法；
(10) 支持多种字体输出，包括汉字的处理；

有两种方式使用 AGG，一种方式是直接使用它，一种方式是为它定义一个封装接口，根据项目的需求编写了一个封装类组。

考虑到方案所使用系统并没有加速硬件，本方案选用了渲染效果好的 AGG 2D 矢量库。对于渲染速度的提高就要对 AGG 进行裁减和优化来改善了。AGG 渲染性能可参考 5.2 节 AGG 渲染测试数据。

3.5.2 图形渲染流程

显示列表中的元素被指定一个深度（depth）值，深度值描述了元素被渲染的顺序。这些元素就是由一条或多条路径组成了矢量形式的几何形状。路径（Path）是由许多直线或曲线首尾相连组成的，两个顶点可以表示直线，两个顶点和一个控制点可以表示曲线（二次贝塞尔曲线）。路径确定后指定该路径的线条样式（Line Style）与填充样式（Fill Style）；再由许多这样的路径组成一个形状（Shape）。Shape 是 Flash

动画中一种非常重要的元素，将这些 Shape 转换为位图的过程就是矢量图形的渲染[27]。

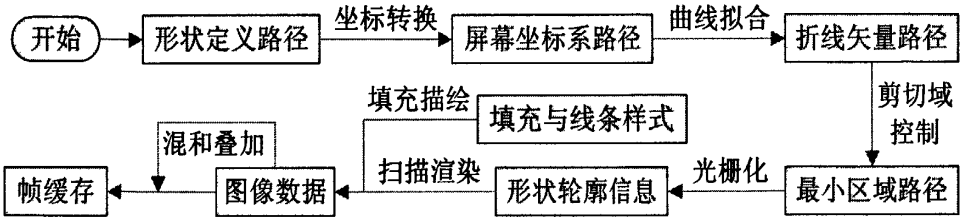

图 3-8 矢量图形渲染流程

形状的渲染是个复杂的过程，图 3-8 描述矢量图形的渲染过程。一般来说，会有以下几个步骤：

(1) 对描述路径的点使用矩阵变换，完成最终屏幕坐标的转换；

(2) 将变换后的路径中的曲线以折线近似，形成一条完整的折线矢量路径；

(3) 用边界框剪切路径，最终形成最少渲染的矢量路径；

(4) 经过光栅化后形状轮廓（Outline）信息就确定下来了，并考虑 Mask 层的蒙板作用，最终确定透明度信息；

(5) 根据形状属性设置填充样式与线条样式，分别进行填充与描绘；

(6) 通过扫描线填充算法，从填充形状的顶端开始，到底端为止，判断扫描线与路径轮廓的交点，并在其间用颜色信息对扫描线进行渲染；

(7) 最后与屏幕上已有的内容进行叠加和混和处理，结果输出到帧缓存。

3.5.3 矢量渲染的优化分析

在 Flash 的 SWF 文件制作完以后，每个形状上直线段与曲线段的数量和图形位置已经确定。通过矩阵运算则可以改变这些信息，又因为图形填充的信息是对图形的描述，当图形的形状改变时，填充的像素数可以根据图形的大小而动态地调整，所以渲染的形状有改变时画面也不会失真[30]。

在 Flash 高分辨率显示的情况下， 由于曲线拟合的时候就会产生较多的折线段（为了更好地近似曲线），在形状渲染时就会需要处理大量数据，特别是在嵌入式平台上解码 SWF 文件，就更加耗时。因此减少需要渲染的部分对降低硬件消耗、提高播放速度具有明显效果。

具统计 Flash 文件中控制标签的数量，发现其中移动对象的操作占了近 90%,而放置对象的操作只占不到 10%。并且每一帧中出现的矢量图形中，特别复杂的图形（主要指轮廓复杂）占 20%左右，而渲染它们所耗的时间超过总耗时的 70%[27]。

综合上述分析，Flash 中的图形元素通常会被反复使用，且是在前后连续的许多帧中被反复使用。帧间复用与位图缓存技术是 SWF 解码器的关键优化技术，这部分将在后面章节详细介绍。

3.6 本章小结

本章详细介绍了嵌入式 SWF 解码器总体的设计与功能实现。首先简单介绍了嵌入式系统运行平台，以及解码器开发的软硬件环境，接着介绍了解码器整体构架与工作流程；然后详细介绍了部分重要功能模块的设计与实现，主要包括标签解析模块和播放逻辑控制模块。最后重点介绍了 SWF 解码器的核心模块——矢量图形渲染引擎，这是整个嵌入式 SWF 文件解码器优化技术的切入点。

第四章 SWF 解码器优化技术

4.1 运算优化

4.1.1 浮点运算处理

在嵌入式领域中，节省成本和减少功耗是系统开发过程中需要考虑的问题。因此很多芯片都是没有浮点运算模块的，该硬件模块通常叫做 FPU（Float Process Unit）。对于拥有 FPU 的处理器芯片而言，程序的浮点运算就相对直接和高效。但是也提高了芯片的成本，增加了系统功耗。通常没有 FPU 支持的情况下，也可以执行浮点运算，运算性能自然有所损失。通常也有两种方式可以实现浮点的处理[32]。

