申请同济大学工学硕士学位论文

基于嵌入式操作系统的 OSD 开发模式研究

培养单位：电子与信息工程学院

一级学科：计算机科学与技术

二级学科：计算机应用

研 究 生：谢鑫君

指导教师：顾伟楠　教授

二〇〇五年二月

A dissertation submitted to
Tongji University in conformity with the requirements for
the degree of Master of Philosophy

School/Department:
Discipline:
Major:
Candidate: Xin-Jun Xie
Supervisor: Prof. Wei-Nan Gu

February, 2005

基于嵌入式操作系统的OSD开发模式研究

谢鑫君　同济大学

学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：

年　　月　　日

经指导教师同意，本学位论文属于保密，在　　年解密后适用本授权书。

指导教师签名：　　学位论文作者签名：

年　　月　　日　　年　　月　　日

同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

签名：

年　　月　　日

摘要

在屏幕控制系统（OSD）是贯穿全部数字电视开发及应用的主线，所有的数字电视控制及其增值业务，都是通过 OSD 作为窗口展现在用户面前的。所以，OSD 开发是数字电视软件开发的关键。

本文就是结合 863 课题“网络化嵌入式支撑技术”的研究成果－－“和欣”嵌入式操作系统，研究基于“和欣”嵌入式操作系统的 OSD 开发模式。“和欣”技术体系所包括的 ezCOM 构件技术、构件运行平台技术开发应用软件所需的集成开发环境，是一个完整的面向构件的应用软件开发平台。本文在对比基于特定芯片 sda55xx 的传统 OSD 开发方式前提下，研究了在“和欣”嵌入式操作系统上，开发基于 ezCOM 的 OSD 中间件，最终形成 OSD 应用程序<－>OSD 中间件<－>“和欣”构件运行平台<－>“和欣”操作系统<－>硬件平台 这一系统架构下的 OSD 开发模式。其中关键的 OSD 中间件部分，完全基于 EZCOM 技术开发，大大提升了 OSD 软件开发技术水平，提高了 OSD 软件生产效率和软件产品质量，将软件工厂的概念引入到了 OSD 软件的开发中来。

“和欣”为数字电视软件的开发提供了强有力的支撑，文章最后，展望了将基于“和欣”嵌入式操作系统的 OSD 开发模式推广到全部数字电视开发中去的美好前景。

关键词：OSD；构件技术；中间件

ABSTRACT

Key Words：OSD；

目录

第 1 章 引言 1

1.1 数字电视的发展状况 1
1.2 数字电视的软件需求 2
1.3 数字电视OSD概述 3
1.4 数字电视及OSD开发的技术架构 3
1.5 本人所作的工作 4

第 2 章 数字电视OSD开发 5

2.1 当前OSD开发模式 5
2.1.1 数字电视OSD开发概述 5
2.1.2 基于SDA55XX的OSD开发 6
2.1.2.1 SDA55xx介绍 6
2.1.2.2 基于SDA55XX的OSD开发系统结构设计 7
2.1.2.3 基于SDA55XX的OSD开发具体实现 8
2.2 当前OSD开发模式问题研究 14
2.2.1 硬件依赖性强，软件移植困难 14
2.2.2 开发困难，周期长 14
2.2.3 缺乏统一的平台 15

第 3 章 基于嵌入式操作系统的OSD开发关键技术 16

3.1 “和欣”嵌入式操作系统 16
3.1.1 数字电视操作系统 16
3.1.2 “和欣”操作系统简介 16
3.1.3 “和欣”灵活内核 17
3.1.4 “和欣”操作系统提供的功能 18

3.1.5 “和欣”操作系统的应用软件开发..... 19
3.1.6 “和欣”操作系统的优势..... 20
3.2 “和欣”构件运行平台.....22
3.2.1 “和欣”构件运行平台简介..... 22
3.2.2 “和欣”构件运行平台的功能..... 22
3.2.3 “和欣”构件运行平台的技术优势..... 24
3.2.4 利用“和欣”构件运行平台编程..... 24
3.3 EZCOM构件技术.....25
3.3.1 EZCOM技术的由来 25
3.3.2 EZCOM构件技术概要..... 27
3.3.3 EZCOM技术的意义 28
3.3.4 EZCOM技术对软件工程的作用..... 28
3.3.5 EZCOM技术在“和欣”技术体系中的作用..... 30
3.3.6 如何用EZCOM技术编程..... 30
第 4 章 基于嵌入式操作系统的OSD开发模式实现方法研究 32
4.1 基于“和欣”嵌入式操作系统的OSD开发系统结构设计.....32
4.1.1 系统结构 32
4.1.2 数字电视开发中间件技术..... 33
4.2 基于“和欣”嵌入式操作系统的OSD开发具体实现.....33
4.2.1 构件描述语言CDL(Component Definition Language)..... 33
4.2.2 OSD中间件介绍 34
4.2.3 OSD 底层构件库 36
4.2.3.1 寄存器构件 36
4.2.3.2 XRAM构件 40
4.2.3.3 DRCS构件 41
4.2.4 OSD 控件库 42
4.2.4.1 OSDBITMAP控件..... 42
4.2.4.2 OSDTEXT控件..... 44

4.2.4.3 OSDSLIDER控件..... 46
4.2.5 OSD中间件开发过程..... 47
4.2.5.1 编写CDL文件生成构件源程序框架并填写实现代码..... 47
4.2.5.2 编译生成构件以及构件的注册..... 52
4.2.5.3 ezCOM构件的使用方法..... 52
4.3 优势分析.....54
4.4 小结.....55
第 5 章 结论与展望 56
5.1 研究结果.....56
5.2 研究展望.....56
致 谢 58
参考文献 59
个人简历　在读期间发表的学术论文与研究成果 60

第 1 章 引言

1.1 数字电视的发展状况

数字电视是继黑白模拟电视，彩色模拟电视之后的第三代电视。 影视数字化从根本上改变了影视的命运：数字化影视创造娱乐设施的新时代。影视节目的制作和播放，由于数字化方式的加入，而变得更加多元化，随机化，全球化和可追求化。

现有的彩色电视包括以下几种不同的制式——欧洲和我国采取的是 PAL 制式，美国和日本采取 NTSC 制式，前苏联采取 SECAM 制式，但利用人的视觉暂留原理顺序扫描、同步扫描的原理却是一样的，因而面临的问题也是相同的，即由于扫描行数的限制而造成的清晰度不够理想。

为了提高电视图像的分辨率，从 70 年代开始，工业发达国家开始了对高清晰度电视系统的研究工作。这个工作最早是从日本的 NHK 开始的，到了 80 年代初获得成效，制作了 1125 线的数字电视机，60 场/秒，2∶1 隔行扫描标准的高清晰度电视，简称 HDTV。到了 90 年代，形成了日本、欧洲、美国三大数字电视制式。日本和欧洲的两种制式出现比较早，图像压缩比较小。采用模拟信号传送，卫星播出方式适合较宽的信道传输；美国的全数字方案吸收了日本和欧洲的优点，采用数字压缩编码和数字通信技术，传送效率高，有效地压缩了宽带，适合于窄频道传输的地面广播，并且对使用相同频道的其它节目不产生干扰，实现了与先行模拟信号电视兼容过渡的根本目的。

数字电视的诞生是为了满足人们的视觉和控制的需要。数字电视的信号不再是模拟信号，而是采用以数字形式进行传输、处理、存储的数字信号。由于数字电视信号的存储和处理电路便于大规模和超大规模的集成，因而其设备比模拟电路设备元件少，便于调整，其重量轻、体积小、功耗少，寿命长且可靠性高，容易与计算机以及其它数字化设备接口，适合于公用数据通讯网，便于实现生产、运行的自动化和视听信息处理的综合化、网络化。计算机技术的发展和介入，使得正处于方兴未艾的电视工业得到了新的支持，带来了又一次巨大变革的历史机遇。

世界各国发展数字电视的情况:美国国家电视网要在 2006 年普及数字电视，全面停止模拟信号。 英国全国由模拟电视向数字电视过渡的时间从 2006 年开始，预计 2010 年结束；日本 2001 年开播 6 套卫星高清晰度数字电视，但地面高清晰度电视要在 2003 年才会在主要的大城市开播。

我国计划在 2005 年将进行数字电视的商业播出，2008 年用数字电视转播奥运会，2015 年停止模拟电视的播放，全面推行数字电视。

数字电视为我们带来的巨大商机：我国的电视机用户在 3 亿 2 千万台左右，如果每年以 10%的数字电视机替代模拟电视机，就会有上百亿美元的市场规模。按 10 年的过渡期计算，每年配 3000 万台电视，年营销金额为 600 亿人民币，仅我国数字电视市场规模就在 1 万亿元左右。

1.2 数字电视的软件需求

数字电视将会形成一个比模拟电视更为庞大的产业网络。这个网络主要是由技术服务商、电视运营商、内容制作商、电视厂家和广大数字电视观众组成。由于数字电视已不再是传统意义上的电视机，而相当于一台拥有 CPU 的电脑，由于数字电视接受的是二进位的数字代码，因此电视节目内容制作商有了更大的空间可以动态地控制这些代码，从而达到制作动态节目的目标，这将是电视史上的一次里程碑。动态的电视节目将为观众带来更为广泛的服务。比如当你躺在自家的沙发上看电视剧的时候，或许你将被电视剧里出现的一副美丽的场景所打动，这时你只要轻按暂停键，这时画面就会停下，同时旅行社提供的去该场景旅游的服务就会出现在你的眼前…。类似的动态服务将比比皆是，动态服务将为我们带来除基本音视频业务之外的数字电视增值业务。由于电视节目的多样化，为内容制作商提供技术支持的技术服务商就应运而生，技术服务商的作用就相当于应用软件开发商，他们利用数字电视的所提供的强大功能开发处大量的数字电视软件，为内容制作商提供强大的支持。而电视运营商将会从内容制作商那里采购所需的电视节目，编排以后提供给电视观众。电视厂家为电视观众提供数字电视，以便观众们能享用丰富的电视节目。

数字电视服务的最大特点是，它除了支持传统的音频、视频业务以外，还能带来电视增值业务，其中包括：视频点播、数据广播、个性化交互电视、远程教育、Internet、三网合一、电视电子商务和日常信息综合服务等服务。而

这一切电视增值业务，都是建立在强大的软件开发能力之上的。而在所有的数字电视要素中，OSD(On Screen Display 在屏控制系统)是贯穿全部数字电视应用和增值业务的主线，所有的增值业务，所有的数字电视设置，都是通过 OSD 作为窗口展现在用户面前的。所以，OSD 开发又是数字电视软件开发的重中之重。

数字电视软件开发的趋势都是将软件建立在中间件的基础上的，中间件是介于数字电视操作系统和数字电视应用软件之间的接口，通过中间件的平滑嫁接，同一应用软件可以在不同的操作系统上使用。国外市场现行的三种数字电视标准都有自己的中间件，美国标准采用"DASE" 中间件，日本标准采用"ISDB-BML" 中间件，欧洲标准采用"DVB-MHP" 中间件。我国将要采用的中国数字电视标准，是一套完全具有自我知识产权的标准。

1.3 数字电视 OSD 概述

早期的屏幕要调整设定，必须使用屏幕下方的旋转钮，可调整的功能较少。但是随着技术的进步，厂商发展出一项新的OSD(On Screen Display)技术，可在屏幕上显示屏幕的相关设定，操作的功能也比较多，现在的 OSD 逐渐淘汰了旋钮调整，而改为按键式，有效避免接触不良、调节精度等状况发生，现在OSD已经成为最普遍的电视及监视器人性化接口。OSD 在屏幕上会以格数或数字显现出设定值以供调整，使用者可利用这样的接口来调整屏幕显示的设定值 (如：亮度、对比、失真...等)，并将调整好的设定值储存下来。现今 OSD 整合的功能越来越多，除了基本的亮度、对比、色彩、移位之外，还包括桶型失真、梯形失真、平行四边形失真等现象的调整。通过OSD的友好界面，用户可以方便并且精确地根据需要使得电视机的各项工作指标达到最佳的使用状态。

1.4 数字电视及 OSD 开发的技术架构

数字电视的硬件部分除了包含传统的显示器和喇叭等装置以外,还包含信道解调器、MPEG2 解码器和 CPU 等。这些硬件加上控制及协调他们共同工作的嵌入式操作系统，就构成了数字电视的基础技术架构。然后通过操作系统所支持的中间件，就可以在数字电视的硬件平台上，增添许多软件功能。例如浏览器，它可以帮助人们通过数字电视上网和观看电视节目。还有一些可以提供增值服

务的应用软件，这些应用软件可以为我们提供视频点播、数据广播、个性化交互电视、远程教育等增值服务。数字电视的硬件和操作系统以及中间件和众多的应用软件将共同组成，我们未来的数字电视。

数字电视的技术架构图如图 1.1 所示：

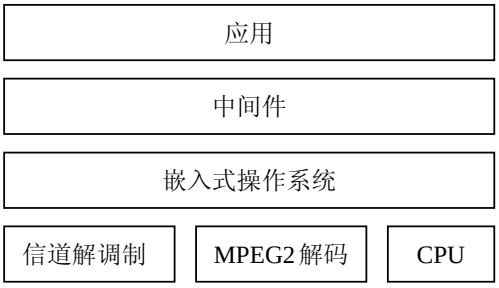

图1.1：数字电视的技术架构图

OSD 开发作为数字电视软件开发最重要的组成部分，其发展的趋势也必定是采用这样的技术架构进行开发。

1.5 本人所作的工作

本文所做的工作，就是结合 863 课题“网络化嵌入式支撑技术”的研究成果－－“和欣”嵌入式操作系统，研究构件技术在嵌入式系统软件开发中的应用，主要工作是在参照开发基于 SDA55XX 芯片的 OSD 系统的经验基础上，采用 EZCOM 技术定义实现 OSD 中间件的结构和主要接口，研究形成了 OSD 应用程序<－>OSD 中间件<－>“和欣”构件运行平台<－>“和欣”操作系统<－>硬件平台 这一系统架构下的 OSD 开发模式。

