江南大学

硕士学位论文

基于S3C44B0X的JPEG图像解码及LCD显示的实现

姓名：蒙智明

申请学位级别：硕士

专业：检测技术与自动化装置

指导教师：屈百达

20070601

摘 要

题目： 基于 S3C44B0X 的 JPEG 图像解码及 LCD 显示的实现

专业： 检测技术与自动化装置

研究生：蒙智明

导师： 屈百达 教授

JPEG标准是一项成熟的图像编码技术，其压缩比例高，失真小，运行速度快，易于实现且适用范围广。已于近年被列为国际标准，得到了广泛的推广和应用。同时随着嵌入式技术的发展，该技术也越来越多的用在多媒体手机、数码相机、高性能扫描仪等多媒体系统中。目前，JPEG编解码多采用专用集成电路来实现，但这种方法成本高，升级拓展困难。本文提出一种新的解决方案，基于高性能的ARM处理器，采用优化算法实现JPEG编解码功能，通过软件升级来实现系统更新和功能拓展。这种方案具有高度的灵活性，较强的适应性且成本较低。

该方案充分利用 ARM 的硬件资源，采用软件编程方法实现 JPEG 图像解码和显示。首先，基于 SAMSUNG 公司的 ARM 处理器 S3C44B0X 及其外围电路构建 JPEG 图像解码和显示的硬件处理平台，编制并调试了相关引导程序、底层驱动程序、解码程序和显示模块程序，为实现 JPEG 图像解码做好软硬件准备。其次，根据 JPEG 标准，编写 JPEG 解码程序。解码过程主要由五个部分组成：标记码信息处理、熵解码、反量化、反离散余弦变换和色彩空间转换。其中对最耗时的 IDCT 部分采用了采用行列分解法，使处理效率大大提高，该算法先将 8*8 的二维 IDCT 转换成 8 点的一维 IDCT，再利用一维快速算法来实现。然后，设计了解码显示系统，使解码图像显示于 LCD 终端。最后，从 S3C44B0X 的硬件特性和 C 语言程序结构方面对系统的解码和显示程序做了进一步优化。

解码速度和图像恢复质量的测试结果表明：解码程序执行效率较高，能满足实时性要求；图像还原质量良好，具有较高的峰值信噪比。该方案可应用于数码相机、数字手持设备等多媒体系统中的静态图像处理，对于M-JPEG、MPEG、MPEG-II、H.26x的研究也有较高的参考价值。

关键词：JPEG，ARM，解码，LCD，S3C44B0X

ABSTRACT

Topic: The Implementation of JPEG Image Decoding and LCD Display Based on S3C44B0X
Major: Checking Technology and Automation Device
Candidate: Meng Zhiming
Supervisor: Prof. Qu Baida

JPEG standard is a mature image coding technology, which is featured with high compress rate, little distortion rate and high run-time speed. And what's more, it is convenient to realize. Adapted as an international standard, it has been applied in many fields. With the development of embedded technology, JPEG image codec technique has been used in many multimedia systems, such as multimedia mobile phone, digital camera, and palmtop high-performance scanner etc. At present, the special ICs are employed to realize JPEG image codec. But its cost is high and it is difficult to upgrade and expand. In this paper, JPEG image codec is realized by employing optimization algorithms based on high-performance ARM processor. It makes the system upgrade and expand easily; it has a higher flexibility, adaptability and lower cost simultaneously.

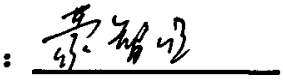
The implementation of JPEG image decoding and display is mainly based on software programming in my scheme, which makes full use of ARM hardware platform. Firstly, A hardware processing platform is constructed on the basis of SAMSUNG's S3C44B0X ARM processor, furthermore, bootloader code, bottom driver and display module program have been developed on this processor, which facilitates the software and hardware for realizing the embedded image processing system. Secondly, according to JPEG standard, decoding programs are designed. The process of decoding consists of five parts: tag code information processing, entropy decoding, inversed quantization, inversed discrete cosine transform and color space converter. As to IDCT which needs a lot of time in the process of decoding, a method named row-column decomposition is adapted, which decompounds the 8*8 two-dimensional into rows and columns with 1-D. then proceeds with fast algorithm of 1-D. The efficiency of IDCT programming could increase greatly. Thirdly, display processing system is developed to perform display on LCD terminal. Finally, the further optimization has been done for the decoding program on the aspects of hardware characteristic of S3C44B0X and C program structure.

A lot of tests have been done on the decoding speed and the picture quality. The results show that the program can perform efficiently and is enough to meet the real-time qualification. The quality of reverting pictures is excellent with a higher S/N ratio. It makes the scheme apply to the static image processor easily and widely in the digital cameras, PDA etc. In additional, it has an important reference value to the R&D of M-JPEG, MPEG, MPEG-II, and H.26x.

Keywords: JPEG; ARM; Decoding; LCD; S3C44B0X

独 创 性 声 明

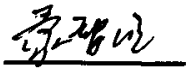
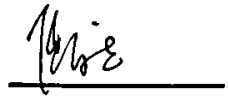
本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含本人为获得江南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：  日期：2007年 6月8日

关于论文使用授权的说明

本学位论文作者完全了解江南大学有关保留、使用学位论文的规定：江南大学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，允许论文被查阅和借阅，可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文，并且本人电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

签名：  导师签名： 

日期：2007 年 6月 8 日

第一章 绪论

1.1 JPEG 概述

JPEG 是“Joint Photographic Experts Group，是联合摄影专家组”的缩写，是适用于黑白及彩色照片，传真和印刷等方面的静止图像编码的标准[1]。JPEG 主要存储颜色变化的信息，特别是亮度的变化。在常用的模式中，用有损压缩方式去处冗余的图像和色彩数据，在获得极高的压缩率的同时能展现十分丰富、生动的图像，可以用较少的磁盘空间得到较好的图像质量。JPEG 还是一种灵活的格式，具有调节图像质量的功能，允许采用不同的压缩比例对文件进行压缩。和具有相同图像质量的其他文件格式(如 BMP、GIF、TIFF)相比，JPEG 是目前静态图像中压缩率比较高的。正是由于这种高压缩比使得 JPEG 格式的文件尺寸较小，下载速度快，使得网页能以较短的下载时间提供大量的美观图像，所以目前各类浏览器都支持 JPEG 图像格式，使得它广泛的用于多媒体和网络程序中。

1.1.1 JPEG 运行方式

由于JPEG是适用范围很广、通用性很强的技术，所以把算法的功能分为有四种运行方式[2]：

1) 基本DCT(Discrete Cosine Transform)顺序：由8*8像素组成的像块，从左到右进行编码处理并按照从上到下顺序进行扫描。编码处理是由二元DCT系数的量化和量化系数的熵编码组成的。基本DCT顺序运行需要基本DCT的缓冲条件最小，因而实现的费用最低。

2) 基于DCT的扩展(渐进模式)：该模式对图像按照由粗到细进行编码，图像重现时由模糊到清晰，处理的顺序及编码处理的基本构成与DCT顺序模式相同，扫描的的顺序也是从上到下，从左到右，但需要对图像进行多次处理扫描。当对图像进行解码时，在第一次扫描后先得到一幅分辨率较低的粗略的图像轮廓，然后在后续的扫描处理后逐渐提高画面质量，得到清晰的图像，直到解码完成。渐进模式传输时间较长，接收端收到的图像是多次扫描由粗糙到清晰的累进过程。

3) 无损模式：无损编码方法能保证解码后精确的回复原图像的采样值，其压缩比低于有损压缩编码方法。使用二维差分脉冲编码调制技术的空间预测算法，可处理较大范围的输入像素精度。由于不使用DCT变换，对接近像素间的差别进行熵编码，从而不产生图像的失真，可以保证重建图像与原始图像完全相同。

4) 分层模式：组合上面三种方式，做成具有多种空间分辨率图像的金字塔结构。适用于对原始图像的滤波和划分像素，依次做成减少空间分辨率的图像时的图像数据结构称为金字塔。因为按空间分辨率由高到低的顺序累积起来是金字塔的形状，所以称为金字塔结构。由低品质图像向高品质图像阶段传送时，使用该方法。图像在多个空间分辨率进行编码，在信道传送速率低，接收端显示器分辨率也不高的情况下，只需要做低分辨率图像解码，不必进行高分辨率解码。

1.1.2 JPEG 编码系统

JPEG 算法有两种不同的压缩方式，一种是基于差分脉冲代码调制的压缩方式DPCM(Differential Pulse Code Modulation)，它是一种可逆的无失真的编码方式；另一种是基于 DCT 的压缩方式，该方式是有失真的不可逆的编码。JPEG 标准中定义了 3 种编码系统[1]：

1) 基于DCT的基本顺序系统：采用基于DCT的顺序编解码算法实现有损图像压缩。重建图像质量达到人眼难以辨别出损失的要求。采用8*8像素自适应DCT算法、量化及Huffman型的熵编码器。

2) 基于DCT的扩展系统：为了满足更为广泛的通信要求，JPEG算法中除了具有基本功能外，还具有扩展功能，其代表性的扩展功能就是累进显示。此外，在扩展功能中还应设置有多种可选择的功能。扩展编码系统使用累进工作方式，采用自适应算术的编码或者Huffman编码过程。

3) 无失真系统：无失真系统是一种完全能恢复原图像的可逆系统。由于DCT编码会产生一定的信息丢失，不适宜用于无失真系统。在JPEG算法中，通过用DPCM脉冲调制代码调制方式实现无失真系统的可逆编码。这是一种有别于DCT的独立的压缩编码方式，有时又称为独立功能。为保证重建图像数据与原始图像数据完全相同，无失真系统采用预测编码及Huffman或算术编码。

在JPEG的三种编码系统中，基于DCT的基本顺序系统是JPEG最基本的压缩系统，JPEG标准的软硬件编码、解码都支持这个过程。另两个系统本课题不再做讨论，本课题所讨论和实现的就是最常用的基于DCT的基本顺序系统。

1.2 课题的意义及研究内容

21 世纪的人类社会将是信息化社会，以信息技术为主要标志的高新技术产业在整个经济中的比重不断增长，多媒体技术及产品是当今世界计算机产业发展的新领域。世界上很多国家，对多媒体技术的研究和应用都给予了极大的重视，并投入了大量的人力、物力开发先进的多媒体信息技术及相关产品，试图占领庞大的多媒体市场。多媒体技术是使计算机具有综合处理声音、文字、图像和视频的能力，它以形象丰富的声、文、图像信息和方便的交互性，极大地改善了人机界面，改变了使用计算机的方式，从而为计算机进入人类生活和生产的各个领域打开了方便之门，给人们的工作、生活和娱乐带来深刻的变化[3]。

随着多媒体技术在各个应用领域不断普及，用户对新产品的期望越来越高，希望它具有更大的图像容量、更高的图像质量和更快的图像处理速度，这些都为图像的存储和处理提出了更高的要求。在数字图像处理可视电话通信、数字电视等应用中，遇到的首要难题就是数据量过大，导致图像传输和存储成问题，单纯靠增加存储器容量，提高信道带宽以及计算机的处理速度等方法来解决这个问题是不现实的，这时最好的解决办法就是对图像进行压缩编码。近十年来，数字图像和数字视频的压缩技术取得

了突破性进展，目前主要有三种方式：其一是纯硬件方式，即采用专用芯片编码，编码速度快，但灵活性差，系统成本造价高；其二是用纯软件方式实现解码算法，解码速度慢，但灵活性好，算法易更新升级，并且造价较低；其三是基于高性能处理器(如ARM、DSP 等)的软件实现，此种方式利用处理器的高速信号处理能力，使软件实现的算法在其上运行时间大大缩短，同时该方案易升级，算法易更新。本课题就是利用第三种方法，在 ARM(Advanced RISC Machines)平台上建立一个高性能的 JPEG 解码及显示系统。

近年来，伴随着因特网和消费电子的兴起，JPEG 标准广泛地应用于消费类电子、网络、军事、工业控制、生物医学、遥感、模式识别和机器人视觉等领域。图像压缩是一个很有发展前途的研究领域，这一领域的突破对于通信和多媒体事业的发展将有深远的影响。

网络、通信、多媒体和信息家电时代的到来，无疑为以 ARM 系列处理器为首的 32 位嵌入式系统应用提供了空前巨大的发展空间。嵌入式系统就是以应用为中心，以计算机技术为基础，软硬件可裁减，适合应用系统对功能、可靠性、成本、体积及功耗严格要求的专用的计算机系统；它是设计完成复杂功能的硬件和软件，并使其紧密耦合在一起的计算机系统[4]。ARM 微处理器具有性能高、成本低和能耗低的特点，适用于工业控制、消费电子、多媒体、成像技术等多种领域。所以在 ARM 开发平台上研究 JPEG 解码具有相当重要的价值和意义。

本文将在深入研究 JPEG 图像压缩标准的基础上，以 SAMSUNG 公司的高端处理器 S3C44B0X 为核心的开发板做为嵌入式硬件平台，构建嵌入式系统，讨论 JPEG 解码及其 LCD 显示的实现，并针对实际的需求，进一步从硬件和软件方面探讨了对整个系统的优化。

1.3 JPEG 的国内外研究现状

数据压缩是个非常活跃的领域，总是不断出现与实践新的方法、思想和技术。JPEG 在图像压缩领域有着广泛的应用，但并不完善。对一个 8*8 像素块进行 DCT 便换有时会导致重建图像中出现块效应[5]。文献[6]为解决量化噪声带来的块效应，提出了通过增加获得不规则图像碎片编码提高量化质量，从而提高解码图像的重构质量。文献[7]，[8]实践了 JPEG 的加密、解密与隐藏技术，加密、解密与隐藏技术是近几年 JPEG 研究的另一个方向。

目前，国内外在嵌入式编解码方面的应用更是异彩纷呈，文献[9]给出了 JPEG 编码在 SoC(System on Chip)上的实时实现，它是为数码相机而设计的，JPEG 图像通过 CMOS 图像传感器获得。文献[10] 根据长途汽车对上车人员进行记录的应用要求，开发了一套基于 EZ-USB 的低端图像数据采集存储及传输系统，采用 OmniVision 公司的 CMOS 图像传感器 OV7620 作为采集芯片，Zoran 公司的 ZR36060 作为数据压缩芯片，Cypress 的带 USB 接口的单片机 AN2131QC 作为总控制芯片和 USB 数据传输芯片。文献[11]中，松下电器产业公司开发出的 SoC 能够进行 JPEG、MPEG－2 和

MPEG－4 等标准的编解码，它配备了支持 SD 卡、DVD、硬盘和 USB 等设备的接口电路，上述编解码处理由对多媒体处理进行了优化的可扩展信号处理器“UniPhier 处理器”执行。文献[12]在基于 Motorola 公司的 ColdFire 系列的 32 位 DSP 处理器 MCF5272 上实现 JPEG 解码算法的设计与实现。文献[13]在 DSP 开发板上实现了 JPEG 图像压缩编码算法。其中的二维 DCT 部分采用行列分解法，一维 DCT 部分使用 DFT 系数方法实现。文献[14]采用以 TI 公司的高速 DSP 芯片 TMS320C6201 为核心的开发板做为图像压缩器的硬件平台，通过自行开发的压缩程序，实现了图像的实时压缩。文献[15]利用 Fujitsu 公司生产的 FR1000 作为嵌入式系统的多核处理器，将 JPEG 图像划分为 4 个部分，分别在 4 个处理器核上进行解码，实现了 JPEG 图像的实时解码；FR1000 将 4 个处理器核集成在 1 枚芯片上，各个处理器核之间共享内存和其它外部设备。文献[16]在 Fujitsu FR1000VDK 开发环境下，实现了一个可在多处理器嵌入式系统环境下运行的 JPEG 解码的 FarmWare 软件产品。

另外，随着可编程逻辑器件技术的发展，人们在利用 FPGA 来实现图像的编解码芯片方面也进行了研究。文献[17]利用 Altera 公司的 Cyclone FPGA 芯片实现了 JPEG 图像的高速编码。文献[18]利用硬件描述语言 VHDL、ALTERA 公司的 QUANTUS II 2.0 EAD 工具平台及上海 TrainSillcon 公司的 OPEN-FPGA4.0 开发板上实现了 JPEG 的编解码。