Linux 内核有一个模块叫 math-emu 的软件模块，就是用整数运算模拟浮点数运算，一般位于 arch/mips/目录下。用工具链编译含有浮点运算的文件时，编译器并不知道目标板上没有 FPU，所以遇到浮点运算的时候还是将其编译成浮点运算指令。但是，编译生成的执行文件最终在执行到浮点运算指令的时候就有问题了。因为芯片没有 FPU，所以浮点指令对于芯片来说是属于不认识或者叫不支持的指令，那么就会产生一个异常。内核在初始化的时候为这个异常设置了异常处理函数，当内核捕捉到这个异常后进入异常处理函数执行。而这个异常处理函数就是内核浮点模拟运算模块的入口，即通过异常处理得到浮点指令，并转化为 math-emu 的输入进行运算[32]。

应用空间使用内核实现的浮点运算模块进行浮点运算，效率不是很高。这主要是基于两个方面，一个是因为需要反复的产生异常从而进行应用空间和内核空间的切换。另一个是因为，内核的浮点模拟模块效率并不高，因为异常处理时并不知道产生异常的指令到底是加减乘除指令，还是浮点比较等指令。所以，需要先用 switch 确定一下该指令到底需要什么运算然后再进行模拟，最后返回运算结果[32]。

另外一种浮点运算的解决方案就是使用软件浮点库。方法就是在编译的时候使用编译选项-msoft-float，然后在链接的时候加上软浮点库。一般完整的平台特异工具链中都会有软浮点库的支持，而软浮点库的性能也同时会针对不同平台而进行特殊指令的优化。这样加上该编译选项后，编译器在编译浮点运算时，并不将浮点运算翻译成浮点运算指令，而是将其在编译阶段就转化成整形数，然后调用相应的软浮点运算库中浮点运算指令模拟函数，这样一条浮点运算指令在同软浮点库链接完成后，就成了一堆整形数运算。这样的可执行文件能够充分的利用芯片的流水线，而且避免了应

用空间和系统空间的切换，并且运算指令的类别判定放在了编译阶段，提高了运行阶段的效率。通常情况下，用软浮点和用内核浮点模拟模块，运算效率是一个数量级的差别，性能提高相当可观[32]。

由于本方案中采用硬件并不支持 FPU，于是在程序编译的时候使用编译选项-msoft-float，使用工具链中提供的软浮点库来模拟浮点运算，达到性能与成本的折中。

4.1.2 矩阵运算优化

在 SWF 解码过程中，使用矩阵关系运算的地方很多，减少矩阵运算的时间消耗很有意义。SWF 文件中的矢量图形对象最终显示在屏幕上的整个过程涉及三个坐标系表示。如果分别用 A，B，C 表示三个坐标系，完成渲染路径的坐标转换可以由下面的公式表示。

(1) 原始文件描述的坐标都以 twips 为基本单位，20 twips = 1 pixel。

$$(Ax \quad Ay) = (x \quad y) \cdot \frac{1}{20} \qquad \text{式 (4-1)}$$

(2) 在 SWF 文件头中定义了原始大小，该坐标系为 B。矢量图形通过 PlaceObject2 标签中定义了矩阵 M 表示形状放置于该坐标的位置信息，而 SWF 文件格式说明[17]中定义了矩阵由 Scale 参数(sx, sy)、Rotate 参数(shx,shy)以及 Translate 参数(tx,ty)组成。

$$(Bx \quad By) = (Ax \quad Ay) \cdot \begin{pmatrix} Msx & Mshx \\ Mshy & Msy \end{pmatrix} + (Mtx \quad Mty) \qquad \text{式 (4-2)}$$

(3) Flash 最终显示屏幕的大小可以与定义屏幕尺寸不一致，而最终显示的屏幕坐标系为 C，用矩阵 N 来表示该坐标系的转换关系。

$$(Cx \quad Cy) = (Bx \quad By) \cdot \begin{pmatrix} Nsx & Nshx \\ Nshy & Nsy \end{pmatrix} + (Ntx \quad Nty) \qquad \text{式 (4-3)}$$

为了避免浮点数的矩阵相乘，于是在坐标转换时采用 twips 为单位，并且将矩阵各元素乘以 65536 取整得到$M'N'$，让大部分运算只涉及整数的乘法。最终的计算表达式为：

$$(Cx \quad Cy) = [((x \quad y)M'N')/20] >> 32 \qquad \text{式 (4-4)}$$

因为路径中的每个点都会参与坐标转换，所以定点化后的矩阵乘法运算效率很高。尤其是在没有浮点运算单元支持的嵌入式系统，定点化工作就显得更为必要。

4.2 帧间复用

由于 Flash 动画的帧间动作的连续性，在上一帧的基础上只需要更新帧缓存中部分区域，就可以得到新的一帧图像，也可以达到减少渲染的目的。由于 SWF 解码过程主要消耗时间的工作是由矢量图形渲染一帧画面的计算，尤其是在嵌入式平台上的 SWF 文件解码器，减少了图形的渲染也就大大地减少了硬件资源的消耗，同时可以提高 SWF 文件播放的实时性。