第 2 章 数字电视 OSD 开发

2.1 当前 OSD 开发模式

2.1.1 数字电视 OSD 开发概述

目前 ,数字电视技术已经基本成熟 ,在美国和欧洲一些发达国家 ,数字电视已经开始普及 ,中国的数字电视发展也已有了明确的时间表 ,相关标准的制订已经或接近完成,随着这一时间表，数字电视软件开发也成为了大家逐渐关注的焦点。而 OSD 菜单因为直接面向用户 ,其质量直接关系数字电视产品的市场前景 ,地位十分重要。在竞争日趋激烈的数字电视的市场中 ,操作简便又美观大方的用户界面是厂家致胜的关键。目前主要的数字电视开发 ,通常是在芯片厂家提供的开发平台上进行的，数字电视的 OSD 部分很多都是以其芯片的 OSG（On Screen Graphic）功能为基础实现的 ，国内的很多开发商都是选择在芯片 OSG 图形驱动上自行开发 OSD 菜单系统。

目前的各类数字电视,基本都是用户命令驱动的 ,其控制软件的核心任务是：对用户通过遥控（或按钮)发出的命令进行分析 ,利用各功能模块的驱动程序 ,对各功能模块实施控制 ,这一切都是通过 OSD 来实现的。所以 ,OSD 设计在数字电视的软件设计中居于核心地位。在数字电视的整个系统中,用户在 OSD 模块的指引下 ,发送控制命令（遥控或键盘信息),指导控制软件从某个ＴＳ流中选择特定的一路节目数据流 ,并进行解码播放。这就涉及到对解调、MPEG2 解复用、视音频解码、键盘或遥控等的控制。

OSD 菜单作为方便的控制手段 ,在数码显示器、大屏幕电视、数字电视等多个领域都得到了广泛的应用 ,虽然基于的 OSG 图形硬件功能差别不大 ,但由于应用的环境差别 ,软件实现的 OSD 菜单系统有很大不同。

下面分别就数码显示器、大屏幕电视 OSD 的特点进行介绍。

显示器调节 OSD 菜单的特点 :主要作用对象是少量模拟量和部分模式选择 ,只用到方向键和确认键就可以实现控制 ,由于控制的对象比较少 ,通常显示区域也很小 ,一般是固定在屏幕中央一小块 OSD 区域 ,并且大多是一层菜单结构。

利用到的 OSD 对象主要是按钮和静态文本（数字量调节显示 ），单选框等。复杂度比较低。

大屏幕彩电 OSD 菜单的特点 :与显示器比较 ，大屏幕彩电的控制对象要多一些 ，除了需要进行一般模拟量控制和部分模式选择外 ，还需要有诸如进度控制等功能 ，一个典型的例子就是自动搜索节目时的进度条显示。由于控制的内容较多 ，OSD 菜单区域比较大 ，有些使用多层菜单结构。使用的 OSD 对象也比较丰富 ，一般都要用到按钮、静态文本、进度条等。很多公司的 OSD 菜单部分还增加了一些小游戏（贪吃蛇、万年历等 ），丰富了 OSD 显示内容。总体来讲 ，大屏幕彩电的 OSD 菜单要比显示器的 OSD 复杂很多。

具体来说，由于数字电视的应用环境的特殊性 ，其所要实现的控制目标多 ，且实现要求更加灵活，所以其 OSD 菜单系统通常要实现按钮、列表框、输入框、进度条、静态文本、状态条等图形控件。使得用户能够通过 OSD 菜单进行设置接收参数、选择特定节目、调节声音图像等功能。

另外传统的 OSD 菜单占据整个屏幕 ，进行 OSD 菜单操作时无法同时观看播放视频节目。针对电视的应用特点，OSD 菜单需要和当前视频节目结合起来，而现在的电视芯片通常都能支持视频缩放和 OSD 层的 ALPHA 混和功能 ，这就为实现 OSD 菜单和缩放视频同时显示的菜单系统提供了可能。

2.1.2 基于 SDA55XX 的 OSD 开发

2.1.2.1 SDA55xx 介绍

SDA55xx微控制器是micronas公司出品的一款主要用于电视机控制以及提供图形图像显示的电视芯片。同时，SDA55XX不进能够提供解码全球图文电视系统WST (World System Teletext)功能，同时支持解码其他诸如视频演放系统VPS（Video Programming System），节目传送控制PDC （Program Delivery Control ）以及用于PAL-plus(欧洲新电视播出制式)播放的宽屏幕信令WSS（Wide Screen Signalling）。SDA55xx的数据分割器以及显示部分支持非常广泛电视制式，包括PAL,NTSC,以及上面提到的VPS,WSS,PDC,TTX和隐蔽字幕数据（closed caption data)。

- **SDA55xx 芯片一般特性**

- 拥有电视相关特性以及高级 OSD 显示的 8051 兼容微控制器

- 通过特殊功能寄存器(Special Function Register, SFR)调整特性
- 支持同时处理 TTX, VPS, PDC 和 WSS (23 行) 数据

- **SDA55xx 芯片显示特性**

- 芯片自带所有西方字符
- 支持图形字符集
- 支持平行显示
- 支持字符缩放，加粗等特性
- 支持字符闪烁频率可调节
- 支持屏幕大小可调节（25 行×33－>64 列）
- 支持字符点阵大小可调节(高 x 宽) 10 x 9 … 16
- 普通模式时支持 256 个自定义字符，增强模式支持 1024 个自定义字符
- 颜色表支持 4096 种颜色组合
- 每个 DRCS（动态可重定义字符集，Dynamically Redefinable Character Set）支持最高 16 种不同颜色
- 1 位色的 DRCS 以及 ROM 自带的字符支持 8 种可选的前景和背景色
- 支持阴影和对比度缩减效果

2.1.2.2 基于 SDA55XX 的 OSD 开发系统结构设计

基于 SDA55XX 的 OSD 开发，由于开发资料只有 micronas 公司提供的关于 SDA55XX 的详细的 DATASHEET 以及其提供的开发环境，包括一套 WINDOWS 下的集成开发环境（WINIDEA）软件，一套模拟器设备，所有的开发，都要从零开始。要开发 OSD 系统，就必须首先熟悉其 DATASHEET，掌握其内部特殊寄存器的功能，内部 XRAM 的分配，DRCS 的结构以及构造方法，这些都是着手开发的前提。根据这些情况，将基于 SDA55XX 的 OSD 开发主要按照以下流程：首先设计 OSD 界面，分析其中包括的主要元素，然后将每个元素的绘制，封装成一个个模块。以上是设计阶段，设计阶段的主要任务是：明确风格 OSD 界面中的元素，定义每个模块的结构等等。其后就到了具体实现阶段，这一阶段，需要实现每个模块的绘制功能，主要工作在：DRCS 的构造，XRAM 的分配以及特殊寄存器的复杂配置。模块绘制功能实现之后，由主程序调用绘制模块完成 OSD 菜单系统。图 2.1 是基本的系统结构。

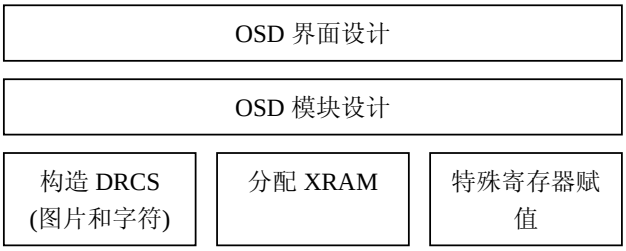

图2.1 基于SDA55XX的OSD开发系统结构图

2.1.2.3 基于 SDA55XX 的 OSD 开发具体实现

1. OSD 界面设计:

OSD 菜单直接面向用户 ,良好的 OSD 菜单在给用户一个清新的感觉的同时，可以让用户轻松地控制自己数字电视。OSD 的质量直接关系到用户对数字电视产品的印象，关系产品的市场前景 ,地位十分重要。在竞争日趋激烈的数字电视的市场中 ,开发出操作简便又美观大方的用户界面是数字电视开发的一个关键任务。

在基于 SDA55XX 的 OSD 开发中，采用了 3 层 OSD 菜单设计，其主要是迎合美观以及实现的方便。3 层菜单包括：最左边的图片信息，左边有 5 张位图，分别代表：图像调节、声音调节、功能设定、频道调节、高级设定。这个是第一层菜单。第一层菜单展开之后，出现绘制在中间的第二层菜单，以图像调节而言，其第二层菜单包括：图像模式调节、亮度调节、对比度调节、清晰度调节、色调调节和高级等 6 个主要成员，成员基本是有纯字符组成。再展开第二层菜单，出现右边具体调节信息的第三层菜单，这里主要由一些数字，进度条构成。用户通过 3 层菜单，可以方便地知道需要调节的功能在哪里可以找到，用户界面友好美观。

图 2.2 是上述 OSD 设计的部分实现。

图2.2 基于SDA55XX的OSD界面设计部分实现效果图

2. OSD 模块设计：

从 OSD 设计中可以看出，每层菜单中的成员，基本都是由一些基本的元素组成的，包括一系列的图片、文字、框图，进度条等。所以有必要将这些元素封装成一个个模块，由此需要实现 DIALOG、BITMAP、FRAME、TEXT、SLIDER 等图形模块。

实现的主要模块包括：

1) DIALOG 控制模块:这是一个控制模块，用于控制整个 OSD 菜单的位置，大小，以及 DIALOG 的颜色信息等。其内部数据结构定义如下：

```
typedef struct OSD_DIALOG_CTRLStruct
{
    WORD    XPos;       //水平方向起始位置
    WORD    YPos;       //垂直方向起始位置
    BYTE    XSize;      //宽度
    BYTE    YSize;      //高度
    BYTE ROM *Colors;   //DIALOG 颜色信息
} OSD_DIALOG_CTRL;
```

2) BITMAP 模块：位图模块，包括 2 色、4 色以及 16 色位图模块，用于在 OSD 菜单中绘制，其内部数据结构定义如下：

```
typedef struct OSD_BITMAP_CTRLStruct
{
    WORD    XPos;       //水平方向起始位置
    WORD    YPos;       //垂直方向起始位置
    BYTE    XSize;      //宽度
    BYTE    YSize;      //高度
    BYTE ROM *Bp_PixelData;    //位图点阵信息
    BYTE ROM *Bp_HwIndexes;    //位图 DRCS 信息在 XRAM 中的索引
    BYTE ROM *Bp_ColorMap;     //位图颜色信息
} OSD_BITMAP_CTRL;
```

3) TEXT 模块：字符模块，其内部保存了整个字符串的位置，大小以及字符点阵等信息，由于 OSD 菜单需要支持多国语言，字符点阵包括多国字符信息。其内部数据结构定义如下：

```
typedef struct OSD_TEXT_CTRLStruct
{
    WORD    XPos;       //水平方向起始位置
    WORD    YPos;       //垂直方向起始位置
    BYTE    XSize;      //宽度
    BYTE    YSize;      //高度
    BYTE ROM *Bp_TextString;   //字符点阵信息
    BYTE ROM *Bp_HwIndexes;    //字符 DRCS 信息在 XRAM 中的索引（这里包括 SDA55XX 自带的字符集索引以及自定义的字符索引。
    BYTE ROM *Bp_ColorMap;     //字符颜色信息
} OSD_TEXT_CTRL;
```

4) FRAME 模块：控制边框的模块，包括整个边框的 4 个角，4 条边，以及中间填充部分的总共 9 部分信息。其内部数据结构定义如下：

```
typedef struct OSD_FRAME_CTRLStruct
```

```
{
    WORD    XPos;       //水平方向起始位置
    WORD    YPos;       //垂直方向起始位置
    BYTE    XSize;      //宽度
    BYTE    YSize;      //高度
    BYTE ROM *Bp_PixelData;    //FRAME 点阵信息
    BYTE ROM *Bp_ColorMap;     //FRAME 颜色信息
    BYTE    B_StartIndex;      //FRAME 的 DRCS 信息在 XRAM 中的起始
索引
    BYTE    B_NumChars;    //该 FRAME 所占用的 DRCS 数量
} OSD_FRAME_CTRL;
```

5) SLIDER 模块：进度条模块，用于显示各类参数的调节过程。其内部数据结构定义如下：

```
typedef struct OSD_SLIDER_CTRLStruct
{
    WORD    XPos;       //水平方向起始位置
    WORD    YPos;       //垂直方向起始位置
    BYTE    XSize;      //宽度
    BYTE    YSize;      //高度
    BYTE ROM *Bp_PixelData;    //进度条 DRCS 点阵信息
    BYTE ROM *Bp_ColorMap;     //进度条颜色信息
    BYTE    B_StartIndex;      //进度条的 DRCS 信息在 XRAM 中的起始
索引
    BYTE    B_NumChars;    //该进度条所占用的 DRCS 数量
    BYTE    Direction;         //进度条的前进方向
} OSD_SLIDER_CTRL;
```

3. 特殊寄存器

SDA55xx 中，数据存储地址空间被分为：256 字节的内部数据 RAM（internal data RAM），128 字节的特殊功能寄存器（Special Function Register）以及扩展数据存储空间(extended data memory)。其中特殊功能寄存器 SFR 用于控

制器件的状态和操作，它实现为静态 RAM 形式。在基于 SDA55XX 的开发方式中，必须详细了解每个 SFR 的作用，很多功能的实现，都是直接控制特殊寄存器的结果。