上面介绍的 JPEG 编解码的实现方法，主要都是利用硬件的方法实现的，其成本较高，比较专一，系统升级比较困难，而且系统不容易拓展；另一方面，随着 ARM 技术的不断发展，ARM 处理器的速度在不断提高，其对数据处理的能力也越来越强，价格也在不断降低。因此我们利用以 ARM 公司的 S3C44B0X 微处理器为核心的开发板设计出了一个成本较低、系统易于升级且容易拓展的嵌入式 JPEG 解码系统。

1.4 论文组织结构

论文全文共由六部分组成：

第一章为绪论，介绍了 JPEG 标准相关内容，课题的意义及研究内容，JPEG 的国内外研究现状。

第二章介绍了基于 ARM 的硬件平台，S3C44B0X 及外围相关电路及接口。

第三章以 JPEG 标准为依据，分析了 JPEG 图像编解码的基本原理和方法，详细论述了图像压缩中所使用的一些关键技术。

第四章着重阐述了软件系统的基础构建与设计。包括运行环境 ADS1.2 的介绍，系统启动引导程序 Bootloader 的裁减、实现，底层驱动程序和 LCD 的驱动程序的设计编写等。

第五章给出了解码芯片总体方案的设计，详细论述了在 PC 机上实现 JPEG 解码的步骤，其次讲解了软件到硬件平台上的移植，然后从软件和硬件角度讲解了优化，最后对解码时间和图像重建质量进行了讨论，最后给出了实验结果。

第六章对论文的全部工作进行了总结和展望。

第二章 ARM 处理器及外围电路

在实现解码之前，首先了解一下系统的硬件知识。硬件是指计算机系统中的各种物理装置，包括控制器、运算器、内存储器、I/O 设备以及外存储器等，它是计算机系统的物质基础。没有硬件，谈不上应用计算机。硬件是软件赖以工作的物质基础。所以，掌握 ARM 处理器结构及外围电路是必要的。

2.1 ARM 微处理器概述和 ARM7 的特点

ARM(Advanced RISC Machines)既可以认为是一个公司的名字，也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。1991 年 ARM 公司成立于英国剑桥，主要出售芯片设计技术的授权。目前，采用 ARM 技术知识产权(IP)核的微处理器，即我们通常所说的 ARM 微处理器，已遍及工业控制、消费类电子产品、通信系统、网络系统、无线系统等各类产品市场，基于 ARM 技术的微处理器应用大约占据了 32 位 RISC 微处理器 75%以上的市场份额，ARM 技术正在逐步渗入到我们生活的各个方面。其中 RISC(Reduced Instruction Set Computer)为精简指令集计算机。

ARM 微处理器目前包括下面几个系列:ARM7 系列、ARM9 系列、ARM9E 系列、ARM10E 系列、SecurCore 系列、Inter 的 Xscale、 Inter 的 StrongARM。

其中，ARM7, ARM9, ARM9E 和 ARM10E 为 4 个通用处理器系列，每一个系列提供一套相对独特的性能来满足不同应用领域的需求。SecurCore 系列专为安全需要而设计，提供了完善的 32 位 RISC 技术的安全解决方案。Xscale 处理器是基于 ARMv5TE 体系结构的解决方案。StrongARM 是采用 ARM 体系结构高度集成的 32 位 RISC 微处理器。

ARM7 系列微处理器为低功耗的 32 位 RISC 处理器，最适合用于对价位和功耗要求较高的消费类应用，具有如下特点[19]:

1) 具有嵌入式 ICE-RT 逻辑，调试开发方便。

2) 极低的功耗，适合对功耗要求较高的应用，如便携式产品。

3) 能够提供 0.9MIPS/MHz 的三级流水线结构。

4) 代码密度高并兼容 16 位的 Thumb 指令集。

5) 对操作系统的支持广泛，包括 Windows CE、 Linux、 Palm OS 等。

6) 指令系统与 ARM9 系列、ARM9E 系列和 ARM10E 系列兼容，便于用户的产品升级换代。

7) 主频最高可达 130 MIPS，高速的运算处理能力能胜任绝大多数的复杂应用。

ARM7 系列微处理器的主要应用领域为:工业控制、Internet 设备、网络和调制解调器设备、移动电话等多种多媒体和嵌入式应用。

ARM7 系列微处理器包括如下几种类型的核：ARM7TDMI，ARM7TDMI-S，ARM720T, ARM7EJ。其中，ARM7TMDI 是目前使用最广泛的 32 位嵌入式 RISC 处理器，本文使用的就是 ARM7TMDI 系列。TDMI 的基本含义为：

T：支持 16 位压缩指令集 Thumb；

D：支持片上 Debug；

M：内嵌硬件乘法器(Multiplier)；

I：嵌入式 1CE 支持片上断点和调试。

2.2 S3C44B0X 芯片概述

2.2.1 S3C44B0X 芯片简介[19]

S3C44B0X是Samsung公司推出的16/32位RISC处理器，为手持设备和一般类型应用提供了高性价比和高性能的微控制器解决方案。

S3C44B0X 使用 ARM7TDMI 内核，采用 0.25μm CMOS 工艺制造。它的低功耗和全静态设计特别适用于对成本和功耗敏感的应用。同时，S3C44B0X 还采用了一种新的总线结构，即 SAMBAII (三星 ARM CPU 嵌入式微处理器总线结构)。S3C44B0X 的另一个出色特性是它的 CPU 内核，是由 ARM 公司设计的 16/32 位 ARM7TDMI RISC 处理器，它集成了 Thumb 代码压缩器和一个 32 位的硬件乘法器，同时还支持 ICE 断点调试，其主频可达 66MHz。其他特点如下：

2.5V ARM7TDMI 内核，带有 8KB Cache；

可选的内部 SRAM；

外部存储器控制器(FP/EDO/SDRAM 控制，片选逻辑)；

LCD 控制器(最大支持 256 色 STN， LCD 具有专用 DMA)；

2 通道通用 DMA，2 通道外设 DMA 并具有外部请求引脚；

2 通道 UART 带有握手协议；

5 个 PWM 定时器和 1 通道内部定时器；

看门狗定时器；

71 个通用 I/O 口；

8 个外部中断源；

8 通道 10 位 ADC；

具有日历功能的 RTC；

片上 PLL 时钟产生器。

2.2.2 S3C44B0X 的系统框图

S3C44B0X 的系统框图如图 2-1 所示。

2.3 核心电路模块设计

2.3.1 系统硬件组成

本系统是以 ARM7 为主要硬件平台，基于 SAMSUNG 公司的 S3C44B0X 处理器而设计的开发板。它包括 S3C44B0X 处理器、8 位 LCD 连接器接口、2MB 的 FLASH、

8MB 的 SDRAM、10MHz 外部时钟，2 个 RS-232 串行口、一个 JTAG 接口、一个并行调试接口、一个 RTC、一个 10/100MB/s 网络接口、一个 USB 接口、一个具有扬声器和麦克的音频接口、IDE(Integrated Drive Electronics)硬盘扩展接口等。图 2-2 是 ARM 开发板实物图。

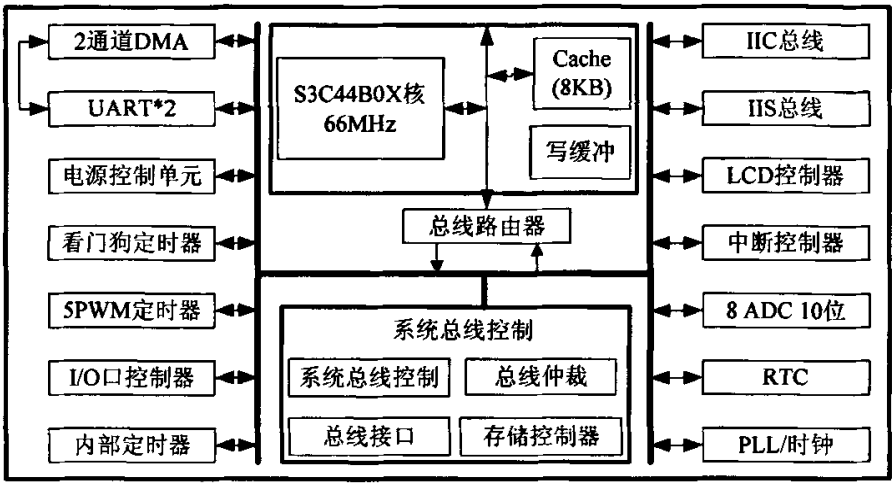


图 2-1 S3C44B0X 系统框图

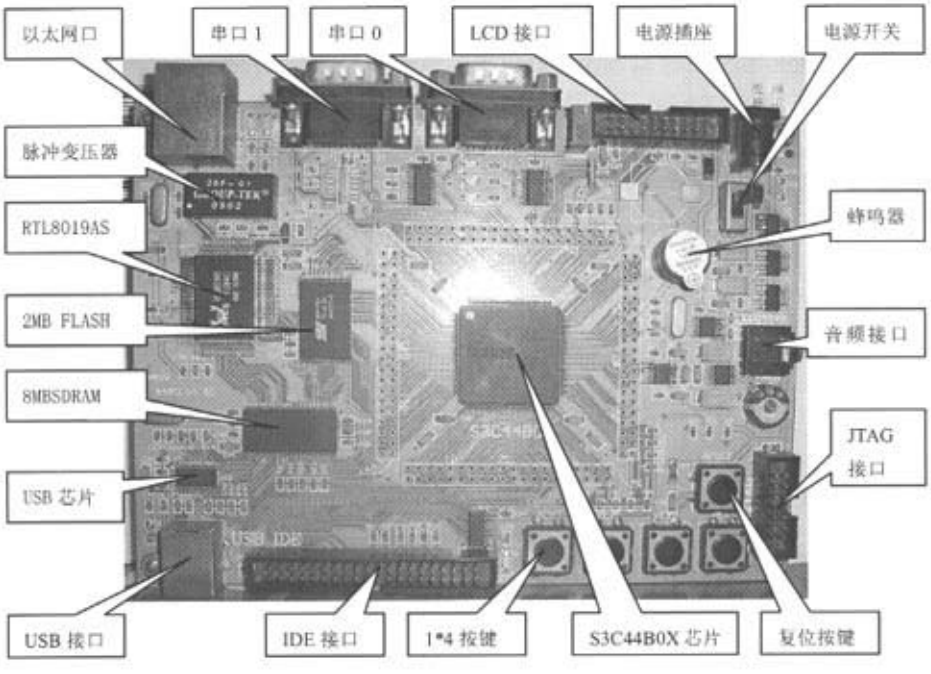


图 2-2 ARM 开发板实物图

2.3.2 电源、时钟、复位电路

采用 DC7.5V 稳压电源进行供电，电源输入后经过板上三个稳压芯片分别产生 3.3V,2.5V 和 1.8V 的电压，3.3V 给 S3C44B0X 的 I/O 端口供电，2.5V 给 ARM 内核供

电。

系统时钟源可以由晶体和外部时钟提供，它的选择控制由 OM[3: 2](在 nRESET 上升沿锁定)来决定。本系统采用 10MHz 晶体作为时钟源，OM[3:2]=00。OM[3:2]可以选择时钟源，OM[3:2]=00 表示选择晶振时钟，OM[3:2]=01 表示选择外部时钟。EXTCLK=VDDIO=3.3V。EXTCLK 引脚表示选择外部时钟时的外部时钟输入信号线，不用时必须接 3.3V。图 2-3 为系统时钟电路。

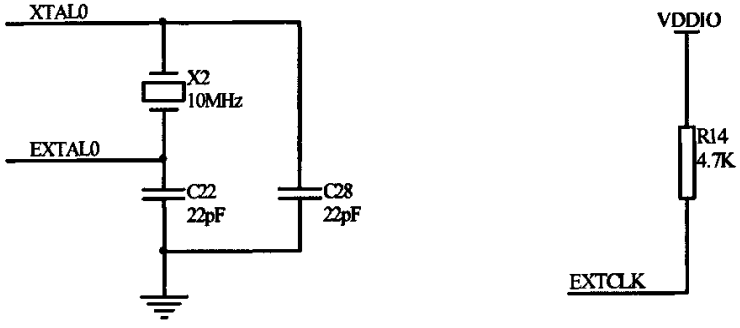


图 2-3　系统时钟电路

系统采用 RC 复位电路，具体电路如图 2-4 所示，该复位电路的工作原理如下：在系统上电时，通过电阻 R26 向电容 C46 充电，当 C46 两端的电压未达到高电平的门限电压时，Reset 端输出为低电平，系统处于复位状态；当 C46 两端的电压达到高电平的门限电压时，Reset 端输出为高电平，系统进入正常工作状态。

当用户按下按钮 S5 时，C46 放电，Reset 端输出为低电平，系统进入复位状态，再重复以上的充电过程，系统进入正常工作状态。两级或门电路用于去按钮引起的抖动和波形整形。通过调整 R26 和 C46 的参数，可调整复位状态的时间。

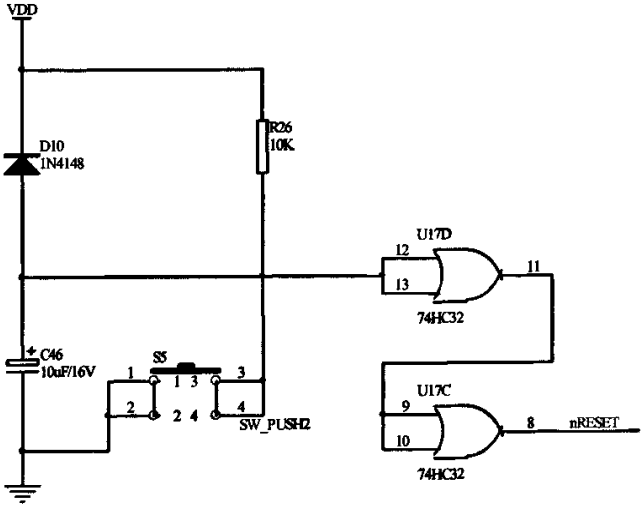


图 2-4 系统复位电路

2.3.3 存储电路

存储系统使用一片 1M*16bit 的 FLASH(SST39VF1601)和一片 4M*16bit 的 SDRAM(HY57V641620HG)。

FLASH 连接电路如图 2-5 所示，处理器通过片选 nGCS0 与片外 Flash 芯片连接。由于是 16bit 的 FLASH,所以用 CPU 的地址线 Al-A20 来分别和 Flash 的地址线 A0-A19 连接。地址空间为 0X0—0X1FFFFF, 其中 0X0—0X3FFFF 用来存放系统的启动引导程序代码,0X40000—0X4FFFF 存放启动程序的参数,0X50000—0X1FFFFF 为用户程序区。

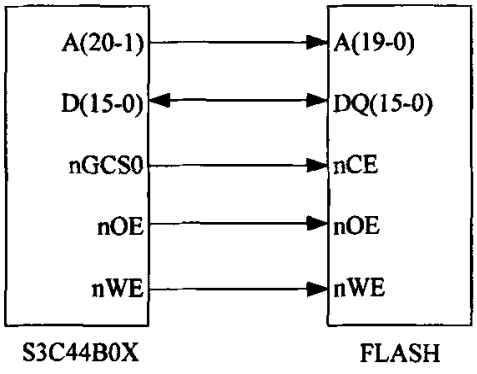


图 2-5 FLASH 连接电路

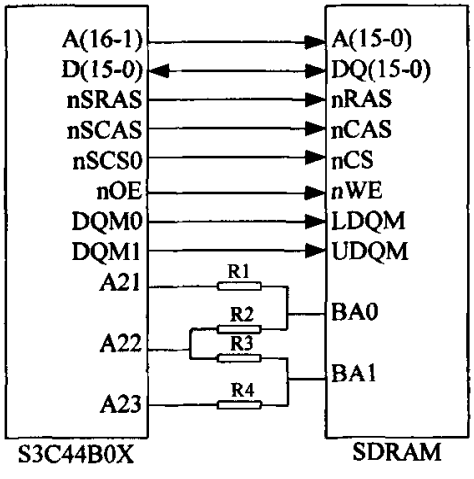


图 2-6 SDRAM 连接电路

SDRAM 连接电路如图 2-6 所示，SDRAM 分成 4 个 BANK，每个 BANK 的容量为 1 M * 16bit。 BANK 的地址由 BA1、BA0 决定，00 对应 BANK0, 01 对应 BANK1, 10 对应 BANK2, 11 对应 BANK3。在每个 BANK 中，分别用行地址脉冲选通 RAS 和列地址脉冲选通 CAS 进行寻址。行、列地址线复用，行地址线为 A[0-11],列地址线为 A[0-7]，地址空间为 0x0C000000-0x0C7FFFFF。