从前面章节说明的矢量图形渲染的过程了解到，需要渲染的形状都定义有一个能完全包围该图形对象的矩形，且该矩形的长宽是符合条件的矩形的最小值。当描述该图形形状的标签被加入显示列表中时，矩形乘以 PlaceObject 标签所带的 M 矩阵以后就得到该形状在 SWF 文件指定大小的显示屏幕上的显示位置。于是就可以知道一帧中所有图形对象的显示范围。这个矩形的显示范围会做为帧间复用的最要操作对象。

前后两帧有变动的形状由 PlaceObject2 和 RemoveObject2 来控制。当执行每帧的这两类标签时，都要将该层的显示区域与影片定义的区域相交的矩形区域加入 m_old_invalidated_ranges 的记录中，如果该层在前一帧也出现过，则该层旧的区域也要被记录下来。然后将该帧所有矩形组合计算出该层的 Clipbounds 记录在显示列表中，形成尽可能大的互不交叠的矩形区域。当显示列表中的元素按深度进行渲染之前，要遍历 clipbounds 记录中的所有矩形，并重新渲染该层在相交矩形中的部分。通常帧之间更新的部分都很小，于是通过这种帧间复用方案可以极大地提高解码效率。

具体复用算法步骤如下：

(1) 当执行当前帧的 PlaceObject ContralTag 时，则判断该层形状的显示区域是否与文件头定义的 SWF 影片区域相交。如果不相交就说明该形状图形不会被显示出来，则可以忽略该层对象，不对其解码；否则，就将该矩形区域加入该对象的 m_old_invalidated_ranges 成员中；

(2) 如果同一层中的形状在前一帧出现过，当前帧中的该形状移动了位置，这时就要将该对象的 m_old_invalidated_ranges 成员修改成前后两个相交矩形的集合一并记录在 m_old_invalidated_ranges 中；

(3) 在显示列表的执行过程中，将各层中的元素对象逐一 display 之前，有一步计算该层形状的 clipbounds，为了得到各层的 clipbounds 就需要知道该层对象的 m_old_invalidated_ranges 以及 SWF 文件影片矩形区域，将其相交部分加入 clipbounds 中；

(4) 在图形渲染部分就只会按照该层的 clipbounds 成员指定区域进行渲染，即可达到帧间复用的效果。

该算法的关键这处就在于计算矩形区域的准确性，并只分析计算出帧间变化的区

域即可。

4.3 位图缓存

4.3.1 引入缓存

缓存技术是一种经典的以空间换取时间的方法，已广泛应用于计算机领域，如代理服务器、网页浏览器等[33][34]。在嵌入式 SWF 解码器里引入缓存技术，以缓解矢量图形渲染过于耗时的问题。

通常 Flash 中的图形元素会在连续的许多帧里被反复使用，这也是在 Flash 解码过程中建立字典的原因。Flash 动画中的图形元素是一个个矢量图形,存储图形的轮廓以及填充样式等信息，在显示时再动态地用渲染引擎生成相应的位图。然而，每次渲染字典中相同矢量形状时，都会完成同样的矢量图形渲染流程，因此存在大量的重复计算。在重复渲染的形状中，很多是保持原位置或平移，只有少部分是进行缩放或旋转。对于仅产生平移的形状放置，就没有必要再渲染该图形，只需要将之前保存的位图使用硬件（简单的 2D 处理）来搬移这块数据到指定位置即可。

为了证明引入位图缓存的方案可行，对大量的 SWF 文件进行了分析，以下是统计结果[27]：

(1) 统计 SWF 文件中控制标签的数量，发现其中移动对象标签占了近 90% ，而放置新对象标签只占不到 10%。

(2) 分析动画中的每一帧，发现当一个图形元素被放置到显示区域后，几乎肯定会出现在此后连续的几帧或几十帧中，它或保持原位置，或被平移、缩放或旋转，由此形成连续的动画效果。甚至有些 SWF 文件制作有要求重复播放、跳至、回放等复杂播放动作的可能。

(3) 每一帧中出现的矢量图形中，特别复杂的图形(主要指轮廓复杂)占 20%左右，而渲染它们所花费的时间超过总耗时的 70%。

通过上述分析，Flash 中的图形元素通常会被反复使用，且是在前后连续的许多帧中被反复使用。因此在解码器中引入缓存技术是提高解码效率的重要途径。

4.3.2 缓存对象

该方案主要是针对高分辨率应用，因此较大的图形渲染占用大量时间，影响播放速度。于是缓存的对象主要是在屏幕上生成的较大图形形状。具体原因有以下几点：

(1) 对于大尺寸的矢量图形渲染，从曲线拟合的过程开始就会直接增强处理的数

据量。尤其在扫描渲染与混和叠加的过程中，还会频繁操作内存，就更加耗时。因此最好减少这类形状渲染的处理。

(2) 在屏幕上越大的形状，产生缩放旋转变化的就越小，最终保存该形状的利用率就越大。尤其某些 Flash 文件，有放置背景图片的习惯，这样就更有利可图了。

(3) 较大的形状不会很多，并且大块的内存区域使用利于缓存区域的管理。

4.3.3 缓存设计

在字典中的每个元素增加了两个数据成员，一个用来指示该形状的缓存首地址，即变量 shapebmp，另一个用来表明该地址缓存数据的位图信息，包括尺寸（长、宽），ARGB 像素格式，还有形状的 M 变换矩阵（即由 PlaceObject2 指定的矩阵）的 Scale 和 Rotate 相关参数。