4. XRAM 分配

SDA55XX 提供了 16KB 的片内扩展存储器。其中包括 1KB 的数据采集存储器，1KB 空间用于用户数据，3KB 是显示 RAM，其余空间用户存储即将显示的 DRCS 数据等。整个 XRAM 分配如图所示。其中 Display-Memory 为 3KB 的显示 RAM，位置可以自定义，用指针 DISPOINT 来指向它的起始地址；GDW（Global Display Word）为 10 字节的数组，用于存储 OSD 显示的一些全局数据，位置可以自定义，用指针 GDWCURPOINT 来指向它的起始地址；CLUT(Color Look Up Table),96 个字节的 OSD 颜色表，每 2 个字节代表一个颜色，加上固化的 16 种颜色，屏幕上一共可以同时显示 64 种不同的颜色，位置可以自定义，用指针 CLUTPOINT 来指向它的起始地址；1 位 2 位 4 位 DRCS 点阵数据存放的位置可以由用户自定义，分别使用 3 个指针 DRCS1POINT、DRCS2POINT 和 DRCS4POINT 来指向它们。

由于 XRAM 的空间中，各部分数据的位置分配，都是用户可以自定义的，除了一点例外，SDA55XX 规定：DISPOINT，CLUTPOINT，GDWCURPOINT 这 3 个指针必须连续存放，DRCS1POINT、DRCS2POINT 和 DRCS4POINT 这 3 个指针也必须连续存放。为了可以读写所有的数据，SDA55XX 使用 2 个 SFR 来寻找 XRAM 数据的入口，分别为 POINTARRAY0 以及 POINTARRAY1。用户必须把指向 3KB 显示 RAM 起始地址的指针（DISPOINT）的地址赋值给 POINTARRAY0,将指向 1 位 DRCS 点阵存放 RAM 的指针（DRCS1POINT）的地址赋值给 POINTARRAY1。这样，显示程序就能寻址到所有显示时需要的数据（包括点阵信息，色表，全局控制数据等等）。

图 2.3 显示了 SDA55xx XRAM 空间分配情况。

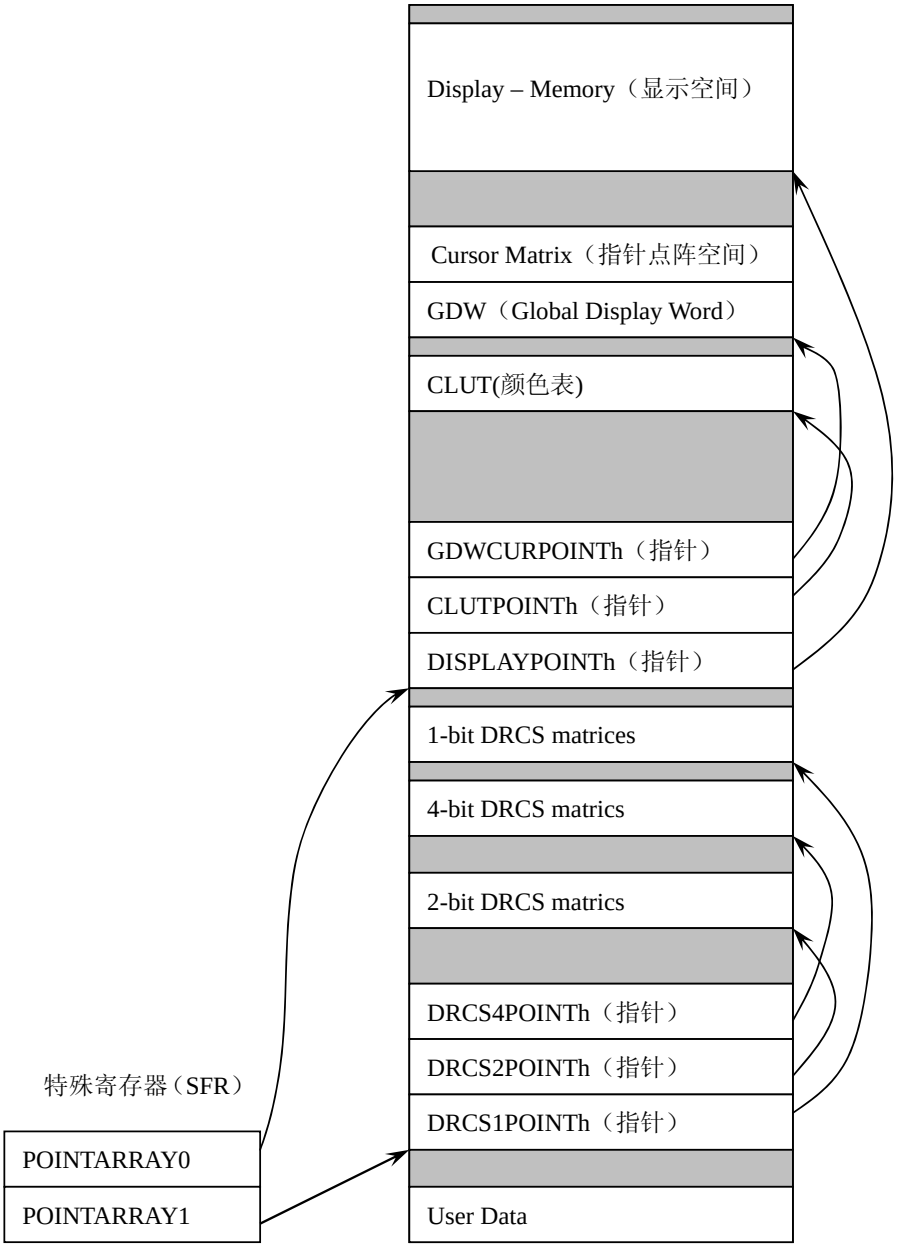

图2.3 SDA55xx XRAM结构示意图

5. DRCS 构造

在 SDA55XX 中，所有显示的 OSD 内容，都是以 DRCS 为单位显示的。每个 DRCS，

是一个矩形点阵，其大小为 10 行×9…16 列（DRCS 的宽度可以自定义控制）。对应于不同的颜色数量，DRCS 又分为 DRCS1,DRCS2,DRCS4，其意思为：没个象素分别用 1，2，4 位来表示，所以 DRCS1 只能显示 2 种颜色，DRCS2 可以显示 4 种颜色，DRCS4 则显示多达 16 种颜色，而正式因为颜色位数的不同，以 10 行×12 列的 DRCS 来说，每个 DRCS1 占用 15 字节的空间，每个 DRCS2 占用 30 字节，DRCS4 占用 60 字节。由于 XRAM 只有 16KB 空间，而每一屏菜单，以 250 行，400 列大小来计算，需要大量的 DRCS 来填充，所以空间的分配和计算，成为 SDA55XX OSD 开发中的一个主要任务。

另外，将 OSD 设计中的所有图片以及文字等素材，转化成 DRCS，也需要大量以及烦琐的工作。其中，由于 DRCS4 占用大量空间，那么，对于 OSD 设计中的所有图片、边框等区域，只要颜色数量在 4 种以下的，就要尽量转化成 DRCS2 和 DRCS1，这一任务，也是 SDA55XX OSD 开发中的一个难点。

2.2 当前 OSD 开发模式问题研究

从前文的论述中可以看到，基于 SDA55XX 的 OSD 开发模式，主要精力都放在了 XRAM 分配以及特殊寄存器赋值等任务上了，而这些任务，都需要非常烦琐和细致的工作。并且，所有这些大工作量的开发都紧紧跟 SDA55XX 的具体细节紧密相关，由此，我们不难看出此开发模式存在的一些主要问题。

2.2.1 硬件依赖性强，软件移植困难

在基于 SDA55XX 的开发模式下，最后生成的 OSD 软件，完全只能在采用 SDA55XX 芯片数字电视中使用。这是由于，除了 OSD 设计以及 OSD 图形模块设计之外，其余关键的实现部分的 XRAM 分配、特殊寄存器设置等，都是完全基于 SDA55XX 的芯片特性开发的，在采用其他芯片的解决方案中，基于 SDA55XX 的所有开发成果，基本都不能移植，必须从头开始开发另外一套方案。这将致使整机厂商只能针对不同的芯片方案进行手工作坊式的软件开发。

2.2.2 开发困难，周期长

由于嵌入式系统有着体积小、功能集中、可靠性高等优点，如何缩短嵌

入式系统的开发周期，降低开发成本，以及提高产品的可靠性已成为嵌入式行业普遍关注的问题。在基于 SDA55XX 的 OSD 开发模式中，软件开发的主要工作量，都放在了详细分析 SDA55XX 的具体 XRAM 分配，寄存器赋值等任务上，对于开发高效，健壮的 OSD 系统，快速投入应用是相当不利的。

2.2.3 缺乏统一的平台

在基于 SDA55XX 的开发模式下，由于没有统一的平台，采用不同芯片的数字电视方案的都有各自的实现，技术不完全开放，造成了诸如一些关键寄存器的地址、设置等原本简单的技术问题被少数人"垄断"的局面，整机厂家不得不高度依赖于国外厂家的技术支持或企业内个别技术人员，而这些资源的变动常常会影响整个产品开发。企业很难形成技术积累，稳步实现技术发展。

第 3 章 基于嵌入式操作系统的 OSD 开发关键技术

从前文的实践研究可以看到，当前的 OSD 开发模式，开发周期长，效率低下，不利于软件工厂化生产。这从一个侧面可以看到，这样的软件开发模式，必然使得数字电视的软件开发跟不上我国数字电视的发展速度，成为发展的瓶颈。从本章开始，将开始探讨和研究基于“和欣”操作系统的 OSD 开发模式，本章将先介绍基于嵌入式操作系统的 OSD 开发关键技术。

3.1 “和欣”嵌入式操作系统

3.1.1 数字电视操作系统

国外数字电视采用的操作系统主要有：Nucleus、PSOS、 OS20(ST)、 Linux、VxWorks 和 Windows CE 等。所有这些操作系统中国都不占有任何产权。同时某些国外厂商也能提供的自己的操作系统，他们采用操作系统与芯片（CPU）捆绑的方式，向我国提供芯片和操作系统一体化的解决方案，但这种操作系统比较"原始"，特点如下：

. 基本上是实时内核 + 设备驱动
. 设备驱动、API 等没有标准
. 缺少网络支持（这些功能都"交"给了"中间件"）
. 图形系统功能弱，或没有（这些功能都"交"给了"中间件"）
. 有些操作系统没有文件系统 （这些功能都"交"给了"中间件"）

3.1.2 “和欣”操作系统简介

“和欣”是 32 位嵌入式操作系统。该操作系统可以从多个侧面进行描述：

32 位嵌入式操作系统。操作系统基于微内核，具有多进程、多线程、抢占式、基于线程的多优先级任务调度等特性。提供 FAT 兼容的文件系统，可以从

软盘、硬盘、FLASH ROM 启动，也可以通过网络启动。“和欣”操作系统体积小，速度快，适合网络时代的绝大部分嵌入式信息设备。

完全面向构件技术的操作系统。操作系统提供的功能模块全部基于 EZCOM 构件技术，因此是可拆卸的构件，应用系统可以按照需要剪裁组装，或在运行时动态加载必要的构件。

从传统的操作系统体系结构的角度来看，“和欣”操作系统可以看成是由微内核、构件支持模块、系统服务器组成的。

微内核：主要可分为 4 大部分：硬件抽象层（对硬件的抽象描述，为该层之上的软件模块提供统一的接口）；内存管理（规范化的内存管理接口，虚拟内存管理）；任务管理（进程管理的基本支持，支持多进程，多线程）；进程间通信（实现进程间通信的机制，是构件技术的基础设施）。

构件支持模块：提供了对 EZCOM 构件的支持，实现了构件运行环境。构件支持模块并不是独立于微内核单独存在的，微内核中的进程间通讯部分为其提供了必要的支持功能。

系统服务器：在微内核体系结构的操作系统中，文件系统、设备驱动、网络支持等系统服务是由系统服务器提供的。在“和欣”操作系统中，系统服务器都是以动态链接库的形式存在。

3.1.3 “和欣”灵活内核

“和欣”操作系统的实现全面贯穿了 EZCOM 思想，EZCOM 构件可以运行于不同地址空间或不同的运行环境。可以把操作系统的内核地址区看成是一段特殊的地址空间，用户可以根据运行时的需求，自主选择将操作系统的某些系统服务构件、文件系统、图形系统、设备驱动构件等运行于内核地址空间或用户地址空间。与传统的操作系统的“大内核”、“微内核”体系结构相比，“和欣”操作系统内核里提供的系统服务，完全可以由用户依据系统自身的需求动态决定。因此称“和欣”操作系统内核为“灵活内核” （Agile Kernel）。

“和欣”灵活内核的体系结构，利用构件和中间件技术解决了长期以来困扰操作系统体系结构设计者的大内核和微内核在性能、效率与稳定性、安全性之间不能两全其美的矛盾。

图 3.1 表示了“和欣”灵活内核及其与系统构件和应用构件的关系：

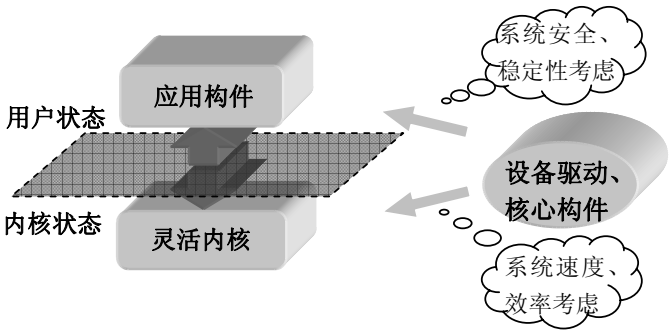

图3.1 “和欣”灵活内核及其与系统构件和应用构件的关系图

3.1.4 “和欣”操作系统提供的功能

从应用编程的角度看，“和欣”操作系统提供了一套完整的、符合 EZCOM 规范的系统服务构件及系统 API，为在各种嵌入式设备的硬件平台上运行 EZCOM 二进制构件提供了统一的编程环境。