2.4 串行接口 URAT

串口电路如图 2-7 所示，系统设计两个串口(URAT0 和 URAT1)与外部通信。PORTC10～PORTC15 分别作为 nRTS1、nCTS1、TXD1、RXD1、nRTS0 和 nCTS0

信号，GPE1 和 GPE2 作为 TXD0 和 RXD0 信号。两个串行接口都采用 MAX232 进行电平转换。

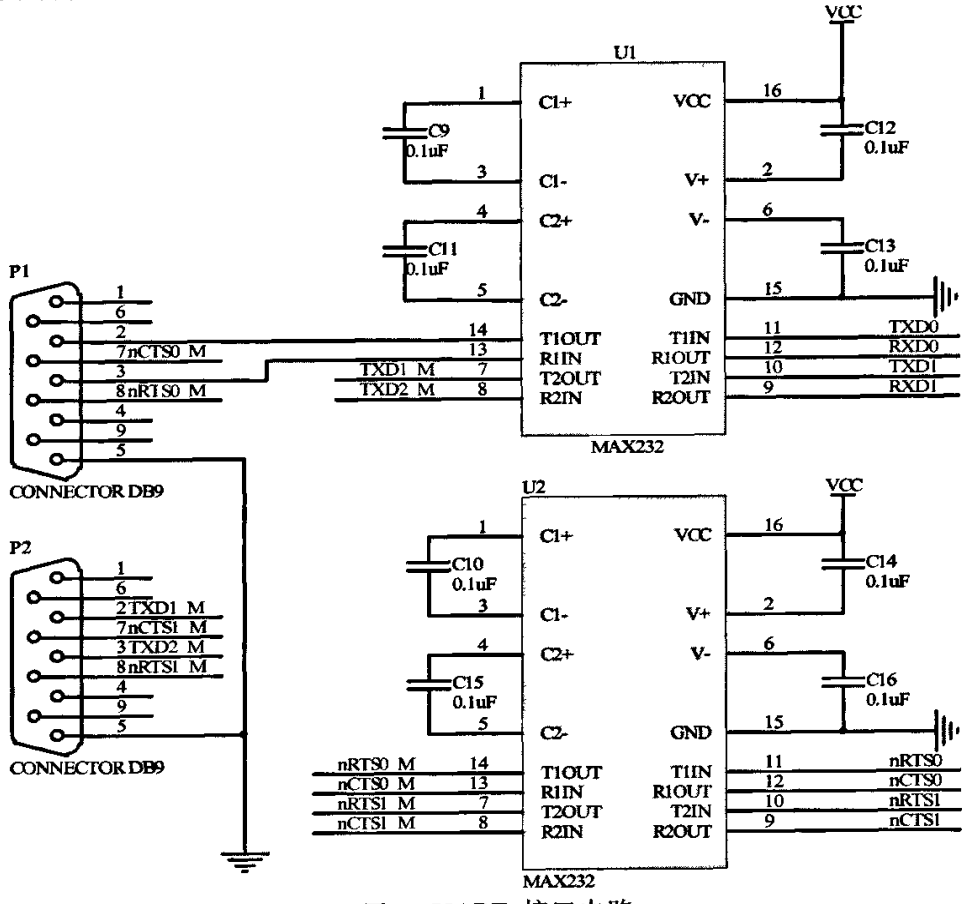


图 2-7 UART 接口电路

2.5 LCD 显示模块

S3C44B0X 中内置 LCD 控制器可以支持 4 级灰度、16 级灰度的黑白 LCD 和 256 级颜色的彩色 LCD 屏；支持 3 种 LCD 驱动器：4 位双扫描，4 位单扫描，8 位单扫描显示模式。内置 LCD 控制器的作用是将定位在系统存储器(SDRAM)中的显示缓冲区中的 LCD 图像数据传送到外部 LCD 驱动器，并产生必须 LCD 控制信号[20]。图 2-8 为 LCD 控制器内部结构框图。其中：VCLK 是 LCD 控制器和 LCD 驱动器之间的像素时钟信号；VLINE 是 LCD 控制器和 LCD 驱动器之间的行同步脉冲信号；VFRAME 是 LCD 控制器和 LCD 驱动器之间的帧同步信号。VM 是 LCD 驱动器的 AC 信号。VD[3：0]和 VD[7：4]是 LCD 像素点数据输出端口。

本课题中LCD选用的是SHARP公司的液晶显示器LM7M632，LM7M632是按照8位单扫描模式工作的。所谓8位单扫描方式，就是显示采用8位并行数据线进行“行”数据连续移位输出，直到整个帧的数据都被移出为止。LCD模块接口信号线的定义如表2-1所示，图2-9为LCD控制器与LCD驱动器的连接电路。在显示系统的硬件电路中，S3C44B0X中的内置LCD控制器与LCD模块LM7M632的连接是关键。

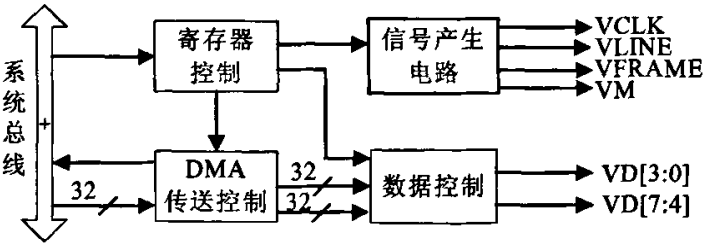


图2-8 LCD 控制器内部结构图

表2-1 LCD模块(LM7M632)接口信号线定义

引脚	引脚描述	引脚	引脚描述
1	YD：扫描开始	8	VDD：logic供电(3.3V)
2	M：交变信号	9	Vcon：对比调节电压
3	LP：输入锁存	10	VSS：地
4	VSS：地	11-14	D0~D3：数据线低四位
5	XCK：时钟	15	VSS：地
6	VSS：地	16-19	D4~D7：数据线高四位
7	DISP：显示开关	20	VSS：地

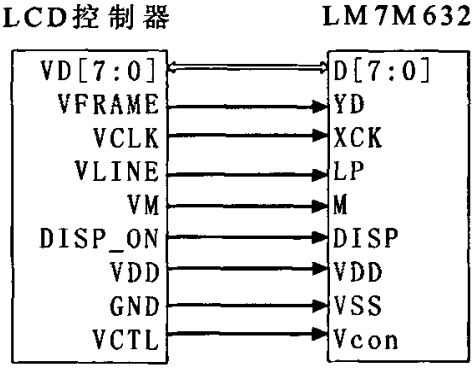


图2-9 LCD控制器与LCD驱动器的连接电路

图2-10给出了LM7M632模块接口时序图。其中，YD是帧(写满整个屏的数据称为1个“帧”)同步信号，该信号启动LCD屏的新一帧的数据。两个YD脉冲之间的时间长度就称之为“帧周期”。根据LCD模块的特性，帧刷新周期为12ms到14ms，频率为70Hz～80Hz。每1帧中包含240个LP脉冲。LP为行(共240行)数据输入锁存信号，该信号启动LCD屏新的一行的数据。也就是行同步脉冲信号。每1行中包括640*3/8个XCK脉冲信号。XCK为行数据输入信号，也就是每一行中像素点数据传输的时钟信号；每组8位的数据在XCK的下降沿处被输入锁存。D0~D7是8位的显示数据输入信号。

在该显示系统中，其显示方式是以直接操作显示缓冲区(SDRAM)的内容进行的，LCD控制器会通过DMA方式从显示缓冲区中获取数据，不需要CPU干预。在256色显

示模式下，显示缓冲区中的一个字节数据代表LCD上的一个点的颜色信息，因此，所需要的缓冲区的大小为640*240*1 字节，每个字节的RGB数据格式为：由3位红色(Bit7~Bit5)、3位绿色(Bit4~Bit2)、2位蓝色 (Bit1~Bit0) 组成。

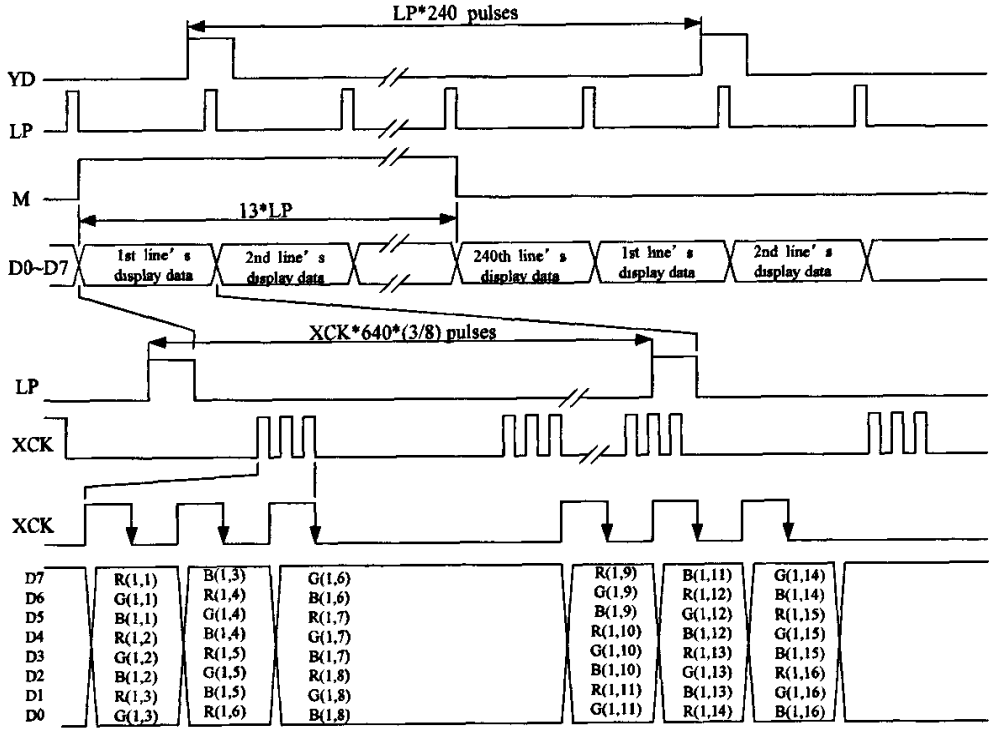


图2-10 LM7M632模块接口时序图

2.6 本章小结

本章主要讲述了核心控制单元的设计。首先讲述了ARM处理器和ARM7特点，然后讲述所选芯片S3C44B0X的特性和它的系统框图，重点对核心电路模块中的电源、时钟和复位电路、存储系统、以及外围电路中通信接口模块中的串行接口进行设计和分析。最后，分析了S3C44B0X中的LCD控制器，给出了LCD控制器和LCD驱动器LM7M632的接口电路，并对液晶屏LM7M632模块接口时序进行了分析。

第三章 JPEG 编解码算法分析

JPEG标准广泛应用于图像存储和图像通讯等方面，这是由于它具有以下优点：

1) 通用性高，能在各种设备中广泛的使用；

2) 压缩效率高，由于充分利用了人眼视觉的低通滤波特性；

3) 恢复的图像质量好，可视性和可懂性都较好；

4) 是多种编码方法的综合，考虑了可逆编码和非可逆编码；

5) 易于实现，执行速度较快。

下面就针对基于 DCT 的基本顺序系统的编解码算法进行阐述。

3.1 JPEG 文件格式[21]

目前广泛使用的是 JPEG 文件交换格式 JFIF (JPEG File Interchange Format) 1.02 版本。JPEG 文件中的字节是按照 Motolora 的格式存储的，也就是正序排列，高位字节在前，低位字节在后。

JPEG 文件大致分为两部分：标记码(tag)和压缩数据。标记码部分给出了 JPEG 图像的大部分信息，如图像的宽、高、Huffman 表、量化表等等。标记码的结构为：

SOI
DQT
DRI
SOF
DHT
SOS
EOI

标记码由两个字节构成，前一个字节是固定值 0XFF，每个标记之前还可以添加任意数目的 0XFF 填充字节。标记码部分类似于 BMP 文件的头信息。绝大多数的 JPEG 只包含以下标记：

· SOI 0X FFD8 Start Of Image 图像开始标记，可作 JPEG 格式的判据(JFIF 还需要 APP0)

·APP0 0X FFE0 APP0 是 JPEG 保留给 Application 所使用的标记码，而 JFIF 将文件的相关信息定义在此标记中。

·APPn 0X FFE1 —0X FFEF reserved for application use 其它应用数据块(n: 1~15)

·DQT 0X FFDB Define Quantization Table 量化表开始

·DRI 0X FFDD Define Restart Interval 重入间隔

·SOF 0X FFC0 Start Of Frame 帧开始

·DHT 0X FFC4 Define Huffman Table Huffman 表开始

·SOS 0X FFDA Start Of Scan 扫描线开始

·EOI 0X FFD9 End Of Image 图像数据结束

下面做进一步解说：

1) 量化表 DQT(一个或多个)

·量化表标识长度 2 字节。

·量化表参数 1 字节(Pq　Tq)高 4 位 Pq 表征量化表数据精度，Pq=0 时，量化表的值 Q[n]为 8bit 表示，Pq=1 时 16bit; Tq 为量化表的编号，为 0-3。在基本系统中，Pq=0，Tq=0，1。

·量化表 Q[n], n=0-63，长度由 Pq 决定。表示量化表的 64 个值得(Zig-zag 形排列)。

2) 帧开始标识 SOF

·SOF 标识长度 2 字节。

·精度每个颜色分量每个像素的位数 1 字节，在基本系统中为 8 位。

·图像高度 2 字节。

·图像宽度 2 字节。

·颜色分量数 1 字节即帧中成分个数，真彩色为 3，灰度图为 1。

·每个颜色分量包括：(每个分量用 3 字节，如真彩则共用 9 字节)

成分编号：1 字节，Y：1. U：2, V：3

采样方式：1 字节，低 4 位是垂直采样率，高 4 位是水平采样率。

量化表号:1 字节

3) Huffman 表 DHT (一个或多个)

·Huffman 表长度 2 字节

·类型编号 1 字节(Tc Th)，基本系统中高 4 位 Tc 表征表类型：0 为 DC Huffman 表，1 为 AC Huffman 表。Th 表示 Huffman 表的编号，基本系统中为 0 或 1。

·位表 L[n]，n=1-16，L[n]表示长度为 n 位的码字的个数。各占 1 位。

·值表 V [t]表示每个 huffman 码字所对应的值。t=1-K, K=SL[n]。

4) 扫描开始 SOS

·SOS 段长度 2 字节

·.颜色分量数 1-3，1 字节

·每个颜色分量包括：(每个颜色分量用 2 个字节)

颜色分量编号：1 字节

huffman 表号：1 字节，低 4 位为 DC 表的编号，高 4 位为 AC 表

·Ss 基本系统中为 0, 1 字节

·Se 基本系统中为 63, 1 字节

·Bf 基本系统中为 0. 1 字节

5) DRI (Define Restart Interval)

此标记需要用到最小编码单元 MCU(Minimum Coding Unit)的概念。我们知道，Y(亮度)分量数据重要，UV(色度)分量的数据相对不重要，所以可以只取 UV 的一部分，以增加压缩比。目前支持 JPEG 格式的软件通常提供两种取样方式 YUV411 和 YUV422，其含义是 YUV 三个分量的数据取样比例。举例来说，如果 Y 取四个数据

单元，即水平取样因子 Hy 乘以垂直取样因子 Vy 的值为 4，而 U 和 V 各取一个数据单元，即 Hu*Vu=1,Hv*Vv=1。那么这种部分取样就称为 YUV411。如图 3-1 所示：

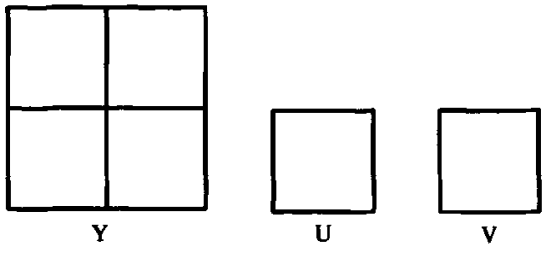


图 3-1 YUV411 的示意图

由上图易知 YUV411 有 50%的压缩比(原来有 12 个数据单元，现在有 6 个数据单元)，同理，YUV422 有 33%的压缩比(原来有 12 个数据单元，现在有 8 个数据单元)。JPEG 标准规定了最小编码单元 MCU，要求 Hy*Vy+Hu*Vu+Hv*Vv=10。