在第一次放置某个元素时，它的缓存起始地址肯定是 NULL。判断该形状是否具备保存的条件（指定为形状范围大于某个阈值），然后在缓存空间找出形状大小的区域，得到首地址保存在该元素的 shapebmp 成员变量内，并将该地址设置为渲染引擎的绘图空间首地址。然后就可以渲染该形状，不过当坐标变换的时候多了一次，即将坐标原点平移到该形状的左上角。这样可以让该形状占用最少的缓存空间，最后将位图形状搬移到指定位置。再遇到需要渲染该元素时，就可以从 shapebmp 成员变量中得到该形状的缓存地址，根据该元素中保存的 Scale 和 Rotate 相关参数判断是否可以直接使用。图 4-1 描述了该缓存优化的工作流程。

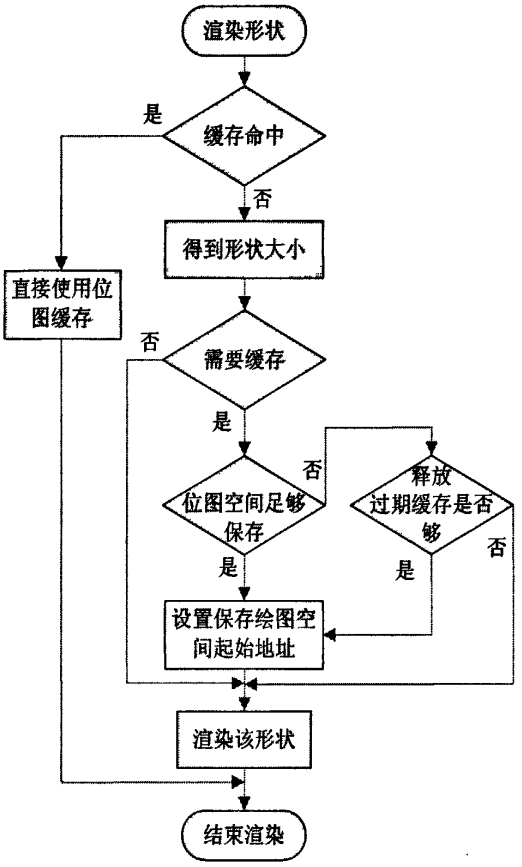

图 4-1 位图缓存的工作流程

4.3.4 缓存释放

当某形状需要缓存，但剩余缓存空间不足以保存该形状位图时，就需要释放一些过期的缓存。淘汰策略以最小化损耗为原则，以最大限度地发挥缓存的作用。在文件解析过程中可以确定每一个元素最后一次使用时的帧号，因此某些缓存元素在过期以后会将它的缓存信息加入一个过期链表 expiryList。当遇到缓存不足时，就会释放 expiryList 指定的缓存，以获得较大的空闲区域。如果 expiryList 已经没有可释放的资源，但还是不够保存某形状，这时就只能直接渲染形状。

4.4 本章小结

矢量图形渲染占用系统资源较多，由于嵌入式环境的 CPU 主频和内存都是有限的，因此该平台上实现 SWF 解码器的技术难点是图形解码和渲染过程中的运算量大，内存消耗大，所以解决问题的方向就是尽量降低硬件消耗。如程序设计时要绝对避免

浮点运算，尽量采用移位运算来代替数学运算等方法来减少运算量。但有时采取这些方法还不够，还要根据应用软件的特点，巧妙地设计数据结构来提高性能。该技术方案利用 SWF 技术的特点，充分发挥平台资源的可用性，基本满足了高分辨率要求下的解码器实现。

本章从技术要点出发提出了三种优化解决方法，并详细说明了方案的可行性：一是优化算法计算，尽量避免浮点数运算和乘法运算；二是充分地利用 Flash 动画帧间动作的连续性，充分地利用帧间复用，以最大限度的降低运算量；三是采用位图缓存技术以及有效的内存管理，大大地减少了矢量图形渲染的数量。

第五章　性能测试与分析

5.1　测试系统说明

本系统采用工作频率400MHz双MIPS内核处理器，512M DDR内存的硬件环境。下图即是系统平台实物照片。该系统支持USB2.0接口，并支持YUV分量输出，分量接电视即可输出。实际产品可以嵌入到电视机内，实现支持SWF解码的目的。