“和欣”操作系统还提供了一组动态链接构件库，这些构件库通常是开发嵌入式应用系统时不可缺少的：

- 图形系统构件库（方便开发图形用户操作界面）；
- 设备驱动构件库（各种输入输出设备的驱动）；
- 文件系统构件库（FAT 兼容，包括对 FLASH 等的支持）；
- 网络系统构件库（TCP/IP 等网络协议支持）。

系统提供的构件库，以及用户开发的应用程序构件都是通过系统接口与内核交互，从这个意义上说，他们处于同样的地位。用户可以开发性能更好或者更符合需求的文件系统、网络系统等构件库，替换这些构件库，也可以开发并建立自己的应用程序构件库。这就是基于构件技术操作系统的优势之一。

此外，为了方便用户编程，在“和欣”SDK 中还提供了以下函数库：

- 与微软 Win32 API 兼容的应用程序编程接口(zeew32 API)；
- 标准 C 运行库（libc）；
- “和欣”提供的工具类函数（zeeutil）。

对程序员来说,“和欣”操作系统提供的用户编程接口与上一节中介绍的“和欣”构件运行平台完全一样。所以，在相互兼容的硬件平台上，不管运行的是“和欣”操作系统还是 Windows 操作系统，应用程序可以不加区分地在其上运行。

“和欣”操作系统实现并支持系统构件及用户构件相互调用的机制，为 EZCOM 构件提供了运行环境。关于 EZCOM 构件的运行环境，其描述与“和欣”构件运行平台是一样的，在此从简。因此，可以把“和欣”操作系统看成是直接运行在硬件平台上的“和欣”构件运行平台。

图 3.2 表示了“和欣”操作系统及其主要构成：

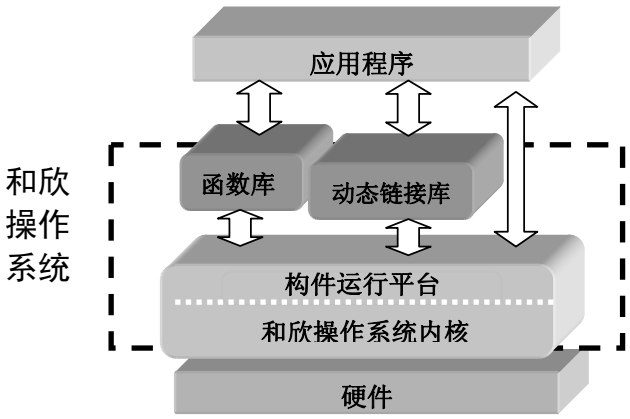

图3.2 “和欣”操作系统及其主要构成

3.1.5 “和欣”操作系统的应用软件开发

“和欣”SDK 提供了应用软件的开发环境和工具。开发“和欣”应用软件的开发环境如图 3.3 所示：

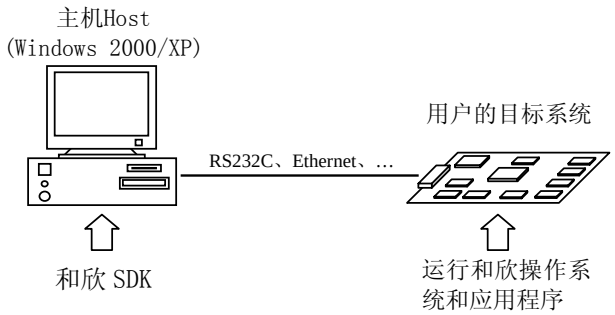

图3.3 “和欣”应用软件的开发环境

开发“和欣”应用软件的过程，如图3.4所示：

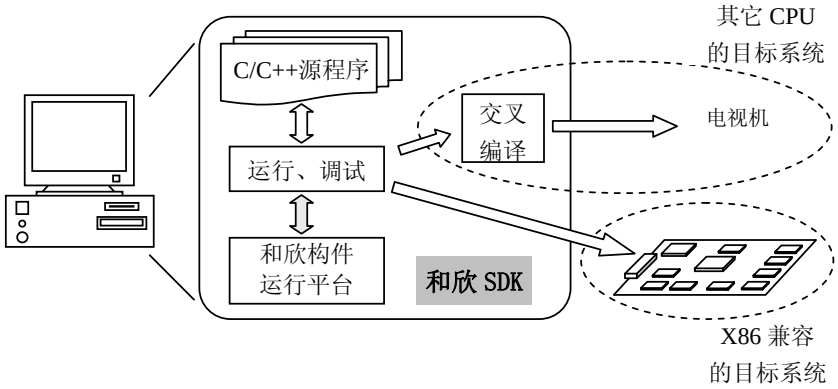

图3.4 开发“和欣”应用软件的过程示意图

3.1.6 “和欣”操作系统的优势

“和欣”操作系统的最大特点就是：

- 全面面向构件技术，在操作系统层提供了对构件运行环境的支持；
- 用构件技术实现了“灵活”的操作系统。

这是“和欣”操作系统区别于其它商用嵌入式操作系统产品的最大优势。

在新一代因特网应用中，越来越多的嵌入式产品需要支持网络服务，而网络服务的提供一定是基于构件的。在这种应用中，用户通过网络获得服务程序，这个程序一定是带有自描述信息的构件，本地系统能够为这个程序建立运行环境，自动加载运行。这是新一代因特网应用的需要，是必然的发展方向。“和欣”操作系统就是应这种需要而开发，率先在面向嵌入式系统应用的操作系统中实现了面向构件的技术。

因此，构件化的“和欣”操作系统可以为嵌入式系统开发带来以下好处：

- 在嵌入式软件开发领域，导入先进的工程化软件开发技术。嵌入式软件一般用汇编语言、C 语言，在少数系统中已经支持了 C++开发，但是由于还没有一个嵌入式操作系统能够提供构件化的运行环境，可以说，嵌

入式软件开发还是停留在手工作坊式的开发方式上。”和欣”操作系统使得嵌入式应用的软件开发能够实现工程化、工厂化生产。

- 可以动态加载构件。动态加载构件是因特网时代嵌入式系统的必要功能。新一代 PDA 和移动电话等移动电子产品，不能再像以前那样由厂家将所有的功能都做好后固定在产品里，而要允许用户从网上获得自己感兴趣的程序。
- 随时和动态地实现软件升级。动态加载构件的功能，同样可以用于产品的软件升级，开发商不必为了添加了部分功能而向用户重新发布整套软件，只需要升级个别构件。
- 灵活的模块化结构，便于移植和剪裁。可以很容易定制成为针对不同硬件配置的紧凑高效的嵌入式操作系统。添加或删除某些功能模块也非常简单。
- 嵌入式软件开发商容易建立自己的构件库。在不同开发阶段开发的软件构件，其成果很容易被以后的开发所共享，保护软件开发投资。软件复用使得系列产品的开发更加容易，缩短新产品开发周期。
- 容易共享第三方软件开发商的成果。面向行业的构件库的建设，社会软件的丰富，使得设备厂家不必亲自开发所有的软件，可以充分利用现有的软件资源，充分发挥自己的专长为自己的产品增色。
- 跨操作系统平台兼容，降低软件移植的风险。在”和欣”开发环境上开发的软件所具有的跨平台特性，使得用户可以将同样的可执行文件不加修改地运行在“和欣”操作系统（嵌入式设备）与 Windows 2000/XP（PC）上。特别是对于需要将 Windows 上的软件移到嵌入式系统以降低产品成本的用户，这一特点不仅可以大大节约软件移植的费用，还可以避免因移植而带来的其它隐患。
- 功能完备的开发环境和方便的开发工具，帮助嵌入式开发人员学习和掌握先进的构件化软件编程技术，提高软件开发效率。应用软件可以在开发环境下开发调试，与硬件研制工作同时进行，缩短产品研制周期。

3.2 “和欣”构件运行平台

3.2.1 “和欣”构件运行平台简介

“和欣”构件运行平台提供了一套符合 EZCOM（详见 2.3 节）规范的系统服务构件及支持构件相关编程的 API 函数，实现并支持系统构件及用户构件相互调用的机制，为 EZCOM 构件提供了编程运行环境。“和欣”运行平台有在不同操作系统上的实现，符合 EZCOM 编程规范的应用程序通过该平台实现二进制级跨操作系统平台兼容。

在“和欣”操作系统中，“和欣”构件运行平台与“和欣”灵活内核共同构成了完整的操作系统。

在 Windows 2000、WinCE、Linux 等其它操作系统上，“和欣”构件运行平台屏蔽了底层传统操作系统的具体特征，实现了一个构件化的虚拟操作系统。在“和欣”构件运行平台上开发的应用程序，可以不经修改、不损失太多效率、以相同的二进制代码形式，运行于传统操作系统之上。

图 3.5 显示了“和欣”构件运行平台在 Windows 2000/XP、“和欣”操作系统中的位置。

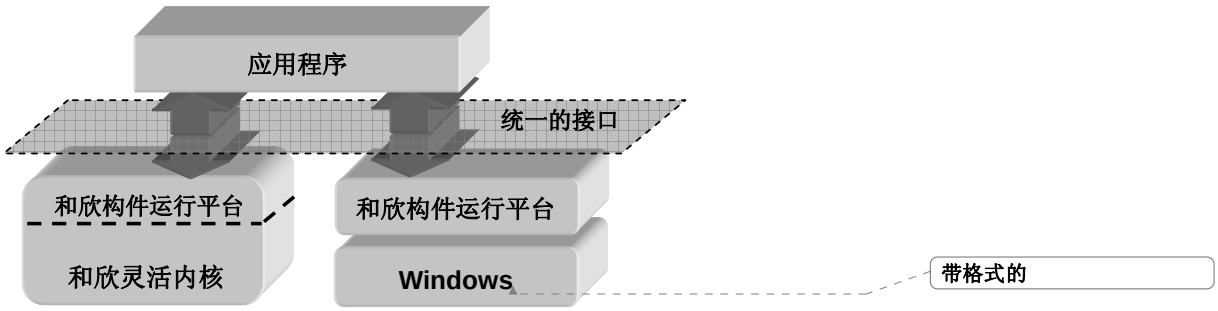

图3.5 “和欣”构件运行平台与Windows 2000/XP、“和欣”操作系统关系图

3.2.2 “和欣”构件运行平台的功能

从“和欣”构件运行平台的定义，知道该平台为 EZCOM 提供了运行环境。从这个意义上，这里说的 EZCOM 技术也可以理解为在运行环境中对 EZCOM 规范

提供支持的程序集合。

从编程的角度看,“和欣”构件运行平台提供了一套系统服务构件及系统 API（应用程序编程接口），这些是在该平台上开发应用程序的基础。

“和欣”操作系统提供的其它构件库也是通过这些系统服务构件及系统 API 实现的。系统提供的这些构件库为应用编程开发提供了方便：

- 图形系统构件库；
- 设备驱动构件库；
- 文件系统构件库；
- 网络系统构件库。

从“和欣”构件运行平台来看，这些构件和应用程序的构件是处于同样的地位。用户可以开发性能更好或者更符合需求的文件系统、网络系统等构件库，替换这些构件库，也可以开发并建立自己的应用程序构件库。

图 3.6 显示出“和欣”构件运行平台的功能及其与构件库、应用程序的关系。

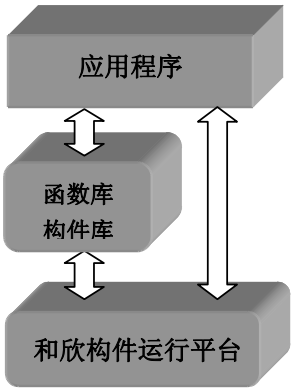

图3.6 “和欣”构件运行平台的功能及其与构件库、应用程序的关系图

从支持 EZCOM 构件的运行环境的角度看，“和欣”构件运行平台提供了以下功能：

- 根据二进制构件的自描述信息自动生成构件的运行环境，动态加载构件；
- 提供构件之间的自动通信机制，构件间通信可以跨进程甚至跨网络；
- 构件的运行状态监控，错误报告等；

- 提供可干预构件运行状态的机制，如负载均衡、线程同步、访问顺序控制、安全（容错）性控制、软件使用权的控制等；
- 构件的生命周期管理，如进程延续（Persistence）控制、事务元（Transaction）控制等；

总之，构件运行平台为 EZCOM 构件提供了对程序员完全透明的运行环境，构件可以运行在不同地址空间，不同环境，甚至跨网络。构件运行平台自动为构件运行提供支持，配置必要的网络协议、针对不同的输入输出设备的协议。程序员不必过多地去关心诸如网络协议转换及构件运行控制等与其它构件互操作时的协调问题，只需专注于自己需要解决的程序算法的实现。从而可以从繁杂庞大的应用环境体系中解放出来，大大提高编程的效率。

“和欣”构件运行平台直接运行二进制构件，而不是像 JAVA 和.NET 那样通过虚拟机在运行程序时解释执行中间代码。因此，与其它面向构件编程的系统相比，具有资源消耗小，运行效率高的优点。

3.2.3 “和欣”构件运行平台的技术优势

作为总结，“和欣”构件运行平台的主要技术优势列举如下：

- 开发跨操作系统平台的应用软件；
- 对程序员透明的 EZCOM 构件运行环境，提高编程的效率；
- 直接运行二进制构件代码，实现软件运行的高效率；
- 构件可替换，用户可建立自己的构件库。

需要说明的是，“和欣”构件运行平台实现的应用软件跨操作系统平台兼容是以具有同样的硬件体系结构为前提的。目前，“和欣”构件运行平台还不能支持不同指令系统的 CPU 间的“跨平台”兼容。

3.2.4 利用“和欣”构件运行平台编程

对程序员来说，编写运行于“和欣”构件运行平台上的程序，运用 EZCOM 技术和跨平台技术的具体方法，体现在对构件库的接口方法、通用 API 函数的调用上。应用程序运行所需要的动态链接库，则是在程序运行时由“和欣”构件运行平台自动加载的。