·DRI 段长度 2 字节

·重入间隔的 MCU 个数，2 字节，必须是一 MCU 行中 MCU 个数的整数，最后一个零头不一定刚好是 Ri 个 MCU。每个重入间隔各自独立编码。

3.2 JPEG 压缩步骤简介

在了解JPEG 解压缩步骤之前，首先对JPEG的标准压缩过程做一些了解。JPEG最基本的压缩就是将图像的每一个区块经由DCT的运算从空间域转换为频率域。由于人们的眼睛对高频的分量部分较不敏感，更由于一般影像高频部份的分量会比低频部分要小得多，所以可用较为少量的粗略影像来表示大量的高频部分，于是可以大幅度地减少要储存或通讯的数据量，而压缩后的信息影像同样能够为人们的视觉感官所接受。图3-2为JPEG编码流程图。

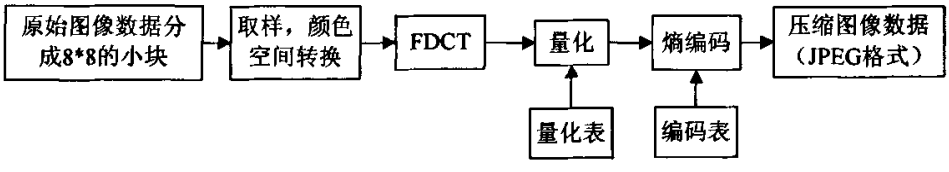


图 3-2 JPEG 编码流程图

首先图像被分割成许多8*8 像素的区块，这些区块再以离散余弦变换FDCT(Forward Discrete Cosine Transform)从空间域转换到频率域，然后这些二维DCT系数，再以8*8 的量化矩阵进行量化处理，由于量化误差的存在，原来的值在进行量化后，无法再完整地恢复回去，因此这个量化处理过程会造成一些图像的失真。一般而言，高频部分的DCT分量系数会比低频部份DCT分量系数的值小，且由于人的视力对图像的高频部分较不敏感，因此在经过量化处理的工作后，会使得二维DCT系数的高频部分产生许多的零值。量化后的二维DCT系数会产生直流DC系数与交流AC系数组成的二维矩阵，必须再以DC与AC系数进行不同的编码。直流DC系数代表一个8*8

的像素块64个图像采样值的平均值。DC系数有两个特点：一是系数的值较大；二是相邻的8*8像素块的DC系数值变化不大。相邻的8*8像素块的DC系数具有很强的相关性，所以JPEG标准对DC系数采用差分脉冲调制编码法(DPCM)进行编码，即不是直接对DC系数本身进行编码，而是对相邻8*8像素块之间的直流DC系数差值(Diff)进行编码。其运算公式如下：

Diff_DC(i)=DC(i)-DC(i-1)

例如，两个相邻 DC 系数分别为 673 和 674，直接传输均需要 10 bit，采用 DPCM 后传输仅需 10 bit 和 1 bit，从而起到压缩的目的。这是由于相邻系数区块的差异不大，且相关性高，故取差值后，值域通常会变小，所以可达到压缩的目的，接着将所得到的 Diff_DC 做霍夫曼编码。DC 系数按＜Size＞＜Value＞数据格式表示：Size 是 Value 的宽度(二进制表示的比特数)，Value 就是它的值。

然后，对AC分量系数进行编码，AC分量系数的编码过程较为复杂，因为量化后系数矩阵右下角63个交流AC系数经压缩，多为0分量，宜采用行程编码。其原理如下：

为了将二维频域矩阵中频域系数排列成一维矢量形式，同时保证低频图像分量在一维排序中先出现，增加连“0”个数，常采用Zig-zag扫描方式。Zig-zag扫描顺序如图3-3所示。扫描之后按＜Run，Value＞数据格式表示：其中Run 表示在非零AC系数前出现零的个数，而Value表该非零AC系数的值，数据最后一个单位为＜EOB＞(End Of Block)，表示接下来的Zig-zag扫描系数皆为零，用＜0,0＞表示。然后将＜Run, Value＞转换为＜Run,Size＞＜Value＞格式，以Value值的大小来参考附录一的编码分类表，以决定Size值的大小，而最终Value值表＜Run, Value＞中Value的值经1的二进制补码后的结果。最后查询附录一的AC霍夫曼(Huffman) 编码表进行编码。

例如经 Zig-zag 扫描后 8*8 二维亮度矩阵转化为一维的序列为：

$15,0,-2,-1,-1,-1,0,0,-1,\underbrace{0,\cdots\cdots,0}_{55个0}$；

由上面的介绍，DC 系数 15 按＜Size＞＜Value＞数据格式表示为：＜4＞＜15＞，将第一个尖括号内的数作为索引查亮度 DC 差值的 Huffman 编码表得到相应的 Huffman 编码，＜4＞对应的码字是 101。再看第二个尖括号，我们直接将这个数写成二进制即可，＜15＞就应当是 1111。注意，如果是遇到负数的话就应该是它的相反数的反码，例如：＜-5＞就应当是 010，＜-1＞就应当是 0。将两部分连接起来就是 DC 系数最后的码字：101 1111。

AC 系数按＜Run, Size ＞数据格式表示为：＜1,-2＞，＜0,-1＞，＜0,-1＞，＜0,-1＞，＜2,-1＞，＜0,0＞。

将＜Run, Value ＞转换为＜Run,Size＞＜Value＞格式表示为：

＜1,2＞＜-2＞，＜0,1＞＜-1＞，＜0,1＞＜-1＞，＜0,1＞＜-1＞，＜2,1＞＜-1＞，＜0,0＞。

进一步转换为：＜1,2＞＜01＞，＜0,1＞＜0＞，＜0,1＞＜0＞，＜0,1＞＜0＞，＜2,1＞＜0＞，＜0,0＞。

查询附录一的 AC 霍夫曼(Huffman)编码表进行编码得到：

11011 01 00 0 00 0 00 0 11100 0 1010

所以，以上序列最终的编码为：101 1111 11011 01 00 0 00 0 00 0 11100 0 1010

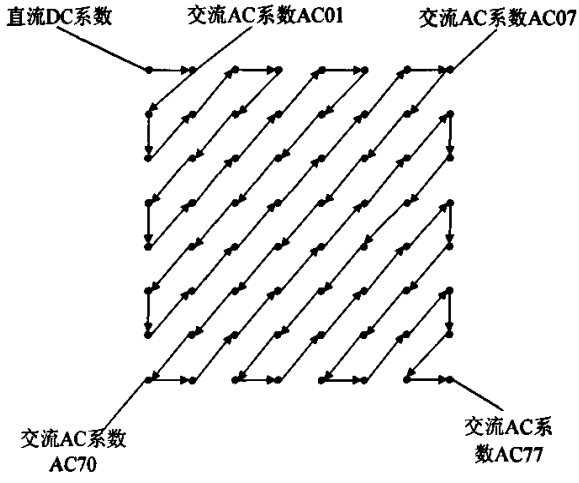


图 3-3 Zig-zag 扫描图

3.3 JPEG 解压缩

JPEG 解压缩的算法流程如图3-4所示，只是简单地把压缩过程给反了过来而已，由于本课题着重在解压缩的部分，故对其做较清楚的阐述。

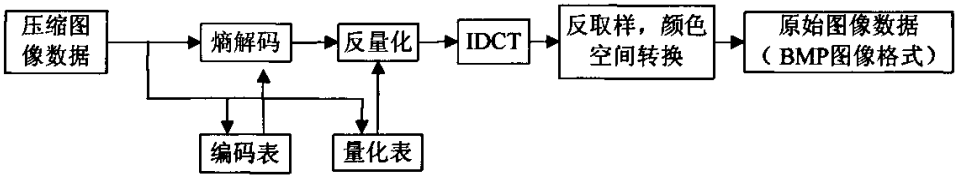


图3-4 JPEG解压缩流程

由图3-4可以看出JPEG解压缩分为以下几步：

1) 使用熵解码将数据还原成直流DC系数和交流AC系数组成的量化图。

2) 将量化区块与加权函数进行计算还原成DCT系数，这个加权函数存于JPEG文件中(标记码部分)，它具有目前数据值的最佳量化效果。

3) 使用反离散余弦转换IDCT，把频率域DCT分量系数转换为空间域表示的数字图像。

4) 最后将YUV，转换成为RGB色彩的BMP格式文件。

3.3.1 熵解码

熵解码是指将经过压缩的图像还原成由直流系数DC和交流系数AC组成的量化数据块的过程。JPEG对直流DC系数和交流AC系数采用不同的方法进行熵解码。熵解码是熵编码的逆过程，在JPEG解压缩算法过程中，由于霍夫曼编码中的编码字具有唯一性，因此可以使用查表方法进行解码。这种可变长度的霍夫曼码表(Huffman table)

已事先定义在JPEG图像的标记码中，编码表对出现频率比较高的符号分配较短的编码，而对出现频率较低的符号分配较长的编码。最后利用熵解码还原由DCPM编码后的数据，得到所要的直流DC系数，另外同样利用熵解码还原由RLE编码后的数据，得到交流AC系数。

我们以上述编码后的霍夫曼位数据串为例，同样分成多个步骤说明解码过程：

步骤1：$(10111111101101000000000111000101 0)_2$，先解DC码，当读入的第一个位$(1)_2$ 时，和附录一霍夫曼码DC表比较后，结果在表中找不到$(1)_2$ 这样一个编码字，接下来再读入下一个位成为$(10)_2$ 进行对比，结果在表中找不到$(10)_2$ 这样一个编码字，然后再读入下一个位成为$(101)_2$ 进行对比，对比结果得差值为数为4，紧接着的4位$(1111)_2$就为DC的值，转化为10进制为15。再解AC码，按照解DC码的方法读入，和附录一霍夫曼码AC表比较，$(11011)_2$对应<1,2>，由于Size为2，所以紧接着的两位代码 $(01)_2$为Value值，所以第一个AC码用<Run,Size><Value>格式表示为<1,2><01>，按照上面的方法，直到读到$(1010)_2$ 即<EOB>为止，最后得到结果为<1,2><01>，<0,1><0>，<0,1><0>，<0,1><0>，<2,1><0>，<EOB>

步骤2：接着，解码第一组<1,2><01>字符串，Size长度值为2时，参考附录一的编码分类表，得到所要的Value值只有-3，-2，2，3 四种可能，但只有-2 经过二进制1的补码转换后为$(01)_2$ ，故解码得<1,-2>，依此类推，最后得到<1,-2>，<0,-1>，<0,-1>，<0,-1>，<2,-1>，<0，0> 。

步骤3：熵解码的最后的一步骤，按照上面的结果写出一维解码序列：15,0,-2,-1,-1,-1,0,0,-1,$\underbrace{0,\cdots\cdots,0}_{55个0}$，然后经过Zig-Zag反扫描，也就是将一维的DCT恢复到8*8二维DCT系数。

3.3.2 反量化

反量化是将在压缩过程中经过FDCT变换后的频率系数还原出来，使用图像文件中8*8的量化表和还原后的二维DCT系数矩阵进行矩阵相乘的运算来进行反量化。由于人眼对低频分量的图像比对高频分量的图像更敏感，每个8*8小方块里面系数的位置愈靠近左上角，它代表的频率愈低，愈靠近右下角，则它代表的频率愈高，如图3-5所示。一般而言，大部份的图像能量会集中在低频部份，也就是转换之后的输出分量系数在低频部份的值较大，而输出系数在高频部份的值很小。

我们由JPEG编码过程可知，在对经过正向DCT变换后得到的8*8个变换系数进行量化时，是利用量化表来完成的。JPEG基本算法包括一套量化表，分别为亮度分量的量化表和色度分量的量化表。这两张量化表是从大量实验中得来的。这套量化表根据心理视觉阈值制作，对8位的亮度和色度的图像处理效果相当不错。量化表是控制JPEG压缩比的关键。基于人眼对高频成分的敏感程度远低于低频成分，并且在图片的像素和像素之间会有一个色彩的过渡过程，大量的图像信息被包含在低频空间中，因此，在量化过程除掉了一些高频分量，损失了高频部分的一些细节。量化表对位于经过DCT变换得到的8*8系数矩阵中不同位置的变换系数采取不同的量化步长。量化

表分为亮度量化表和色度量化表，如表3.1、3.2所示。量化结果就是正向DCT变换后得到的8*8个变换系数与量化矩阵相除的结果。而反量化同样利用这两个量化矩阵。反量化运算就是将上面经熵9解码得到数组中的元素与量化表中对应位置元素相乘：

$$F(u,v)=C(u,v)\times Q(u,v) \tag{3.1}$$

其中，$C(u,v)$代表熵解码输出，$Q(u,v)$代表相应的量化矩阵。$F(0,0)$就是直流系数DC，其余63个都是交流系数AC。

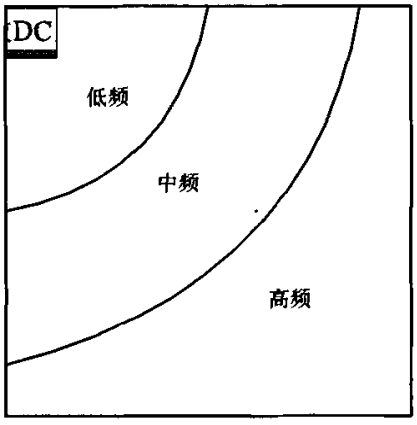


图3-5 8*8块的频率分布图

表3.1 亮度量化矩阵表

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
39	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

表3.2 色度量化矩阵表

17	18	24	47	66	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

3.3.3 反离散余弦变换

反离散余弦变换就是将量化之后的8*8个分量系数转换为8*8个像素点。此转换把频率域DCT分量系数转换到空间域表示的数字图像， 8*8矩阵二维IDCT的公式为：

$$f(x,y)=\frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\left[C(u)C(v)F(u,v)\cdot\cos\frac{(2x+1)v\pi}{16}\cos\frac{(2y+1)u\pi}{16}\right] \tag{3.2}$$

式中x,y代表图像数据矩阵内某个数值的坐标位置，$0\le x,y\le 7$；

$f(x,y)$代表图像数据矩阵内(x,y)位置上的像素值，$-128\le f(x,y)\le 127$；

$F(u,v)$代表矩阵内的(u,v)位置上DCT变换后的频率系数，$-1024\le F(u,v)\le 1023$；

u,v 代表DCT变换后矩阵内某个数值的坐标位置，$0 \le u,v \le 7$；

当 $u,v = 0$ 时 $C(u),C(v) = 1/\sqrt{2}$ ，否则 $C(u),C(v) = 1$ 。

直接运用式3.2计算，会有很大的运算量，必然影响JPEG的解码速度，为此人们找到了一系列快速算法。其中常用的有查表法和行列分解法。查表法就是预先将计算过程中所需乘法运算的系数提前算好，并以数组的形式放到内存中，在运算过程中，尽量减少乘法运算的次数，甚至不用乘法运算，只进行必要的加法运算，从而完成整个IDCT运算。查表法的运算时间与变换对象的个数成正比，因为在整个运算过程中不使用乘法运算只有加法运算，因此有较快的速度。文献[22]，[23]中提出了基于查表法实现二维DCT算法。本系统若采用二维查表法，粗略估计需要400K以上空间，这400K以上空间对资源有限的嵌入式系统，非常浪费。行列法是利用二维IDCT具有的可分离性，可以把二维IDCT变换成水平方向和垂直方向的两个一维IDCT进行计算。对8*8的二维数组进行IDCT的计算可以转化为先对该数组的行分别进行8次一维IDCT，再对列分别进行8次一维IDCT，这样就简化了计算复杂度，它是目前应用较多的方法。其转化如下：

$$f(x,y) = \frac{1}{2}\left[\sum_{u=0}^{7} C(u)g(x,v)\cos\frac{(2y+1)u\pi}{16}\right] \tag{3.3}$$

$$g(x,v) = \frac{1}{2}\left[\sum_{v=0}^{7} C(v)F(u,v)\cos\frac{(2x+1)v\pi}{16}\right] \tag{3.4}$$