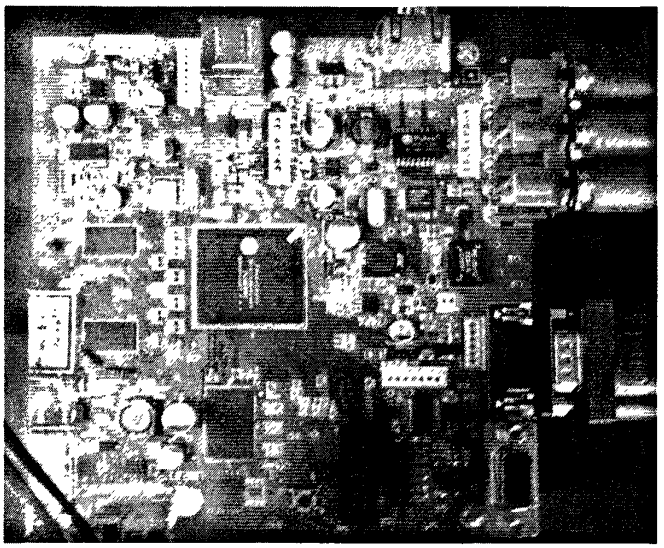

图5-1　开发系统实物图

该方案实现了SWF解码播放，并支持SWF V7功能，以及部分支持SWF V8和SWF V9的功能。以下就分别从图形渲染与性能方面进行了测试，说明该方案解码器的功能与性能效果。

5.2　图形渲染测试

前面章节也说明了该方案采用AGG作为图形渲染库。为了说明该方案的优势所在，特意独立出图形渲染过程作为测试，说明本解码器具有高质量的图形渲染。图5-2是GDI+和AGG的渲染品质的比较。

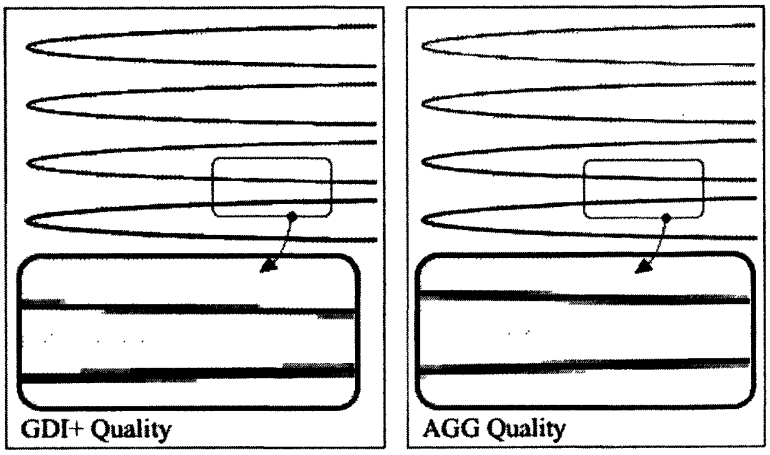

图 5-2 GDI+与 AGG 渲染品质比较[29]

AGG 采用了反锯齿（Anti-aliasing）特性。计算机图形显示是由屏幕象素点来表示的，这被称为点采样（point sampling)。而有一些象素是位于物体的边缘，则该象素内部的色彩是有一定比例的，物体边缘两边却呈现出不同的颜色，点采样科技将会使得整个象素呈现出边缘两边的某一种颜色。而这样对物体边缘的着色无论是哪一种色，由于象素间色彩的忽然跳变，都自然而然的会呈现出锯齿状。一个更好的办法就是将边缘两的颜色进行混合创造出别一种颜色来填充边缘象素，即可有效地改善边缘的锯齿效果。而这种混合的算法会有很多种，可以根据边缘象素被物体内覆盖的比例进行混合。AGG 还利用了亚象素精度的技术，图 5-3 中显示了使用简单的 Bresenham 差值放大后的结果[29]。

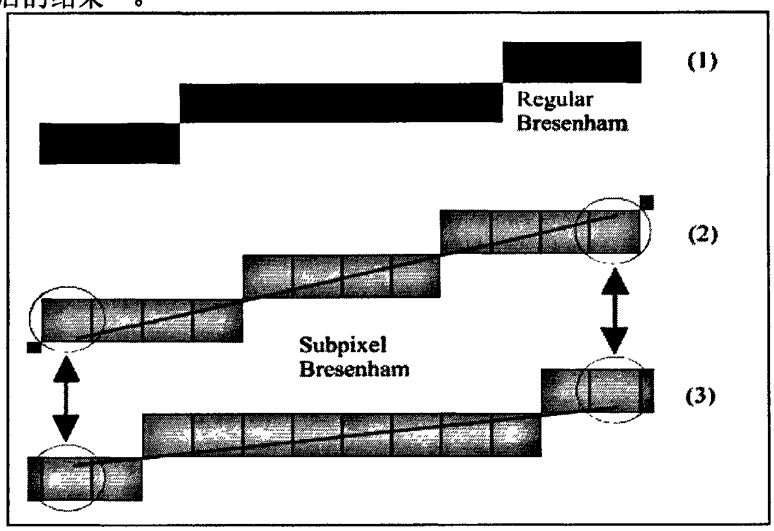

图 5-3 亚象素精度技术[29]