图 3.7 简明地表示了编写运行于“和欣”构件运行平台上的应用程序所需

的相关要素之间的关系的示意。

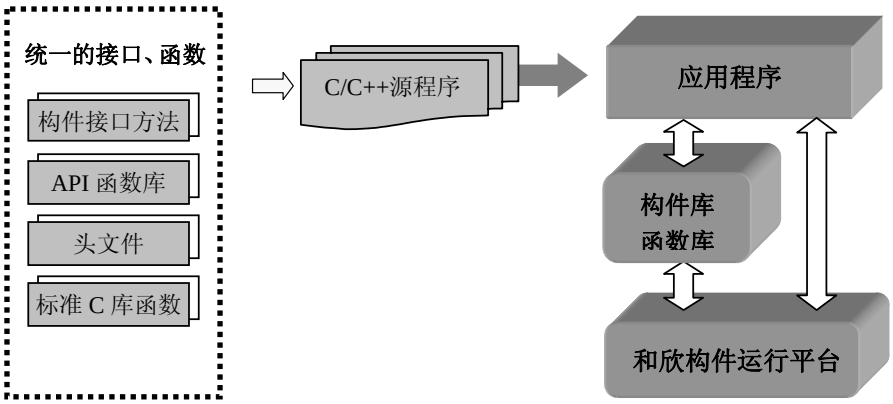

图3.7 编写“和欣”构件运行平台上的应用程序相关要素示意图

3.3 EZCOM 构件技术

3.3.1 EZCOM 技术的由来

80 年代以来，目标指向型软件编程技术有了很大的发展，为大规模的软件协同开发以及软件标准化、软件共享、软件运行安全机制等提供了理论基础。其发展可以大致分为以下几个阶段。

- 面向对象编程

通过对软件模块的封装，使其相对独立，从而使复杂的问题简单化。面向对象编程强调的是对象的封装，但模块（对象）之间的关系在编译的时候被固定，模块之间的关系是静态的，在程序运行时不可改变模块之间的关系，就是说在运行时不能换用零件。其代表是 C++语言所代表的面向对象编程。

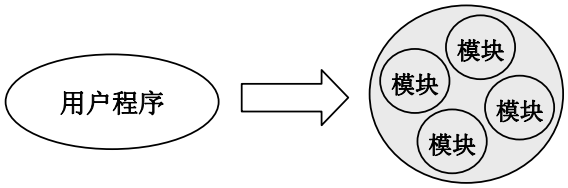

图3.8 面向对象编程的运行模型，模块之间的关系固定

- 面向构件编程

为了解决不同软件开发商提供的构件模块（软件对象）可以相互操作使用，构件之间的连接和调用要通过标准的协议来完成。构件化编程模型强调协议标准，需要提供各厂商都能遵守的协议体系。就像公制螺丝的标准一样，所有符合标准的螺丝和螺母都可以相互装配。构件化编程模型建立在面向对象技术的基础之上，是完全面向对象的，提供了动态构造部件模块（运行中可以构造部件）的机制。构件在运行时动态装入，是可换的。其代表是 COM 技术。

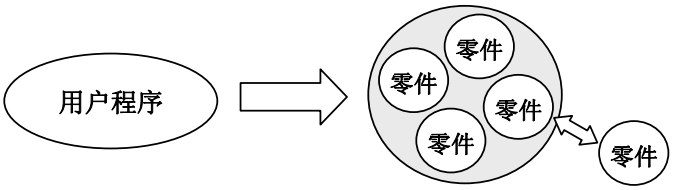

图3.9 构件化程序的运行模型，运行时零件可替换

- 面向中间件编程

由于因特网的普及，构件可来自于网络，系统要解决自动下载，安全等问题。因此，系统中需要根据构件的自描述信息自动生成构件的运行环境，生成代理构件即中间件，通过系统自动生成的中间件对构件的运行状态进行干预或控制，或自动提供针对不同网络协议、输入输出设备的服务（即运行环境）。中间件编程更加强调构件的自描述和构件运行环境的透明性，是网络时代编程的重要技术。其代表是 EZCOM、JAVA 和.NET（C#语言）。

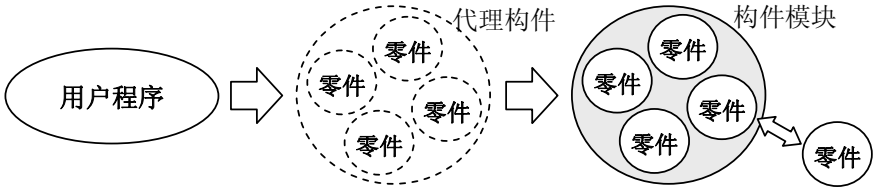

图3.10 中间件运行环境的模型，动态生成代理构件

在这样的发展过程中，人们逐步深化了对大规模软件开发所需的科学模型、网络环境下软件运行必要机制的理解，使软件技术达到了更高的境界，实现了：

构件的相互操作性。不同软件开发商开发的具有独特功能的构件，可以确保与其他人开发的构件实现互操作。

软件升级的独立性。实现在对某一个构件进行升级时不会影响到系统中的其它构件。

编程语言的独立性。不同的编程语言实现的构件之间可以实现互操作。

构件运行环境的透明性。提供一个简单、统一的编程模型，使得构件可以在进程内、跨进程甚至于跨网络运行。同时提供系统运行的安全、保护机制。

EZCOM 技术就是在总结面向对象编程、面向构件编程技术的发展历史和经验，为更好地支持面向以 Web Service（WEB 服务）为代表的下一代网络应用软件开发而发明的。EZCOM 很大程度地借鉴了 COM 技术，保持了和 COM 的兼容性，同时对 COM 进行了重要的扩展。

3.3.2 EZCOM 构件技术概要

EZCOM 构件技术是面向构件编程的编程模型，它规定了一组构件间相互调用的标准，使得二进制构件能够自描述，能够在运行时动态链接。

EZCOM 兼容微软的 COM。但是和微软 COM 相比，EZCOM 删除了 COM 中过时的约定，禁止用户定义 COM 的非自描述接口；完备了构件及其接口的自描述功能，实现了对 COM 的扩展；对 COM 的用户界面进行了简化包装，易学易用。

从上面的定义中，我们可以说 EZCOM 是微软 COM 的一个子集。EZCOM 很大程度地借鉴了 COM 技术，保持了和 COM 的兼容性，同时对 COM 进行了重要的扩展。在“和欣”SDK 工具的支持下，使得高深难懂的构件编程技术很容易被 C/C++程序员理解并掌握。EZCOM 中的“ez”源自与英文单词“easy”，恰如其分地反映了这一特点。

EZCOM 技术是在总结面向对象编程、面向构件编程技术的发展历史和经验，

为更好地支持面向 Web Service（WEB 服务）的下一代网络应用软件开发而开发的。

EZCOM 的编程思想是“和欣”技术的精髓，它贯穿于整个技术体系的实现中。

为了在资源有限的嵌入式系统中实现面向中间件编程技术，同时又能得到 C/C++的运行效率，EZCOM 没有使用 JAVA 和.NET 的基于中间代码—虚拟机的机制，而是采用了用 C++编程，用“和欣”SDK 提供的工具直接生成运行于“和欣”构件运行平台的二进制代码的机制。用 C++编程实现构件技术，使得更多的程序员能够充分运用自己熟悉的编程语言知识和开发经验，很容易掌握面向构件、中间件编程的技术。在不同操作系统上实现的“和欣”构件运行平台，使得 EZCOM 构件的二进制代码可以实现跨操作系统平台兼容。

3.3.3 EZCOM 技术的意义

对于面向 WEB 服务的应用软件开发，以及开发操作系统这样的大型系统软件而言，采用 EZCOM 构件技术具有以下的意义：

- 不同软件开发商开发的具有独特功能的构件，可以确保与其他人开发的构件实现互操作。
- 实现在对某一个构件进行升级时不会影响到系统中的其它构件。
- 不同的编程语言实现的构件之间可以实现互操作。
- 提供一个简单、统一的编程模型，使得构件可以在进程内、跨进程甚至于跨网络运行。同时提供系统运行的安全、保护机制。
- EZCOM 构件与微软的 COM 构件二进制兼容，但是 EZCOM 的开发工具自动实现构件的封装，简化了构件编程的复杂性，有利于构件化编程技术的推广普及；
- EZCOM 构件技术是一个实现软件工厂化生产的先进技术，可以大大提升企业的软件开发技术水平，提高软件生产效率和软件产品质量；
- 软件工厂化生产需要有零件的标准，EZCOM 构件技术为建立软件标准提供了参考，有利于建立企业、行业的软件标准和构件库。

3.3.4 EZCOM 技术对软件工程的作用

大型软件的生产，已经不是传统的手工作坊式的开发所能够胜任的。所谓

的软件工厂化生产，是软件工程研究几十年来追求的理想。实现这样的目标，需要有一个科学的编程模型，为所有参与项目开发的软件工程师开发的软件能够正确地对接运行提供技术上的保障。这些软件可以是由一个公司的不同部门提供，也可以是不同企业，不同地点、不同时间生产的。COM 技术在微软经过十多年的应用与提炼，被证明是行之有效的编程模型，基于 COM 技术，微软造就了 Windows NT 以及配套应用软件这样的大型软件产品。汲取了 COM 技术的精华，并对其进行了扩展和简易包装的 EZCOM 技术，将成为软件工厂化生产的利器。

EZCOM 的重要特点就是：构件的简易化包装；构件的相互操作性；软件升级的独立性；编程语言的独立性；进程运行透明度。在实际的编程应用中，EZCOM 技术的优势体现在以下方面：

（1）易学易用

基于 COM 的构件化编程技术是大型软件工程化开发的重要手段。但是微软 COM 的繁琐的构件描述体系令人望而生畏。EZCOM 的开发环境“和欣”SDK 提供了结构简洁的构件描述语言和自动生成辅助工具等，使得 C++程序员可以很快地掌握 EZCOM 编程技术。

（2）可以动态加载构件

在网络时代，软件构件就相当于零件，零件可以随时装配。EZCOM 技术实现了构件动态加载，使用户可以随时从网络得到最新功能的构件。

（3）采用第三方软件丰富系统功能

EZCOM 技术的软件互操作性，保证了系统开发人员可以利用第三方开发的，符合 EZCOM 规范的构件，共享软件资源，缩短产品开发周期。同时用户也可以通过动态加载第三方软件扩展系统的功能。

（4）软件复用

软件复用是软件工程长期追求的目标，EZCOM 技术提供了构件的标准，二进制构件可以被不同的应用程序使用，使软件构件真正能够成为“工业零件”。充分利用“久经考验”的软件零件，避免重复性开发，是提高软件生产效率和软件产品质量的关键。

（5）系统升级

传统软件的系统升级是一个令软件系统管理员头痛的工程问题，一个大型软件系统常常是“牵一发而动全身”，单个功能的升级可能会导致整个系统需要重新调试。EZCOM 技术的软件升级独立性，可以圆满地解决系统升级问题，个别

构件的更新不会影响整个系统。

（6）实现软件工厂化生产

上述几个特点，都是软件零件工厂化生产的必要条件。构件化软件设计思想规范了工程化、工厂化的软件设计方法，提供了明晰可靠的软件接口标准，使软件构件可以像工业零件一样生产制造，零件可用于各种不同的设备上。

（7）提高系统的可靠性、容错性

由于构件运行环境可控制，可以避免因个别构件的崩溃而波及到整个系统，提高系统的可靠性。同时，系统可以自动重新启动运行中意外停止的构件，实现系统的容错。

（8）有效地实现系统安全性

系统可根据构件的自描述信息自动生成代理构件，通过代理构件进行安全控制，可以有效地实现对不同来源的构件实行访问权限控制、监听、备份容错、通信加密、自动更换通信协议等等安全保护措施。

3.3.5 EZCOM 技术在“和欣”技术体系中的作用

EZCOM 构件技术是“和欣”技术体系的精髓，它贯穿于整个技术体系的实现中。可以说“和欣”操作系统是基于 EZCOM 实现的，同时它也是全面面向 EZCOM 构件技术的。

- 构件化的操作系统内核，以及“灵活内核”体系结构；
- 构件化的图形系统、设备驱动、文件系统等操作系统的系统服务；
- 支持应用软件跨平台二进制兼容的“和欣”构件运行平台；
- 构件化的应用软件，支持 WEB 服务，支持动态加载。

在面向嵌入式系统应用的操作系统中全面导入构件技术，对于开发功能越来越复杂的应用系统有着重要的意义。功能丰富的嵌入式设备将更多地要和网络发生关系，要支持WEB服务，前一节中所介绍的软件工程的意义，对于嵌入式系统开发来说同样是重要的。

3.3.6 如何用 EZCOM 技术编程

EZCOM 是一个面向构件的编程模型，也可以说是一种编程思想，它表现为一组编程规范，包括构件、类、对象、接口等定义与访问构件对象的规定。

EZCOM 的实现，是由一个配套的开发环境和运行环境完成的。

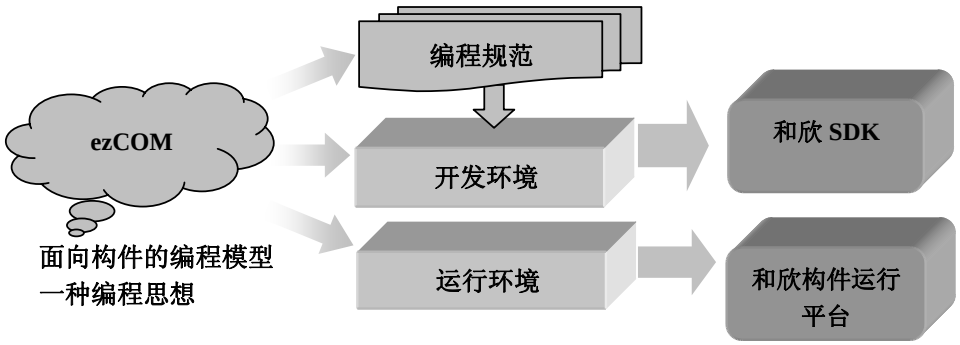

图3.11 使用ezCOM编程示意图

第 4 章 基于嵌入式操作系统的 OSD 开发模式实现方法研究

从本章开始，将着重研究基于第三章介绍的和欣操作系统及其 EZCOM 构件技术之上的 OSD 开发模式的实现方法。

4.1 基于“和欣”嵌入式操作系统的 OSD 开发系统结构设计

4.1.1 系统结构

“和欣嵌入式操作系统 +中间件”构成了基于嵌入式操作系统的 OSD 系统的开发运行环境。嵌入式操作系统及设备的上层应用接口通过驱动程序控制相应设备，并对中间件提供系统服务。在此开发模式中，OSD 中间件基于操作系统为应用软件提供编程接口，使得上层应用可以使用统一的开发的接口。这样，上层应用只要是遵循中间件提供的接口开发的，那么底层设备的变动将不再影响到上层的实现了。