就一维的IDCT，运算起来计算量也比较大，也出现了很多的快速算法。文献[24~30]中已有很多不同的快速算法实现一维IDCT。这些算法各有各的优点，根据本文的实现目标和硬件特点，本文中的二维IDCT算法采用行列分解法，对一维IDCT采用chen-wang快速算法[31]。

经过反离散余弦变换后的各个像素值的范围是－128～127，而图像的像素值是无符号的，其范围是0～255，因此，需要对IDCT变换后输出的像素值再加上128，则像素块返回原有的电平。如果经过IDCT变换后得到像素值若超出了范围0～255，当小于0时，按0处理；当大于255时，按255处理。

3.3.4 反取样及色彩空间转换

从解码流程图可以看出，解码结果输出的是BMP文件，BMP文件使用的颜色模型是RGB模型，而JPEG文件使用的颜色模型是YUV颜色模型，因此需要将YUV颜色模型转化为RGB模型。YUV模型表示图像的优点是亮度信号和色度信号U、V是分离的。由于人的眼睛对图像上的亮度Y的变化远比色度UV的变化敏感的多，因此在对图像编码时，JPEG就是通过对图像亮度和色度的不同的取样比来对亮度和色度取样的。若在压缩过程中，采用的是Y ：U：V＝4：1：1 的取样方式，即就是从一个16*16大小的Y亮度区块中有4个8*8的像素块，在取样时是全数保留，而在色度UV取样过

程中，是从16*16大小的区块中分别抽取出一个8*8的像素块，这6个8*8的像素块就称为最小编码单元MCU，因此，解码时，必须还原原来的一个Y块对应一个U块和一个V块，这就是反取样。反取样如图3-6所示。

下面是在RGB色彩空间与YUV色彩空间的转换公式。式3.4是RGB到YUV的转换，式3.5是YUV到RGB的转换。

$$\left.\begin{aligned} Y&=0.299\ R+0.587\ G+0.114B \\ U&=-0.1687\ R-0.3313G+0.5B+128 \\ V&=0.5\ R-0.4187G-0.0813B+128 \end{aligned}\right\} \tag{3.5}$$

$$\left.\begin{aligned} R&=Y+1.402\ (U-128) \\ G&=Y-0.34414(V-128)-0.71414\ (U-128) \\ B&=Y+1.772\ (V-128) \end{aligned}\right\} \tag{3.6}$$

应用公式3.6，将YUV数据转化为24位RGB数据。

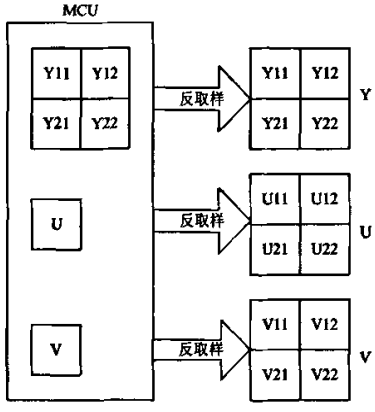


图3-6 YUV411反取样图

3.4 BMP 文件格式

JPEG解码之后在WINDOWS环境下存储格式为BMP文件格式，它是WINDOWS采用的图像文件存储格式，BMP 文件大体上分成四个部分[32]，如图 3-7 所示。

位图文件头 BITMAPFILEHEADER
位图信息头 BITMAPINFOHEADER
调色板 Palette
实际位图数据 ImageData

图 3-7 Windows 位图文件结构示意图

1) 位图文件头 BITMAPFILEHEADER，是一个结构，其定义如下：

```
typedef struct tagBITMAPFILEHEADER {
        WORD        bfType;
        DWORD       bfSize;
```

```
    WORD        bfReserved1;
    WORD        bfReserved2;
    DWORD       bfOffBits;
} BITMAPFILEHEADER;
```

这个结构的长度是固定的，为14个字节(WORD为无符号16位整数，DWORD为无符号32位整数)，各个域的说明如下：

bfType：指定文件类型，必须是0x424D，即字符串"BM"，也就是说所有.bmp文件的头两个字节都是"BM"。

bfSize：指定文件大小，包括这14个字节。

bfReserved1，bfReserved2为保留字，不用考虑。

bfOffBits：为从文件头到实际的位图数据的偏移字节数，即图3-7中前三个部分的长度之和。

2) 位图信息头 BITMAPINFOHEADER，也是一个结构，其定义如下：

```
typedef struct tagBITMAPINFOHEADER{
    DWORD   biSize;
    LONG    biWidth;
    LONG    biHeight;
    WORD    biPlanes;
    WORD    biBitCount
    DWORD   biCompression;
    DWORD   biSizeImage;
    LONG    biXPelsPerMeter;
    LONG    biYPelsPerMeter;
    DWORD   biClrUsed;
    DWORD   biClrImportant;
} BITMAPINFOHEADER;
```

这个结构的长度是固定的，为40个字节(LONG为32位整数)，各个域的说明如下：

biSize：指定这个结构的长度，为40。

biWidth：指定图像的宽度，单位是像素。

biHeight：指定图像的高度，单位是像素。

biPlanes：必须是1。

biBitCount：指定表示颜色时要用到的位数，常用的值为1(黑白二色图), 4(16色图), 8(256色), 24(真彩色图)。

biCompression：指定位图是否压缩，有效的值为BI_RGB，BI_RLE8，BI_RLE4，BI_BITFIELDS(都是一些Windows定义好的常量)。要说明的是，Windows位图可以采用RLE4，和RLE8的压缩格式，但用的不多。我们今后所讨论的只有第一种不压缩的情况，即biCompression为

BI_RGB的情况。

biSizeImage指定实际的位图数据占用的字节数。

biXPelsPerMeter：指定目标设备的水平分辨率，单位是每米的像素个数。

biYPelsPerMeter：指定目标设备的垂直分辨率，单位同上。

biClrUsed：指定本图像实际用到的颜色数，如果该值为零，则用到的颜色数为2的biBitCount次方。

biClrImportant：指定本图像中重要的颜色数，如果该值为零，则认为所有的颜色都是重要的。

3) 调色板(Palette)，当然，这里是对那些需要调色板的位图文件而言的。有些位图，如真彩色图，是不需要调色板的，BITMAPINFOHEADER 后直接是图像数据。

调色板实际上是一个数组，共有 biClrUsed 个元素(如果该值为零，则有 2 的 biBitCount 次方个元素)。数组中每个元素的类型是一个 RGBQUAD 结构，占 4 个字节，其定义如下：

```
typedef struct tagRGBQUAD {
    BYTE    rgbBlue;      /*该颜色的蓝色分量*/
    BYTE    rgbGreen;     /*该颜色的绿色分量*/
    BYTE    rgbRed;       /*该颜色的红色分量*/
    BYTE    rgbReserved;  /*保留值*/
} RGBQUAD;
```

4) 实际图像数据，对于用到调色板的位图，图像数据就是该像素颜在调色板中的索引值，对于真彩色图，图像数据就是实际的R，G，B值。下面就2色，16色，256色位图和真彩色位图分别介绍。

对于2色位图，用1位就可以表示该像素的颜色(一般0表示黑，1表示白)，所以一个字节可以表示8个像素。

对于16色位图，用4位可以表示一个像素的颜色，所以一个字节可以表示2个像素。

对于256色位图，一个字节刚好可以表示1个像素。

对于真彩色图，三个字节才能表示1个像素。

3.5 本章小结

本章主要讲解了 JPEG 的编解码过程，侧重点是解码过程，包括熵解码、反量化、反离散余弦变换、反取样及颜色空间转换。另外,还讲解了 JPEG 的标记码，标记码是解码时必需首先要处理的，因为它包含图像的信息。最后讲述了 BMP 文件格式，BMP 文件的前两部分完全可以由 JPEG 的标记码部分生成。

第四章 软件系统的基础构建与设计

嵌入式系统最终是要完成用户制定的任务、运行用户编写的程序，只有硬件环境和用户的应用程序是不够的。我们必需对系统的硬件进行相应的配置，使硬件进行正常的工作。这就关系到软件系统的基础构建与设计。本系统的基础构建包括 Bootloader 程序的编写、I/O 端口配置、串行口驱动、1*4 按键驱动、LCD 驱动等。本章着重讲解这些底层程序的编写及其运行环境。

4.1 交叉开发软件 ADS 介绍[36]

作为嵌入式系统应用的 ARM 处理器，其应用软件的开发属于跨平台开发，因此，需要一个交叉开发环境。交叉开发是指在一台通用计算机上进行软件的编辑编译，然后下载到嵌入式设备中进行运行调试的开发方式。用来开发的通用计算机可以是 PC 机、工作站等，运行通用的 Windows 或 Unix 操作系统。开发计算机一般称宿主机，嵌入式设备称目标机。在宿主机上编译好程序，下载到目标机上运行，交叉开发环境提供调试工具对目标机上运行的程序进行调试。

交叉开发环境一般由运行于宿主机上的交叉开发软件和宿主机到目标机的调试通道组成。运行于宿主机上的交叉开发软件最少必须包含编译调试模块，其编译器为交叉编译器。宿主机一般为基于 x86 体系的台式计算机，而编译后的代码必须在 ARM 体系结构的目标机上运行，这就是所谓的交叉编译。在宿主机上编译好目标代码后，通过宿主机到目标机的调试通道将代码下载到目标机，然后由运行于宿主机的调试软件控制代码在目标机上进行调试。ADS1.2 就是为了实现此目的而开发的。

ADS1.2(ARM Developer Suite)，是 ARM 公司推出的新一代 ARM 集成开发工具。ADS 由命令行开发工具，ARM 实时库，GUI 开发环境(Code Warrior 和 AXD),适用程序和支持软件组成。GUI 开发环境包括 CodeWarrior for ARM 和扩展调试器 AXD。

CodeWarrior 是一套完整的集成开发工具，为管理和开发项目提供了简单多样化的图形用户界面。CodeWarrior 是专为基于 ARM RISC 的处理器而设计的，充分发挥了 ARM RISC 的优势，它可加速并简化嵌入式开发过程中的每一个环节，使得开发人员只需通过一个集成软件开发环境就能研制出 ARM 产品。CodeWarrior 集成开发环境用户可以使用 ADS 的 CodeWarrior IDE 为 ARM 和 Thumb 处理器开发用 C，C++，或 ARM 汇编语言的程序代码。用 CodeWarrior 开发一般流程为建立工程，为工程建立或添加文件，编译连接工程，最后调试工程。

AXD 调试器为 ARM 的扩展调试器(即 ARM eXtended Debugger)，是 ADS 软件中独立于 CodeWarrior IDE 的图形软件，支持硬件仿真和软件仿真。ARMulator 是一个 ARM 指令集仿真器，集成在 ARM 的调试器 AXD 中，它能对 ARM 的指令集的仿真，为 ARM 和 Thumb 提供精确的模拟，是调试的时候最常用的一种调试工具，用户可以在硬件尚未做好的情况下，开发程序代码。硬件仿真时要在 AXD 的仿真目标配置中选择 ADP，才能进行硬件的仿真调试。本文就是先对程序进行硬件仿真调试，

调试通过后再将程序下载到硬件系统中，结合硬件系统进行整体调试。要用 AXD 进行代码调试，首先要把编译链接工程后生成的含有调试信息的.axf 映像文件装载到目标内存中。

组成 ARM 交叉开发环境的宿主机到目标机的调试通道一般有以下 3 种：

1) 在线仿真器 ICE

在线仿真器 ICE(In Circuit Emulator)是一种模拟 CPU 的设备。它使用仿真头完全取代目标板上的 CPU，可以完全仿真 ARM 芯片的行为，提供更加深入的调试功能。在与宿主机连接的接口上，在线仿真器也是通过串行端口或并行端口、网口、USB 口通信。在线仿真器为了能够全速仿真时钟速度很高的 ARM 处理器，通常必须采用及其复杂的设计和工艺，因而其价格比较昂贵。在线仿真器通常用在 ARM 的硬件开发中，在软件的开发中较少使用。其价格昂贵，也是在线仿真器难以普及的原因。

2) 基于 JTAG 的 ICD

JTAG 的 ICD(In Circuit Debugger)也称为 JTAG 仿真器，是通过 ARM 芯片的 JTAG 边界扫描口进行调试的设备。JTAG 仿真器通过 ARM 处理器的 JTAG 调试接口与目标机通信，通过并口或串口、网口、USB 口与宿主机通信。JTAG 仿真器价格比较便宜，连接比较方便。通过现有的 JTAG 边界扫描口与 ARM CPU 核通信，属于完全非插入式(即不使用片上资源)调试。它无需目标存储器，不占用目标系统的任何应用端口。通过 JTAG 方式可以完成：读/写 CPU 的寄存器，访问控制 ARM 处理器内核；读/写内存，访问系统中的存储器；访问 ASIC 系统；访问 I/O 系统；控制程序单步执行和实时执行；实时地设置基于指令地址值或数据值的断点。

3) Angel 调试监控软件

Angel 调试监控软件也称为驻留监控软件，是一组运行在目标板上的程序，可以接收宿主机上调试器发送的命令，执行诸如设置断点、单步执行目标程序、读/写存储器、查看或修改寄存器等操作。宿主机上的调试软件一般通过串行端口、以太网口、并行端口等通信端口与 Angel 调试监控软件进行通信。与基于 JTAG 的调试不同，Angel 调试监控程序需要占用一定的系统资源，如内存、通信端口等。驻留监控软件是一种比较低廉有效的调试软件，不需要任何其它硬件调试和仿真设备。Angel 调试监控程序的不便之处在于它对硬件设备的要求比较高，一般在硬件稳定之后才能进行应用软件的开发；同时它占用目标板上的一部分资源，如内存、通信端口等，而且不能对程序的全速运行进行完全仿真，在一些要求严格的情况下不是很适合。

基于 JTAG 仿真器的调试是目前 ARM 开发中采用最多的一种方式，本系统调试就采用此种方式。

4.2 系统启动程序 Bootloader

在32位处理器应用系统中，多数硬件模块都是可配置的，需要由软件来设置其需要的工作状态。因此在用户的应用程序之前，需要由专门一段代码Bootloader来完成对系统的初始化。由于Bootloader代码直接面对处理器内核和硬件控制器进行编程，

一般都是用汇编语言。Bootloader程序是硬件上电复位后首先要执行的部分，相当于PC机中的BIOS (Basic Input /Output System)，它存放于目标平台的非易失存储介质FLASH中，它依赖于实际的硬件和应用环境，因此要为嵌入式系统建立一个通用、标准的Bootloader是非常困难的[33-35]；它也依赖于具体的嵌入式板级设备的配置，这也就是说，对于两块不同的嵌入式主板而言，即使它们是基于同一CPU 而构建，要想让运行在一块板子上的Bootloader程序也能运行在另一块板子上，通常都需要修改Bootloader的源程序[37]。它可以完成硬件初始化、栈寄存器的设置、全局变量的初始化或清0、RAM中运行的模块的加载、堆参数的初始化等。Bootloader它完成这些工作后,再把控制权交给C的main函数。另外,为了避免产生混淆,我们还必须给main 函数另外取一个名字Main。否则,编译器将会给main 函数生成一大堆初始化代码，导致C程序的主入口与系统引导模块的接口错误。系统引导模块完成各种初始化工作后，用一条跳转指令进入C的主入口Main，控制权从此移交给了C应用程序。本系统根据完成目标和硬件平台对开发板所代的启动代码Bootloader进行了裁减和修改，实现了系统的启动引导程序Bootloader。系统启动流程如图4-1所示：

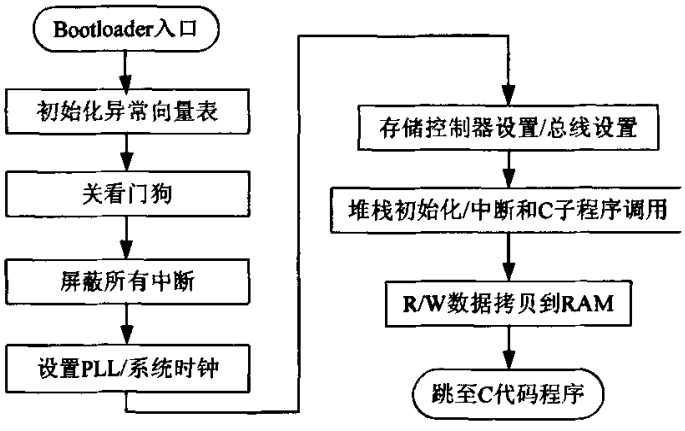


图 4-1 Bootloader 程序流程图

系统上电或复位后，先执行启动模式，ARM从地址0x0处开始执行，对硬件进行初始化。初始化主要包括的内容(其中/*.....*/为程序注释部分)：

1) 初始化中断向量表

```
AREA Boot ,CODE , READONLY   /* 代码区，只读*/
ENTRY:                       /*指定汇编程序的入口点*/
b ResetHandler    /* S3C44B0X复位后从此处执行，下面的代码用于出现异
                  常时CPU 就会根据以下的语句自动跳转到对应的异常处
                  理程序处*/
b HandlerUndef    /*未定义异常向量*/
b HandlerSWI      /*软中断向量*/
b HandlerPabort   /*取指异常向量*/
b HandlerDabort   /*取数据异常向量*/
```

```
b .                 /*保留*/
b HandlerIRQ        /*中断向量*/
b HandlerFIQ        /*快速中断向量*/
bl   Main           /*跳转到C入口*/
end
```