如图中（2）和（3），由于利用了亚像素精度，虽然线的起始点和结束点是同样的，但是可以表示两种不同的线。而如果不利用亚像素精度，同样的起始点和结束点

则只能表示一种线，如（1）。如果同时利用亚像素精度和反锯齿，就能够获得更好的效果。这也就是 AGG 渲染图形的优势所在。

图 5-4 是对 Tiger_project 的虎头形状进行渲染测试的效果图。

图 5-4　AGG 渲染效果测试

5.3　功能实测

对于该解码器方案的功能测试，则采用大量的 SWF 文件作为测试样例，首先测验解码器是否支持版本号为 7 的 SWF 格式的所有标签（TAG），这是 SWF 解码器主要功能的一个体现。其次要测验解码器对其他 SWF 版本的标签支持情况，以及在遇到暂不支持的标签时能否继续工作，目的是检测 SWF 解码器的兼容性。

测试结果表明，该解码器能很好地支持 SWF v7 的绝大部分标签，同时也支持部分的 SWF v8 和 SWF v9 的特殊功能标签。在测试过程中，虽然也遇到对一些高级的标签支持不是很好的状况，有的甚至会导致程序崩溃，但这也为之后的调试和稳定测试提供了指导。由于支持的标签数目很多，这里就不一一列举了。此外，该 SWF 解码器方案也同时支持 FLV, H263, MP3 等流格式数据解析。

由于该解码器主要是基于高清数字电视或机顶盒上的应用，操作方式就跟一般的 PC 机不同（电视不用鼠标，而用遥控器进行交互操作），所以对于遥控器的模拟鼠标的测试也要确保操作的正确性。方案设计时采用遥控器的“UP”键作为前一个交互操作的按键或位置处，使用“DOWN”键作为后一个交互位置，即模拟鼠标的移动。

同时用遥控器上“确定键”来模拟鼠标的左键，完成确认信息的输入。由于平台的开发中已经提供了基于遥控器输入法设计，所以可以满足特殊 Flash 动画输入文字的需求。通过测试表明，这种方式模拟鼠标可以达到大部分 Flash 操作的需求。图 5-5 为播放 Flash 画面的实物照片。

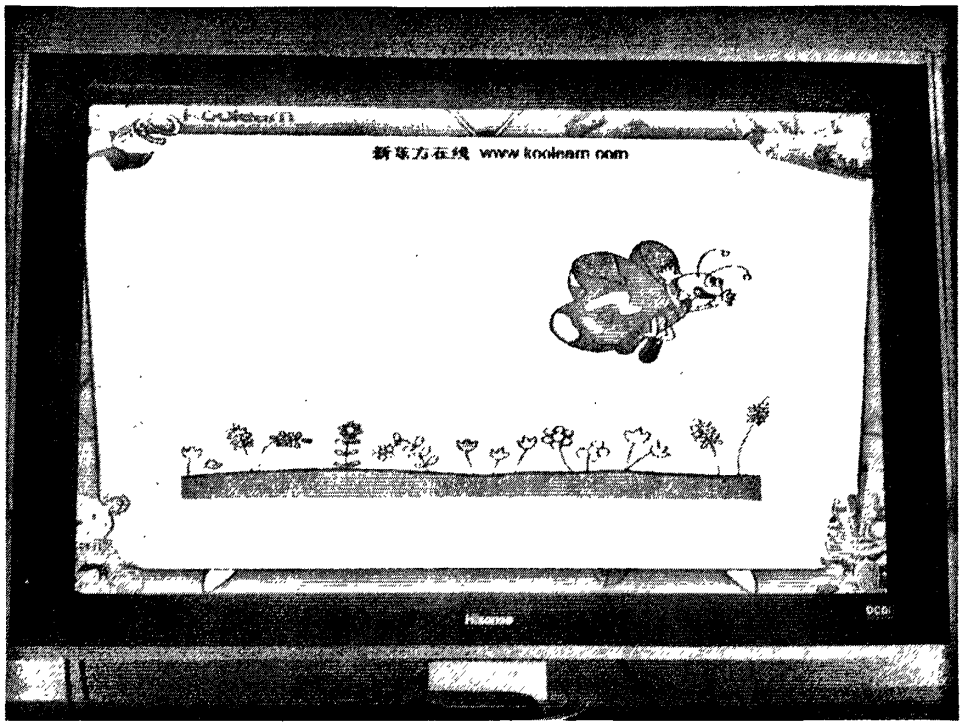

图 5-5 开发系统播放效果实物照片

5.4 性能实测

由于 Flash 也是实时性的流播放媒体，性能上可以从播放的帧率来衡量，即每秒种多少帧画面显示。前面章节也较详细地分析了 SWF 解码的过程，从中可以了解到：Flash 播放速率的主要受到矢量图形渲染快慢的制约。尤其是在嵌入式平台上，受到很多性能限制，常规的方法是无法达到 Flash 的流畅播放的。因此就必须在解码过程中对各过程的细节进行优化调整，并充分发挥硬件平台的资源，利用 SWF 解码的特点，采用位图缓存、帧间复用等技术，来减少图形的渲染，提高每帧画面显示的速度。这些技术细节已在第四章中各节详细说明。

下面是采用这些优化方法前后的性能比较。其中包括：Total 性能、Parser & Flow 性能、Shape Render 性能。分别表示总体测试结果、解析和流程控制部分性能、以及形状渲染部分性能。测试过程中采用 1080p 的分辨率完成测试结果。