基于“和欣”嵌入式操作系统的 OSD 开发模式采用立体层次结构，OSD 中间件作为系统构件运行在操作系统中间件运行平台上，使得它可以不断的根据不同的功能需求和所支持应用的类型增加新的功能和要求。图 4.1 描述了这种基于“和欣”嵌入式操作系统的 OSD 开发模式的系统结构。

图4.1 基于“和欣”嵌入式操作系统的OSD开发模式的系统结构图

4.1.2 数字电视开发中间件技术

数字电视中间件是指位于操作系统与应用程序之间的软件部分，它以应用程序接口的形式存在，所有的应用程序基于 API 进行开发，能够支持丰富的应用。采用中间件系统，可以跨越技术、标准等复杂的内容，用简单的方法定制具有自己特色的应用软件，从而在提高开发效率、减少开发成本的同时能够跟上技术的发展，将应用的开发变得更加简捷，使产品的开放性和可移植性更强，可以大大降低数字电视的开发成本。

本系统中的 OSD 中间件，也是属于数字电视中间件的一种实现，OSD 中间件完全基于 EZCOM 技术实现，这将在下文进行详细阐述。

4.2 基于“和欣”嵌入式操作系统的 OSD 开发具体实现

“和欣”操作系统基于与微软的 COM 构件技术兼容的 ezCOM 构件编程技术，系统架构相对灵活，局部部件（硬件驱动）的替换将不会影响到系统和应用的整体运行。因此，移植到新的硬件平台时只要对某些局部的模块（构件）做一些替换。这样的体系结构使得“和欣”的移植性大大优于其他商业推广的嵌入式操作系统。

4.2.1 构件描述语言 CDL(Component Definition Language)

“Elastos”操作系统最显著的一个特点就是它是一个完全面向构件技术的操作系统，因此也带来了软件升级维护的巨大方便。为了充分发挥构件化开发的优点，在设计时，对系统最终要提供的功能进行详细分析，将每个基本的功能都封装成独立的构件，供高层应用调用，在描述具体构件时，对 Elastos 提供的一些重要接口，机制进行分析。构件的描述使用的是 ezCOM 的构件描述语言 CDL(Component Definition Language)。CDL 摈弃了 IDL 中不合理的地方，做了一些扩展。

开发一个 ezCOM 构件时，第一步就是写一个 CDL 文件，然后进行编译，自动生成工具就可以自动生成程序框架，用户需要做的就是添加处理自己事务逻

辑的代码。实现如图 4.2 所示：

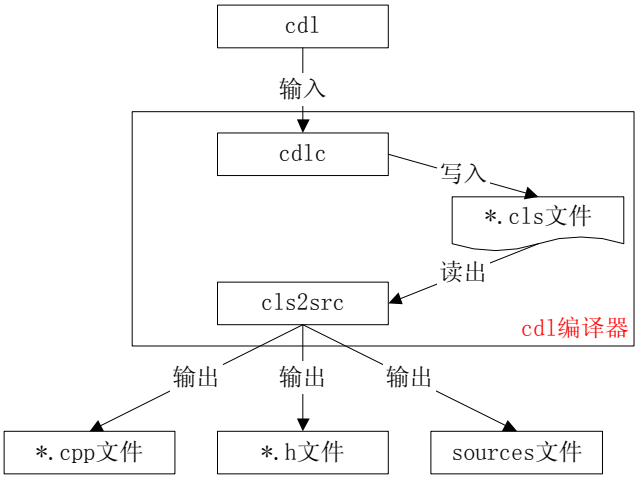

图4.2 使用CDL生成程序框架示意图

4.2.2 OSD 中间件介绍

- **OSD 中间件概述**

在“和欣”上实现 OSD 系统的的最简单方法是使用 OSD 中间件。OSD 中间件采用了模块化 ezCOM 构件技术，对于各类底层操作进行了封装，把应用程序的编程工作量减到最低。

- **OSD 中间件的基本功能**

  - OSD 中间件为应用程序提供基本的寄存器、内存读写功能。
  - OSD 中间件提供位图、文字、进度条等 OSD 菜单基本控件。
  - OSD 中间件由于采用了可扩展的构件结构，可以方便的添加功能模块，而不会影响用户已有程序的运行。

- **OSD 中间件的特点**

  - 面向构件的体系结构，提供了基本的 OSD 构件库，能够灵活地实现各种 OSD 菜单。

- 自动生成程序框架，用户编程可以灵活调用各类控件，大大简化了开发难度并减少了编程工作量。
- 接口方法友好，易于掌握和使用。

- **OSD 中间件的结构**

OSD 中间件给 OSD 应用提供服务，应用程序既可以灵活调用 OSD 控件，又可以进行 OSD 底层调用。

下面请看 OSD 中间件系统结构图及其与 OSD 应用和“和欣”构件运行平台的关系：

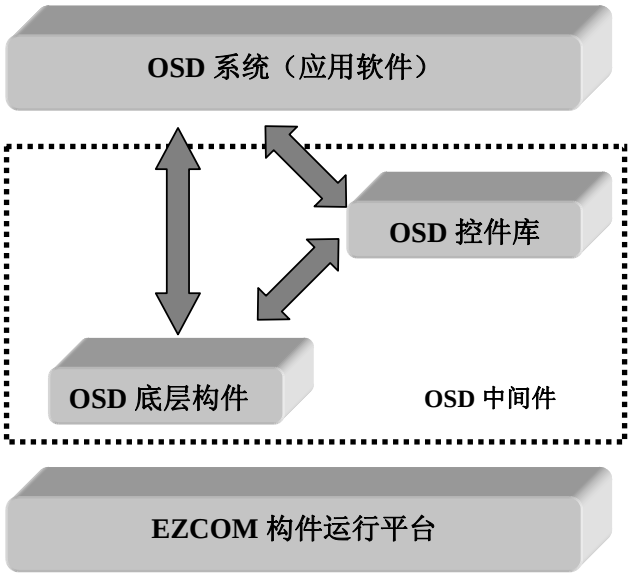

图4.3 OSD中间件系统结构图

由图 4.3 可见，OSD 中间件由两部分组成，分别是底层构件库和 OSD 控件库。底层构件库负责各类基础寄存器读写，RAM 操作；底层构件提供的功能既可以由 OSD 应用直接调用，又为 OSD 控件的实现提供支持。

OSD 控件库提供了很多控件，以方便用户构建 OSD 菜单。譬如位图（BITMAP）、字符（TEXT）、框架（FRAME）、进度条（SLIDER）等等控件。控件集是构建在 OSD 底层构件之上，它们的实现使用了 OSD 底层构件提供的基本的寄存器和 RAM 操作。不过在编写简单的 OSD 应用时，控件库并不是必需的，如果用户不需要使

用控件来搭建他们的 OSD 菜单，那么就不需要使用控件集。

下面对底层构件以及控件库，进行具体描述。

4.2.3 OSD 底层构件库

OSD 底层构件负责各类特殊寄存器的读写，XRAM 操作，同时提供 DRCS 构造功能。所以其实现，主要是由 3 个构件组成：寄存器构件，XRAM 构件，DRCS 构件。

4.2.3.1 寄存器构件

寄存器构件用于寄存器的读写操作。

以下是 EZOSDReg 构件的 CDL 内容，定义了寄存器构件中的类、接口以及接口方法等信息。

```
[
    uuid(e4ee9b0c-d93b-40ee-834f-4ba16542fd80),
    urn(http://www.koretide.com/ezcom/EZOSDReg.dll)
]
component EZOSDReg
{
    //接口 IOSDReg
    [uuid(078ebd60-fd5e-48f8-ac91-31f0951ee3d1)]
interface IOSDReg {
   HRESULT RegInit();
         HRESULT SetRegBit([in]int regAddress,[in]BYTE bValue);
         HRESULT SetRegByte([in]int regAddress,[in]BYTE BValue);
         HRESULT SetRegWord([in]int regAddress,[in]WORD WValue);
         HRESULT SetRegDWord([in]int regAddress,[in]DWORD DWValue);
         HRESULT GetRegBit([in]int regAddress,[out]BYTE* bValue);
         HRESULT GetRegByte([in]int regAddress,[out]BYTE* BValue);
         HRESULT GetRegWord([in]int regAddress,[out]WORD* WValue);
         HRESULT GetRegDWord([in]int regAddress,[in]DWORD* DWValue);
```

```
    }
    //类 COSDReg
    [uuid(e945a3fe-a939-4612-a621-0ce810f415b1)]
    class COSDReg{
        interface IOSDReg;
    }
}
```

- **构件**

在 CDL 中使用 component 关键字定义构件，在一个 CDL 中只能有一个构件的定义。定义如下：

```
//构件 EZOSDReg 的定义
component EZOSDReg
{
    ……
}
```

- **构件属性**

构件属性对构件起修饰作用，其中 uuid 属性和 urn 属性是必须的。构件属性中的 uuid 称为构件 ID，它用于唯一标识一个构件；urn 指定构件所存在的 dll 文件在 Internet 上的位置。

```
[
    uuid(e4ee9b0c-d93b-40ee-834f-4ba16542fd80),
    urn(http://www.koretide.com/ezcom/EZOSDReg.dll)
]
```

- **构件定义**

构件定义中主要包括接口的定义和类的定义，还可以包括其它一些类型定义。在 EZOSDReg 构件定义部分中定义了一个接口：IOSDReg；一个类：COSDReg。

- **接口：**在 CDL 中使用 interface 关键字定义接口。具体如下：

```
//接口 IOSDReg 定义
interface IOSDReg {
    ……
    ……
}
```

- **接口属性：**

接口属性对接口起修饰作用,其中的uuid属性是必须的。接口属性中的uuid称为接口 ID（IID），它用于唯一标识一个接口。

```
//接口 IOSDReg 属性
[uuid(078ebd60-fd5e-48f8-ac91-31f0951ee3d1)]
```

- **接口定义**

接口定义主要包括接口方法的声明。在 CDL 文件中最重要的当属接口的定义。因为在构件中，能够与外界打交道的只有接口。一个没有接口方法的接口在现实中没有意义。可以使用 ezCOM 提供智能指针完成对接口方法的操作。

IOSDReg 接口中的方法声明为：

```
interface IOSDReg {
    HRESULT RegInit();
    HRESULT SetRegBit([in]int regAddress,[in]BYTE bValue);
    HRESULT SetRegByte([in]int regAddress,[in]BYTE BValue);
    HRESULT SetRegWord([in]int regAddress,[in]WORD WValue);
    HRESULT SetRegDWord([in]int regAddress,[in]DWORD DWValue);
    HRESULT GetRegBit([in]int regAddress,[out]BYTE* bValue);
    HRESULT GetRegByte([in]int regAddress,[out]BYTE* BValue);
    HRESULT GetRegWord([in]int regAddress,[out]WORD* WValue);
```

```
        HRESULT GetRegDWord([in]int regAddress,[in]DWORD* DWValue);
    }
```

接口中的方法定义必须遵守如下规定：

a) 接口方法必须使用 HRESULT 作为返回值；

b) 接口方法参数必须显式指明接口方法的参数是输入参数（in）或输出参数（out）或作为返回值使用（retval）。

从接口定义可以看到，寄存器构件的主要接口方法，包括按地址对寄存器实施按位（BIT），按字节(BYTE)，按字(WORD)，按双字(DWORD)读写的操作,同时提供一个 RegInit()方法用于寄存器重置。

- **类：**在 CDL 中使用 class 关键字定义类

```
    class COSDReg{
    ……
    }
```

在 CDL 文件中，类的定义非常重要。与 C++中对象的概念相似，客户程序与 ezCOM 构件进行交互的实体是类对象，所以必须要有类定义的存在。

- **类的属性**

类属性对类起修饰作用，其中的 uuid 属性是必须的。类属性中的 uuid 称为类 ID（CLSID），用来唯一标识一个类。

```
    //类 COSDReg
    [uuid(e945a3fe-a939-4612-a621-0ce810f415b1)]
```

- **类的定义**

类定义中主要包括接口的声明。表示该类对象提供的接口。

```
class COSDReg{
        interface IOSDReg;
}
```

4.2.3.2 XRAM 构件

XRAM 构件用于存储器的读写操作。

以下是 EZOSDXram 构件的 CDL 内容，定义了 XRAM 构件中的类、接口以及接口方法等信息。

```
[
    uuid(aea76946-6a2c-4f8f-ada6-d44a2b7db7d6),
    urn(http://www.koretide.com/ezcom/EZOSDXram.dll)
]
component EZOSDXram
{
    //接口 IOSDXram 属性
[uuid(f7481d8c-45bc-4de4-b32a-ca3790e0d076)]
    //接口 IOSDXram 定义
interface IOSDXram{
   //接口 IOSDXram 方法,包括 Xram 的按位置读以及写操作，读写要求指定读写的字节数。
       HRESULT  SetXramValue([in]int  XramAddress,[in]int  iByteCount,
[in]EzVar bValue);
       HRESULT  GetXramValue([in]int  XramAddress,[in]int  iByteCount,
[out]EzVar * DWValue);
   }
    //类 COSDXram 属性
[uuid(b1cd2257-d98b-493c-9689-2632400b713a)]
    //类 COSDXram 定义
```

```
    class COSDXram{
        interface IOSDXram;
    }
}
```