2) 禁止看门狗

```
ResetHandler:                       /*上电后跳转到此处开始执行*/
Ldr r0,=WTCON                       /*禁止看门狗*/
ldr r1,=0x0
str r1,[r0]
```

3) 屏蔽所有中断

```
ldr r0,=INTMSK                      /* 屏蔽所有中断请求*/
ldr r1,=0x07ffffff
str r1,[r0]
```

4) 设置时钟控制寄存器

```
ldr r0,=LOCKTIME
ldr r1,=0xfff
str r1,[r0]
.if PLLONSTART
ldr r0,=PLLCON              /* 设置PLL*/
ldr r1,=((M_DIV＜＜12)+(P_DIV＜＜4)+S_DIV)
str r1,[r0]
.endif
ldr r0,=CLKCON
ldr r1,=0x7ff8              /*所有单元时钟允许*/
str r1,[r0]
```

5) 初始化内存寄存器，指定内存总线宽度和数据格式。

```
ldr r0,=SMRDATA
ldmia r0,{r1-r13}
ldr r0,=0x01c80000          /* BWSCON 存储控制寄存器地址*/
stmia r0,{r1-r13}           /* r1 - r13 寄存器内容写入*/
```

6) 拷贝读写区域数据

在对必要的硬件初始化完毕后，就要为核心代码准备RAM空间，拷贝读写区域数据，将系统需要读写的数据和变量从ROM 拷贝到RAM里。这里包括RO、RW和ZI这3个段设置相应的内存映射向量，BootLoader 要将RO段复制到RW中，并将ZI 段清零。 代码如下：

```
IMPORT | Image   $$ RO $$   Base|      /*ROM 的开始地址*/
```

```
IMPORT | Image   $$ RO $$   Limit |    /* ROM 的大小*/
IMPORT | Image   $$ RW $$   Base|     /*RW 区的开始地址*/
IMPORT | Image   $$ RW $$   Limit |   /*RW 区的大小*/
IMPORT | Image   $$ ZI $$   Base |    /* Zero Ini2tialised 区的开始地址*/
IMPORT | Image   $$ ZI $$   Limit|    /*ZI 区的大小*/
LDR    r0 , = | Image $$ RO $$Limit|
LDR    r1 , = | Image $$ RW $$ Base|
LDR    r3 , = | Image $$ ZI $$ Base|
CMP r0, r1                          /* 检查是否相同*/
BEQ F1                              /* 相同则跳过拷贝操作*/
F0:
CMP r1, r3                          /* 执行拷贝操作*/
LDRCC r2, [r0], #4
STRCC r2, [r1], #4
BCC F0
F1:
LDR r1, =Image_ZI_Base              /* 零数据准备区起始地址*/
MOV r2, #0
F2:
CMP r3, r1      /*执行数据区清零*/
STRCC r2, [r3], #4
BCC F2
```

7) 设置堆栈

```
ldr sp, =SVCStack
```

由于ARM有7种执行状态都要有自己独立的物理堆栈寄存器R13，在用户应用程序的初始化部分，一般都要初始化每种模式下的R13，使其指向该运行模式的栈空间，这样，当程序的运行进入异常模式时，可以将需要保护的寄存器放入R13所指向的堆栈，而当程序从异常模式返回时，则从对应的堆栈中恢复，采用这种方式可以保证异常发生后程序的正常执行。

```
bl InitStacks                       /* 跳转至其它堆栈初始化程序并返回*/
```

8) 跳转到应用程序Main函数

接下来执行BL Main 转到由C语言编写的核心程序,嵌入式操作系统的内核就可通过该C 程序加载到RAM，接管对系统的控制权。本系统是无操作系统的嵌入式系统，执行该语句就直接跳转到应用程序，即就是解码程序。

4.3 驱动设计

驱动程序为应用程序设计者提供了使用CPU内部资源及外围设备的接口，把操作

系统(软件)和硬件设备(硬件)分离开来。当外围设备改变的时候，只需要更换相应的驱动程序，不必修改操作系统的内核以及运行在操作系统中的软件，这样大大提高了软件的可移植性和可重用性。对于没有操作系统的嵌入式系统来说，把底层软件和硬件设备分离开来，当外围设备改变的时候，只需要更换相应的驱动程序，不必修改应用程序。

由于S3C44B0X集成了大部分常用设备的接口控制器，而且有些外部设备也是通过S3C44B0X的总线直接扩展的。所以对驱动开发主要是对S3C44B0X中相关CPU寄存器的操作。S3C44B0X中的CPU寄存器占据了Bankl中的0x01c00000--0x02000000的地址空间。文献[38]已给出各种内部控制器的相关寄存器的默认地址定义，这给设备驱动开发带来很大的方便，几乎所有的外设驱动都可以用标准C编写。在程序中，专门为寄存器开辟了一个头文件44b.h。

下面介绍UART，LCD和4*1键盘的驱动设计以及具体接口函数的编写设计。

4.3.1 UART 串行口程序设计

S3C44B0X有两个带DMA和中断的UART(Universal Asynchronous Receiver /Transmitter)，每个S3C44B0X的UART通用异步收发器提供两个独立的异步串行I/O端口，每个都可以在中断和DMA两种模式下工作。它们的支持的最高波特率为115200b/s。每个UART通道包含2个16字节FIFO(First In First Out)分别提供给接收和发送。它的作用是完成嵌入式系统与PC机之间的通信。

S3C44B0X的UART可以进行以下参数的设置：可编程的波特率，红外收/发模式，1或2个停止位，5位、6位、7位或8位数据宽度和奇偶位校验。

每个UART包含一个波特率产生器、发送器、接收器和控制单元。波特率发生器以MCLK作为时钟源。发送器和接收器包含16字节的FIFO和移位寄存器。要被发送的数据，首先被写入FIFO然后拷贝到发送移位寄存器。然后它从数据输出端口(TxDn)依次被移位输出。被接收的数据也同样从数据接收端口(RxDn)移位输入到移位寄存器，然后拷贝到FIFO中。

每个UART的波特率发生器为传输提供了串行移位时钟，时钟源可选为S3C44B0X的内部系统时钟。波特率的时钟通过一个16位分频器分频后产生，16位分频器的值由UBRDIVn寄存器具体说明。

UBRDIVn=(round_off)(MCLK/(波特率*16))-1；

本系统中UBRDIVn=(round_off)(60000000/(115200*16))-1=32；

UART的操作，包括数据发送、数据接收、中断发生、波特率发生、回送模式和自动流控制等内容。

S3C44B0X的UART控制器的功能较为复杂，本驱动程序的编写采用传统的查询——等待方式，使用其简单功能。按照软件可复用、可移植的设计思想，将串行口的驱动程序分解为如下三个核心接口函数：

Uart_Init(int mclk, int baud)：功能是初始化串口，设置波特率。

char Uart Getch(void)：功能是等待并接收串口数据，收到数据时返回，否则一直等待。

void UartSendByte(int data) 功能是发送串口数据，发送不成功便阻塞。

在以上三个核心函数的基础上可以产生其它接口函数，作简要说明：

void Uart_TxEmpty(int ch)； 等待直到串口发送缓冲区内部的数据发送完毕

char Uart_GetKey(void)； 从串口接收一个字符，如果没有收到数据返回0

void Uart_GetString(char *string)； 从串口获取一个字符串

void Uart_SendString(const char *pt)； 向串口送出一串字符

void Uart_Printf(const char *fmt,...)； 以标准输出格式向串口输出各种信息

串行口发送数据、接收数据的流程图如图 4-2、4-3 所示。

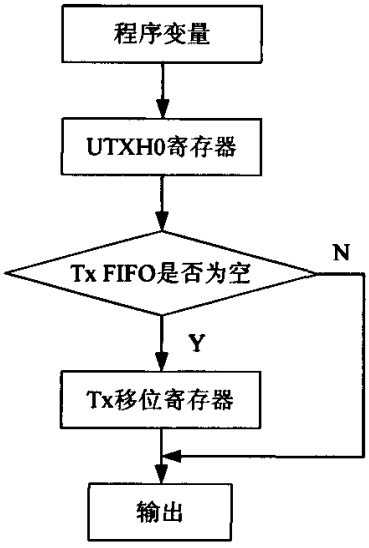


图 4-2 数据发送流程图

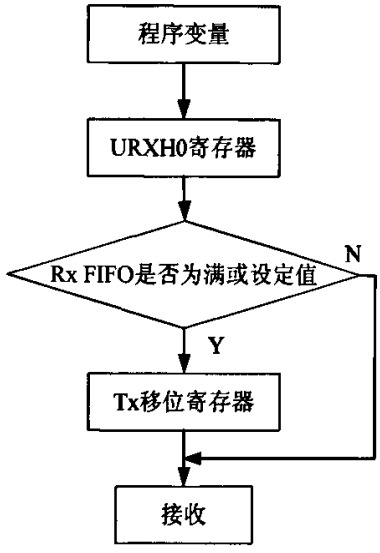


图 4-3 数据接收流程图

4.3.2 LCD 驱动程序设计

LCD通常有两种使用方式，一种是使用LCD本身自带的驱动控制器或者外部扩展的控制器，另一种就是使用处理器内部集成的LCD控制器来驱动LCD显示。本设计采用后一种方式，使用S3C44B0X内部集成的LCD控制器来驱动一个640*240点阵、256彩色的LCD模块。

LCD驱动过程包括：初始化端口、申请显示缓冲区、初始化LCD控制寄存器。

初始化端口：由S3C44B0的端口定义知道，S3C44B0X的PC口和PD口作为LCD驱动接口，因此需要设置PC口工作在第3功能状态，PD口工作在第2功能状态。

申请显示缓冲区：显示缓冲区就是在系统存储器中划出一块区域，用来存放要显示的图像数据。将要显示的图像数据直接放入显示缓冲区就能直接在LCD显示屏上显示出所显示的图像。因此，可以通过直接修改显示缓冲区的内容来完成显示的目的。显示缓冲区的大小与LCD显示屏的屏幕大小相同，为LCD显示屏的象素点的个数。显示缓冲区中的一个字节数据代表LCD上的一个象素点的信息颜色。

在点亮LCD之前，还应该对LCD控制器相关的寄存器进行初始化，使LCD控制器的配置与外接LCD显示模块特性相匹配，按照用户指定的工作方式运行。LCD初始化包括设置LCD分辨率、扫描频率、显示模式、产生控制信号和控制时序等。

下面给出LCD驱动，程序中的相关符号，我们在lcd.h头文件中已经定义了。其中/*……*/为注释部分。

1) I/O口LCD功能设置

设置I/O口控制寄存器的语句如下：

```
rPDATC = rPDATC &～ (1 << 8) | (1 << 8);          /*LCD使能*/
rPCONC = rPCONC &～ ( 0xff << 8) | ( 0xff <<8);   /* 配置VD[7：4 ]*/
rPCOND = 0xaaaa;                  /*配置VD[3：0 ]，VCLK，VLINE，VM，VFRAME*/
```

2) LCD初始化程序

LCD在显示之前必须对其控制器进行初始化，详细的LCD初始化代码如下所示：

```
void lcd_init ()
{
/*申请大小为640 ×240的缓冲区，其中ARRAY_SIZE_COLOR = 640 ×240*/
frameBuffer256 = (unsigned char * ) malloc (ARRAY_SIZE_COLOR) ;
/*初始化LCD控制器*/
rLCDCON1=(0)|(2<<5)|(1<<7)|(0<<8)|(0x03<<10)|(CLKVAL_COLOR<<12);
rLCDCON2=(LINEVAL)|(HOZVAL_COLOR<<10)|(1<<21);
rLCDSADDR1=(0x03<< 27) | ( ( (U32) f rameBuffer256 >>22) << 21) | M5D( (U32) f rameBuffer256 >>1) ;
rLCDSADDR2= M5D((((U32)frameBuffer256+(SCR_XSIZE*LCD_YSIZE))>>1)) | (MVAL<<21);
rLCDSADDR3 = ( ( LCD_ XSIZE ) / 2 ) |( ( (SCR_XSIZE-LCD_XSIZE) / 2)<<9) ;
/*设置红绿蓝查表寄存器*/
rREDLUT    = 0xfdb96420 ;
rGREENLUT = 0xfdb96420 ;
rBLUELUT   = 0xfb40 ;
rDITHMODE = 0x0 ;
/*设置抖动模式寄存器*/
rDP1_2 = 0xa5a5 ;
rDP4_7 = 0xba5da65 ;
```

```
rDP3_5 = 0xa5a5f ;
rDP2_3 = 0xd6b ;
rDP5_7 = 0xeb7b5ed ;
rDP3_4 = 0x7dbe ;
rDP4_5 = 0x7ebdf ;
rDP6_7 = 0x7fdfbfe ;
rLCDCON1=(1)|(2<<5)|(MVAL_USED<<7)|(0<<8)|(0x03<<10)|
(CLKVAL_COLOR <<12);
}
```

另外程序还给出了和LCD相关的程序，现说明如下：

void clrscreen(void)：清屏。

void lcd_put_pixel(int x,int y,unsigned char c)：在LCD屏上打点，x,y为LCD屏的坐标，c为打点色彩像素值。

4.3.3 1*4 按键驱动

在本文中，按键的目的是为了切换LCD终端的显示图像。我们在FLASH里面烧些了3幅640*240像素的JPEG图像，解码显示完一幅图像之后，可按下按键0、1、2、3进入下一幅图像处理过程。下面是按键驱动程序。

```
rPDATG=0xff;
rPCONG=0xffff;    /*rPCONG=0xffff,KEY0~KEY3定义为中断*/
rEXTINT=0x0; /*中断控制寄存器，低电平触发*/
BYTE GetKey(void)
{
   BYTE whichkey=0xf;
   whichkey=rPDATG>>4&0xf;
      if(whichkey!=rPDATG>>4&0xf)
      whichkey=0xff;
   switch(whichkey)
    {
      case 0xe:       /*1键按下*/
            return 1;
      case 0xd:       /*2键按下*/
          return 2;
      case 0xb:       /*3键按下*/
          return 4;
        case 0x8:       /*3键按下*/
            return 8;
```

```
        default:
            return 0xff;
    }
}
```

4.3.4 I/O 口的配置初始化

S3C44B0X共有71个多功能复用的I/O，大多数I/O的功能在硬件系统设计时已经确定下来，并且在系统正常运行时，将不能被改变。它们包含在A~G 7个端口中。每组端口都可以通过软件配置寄存器来满足系统和设计的需要，在运行程序之前必须先对每一个用到的引脚的功能进项设置。那些复用功能没有被使用的I/O，往往被配置成普通的数据输入输出端口。

S3C44B0X的71个I/O分为7组，

1) 一组10位的输出端口(Port A)；
2) 一组11位的输出端口(Port B)；
3) 一组16位的输入/输出端口(Port C)；
4) 两组9位的输入/输出端口(Port D和Port G)；
5) 两组9位的输入/输出端口(Port E和Port F)；

针对每一组端口，芯片分别有七个寄存器(PCONA~PCONG)来配置其功能，I/O的输入输出也由操作对应7个寄存器来实现。其功能用函数Port_Init()来实现。