表 5-1 是未作优化的原始 Flash 播放性能：

表 5-1 原始软件测试结果

File Samples	Total	Parser & Flow	Shape Render	
	帧率（FPS）	帧率（FPS）	Time(50F)	Percent
Hats.swf	1.612	1.972	5.66	18.24629
Buttfly.swf	0.121	0.279	232.37	56.45255
Libai.swf	0.119	0.421	300.15	71.6263

表 5-2 是经过设计优化后的测试结果：

表 5-2 解码优化测试结果

File Samples	Total	Parser & Flow	Shape Render	
	帧率（FPS）	帧率（FPS）	Time(50F)	Percent
Hats.swf	9.5	10.785	0.626	11.89662
Buttfly.swf	5.425	7.465	2.52	27.33189
Libai.swf	2.847	5.967	9.197	52.33739

测试中选用了三个具有代表性的 SWF 文件：Hats.swf 文件是相对比较简单，层数限制在 10 层以下，并且各形状的路径边数 Paths 小于 50；Buttfly.swf 文件表现层数在 30 层以下，并且形状平均路径边数 Paths 小于 300；Libai.swf 文件的层数在 50 层以上。

从以上测试结果数据表明：SWF 解码过程经过优化处理后，简单的 Flash 动画就可以达到正常要求，并且耗时主要限制在解析标签与流程控制部分，显然简单 Flash 的形状渲染部分已经不是主要的消耗。然而，由于受到了硬件性能的限制，该方案对于形状复杂或本身层数较多的 Flash 文件的处理，尚未达到理想状态，还需要进一步改进。

5.5 本章小结

本章在实现 SWF 解码器系统的基础上，对系统性能进行了测试。首先介绍了系统的测试环境，并对解码器的矢量图形渲染引擎所使用的 AGG 图形库进行测试，表明了该方案在图形显示上的品质级别。之后通过使用大量的 SWF 文件对解码器进行测试，获得解码器功能及性能上的测试数据结果。论文主要提出了嵌入式平台上的 SWF 解码器的关键优化技术，本章也从测试数据的角度分析说明了这些关键的优化技术的使用对于嵌入式平台上开发 SWF 解码器是至关重要的，最后分析指出了该

SWF 解码器目前还存在的不足及其原因。

第六章 结束语

随着互联网的飞速发展，Flash 技术在信息领域的应用越来越广，嵌入式 Flash 播放器作为一种重要的产品应用也越来越受到社会和企业的重视。本论文提出了一种可行的嵌入式 SWF 解码器，通过采用各种优化方法，使解码器具有了较好的性能指标。但限于作者自身能力和时间的关系，本课题的研究内容还存在一些不足之处，在一些方向上需要开展进一步的研究和改进工作，这里提出为随后的开发与研究工作指明方向：

(1) 继续完善支持较新的 SWF 标签，支持较高版本的 SWF 解析；

(2) 采用 OpenVG 的矢量渲染方式，并逐步采用硬件加速来提高矢量图形的渲染速度；

(3) 细分解码部分，并合理采用多核协同工作完成 SWF 解码，优化处理速度。以下为多核优化方法的总体思路设计：

该方法是结合解码器使用的硬件环境考虑的。解码器的硬件平台是 PowerLayer 公司的SOC芯片，该芯片内部具有两颗MIPS24k核，一个DSP核，2D Graphics Engine，以及图像增强等功能单元。其硬件资源十分丰富，对于 Flash 应用主要的优势在于，多核协同支持，简单的 2D 图形加速，以及使用高性能图像增强技术来弥补在 Scale 低分辨率 Flash 过程中的画面损失。

该平台中 CPU0 主要运行着嵌入式 Linux 操作系统，CPU1 主要完成 DSP 解码器的控制，因此考虑并行处理以提高效率。将最耗时的矢量图形渲染部分独立由 CPU1 来完成，并且利用 CPU1 对 DSP 的控制，在 DSP 上实现高效渲染算法。该方案采用了 Ringbuffer 的共享内存方式进行数据通信。图 6-1 表示两个 CPU 间的并行处理流程。

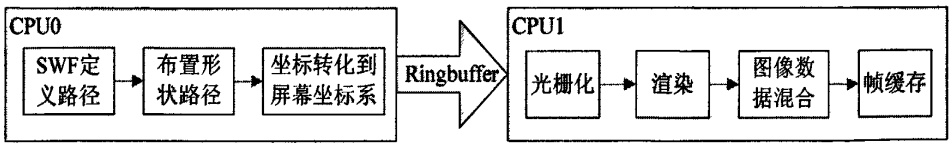

图 6-1 并行处理数据处理流程

协同处理的关键在于存储在 Ringbuffer 中的数据形式，考虑到渲染过程的复杂，因此 传到 Ringbuffer 中的数据是经过坐标变换后的路径信息。CPU1 在将显示数据输出显示后会设置一些控制参数以实现音视频的同步和事件的同步。