4.2.3.3 DRCS 构件

DRCS 构件用于 DRCS 的构造操作。

以下是 EZOSDDrcs 构件的 CDL 内容，定义了 DRCS 构件中的类、接口以及接口方法等信息。

```
[
    uuid(3b04568e-6bcc-43a1-9e25-6c389837d28e),
    urn(http://www.koretide.com/ezcom/EZOSDDrcs.dll)
]
component EZOSDDrcs
{
    //接口 IOSDDrcs 属性
    [uuid(fac60254-54a7-40c2-9c3c-00750fb2fcf7)]
    //接口 IOSDDrcs 定义
    interface IOSDDrcs{
        //接口 IOSDDrcs 方法,目前包括了从位图、文字、进度条、数字和边框构造 Drcs 的方法。
        HRESULT ConvertBitmap([in] EzStr bitmapFileName,[in]int drcsType,[out]EzVar * drcsPixMatrix, [out]EzVar * drcsColorMap);
        HRESULT ConvertText([in] EzStr textStr,[in]int drcsType,[out]EzVar * drcsPixMatrix);
        HRESULT ConvertSlider([in] EzStr sliderName,[in]int drcsType,[out]EzVar * drcsPixMatrix, [out]EzVar * drcsColorMap);
        HRESULT ConvertNumber([in] EzStr numberStr,[in]int drcsType,[out]EzVar * drcsPixMatrix);
```

```
        HRESULT      ConvertFrame([in]      EzStr      frameName,[in]int
drcsType,[out]EzVar * drcsPixMatrix, [out] EzVar * drcsColorMap);
    }
    //类 COSDDrcs 属性
    [uuid(ed75a861-36d9-443a-b295-7a76e8131c13)]
    //类 COSDDrcs 定义
    class COSDDrcs{
        interface IOSDDrcs;
    }
}
```

4.2.4 OSD 控件库

OSD 控件库提供了很多控件，以方便用户构建 OSD 菜单。譬如位图（BITMAP）、字符（TEXT）、框架（FRAME）、进度条（SLIDER）等等控件。控件集是构建在 OSD 底层构件之上，它们的实现使用了 OSD 底层构件提供的基本的寄存器和 RAM 操作。

下面将详细列举 OSD 控件库中最主要的 3 个控件：OSDBITMAP, OSDTEXT, OSDSLIDER。

4.2.4.1 OSDBITMAP 控件

OSDBITMAP 控件主要负责控制 OSD 菜单中位图的绘制。

以下是 EZOSDBitmapCtl 构件的 CDL 内容，定义了 OSDBITMAP 构件中的类、接口以及接口方法等信息。

```
[
    uuid(7ddb5051-aa7f-4fe1-8d06-4acf592fff9f),
    urn(http://www.koretide.com/ezcom/EZOSDBitmapCtl.dll)
]
component EZOSDBitmapCtl
{
```

```
    //接口 IOSDBitmapCtl 属性
    [uuid(e46d7d58-d27b-49e3-a68d-e1ebd46f6c19)]
    //接口 IOSDBitmapCtl 定义
    interface IOSDBitmapCtl{
        //接口 IOSDBitmapCtl 方法,包括对 Bitmap 的绘制、颜色获取、点阵信息修改等等操作。
    //从(xposition,yposition)处开始绘制位图，图像大小由 width 和 height 两个参数决定。
    HRESULT DrawBitmap([in] EzStr bitmapFileName, [in]int drcsType, [in]int Xposition, [in]int Yposition，[in]int width, [in]int height);
    //擦除位图，并且使用指定颜色覆盖
    HRESULT EraseBitmap([in]int Xposition, [in]int Yposition，[in]int width, [in]int height, [in]UINT Color);
    //获取位图宽度
    HRESULT GetBitmapWidth([out]int *pWidth);
    //获取位图高度
    HRESULT GetBitmapHeight([out]int *pHeight);
    //获取位图颜色数量
    HRESULT GetBitmapColorCount([out]int *pColorCount);
    //获取位图中指定像素的颜色。(x: 要检索的像素的横坐标, y: 要检索的像素的纵坐标，pColor: 指向要获取的像素颜色的指针).
    HRESULT GetBitmapPixel([in] int x, [in] int y,[out] UINT *pColor);
    //设置位图中指定像素的颜色。(x: 位图中指定像素的横坐标, y: 位图中指定像素的纵坐标，pColor: 指定给像素的颜色).
    HRESULT SetBitmapPixel([in] int x, [in] int y,[in]UINT Color);}
    //类 COSDBitmapCtl 属性
    [uuid(2fc093d3-f77b-4bb0-8b7c-ac44a9532f43)]
    //类 COSDBitmapCtl 定义
    class COSDBitmapCtl{
        interface IOSDBitmapCtl;
    }
```

```
}
```

从 OSDBITMAP 控件的 CDL 文件可以看到，OSDBITMAP 控件通过使用接口方法 DrawBitmap([in] EzStr bitmapFileName, [in]int drcsType, [in]int Xposition, [in]int Yposition，[in]int width, [in]int height)绘制位图至 OSD 菜单，并且可以通过参数调节位图位置和大小，位图的颜色以及点阵信息，是通过调用 OSD 基础控件的 DRCS 控件获取的。调用 EraseBitmap（）方法可以擦除所绘制的位图，并且可以指定颜色来覆盖原来位图的位置。OSDBITMAP 控件又可以读写各类关于位图的信息（包括宽度，高度，位图的点阵等）。使用 OSDBITMAP 控件,在绘制位图的时候，不用关心内部的 DRCS 构造，XRAM 分配等问题，直接调用 DrawBitmap()方法就可以获得需要的效果，大大减少了手动编写的程序量。

4.2.4.2 OSDTEXT 控件

OSDTEXT 控件主要负责控制 OSD 菜单中字符串的绘制。

以下是 EZOSDTextCtl 构件的 CDL 内容，定义了 OSDTEXT 构件中的类、接口以及接口方法等信息。

```
[
    uuid(2e9e9d8a-438d-4f69-986f-b2eb19064cc4),
    urn(http://www.koretide.com/ezcom/EZOSDTextCtl.dll)
]
component EZOSDTextCtl
{
    //接口 IOSDTextCtl 属性
    [uuid(4221e214-adbf-4320-bd12-ec245a11c678)]
    //接口 IOSDTextCtl 定义
    interface IOSDTextCtl{
```

```
    //接口 IOSDTextCtl 方法,包括对 Text 的绘制、颜色设置、字符大小等等操作。
    //从(xposition,yposition)处开始绘制字符串，并由参数决定字符大小。
    HRESULT DrawText([in] EzStr textStr,[in]int drcsType [in]int Xposition, [in]int Yposition，[in]int textSize，[in]UINT textColor);
    //擦除字符，并且使用指定颜色覆盖
    HRESULT EraseText([in]int Xposition, [in]int Yposition，[in]int width, [in]int height, [in]UINT Color);
    //获取字符大小，字符大小一共有 3 类，小，中，大，分别是小字符变大一倍成为中字符，中字符变大一倍成为大字符。
    HRESULT GetTextSize([out]int *pTextSize);
    //设置字符大小，使得字符具有 double 等效果。
    HRESULT SetTextSize([in]int * textSize);
    //获取字符颜色
    HRESULT GetTextColor([out]UINT *pColor);
    //设置字符颜色
    HRESULT SetTextColor([in]UINT *pColor);
    }
    //类 COSDTextCtl 属性
    [uuid(f27016cb-f7b1-43e4-9e65-5ddead953752)]
    //类 COSDTextCtl 定义
    class COSDTextCtl{
        interface IOSDTextCtl;
    }
}
```

OSDTEXT 控件通过使用接口方法 DrawText([in] EzStr textStr,[in]int drcsType [in]int Xposition, [in]int Yposition,[in]int textSize,[in]UINT textColor)绘制字符至 OSD 菜单，并且可以通过参数调节字符位置、大小以及字符的颜色，字符的点阵信息，是通过调用 OSD 基础控件的 DRCS 控件获取的。调用 EraseText（）方法可以擦除所绘制的字符，并且可以指定颜色来覆盖原来字符的位置。OSDTEXT 控件同时提供控制字符颜色、大小等方法。

4.2.4.3 OSDSLIDER 控件

OSDSLIDER 控件主要负责控制 OSD 菜单中进度条的绘制。

以下是 EZOSDSliderCtl 构件的 CDL 内容，定义了 OSDTEXT 构件中的类、接口以及接口方法等信息。

```
[
    uuid(c07d36d7-ec8a-4d03-a6bd-efff16fb11c6),
    urn(http://www.koretide.com/ezcom/EZOSDSliderCtl.dll)
]
component EZOSDSliderCtl
{
    //接口 IOSDSliderCtl 属性
    [uuid(ff1170d6-5155-4d2b-8c99-a9d5b075a35c)]
    //接口 IOSDSliderCtl 定义
    interface IOSDSliderCtl{
    //接口 IOSDSliderCtl 方法,包括对 Slider 的绘制、颜色设置、当前值设置等等操作。
    //从(xposition,yposition)处开始绘制进度条，由参数决定进度条长度以及当前进度条的值。
    HRESULT DrawSlider([in] EzStr sliderName,[in]int drcsType,[in]int Xposition, [in]int Yposition , [in]int sliderLength , [in]int currentValue);
    //擦除进度条，并且使用指定颜色覆盖
    HRESULT EraseSlider([in]int Xposition, [in]int Yposition, [in]int width, [in]int height, [in]UINT Color);
    //获取进度条当前的值。
    HRESULT GetSliderValue([out]int *pSliderValue);
    //设置进度条的值。
    HRESULT SetSliderValue([in]int * sliderValue);
```

```
    //获取进度条颜色数量
    HRESULT GetColorCount([out]int *pColorCount);
    //获取进度条中指定像素的颜色。(x: 要检索的像素的横坐标, y: 要检索的像素的纵坐标，pColor: 指向要获取的像素颜色的指针).
    HRESULT GetSliderPixel([in] int x, [in] int y,[out] UINT *pColor);
    //设置进度条中指定像素的颜色。(x: 位图中指定像素的横坐标, y: 位图中指定像素的纵坐标，pColor: 指定给像素的颜色).
    HRESULT SetSliderPixel([in] int x, [in] int y,[in]UINT Color);
    }
    //类 COSDSliderCtl 属性
    [uuid(abae2b48-77e5-48a7-9fe9-4cfb2187be25)]
    //类 COSDSliderCtl 定义
    class COSDSliderCtl{
        interface IOSDSliderCtl;
    }
}
```

OSDSLIDER 控件通过使用接口方法 DrawSlider([in] EzStr sliderName,[in]int drcsType,[in]int Xposition, [in]int Yposition，[in]int sliderLength，[in]int currentValue)绘制进度条至 OSD 菜单，并且可以通过参数调节进度条位置、长度以及当前值，进度条的点阵信息，是通过调用 OSD 基础控件的 DRCS 控件获取的。调用 EraseSlider（）方法可以擦除所绘制的进度条，并且可以指定颜色来覆盖原来进度条的位置。进度条使用最多的就是设置和提取当前进度条的值，OSDSLIDER 控件通过 GetSliderValue（）和 SetSliderValue()非常方便地就能改变当前进度条的值。同时 OSDSLIDER 也提供颜色控制等方法。

4.2.5 OSD 中间件开发过程

4.2.5.1 编写 CDL 文件生成构件源程序框架并填写实现代码

开发 ezCOM 构件的第一步是编写 CDL 文件。以上已经详细介绍了 OSD 中间件的构件 CDL 结构以及由其定义的构件中的类、接口、方法及其参数。

在编写完 CDL 文件后，在“和欣”SDK 开发环境下，使用 emake 工具可以生成构件源程序框架。这将减少程序员的输入量，并且可以有效避免此环节的拼写错误。具体用法是执行下面的语句：

```
emake <cdlfile>
```

其中<cdlfile>为 CDL 文件路径名。执行该命令后，将在当前目录生成相应的头文件、cpp 文件和 sources 文件，其中头文件和 cpp 文件为程序框架文件。sources 文件（无扩展名）是位于编译目录下的一个文本文件，它为编译当前目录设置了宏定义，包括当前目录的源程序文件，产生的目标文件类型、目标文件名称等等。还有一些关于编译器和链接器的 FLAG 的宏定义，所有这些宏控制着编译器和链接器的行为。头文件和 cpp 文件的文件名由 CDL 文件中定义的类名指定。当 CDL 文件中定义了多个类时，将生成对应的多个头文件和.cpp 文件。

在“和欣”SDK 中执行如下语句，将生成前文介绍的 EZOSDDrcs 构件的源程序框架：

```
emake ezOSDdrcs.cdl
```

这条语句将生成将 COSDDrcs.h 文件、COSDDrcs.cpp 文件和 sources 文件。sources 文件生成后一般不需要修改，下面是生成的 sources 文件的内容：

```
TARGET_NAME= ezOSDdrcs
TARGET_TYPE= dll

SOURCES= \
    ezOSDdrcs.cdl \
    COSDDrcs.cpp \

ELASTOS_LIBS = \
    elastos.lib \
    elacrt.lib \
```