4.3.5 内存管理接口

内存管理是一个复杂的课题。从广义的角度来说,磁盘文件系统、内存、片内高速Cache 等都属于这个范畴。用C语言进行嵌入式程序设计时，与在一般的PC机上编程一样，动态内存的分配管理是无法避免的，若内存管理不善(如内存泄漏)，就有可能导致程序出错甚至崩溃，这一问题在嵌入式软件编程中显得尤为突出。但是在编程和调试时，往往无法察觉这样那样的内存问题，而在投入运行时才会显露出问题。C语言中动态内存分配与释放主要由malloc和free 两个标准库函数实现。malloc从系统空闲内存中分配合适的内存块，free 函数完成内存块的回收。这两个函数一般需要操作系统内核的支持,在ARM裸平台上，不能直接调用。为此，我们将要自己编写malloc()和free()两个函数，实现动态存储管理的功能。所以在嵌入式系统中往往会重新定义内存管理函数。

下面是针对本硬件平台设计的malloc 和free。

```
void *mallocPt=Image$$RW$$Limit;
void*malloc(unsigned nbyte)
{
    void *returnPt=mallocPt;
    mallocPt= (int *)mallocPt+nbyte/4+((nbyte%4)>0);
    if( (int)mallocPt>HEAPEND)
```

```
        {
            mallocPt=returnPt;
            return NULL;
        }
          return Pt;
    }

    void free(void *pt)
    {
        mallocPt=pt;
    }
```

4.3.6 延时函数设计

延时函数在嵌入式应用中常常用到。在本系统中，图像开始显示到完全清晰显示需要一段时间。我们是利用看门狗定时器实现延时的。下面是延时函数。参数time表示time个100μm。

```
void Delay(int time)
{
    int i, adjust=0;
    if(time==0)
    {
        time=200;
        adjust=1;
        delayLoopCount=400;
        rWTCON=((MCLK/1000000-1)<<8)|(2<<3);      /*禁止看门狗定时器*/
        rWTDAT=0xffff;
        rWTCNT=0xffff;
        rWTCON=((MCLK/1000000-1)<<8)|(2<<3)|(1<<5); /*启动看门狗定时器
*/
    }
    for(;time>0;time--)
        for(i=0;i<delayLoopCount;i++);
            if(adjust==1)
        {
         rWTCON=((MCLK/1000000-1)<<8)|(2<<3); /*禁止看门狗定时器*/
         i=0xffff-rWTCNT;
           delayLoopCount=(delayLoopCount* time*100) /(i*64);
```

```
        }
}
```

本程序必须先执行Delay(0)语句，Delay(0)的作用是校准delayLoopCount。本程序的原理是：在执行双重循环之前打开定时器，在执行双重循环之后关闭定时器，寄存器rWTCNT每减少1，花费时间为64μm(第五章里面有计算过程)，所以就有关系式：

$$\frac{64*i}{\text{delayLoopCount}*\text{time}}=\frac{100}{\text{delayLoopCount}'} \tag{4.1}$$

delayLoopCount′为定时器运行100μm，for循环执行的次数。在后面执行Delay(time)仅仅是执行双重循环，delayLoopCount的值为上面4.1式delayLoopCount′的值，即就是校准之后的值。

4.4 本章小结

本章重点阐述了嵌入式系统软件的设计与实现，它包括系统引导程序Bootloader，UART驱动、LCD驱动、I/O口的配置初始化、内存管理接口函数及延时函数的设计。它为下一步JPEG图像的解码、显示提供了一个基础平台。另外还介绍了一下交叉编译环境ADS1.2。

第五章 JPEG 解码及 LCD 显示在 ARM 平台上的实现

完成底层软件开发之后，就要进行 JPEG 解码工作、显示工作的实现。本章将完成此项工作，并在此基础上，从软件和硬件两个方面讨论了对整个系统的优化，最后对解码测试结果、显示结果进行了分析。

5.1 总体方案设计

就解码器的设计方法而言，我们首先在 PC 机上，利用 C 语言对解码器进行原理验证，实现了基本的解码功能。此时，对于由 C 语言为基础实现的解码器，除了 PC 机外，在其它一些常用的处理器平台上，也能够正常运行并进行编码。但由于就具体的某个应用平台来讲，每个处理器有其各自不同的特点，解码器的性能可能会有很大的差异。所以，为了充分发挥处理器的优势和进一步提高解码器的性能，在实际应用时，还必须根据处理器的特点对软件进行必要的修改、优化等处理。由于本课题中的解码器最终的运行平台是 S3C44B0X，所以，在实现了编码器的原理验证后，还要将它移植到 S3C44B0X 上进行运行，并根据 S3C44B0X 的硬件环境和系统资源的特点采取了大量的优化和改进措施，使其实现了在 ARM 环境下的实时解码和显示。

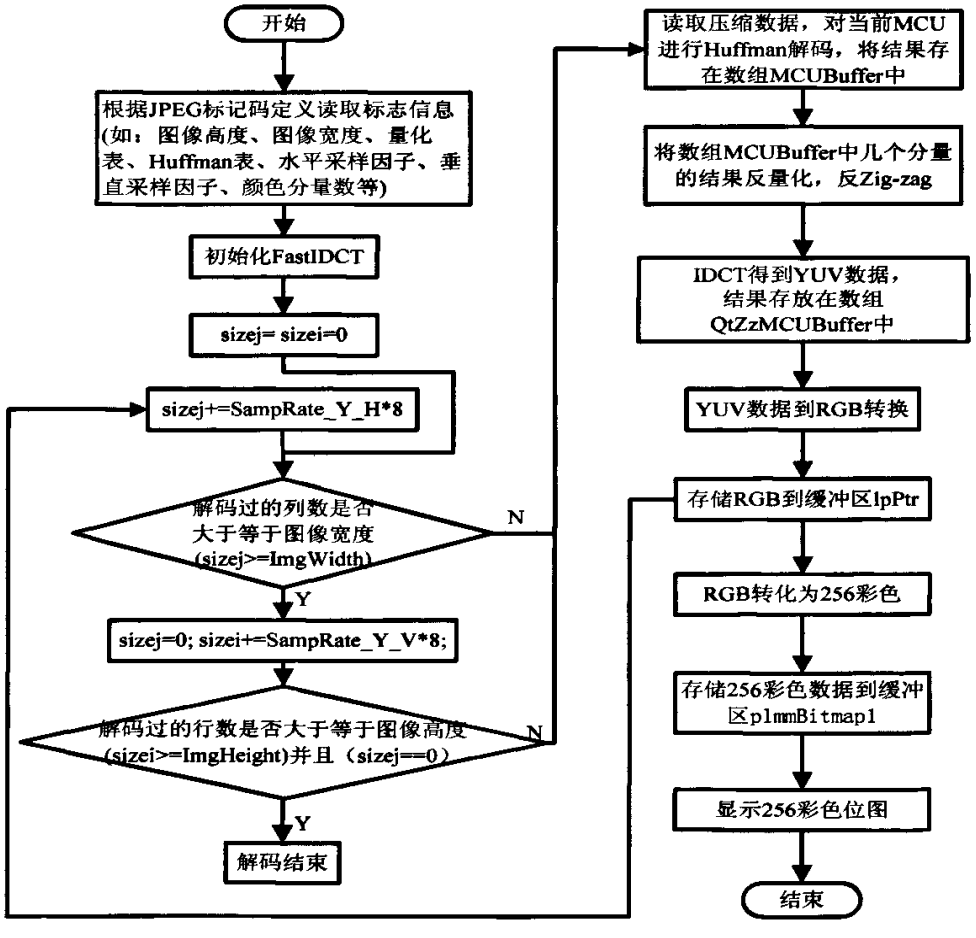


图 5-1 解码、显示流程图

对于嵌入式的实时解码的应用，最重要的问题之一是解码的实时性，即较低的延迟和较快的速度。在此基础上，尽可能追求更好的图像质量。所以，进行嵌入式的实时解码器设计，很多情况下需要在解码器的性能和解码速度之间进行恰当的折衷。

具体方案是用 tftpd32 软件将 JPEG 文件通过网口下载到 FLASH 某段地址(如0x60000)开始的一段空间(可以是几幅图像，对应几个地址)，程序开始从这一位置读取 JPEG 数据；具体流程如图 5-1 所示。

5.2 基于 PC 机的 JPEG 解码器的实现

上面我们谈到，要实现在ARM平台上的解码，首先在PC机上对解码程序进行验证。PC机的JPEG解码器的主要算法包括熵解码算法、反量化算法、反离散余弦算法，而熵解码算法、反离散余弦算法是解码器设计的核心，直接决定了解码器的性能。实际上，解码器的绝大部分的处理过程和处理时间都用于实现这些算法，其算法的复杂程度和优化程度最终决定着解码器系统的性能指标。所以算法的选择是非常关键的。

由于嵌入式系统应用对于解码器实时性的要求，以及它可用的系统资源的限制，在解码器的设计过程中尽量采用了一些目前被广泛采用的较为成熟的算法。相对而言，这些算法实现起来复杂程度不是很高，因而对系统资源也没有太高的要求。同时，由于这些算法的成熟性，所以有一些比较好的快速算法可以采用。当然，从解码效率或图像的质量上来看，这些算法不一定是最好的，但从解码器的实时性以及总体性能的综合考虑来看，这些算法的选择是比较合理的。所以在PC机上设计解码器，必须考虑代码的最终应用平台。

由JPEG编码算法可知，每一幅位图图像在JPEG编码的时候都被划分成了若干个MCU，每个MCU可以由一个或两个或四个8*8的分块组成。在解码的时候，需要按照从左到右、从上到下顺序将压缩数据按分块为单位进行Huffman解码、量化、IDCT变换以及颜色空间转换，将数据存放到一个缓冲lpPtr中，重复上面过程直到结束，得到每一个像素的位图数据，从而完成解码。需要特别注意的是：一、解码从开始到得到YUV数据都是以单个MCU为单元的；二、缓冲lpPtr存放所用MCU经过颜色空间转换的数据，即RGB数据。

5.2.1 熵解码算法的实现

熵解码是解码过程中非常重要的一环，JPEG算法中提供了标准的Huffman码表，也允许用户选择自适应的Huffman码编码，在这种情况下，首先要统计输入图像的特性，先生成码树，再做反推得到各级Huffman码表。本平台用自适应的Huffman码表。

由第三章JPEG文件格式可知，JPEG标记码里定义了一张表来描述Huffman树，定义在DHT标记面。Huffman代码的长度限制在16bit内。一般一个JPEG文件里会有2类Huffman表：一个用于DC一个用于AC (实际有4个表，亮度的DC，AC两个，色度的DC，AC两个) 这张表是这样保存的：前面的16字节对应长度为1到16的Huffman码字的个数，接下来是这16个数字之和个字节，对应字节就是对应Huffman码字的等价数

字。

在Huffman解码端收到的是由变长码VLC和变长整数VLI组成的数据流。为了从这数据流中恢复编码前的DCT系数，我们在程序中定义了三张解码码表，分别是：huf_min_value[i]：码长为i的最小码字；huf_max_value[i]：码长为i的最大码字；code_len_table[i]：码长为i的码字的个数。这3张码表在子程序InitTag()中条件分支case M_DHT之后已经生成。

利用huf_min_value [i]、huf_max_value [i]、code_len_table[i]进行Huffman解码。图5-2表示解一个码字的流程，在程序中用子程序DecodeElement()实现，解码输出是一个8位的值value。

子程序HufBlock(BYTE dchufindex, BYTE achufindex)实现一个或Y或U或V块的解码，参数dchufindex对应或Y或U或V块的DC Huffman索引，参数achufindex对应或Y或U或V块的AC Huffman索引。

子程序DecodeMCUBlock()实现一个MCU的Huffman解码。将当前解码数据存储到数组MCUBuffer中。

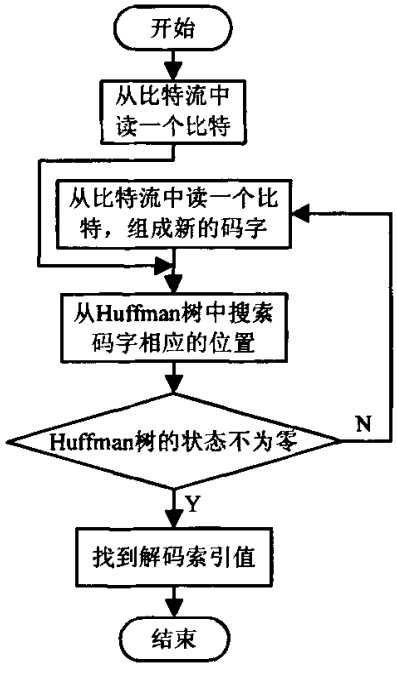


图5-2 Huffman解码解一个码字的流程

5.2.2 反量化和逆 Zig-zag 排序

JPEG的标记码里包含了量化表，定义在标记DQT后，一般来说，一个DQT定义一张量化表，JPEG文件里一般含有两个量化表，一个亮度Y分量的量化表，一个色度U分量和V分量共用的量化表。一般在JPEG文件里都有自带的量化表，也可以采用JPEG标准推荐的量化表。反量化熵解码出来的系数即是对解码系数乘上对应的量化阶矩，如公式3.1所示。

反量化之后，要对其系数矩阵进行逆Zig-zag变换，反Zig-zag变换后的数据流再送IDCT模块进行反离散余弦变换。

我们在JPEG.H文件中定义了一个Zig_Zag[8][8]数组来完成逆Zig_Zag变换。

```
Zig_Zag[8][8]={  {0,1,5,6,14,15,27,28},
                 {2,4,7,13,16,26,29,42},
                 {3,8,12,17,25,30,41,43},
                 {9,11,18,24,31,40,44,53},
                 {10,19,23,32,39,45,52,54},
                 {20,22,33,38,46,51,55,60},
                 {21,34,37,47,50,56,59,61},
                 {35,36,48,49,57,58,62,63}};
```

5.2.3 反离散余弦变换的选择和实现过程

IDCT一直是JPEG解码的关键技术，这是因为IDCT的运算量很大，在JPEG解码过程中所占时间最多，因而在一些高速或实时场合能否快速实现IDCT就成为一个关键因素。本文中的二维IDCT算法采用行列分解法，对一维IDCT采用chen-wang快速算法。下面给出8点一维IDCT的公式：

$$f(x)=\frac{1}{2}\sum_{u=0}^{7}C(u)\cdot F(u)\cdot\cos\frac{(2x+1)\pi u}{16} \tag{5.1}$$

其中 $C(u)=\begin{cases}\frac{1}{\sqrt{2}}, & (u=0)\\ 1, & (u>0)\end{cases}$

由(5.1)式可以看出，8点的一维IDCT需要计算64次乘法，56次加法，计算量很大。可以对算法作一些变换。用向量 f 表IDCT的结果，F 表DCT的输入，8点一维IDCT可以写成矩阵形式，如式(5.2)所示。

$$\begin{bmatrix}f0\\f1\\f2\\f3\\f4\\f5\\f6\\f7\end{bmatrix}=\frac{1}{2}\cdot\begin{bmatrix}c4 & c1 & c2 & c3 & c4 & c5 & c6 & c7\\ c4 & c3 & c6 & -c7 & -c4 & -c1 & -c2 & -c5\\ c4 & c5 & -c6 & -c1 & -c4 & c7 & c2 & c3\\ c4 & c7 & -c2 & -c5 & c4 & c3 & -c6 & -c1\\ c4 & -c7 & -c2 & c5 & c4 & -c3 & -c6 & c1\\ c4 & -c5 & -c6 & c1 & -c4 & -c7 & c2 & -c3\\ c4 & -c3 & c6 & c7 & -c4 & c1 & -c2 & c5\\ c4 & -c1 & c2 & -c3 & c4 & -c5 & c6 & -c7\end{bmatrix}\cdot\begin{bmatrix}F0\\F1\\F2\\F3\\F4\\F5\\F6\\F7\end{bmatrix} \tag{5.2}$$

其中 $ci=\cos\left(\frac{i\cdot\pi}{16}\right),\ (1\le i\le 7)$

观察系数矩阵可以发现，上半矩阵和下半矩阵存在一定的对称性。定义向量

P和Q，

$$P=\begin{bmatrix} c4 & c2 & c4 & c6 \\ c4 & c6 & -c4 & -c2 \\ c4 & -c6 & -c4 & c2 \\ c4 & -c2 & c4 & -c6 \end{bmatrix}\cdot\begin{bmatrix} F0 \\ F2 \\ F4 \\ F6 \end{bmatrix} \tag{5.3}$$

$$Q=\begin{bmatrix} c1 & c3 & c5 & c7 \\ c3 & -c7 & -c1 & -c5 \\ c5 & -c1 & c7 & c3 \\ c7 & -c5 & c3 & -c1 \end{bmatrix}\cdot\begin{bmatrix} F1 \\ F3 \\ F5 \\ F7 \end{bmatrix} \tag{5.4}$$

$$\begin{bmatrix} f0 \\ f1 \\ f2 \\ f3 \end{bmatrix}=\frac{1}{2}\cdot P+\frac{1}{2}\cdot Q \tag{5.5}$$

$$\begin{bmatrix} f7 \\ f6 \\ f5 \\ f4 \end{bmatrix}=\frac{1}{2}\cdot P-\frac{1}{2}\cdot Q \tag{5.6}$$