由于时间和条件所限，以上部分尚未完全实现，今后需要做进一步的研究。

参考文献

[1] 张卫，赵志峰.高清互动电视业务发展初探.中国有线电视.2008 年 11 期.
[2] 方向忠.高清数字电视发展的相关问题探讨.广播电视技术.2004 年 4 期.
[3] 鞠志明.浅谈数字机顶盒.中国有线电视.2007 年 11 期.
[4] 闫书元.数字高清技术与高清机顶盒设计浅谈.2005 国际有线电视技术研讨会论文集.2005.
[5] U Reimers. Digital Video Broadcasting. Communication Magazine, 1998.
[6] 刘菲，于文超.Flash 动画的标签信息提取.电脑知识与技术.2009 年 16 期.
[7] 刘丹.浅谈 Flash 动画.科技信息.2007 年 23 期.
[8] 万明民，顾景文.网络矢量图形 SWF 与 SVG 的比较分析.哈尔滨工业大学学报.2003 年 08 期.
[9] 刘海疆，周培祥，孙炜.网络多媒体应用技术.清华大学出版社.2006: 113.
[10]李秀.计算机文化基础.清华大学出版社.2005: 214.
[11]关雪梅.嵌入式技术应用研究.光盘技术.2009 年 10 期.
[12]朱田.嵌入式 SWF 解码器实现的若干关键技术[学位论文].华中科技大学.2007.
[13]赵迎军.点位图与矢量图.数码印刷.2003 年 03 期.
[14]滕召荣，蒋天发.邻域平均法对矢量图平滑处理.现代电子技术.2009 年 14 期.
[15]李春卉.矢量图形标准与规范研究现状.情报科学.2007 年 04 期.
[16]贺雪晨.多媒体实用教程.清华大学出版社.2005: 180
[17]Adobe Corporation. Flash File Format Specification [DB/OL]. http://www.adobe.com.
[18]Jacqueline Emigh. New Flash player rises in the Web-video market. Computer. 2006.
[19]唐凤仙，吴祥业.基于 swf 格式的积件资源库的设计与实现.河池学院学报.2005 年 05 期.
[20]沈静，何必仕，周丽.移动 Flash 播放器的设计与实现.计算机系统应用.2009 年 02 期.
[21]Meng Xiang-zeng, Liu Lei. On Retrieval of Flash Animations based on Visual Features, Lecture Notes of Computer Science[J]. 2007.
[22]张连明，霍迎辉.嵌入式系统的设计与开发.现代电子技术.2003 年 18 期.
[23]A Sangiovanni-Vincentelli, G.Martin. Platform-based design and software design methodology for embedded systems. Design & Test of Computers, 2001.
[24]赵苍明，穆煜.嵌入式 Linux 应用开发教程.人民邮电出版社.2009: 25.
[25]孙琼.嵌入式 Linux 应用程序开发详解.人民邮电出版社.2006: 121-126.
[26]田华健.一种嵌入式 SWF 解码器的设计与实现[学位论文].武汉科技大学.2008.
[27]王春.嵌入式 Flash 播放器研究与实现[学位论文].电子科技大学.2008.
[28]鲁力,李欣.基于 AGG 算法库的通用图形接口设计.微计算机信息.2009 年 06 期.
[29]Maxim Shemanarev. AGG Reference Manual. [DB/OL]. http://www.antigrain.com

[30]Lee SY, Choi BU. Vector Graphic Reference Implementation for Embedded System. Technologies for E-learning and Digital Entertainment. 2000.
[31]石学林，张兆庆，武成岗.定浮点数据算术及其优化.计算机科学.2005 年 06 期.
[32]MIPS 架构下 Linux 软浮点研究. [DB/OL].http://blog.163.com/liyun1982-1982/
[33]胡贯荣，阳富民.嵌入式浏览器缓存策略设计与实现.计算机工程与设计.2005 年 26 期.
[34]王文林，廖建新，朱晓民.XML 语音平台缓存技术综述.通信学报.2007 年 28 期.

致 谢

本论文的完成，是我硕士的学业与研究工作的一个总结，在这里，谨向所有帮助、关心过我的老师、同学、以及亲友们致以诚挚的感谢。

衷心地感谢丁玉珍老师在我就读研究生期间对我的教导和培养，感谢她悉心指导和深切关怀。丁老师严谨的治学态度、耐心的教育方式、平易近人的作风都深深地影响着我。

诚挚地感谢门爱东老师和杨波老师在我的学业中给予的无私帮助和指导。

感谢PowerLayer公司的连达、王庆工程师的支持和帮助，在实习期间给予我很多指导。

感谢我的同窗韩霜、王妍、卫飞宇、文闻、姚远、蒋飞和刘雨等硕士，大家相互鼓励、相互支持，在课题研究之间充分交流，让我充分体会到一个团体的协作精神。同时感谢宿舍的姐妹们，和你们一起学习、生活使我的研究生生涯很丰富、很充实，这里祝你们学业有成。

深深地感谢我的父母，他们对我的支持和关怀使得我以顺利地进行我的学业。

最后，再一次衷心感谢所有帮助过我的老师、同学、朋友和家人。

作者攻读学位期间参加的科研项目与发表的学术论文

发表的学术论文：

[1] 陈颖.嵌入式SWF文件解码器的高分辨率应用解决方案.电视技术. 2010年04月. 第一作者