TARGET_NAME 指定生成构件的名字，此名字必须与 CDL 文件名相同，否则不能通过编译。生成构件的类型默认为 dll，由 TARGET_TYPE 指定。

另外 COSDDrcs.h 的内容如下，这个文件一般不需要修改：

```
#if _MSC_VER > 1000
#pragma once
#endif

#if !defined(_COSDDRCS_H_)
#define _COSDDRCS_H_

#include "_COSDDrcs.h"

class COSDDrcs : public _COSDDrcs
{
public:
    STDMETHODIMP ConvertBitmap(
        /* [in] */ EzStr bitmapFileName,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix,
        /* [out] */ EzVar * drcsColorMap);

    STDMETHODIMP ConvertText(
        /* [in] */ EzStr textStr,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix);

    STDMETHODIMP ConvertSlider(
        /* [in] */ EzStr sliderName,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix,
        /* [out] */ EzVar * drcsColorMap);

    STDMETHODIMP ConvertNumber(
```

```
        /* [in] */ EzStr numberStr,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix);

    STDMETHODIMP ConvertFrame(
        /* [in] */ EzStr frameName,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix,
        /* [out] */ EzVar * drcsColorMap);

private:
    // TODO: Add your private member variables here
};
#endif //!(_COSDDRCS_H_)
```

程序自动生成的框架，最重要的、也是需要手动填写实现代码的为 COSDDrcs.cpp 文件，其内容如下：

```
#include "COSDDrcs.h"
#include "_COSDDrcs.cpp"

DECLARE_CLASSFACTORY(COSDDrcs)

HRESULT COSDDrcs::ConvertBitmap(
        /* [in] */ EzStr bitmapFileName,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix,
        /* [out] */ EzVar * drcsColorMap)
{
    // TODO: Add your code here
    return E_NOTIMPL;
}
```

```
HRESULT COSDDrcs::ConvertText(
        /* [in] */ EzStr textStr,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix)
{
    // TODO: Add your code here
    return E_NOTIMPL;
}

HRESULT COSDDrcs::ConvertSlider(
        /* [in] */ EzStr sliderName,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix,
        /* [out] */ EzVar * drcsColorMap)
{
    // TODO: Add your code here
    return E_NOTIMPL;
}

HRESULT COSDDrcs::ConvertNumber(
        /* [in] */ EzStr numberStr,
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix)
{
    // TODO: Add your code here
    return E_NOTIMPL;
}

HRESULT COSDDrcs::ConvertFrame(
        /* [in] */ EzStr frameName,
```

```
        /* [in] */ int drcsType,
        /* [out] */ EzVar * drcsPixMatrix,
        /* [out] */ EzVar * drcsColorMap)
{
    // TODO: Add your code here
    return E_NOTIMPL;
}
```

这样，在相应的// TODO: Add your code here 处，添加相应的实现代码，就完成了构件的编写工作。

4.2.5.2 编译生成构件以及构件的注册

在填写完实现代码后，使用 emake 工具，编译源代码即可在镜像目录生成 ezCOM 构件。与生成源程序框架不同的是，这次 emake 命令不需带参数。在上面的 EZOSDDrcs 构件的源代码框架中，填入实现代码后，执行 emak 就能在相应的 bin 目录下生成 ezOSDdrcs.dll 文件，这个就是最终生成 EZCOM 构件。

在 Windows 2000 端，当客户程序创建构件中某个类的对象时，需要通过 CLSID 找到包含该类的构件（.dll）的位置，并由系统加载到内存中。该 CLSID 的信息和构件位置的信息保存在注册表中。当自己编写构件时，在编译构件的过程中，emake 工具会自动将该构件注册到注册表中。当从其它的途径（如购买）获得并需要使用一个 ezCOM 构件后，就需要注册该构件。在 Windows 2000 端，可以使用“和欣”SDK 提供的 regcom 工具注册构件。例如：注册 EZOSDDrcs 构件，则可在“和欣”开发环境中执行以下命令：

```
regcom /l EZOSDDrcs.dll
```

即可在 Windows 2000 下注册 EZOSDDrcs 构件，完成以上操作，就开发了一个完整的 ezCOM 构件。

4.2.5.3 ezCOM 构件的使用方法

在 ezCOM 构件注册到操作系统中后，就可以使用该构件了。下面通过 EZOSDDrcs 构件的客户程序代码来说明使用方法：

```
#import <EZOSDDrcs.dll>

int __cdecl main()
{
    EzVar  drcsPixMatrix;
    EzVar  drcsColorMap;
    COSDDrcsRef cOSDDrcs;    //创建类智能指针
    HRESULT hr = cOSDDrcs.Instantiate();  //创建类 CHello 的对象
    if (!cOSDDrcs.IsValid()) {
        assert(0 && "Can't create cOSDDrcs");
        return 1;
    }
    hr      =      cOSDDrcs.ConvertBitmap(EZCSTR("TVOSDAudioPic"),4,
&drcsPixMatrix, &drcsColorMap);     //使用类智能指针 cOSDDrcs 调用
ConvertBitmap()方法
    if (FAILED(hr)) {
        return 1;
    }
    return 0;
}
```

在程序中使用 import 语句引用构件，例如：

```
#import <EZOSDDrcs.dll>
```

智能指针（Smart Pointer）的使用方法:上面的构件使用方法中，参数 cOSDDrcs 为类 COSDDrcs 的类智能指针。当使用 import 引用了构件 hello 时，就可以使用类 COSDDrcs 提供的智能指针。通过类智能指针 cOSDDrcs，程序可以调用类 COSDDrcs 对象的所有接口中的方法，例如：

```
    hr      =      cOSDDrcs.ConvertBitmap(EZCSTR("TVOSDAudioPic"),4,
&drcsPixMatrix, &drcsColorMap);     //使用类智能指针 cOSDDrcs 调用
ConvertBitmap()方法
```

类智能指针cOSDDrcs使用点操作符“.”调用接口IOSDDrcs中的方法 ConvertBitmap()。

从CDL的编写，到自动生成程序框架，再到填写实现代码，编译生成构件并且注册，最后到使用，再整个过程中，都体现了基于EZCOM构件技术的OSD中间件开发过程相当简单易用，可以简单快速地开发出自己的中间件。

4.3 优势分析

“和欣”技术体系所包括的 ezCOM 构件技术、构件运行平台技术开发应用软件所需的集成开发环境，是一个完整的面向构件的应用软件开发平台，可以让熟悉 C/C++的程序员很容易地掌握面向构件的编程技术。OSD 中间件完全基于 EZCOM 技术实现，它能够高效地运行在科泰世纪的″“和欣”″ 操作系统上，为 OSD 程序提供一个通用的软件平台，为用户提供一个完整的中间件解决方案。

采用 OSD 中间件的 OSD 开发，用户程序构建在 OSD 中间件之上，能够支持丰富的应用。采用 OSD 中间件系统进行开发，隔离了很多具体细节的实现，用简单的方法定制具有自己特色的 OSD 软件，从而在提高开发效率、减少开发成本的同时能够跟上技术的发展，将应用的开发变得更加简捷，使产品的开放性和可移植性更强。

在基于 SDA55XX 的 OSD 开发中，所有模块的实现，都要完全依赖具体的实现，主要精力都放在了 XRAM 分配以及特殊寄存器赋值等任务上了，而这些任务，都需要非常烦琐和细致的工作。同时其对于硬件的依赖性强，基于 sda55xx 芯片开发的 OSD 程序，只能在采用 SDA55XX 芯片数字电视中使用，对于服务商和制造商而言，致力于多种产品的开发，往往希望开发的软件产品能够跨平台运行，这种开发模式中，这显然是很难办到的。而架构在“和欣”嵌入式操作系统上的，通过 OSD 中间件的 OSD 开发，芯片相关部分被操作系统隔离，用户开发提升到以 ezcom 构件技术开发的 OSD 中间件之上，显然，只要 OSD 中间件不变，用户开发的软件产品，对于具体硬件的依赖程度是很低的。

由于嵌入式系统有着体积小、功能集中、可靠性高等优点，如何缩短嵌入式系统的开发周期，降低开发成本，以及提高产品的可靠性已成为嵌入式行业普遍关注的问题。在基于 SDA55XX 的 OSD 开发模式中，软件开发的主要工作量，都放在了详细分析 SDA55XX 的具体 XRAM 分配，寄存器赋值等任务上，对于开发高效，健壮的 OSD 系统，快速投入应用是相当不利的。反之，架构在“和欣”

嵌入式操作系统上的，通过 OSD 中间件的 OSD 开发，所有以前的积累都可以通过快速修改应用到新的开发任务中，这比重新开发，不仅速度上是一个提升，特别是对于最终产品的健壮性，都有非常大的提升作用，这是因为，很多以前开发的软件产品，在多次应用之中不断得到强壮的结果。这对于开发速度和开发效率，是非常有利的。　在基于 SDA55XX 的开发模式下，由于没有统一的平台，采用不同芯片的数字电视方案的都有各自的实现，技术不完全开放，造成了诸如一些关键寄存器的地址、设置等原本简单的技术问题被少数人"垄断"的局面，整机厂家不得不高度依赖于国外厂家的技术支持或企业内个别技术人员，而这些资源的变动常常会影响整个产品开发。这对于企业很难形成技术积累，稳步实现技术发展。而“和欣”操作系统，为软件开发提供了一个统一的平台，使得应用程序不再依赖各个不同的硬件平台而正常的工作，操作系统使得具体芯片相对于应用程序而言，成为了透明。另外，基于特定芯片开发的软件模块，接口都具有很大的差异性，而在数字电视业务的不断拓展过程中，指定统一的应用程序接口成为大势所趋，这无疑是其弊端之一。基于 OSD 中间件的 OSD 开发，所有接口都由中间件统一定义（当然，定制统一的 OSD 中间件接口，也是 OSD 中间件开发的主要任务之一），统一的接口，统一的平台，这都是摒除手工作坊式软件开发的有效手段和前提。

4.4 小结

本节主要叙述了基于嵌入式操作系统 OSD 开发模式实现方法，详细讨论了 OSD 应用程序－OSD 中间件－“和欣”构件运行平台－“和欣”操作系统－硬件平台这一系统架构下的 OSD 开发模式，包括主要定义了 OSD 中间件的主要结构以及中间件中主要构件的结构，并且讨论了使用这些构件进行开发的方法。最后通过对比两种中开发模式，详细阐述了基于嵌入式操作系统 OSD 开发模式的巨大优势。

第 5 章 结论与展望

5.1 研究结果

本文结合 863 课题“基于构件、中间件技术的因特网操作系统及跨操作系统的构件、中间件运行平台”，展开了对基于“和欣”操作系统实现构件化的 OSD 软件的开发研究。研究了在“和欣”嵌入式操作系统上，通过开发基于 ezCOM 的 OSD 中间件，最终形成 OSD 应用程序<—>OSD 中间件<—>“和欣”构件运行平台<—>“和欣”操作系统<—>硬件平台 这一系统架构下的 OSD 开发模式，并且实现了一个完全构件化的系统架构。

本文首先分析了基于芯片 SDA55XX 的 OSD 开发模式，通过对比这一模式，引出了对新的开发模式的研究，然后在运用“和欣”操作系统、ezCOM 技术以及“和欣”构件运行平台的基础上，提出了基于“和欣”操作系统的 OSD 开发模式，本开发模式的核心是 OSD 中间件的定义以及实现。文中详细阐述了 OSD 中间件与上层 OSD 应用程序以及下层的构件运行平台，“和欣”操作系统以及硬件平台的关系。采用 OSD 中间件系统进行开发，隔离了很多具体细节的实现，用简单的方法定制具有自己特色的 OSD 软件，从而在提高开发效率、减少开发成本的同时能够跟上技术的发展，将应用的开发变得更加简捷，使得产品的开放性和可移植性更强。

5.2 研究展望

数字电视的发展的脚步正在不断地加快，2005 年我国将进行数字电视的商业播出，2008 年用数字电视转播奥运会，2015 年停止模拟电视的播放，全面推行数字电视，这一时间表，为像我国这样拥有 3 亿多电视用户地国家来说，是巨大的发展机会。

数字电视的发展，意味着数字电视软件开发也必须跟上数字电视的发展普及速度，而软件开发方式的优劣，很大程度上决定了软件开发的效率和质量。传统的手工作坊式的软件开发，应用到数字电视的开发中，显然是不能适应其

快速发展及其复杂的需求的。

本文研究的基于和欣嵌入式操作系统的 OSD 开发模式，无论是在软件复用、系统可升级性、软件可靠性、容错性方面都拥有大的优势，ezCOM 技术提供了构件的标准，二进制构件可以被不同的上层应用程序使用，使软件构件真正能够成为"工业零件"。充分利用"久经考验"的软件零件，避免重复性开发，能有效提高软件生产效率和软件产品质量。

这一开发模式，完全可以应用到数字电视的其他开发中去，直到达成在数字电视开发中实现软件工厂化生产，这将需要更多艰苦卓绝的研究和实践。

致 谢

参考文献

[1] 科泰世纪科技有限公司 和欣资料大全 2003
[2] 潘爱民 《COM 原理和应用》 清华大学出版社 1999
[3] Bruce Eckel 《Thinking in C++》 McGraw Hill 2000
[4] 贾育 基于构件开发方法的概念、目标和意义 http://www.csdn.net/develop/read_article.asp?id=15398
[5] Joan Daemen 谷大武、徐胜波译 《The Design of Rijindael AES:The Advanced Encryption Standard》 清华大学出版社 2003 年 4 月
[6] Michael Welschenbach 赵振江译《Kryptographie in C and C++ Zweite,uberarbeitete and erweiterte Auflage》 电子工业出版社 2003 年 7 月
[7] William Stallings 魏迎梅、王涌译 《操作系统——内核与设计原理》（第四版） 2001 年 6 月
[8] Don Box:《Effective COM》, 2002

个人简历　在读期间发表的学术论文与研究成果

个人简历：

谢鑫君，男，1980 年 1 月生。

2002 年 7 月毕业于同济大学计算机系 获学士学位。

2002 年 9 月入同济大学读硕士研究生。

已发表论文：

[1]