向量P的系数矩阵也存在一定的对称性，可以用类似的方法继续分解：

最后，可以得到chen-Wang算法的计算流程，如图所示：

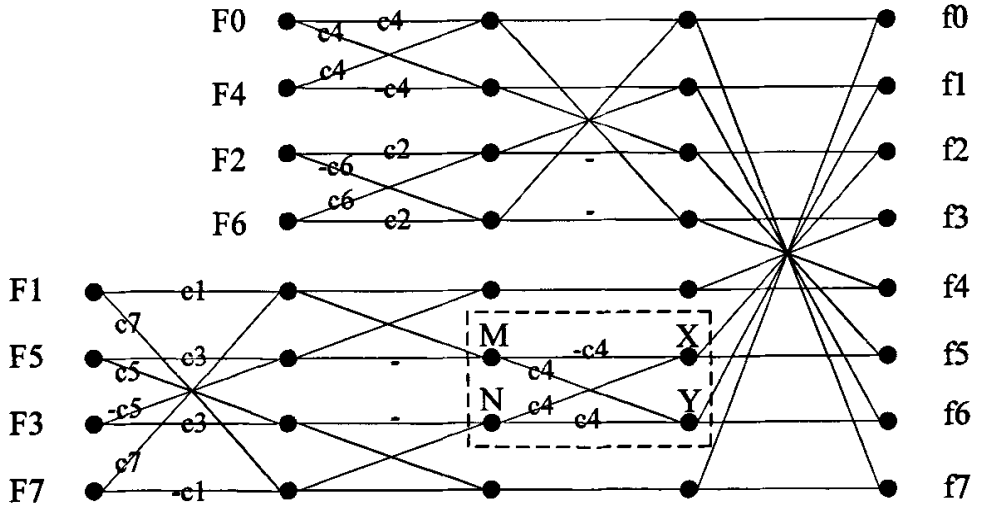


图5-3 IDCT快速算法蝶形图

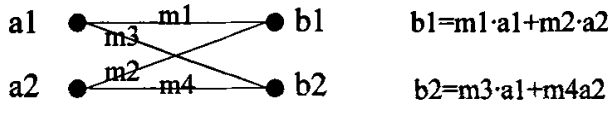


图5-4 图5-3图例说明

图5-3中虚线框的运算为：X=-c4·M+c4N, Y=c4·M+c4N,其中c4和M、N的系数*ci* 相乘，积化和差，得到新的系数。

由流程图5-3不难算出，运行8点的IDCT运算一共需要11次乘法和29次加法。可见快速算法比直接算法效率有很大的提高。

程序中函数 Fast_IDCT(int * block)实现一个8*8块的快速IDCT；函数idctrow(int * blk) 实现8*8块的行变换，函数idctcol(int * blk) 实现8*8块的列变换。IDCT的结果存放在数组QtZzMCUBuffer中。在程序中，我们对系数*ci* 做了相应的处理，避免了浮点运算。

5.2.4 反取样和色彩空间转化

由于我们最终要将解码结果显示出来，而显示设备LM7M632只支持RGB模式，不支持YUV模式，所以反离散余弦变换之后得到YUV数据，必须利用公式3.6对IDCT结果进行色彩空间转换，并将所有MCU块的转换结果RGB数据保存入lpPtr缓冲区中，同时根据BMP文件格式，可生成BMP文件。利用公式3.5对IDCT结果进行色彩空间转换的同时就完成了反取样。这里需要注意的是BMP文件和JPEG文件的扫描方式不同：JPEG文件是从左到右、从上到下扫描，而BMP文件是从左到右、从下到上扫描的，所以在程序处理时要格外关注，否则，解出来的图像正好是原图像的水平镜像。

5.3 24 位 RGB 到 256 彩色的转换

我们的显示设备LM7M632不支持24位真彩色显示，支持RGB332显示(256彩色)，这就要将24位RGB真彩色转化为RGB332模式。其转换由函数void tranRGBto256()实现。具体程序如下：

```
void tranRGBto256()
{
    int i,iRed,iGreen,iBlue;
    plmmBitmap1=(BYTE*)(malloc(sizeof(BYTE)*(ImgWidth*ImgHeight)));
     for (i = 0; i 〈ImgWidth*ImgHeight; i++) {
          iBlue= *lpPtr++;
          iGreen =*lpPtr++;
          iRed= *lpPtr++;
          iRed = (iRed*7+127)/255;
          iGreen = (iGreen*7+127)/255;
          iBlue = (iBlue+42)/85;
          plmmBitmap1[i]=iBlue+(iGreen＜＜2)+(iRed＜＜5); }
}
```

5.4 软件的移植

所谓移植，就是使一个实时操作系统能够在某微处理器平台或微控器上运行[39]。现在要将JPEG解压缩程序从PC机移植到ARM微处理器平台。由于在Visual C++和ADS1.2中，数据类型完全相同，不同的只是这些代码被不同的编译器生成不同的目标文件，在不同的硬件平台上运行，所以移植相对比较容易。

在移植之前，必须对原程序做一些修改，去掉那些读JPEG文件和写BMP文件的语句，这是因为本系统无操作系统，它不支持文件系统格式。

首先，在交叉编译环境ADS下建立工程，将Bootloader代码、UART代码、初始化代码、LCD驱动代码、色彩转换代码、主程序代码等添加进工程文件，设置好运行环境。运行环境设置主要包括程序入口地址0xc008000和编译生成二进制文件名decode.bin及路径等。

然后编译工程文件，调试修改程序，直到没有错误为止，这时生成了二进制文件decode.bin，将此文件烧入FLASH中(以0x50000为起始地址)，占用空间为该文件大小，此处特别注意，烧文件之前先要清除FLASH中该位置空间。

另外，由于此开发板默认程序入口地址为0xC008000,所以在超级终端中必需设置一句命令：setenv bootcmd cp 0x50000 0xc008000 1c47;go 0xc008000。其目的在于启动时将刚才烧入FLASH的代码复制到以0xC008000地址为首的地址空间中，以便程序能够自动加载运行，其中1c47表示decode.bin代码占用0x1c47个字节。然后重新启动开发板，解码程序将自动运行，我们可以在LCD屏幕上看见解压缩之后的图像。

我们可以利用1*4键盘实现几幅JPEG图像的解码与显示，当按下1*4键盘的任一键时，系统进入下一幅图片的解码与显示。

5.5 优化

在嵌入式实时系统中，对程序的运行速度要求很高，因此在此平台上的程序在保证正确性的前提下，最大限度地提高运行速度是我们追求的另一目标。而在实际程序开发中，算法逻辑实现阶段常常仅仅保证了逻辑上的正确性，没有考虑运行平台的特点，代码没有达到平台最优，或运行速度达不到系统的要求。所以在ARM上的程序开发中，除了算法实现阶段外，代码优化也必不可少。众所周知，优化代码是要花费时间，而且会降低原代码的可读性。所以通常只对经常被调用且对性能影响较大的函数进行优化。下面从 C 语言和硬件方面对程序进行了优化，从而提高了程序的运行速度。

5.5.1 从基本的C数据类型优化

ARM处理器内部是32位寄存器和32位的数据处理操作，其体系结构是RISC load/store结构，换句话说，数据在使用前必须先将其从内存装载到内部寄存器，任何算术或者逻辑指令都不能直接在存储器里进行数据操作。

8位或16位数据装载/存储ARM寄存器之前，先要扩展成32位。无符号数把0作为

扩展位，有符号数就照符号位扩展。这就意味着装载int类型的数据无需花费多余的指令时间进行位扩展。同样在存储时，8位或16位的数据必须放在寄存器的低8位或低16位。而一个int类型的传送，存储时就不要花费额外的指令时间了。表5.1给出了在ADS环境下，C编译器数据类型映射。

所以，对于存放在存储器中的局部变量，除了8位或16位的算术模运算外，尽量不要使用char和short类型，而要使用有符号或无符号int类型。

表5.1 C编译器数据类型映射

C数据类型	表示的意义
char	无符号8位字节数据
short	有符号16位半字数据
int	有符号32位字数据
long	有符号32位字数据
long long	有符号64位双字数据

5.5.2 避免使用浮点运算

S3C44B0X处理器硬件上不支持浮点运算[40]。这样在一个对价格敏感的嵌入式应用系统中，可节省空间和降低功耗；但这并不意味着不能计算浮点运算，因为C编译器在软件上支持浮点运算。实际上，C编译器要把每一个浮点操作转换为一个子程序调用。C库函数中的子程序使用整形运算来模拟浮点操作。这些代码是用高度优化的汇编语言编写的。尽管如此，浮点运算执行起来还是要比相应整形运算慢的多。

在头文件JPEG.H中，我们定义了W1、W2、W3、W5、W6、W7，它是为解码中IDCT准备的。实际上，它是在IDCT之前对每一个系数左移11位，即扩大2^{11}，在IDCT之后，再右移11位，从而避免了浮点运算。

5.5.3 充分利用寄存器分配[41]

编译器会试图对C函数中的每一个局部变量分配一个寄存器。如果几个局部变量不会交替使用，那么编译器会对它们分配同一个寄存器。当局部变量多于可用的寄存器时，编译器会把多余的变量存储到堆栈。由于这些变量被写入了存储器，所以被称为溢出或者替换变量，就像虚拟存储器的内容被替换到硬盘一样。与分配在寄存器中的变量相比，对溢出变量的访问要慢得多。

理论上，C编译器可以分配14个变量到寄存器而不溢出。实际上，一些编译器对某些寄存器有特定的用途(如用r12作为临时过渡寄存器使用，r13用作堆栈指针，r14用作连接寄存器(保存子程序的返回地址))，编译器就不再分配任何变量给它了。因此，应尽量限制函数内部循环所用局部变量的数目，最多不超过12个，这样，编译器就可以把这些变量都分配给ARM寄存器。

5.5.4 避免使用除法

ARM硬件上不支持除法指令[42]。编译器是通过调用C库函数来实现除法运算的。

除法和模运算(/和%)执行起来比较慢，所以应尽量避免使用。如果不能避免除法运算，那么就尽量使除数和被除数都是无符号的整数，有符号的除法执行起来会更慢，因为他们先要取得除数和被除数的绝对值，在调用无符号除法运算，最后再确定结果的符号。一般情况下，对于重复对同一个除数的除法，我们可以把除法变为乘法运算，即乘以它的倒数。

5.5.5 从函数调用方面优化[43]

ARM过程调用标准ATPCS(ARM-Thumb Procedure Call Standard)定义如何通过寄存器传递函数参数和返回值。ATPCS定义了寄存器组中的(r0~r3)作为参数传递和结果返回寄存器， 如果参数数目超过四个，则使用堆栈进行传递。我们知道内部寄存器的访问速度是远远大于存储器的，所以要尽量使参数传递在寄存器里面进行，即应尽量把函数的参数控制在四个以下，也可以将几个相关的参数组织在一个结构体中，用传递结构体指针来代替多个参数。这是理解ATPCS后，应该实现的一种编程风格。

另外，在调用函数时，如果函数体很小，只用到很少的寄存器(很少的局部变量)，可以把调用函数和被调用函数放在同一个C文件中，并且要先定义，后调用，这样编译器就知道了被调用函数生成的代码，并以此对函数调用进行优化。

5.5.6 避免指针别名[42]

当两个指针指向同一个地址对象时，这两个指针被称作该对象的别名。如果对其中一个指针进行写入，就会影响从另一个指针的读出。在一个函数中，编译器通常不知道哪一个指针是别名，哪一个不是；或哪一个指针有别名，哪一个没有。所以，编译器认为，对任何一个指针的写入，都将会影响从任何其它指针的读出，但这样会明显降低代码执行的效率。因此不要依赖编译器来消除包含存储器访问的公共子表达式，而应建立一个新的局部变量来保存这个表达式的值，这样可以保证只对这个表达式求一次值。

5.5.7 使用高速缓存机制

该嵌入式系统使用了高速缓存机制，因为直接访问内存会使指令执行时间变长。带有Cache的ARM内核采用了两种总线结构：冯·诺依曼结构和哈佛结构。这两种总线结构的区别在于，是否在内核与主存之间将指令和数据通道分离。在使用冯·诺依曼结构的处理器内核中，只有一个数据和指令公用的Cache被称作统一Cache(或混合Cache)，它可以存储指令和数据。S3C44B0X就是冯·诺依曼结构，所以它是统一的Cache。

由于S3C44B0X处理器有8KB Cache(高速缓冲存储器)，可以以3种方式使用，第一是8KB的存储空间作为8KB的统一(指令或数据)Cache；第二，内部存储器可以用作一个4KB的统一Cache和一个4KB的内部SRAM；第三，内部存储器可以整个地用作8KB的内部SRAM。

Cache即高速缓冲存储器，是位于CPU与主存之间一种容量较小，但速度很高的

存储器。由于CPU在进行运算时，所需的指令和数据都是从主存中提取的，而CPU运算速度要比主存读写速度快得多，这样极其影响整个系统的性能。采用Cache技术，即在Cache中存放CPU常用的指令和数据，然后将这些数据和指令以一定的算法和策略从主存中调入，使CPU可以不必等待主存数据而保持高速操作。这样就满足了嵌入式系统实时、高效的要求。Cache有两个主要组成部分：Cache控制器和Cache存储器。Cache存储器是一个专用的存储器阵列；而Cache控制器是一种硬件，它将主存中的数据或代码自动拷贝到Cache存储器中，Cache控制器在不为应用软件所知情况下，自动完成搬移工作，所以，同一个应用软件不用修改，就可以在Cache和没有Cache的系统中运行。

S3C44B0X的Cache提供整个对Cache进行使能/禁止的功能。可以通过设置SYSCFG中的CM位为01或11来使能Cache，或者将SYSCFG[2:1]设置为00来禁止Cache。如果Cache被禁止，指令和数据将一直从外部存储器中获取。另外，S3C44B0X提供2个非Cache区域。每个非Cache区域给出了每个非Cache区域的开始地址和结束地址。在非Cache区域中，当发生了Cache读未命中，Cache区域中的数据不会更新。有时，如果将数据区域被设计为非Cache区域，程序执行速度将会提高，因为大多数变量不会被再利用。

下面代码使能Cache及设置不使用高速缓存的区域：

```
#define rSYSCFG(*(volatile unsigned *)0x1c00000)
#define WRBUFOPT (0x8)      /*允许写缓冲操作*/
#define SYSCFG_0KB (0x0|WRBUFOPT)
#define SYSCFG_4KB (0x2|WRBUFOPT)
#define SYSCFG_8KB (0x6|WRBUFOPT)
#define rNCACHBE0 (*(volatile unsigned *)0x1c00004)
#define rNCACHBE1 (*(volatile unsigned *)0x1c00008)
#define Non_Cache_Start(0x2000000) /*不能Cache访问的区域开始地址 */
#define Non_Cache_End(0xc000000) /*不能Cache访问的区域结束地址 */
rSYSCFG=SYSCFG_8KB;        /*使用8K字节的指令缓存*/
rNCACHBE0=((unsigned int)(Non_Cache_End>>12)<<16)|(Non_Cache_Start>
>12);  /*在上面的数据区域不使用高速缓存*/
```

5.6 解码及显示结果分析

就本课题而言，我们最关心的就是解码速度和图像质量。将优化之后的代码程序移植到ARM平台上之后，需要测试程序在实际环境中的运行时间和效果。

为了测一幅JPEG图像的解码时间，我们利用了看门狗定时器，编写了两个函数来实现此功能，这两个函数分别是Timer_Start(int divider)和Timer_Stop()。Timer_Start(int divider)启动定时器，而Timer_Stop()使定时器停止。在解码之前启动看门狗定时器，解码之后禁止看门狗定时器。

```
void Timer_Start(int divider)
{
  rWTCON=((MCLK/1000000-1)<<8)|(divider<<3); /*禁止看门狗定时器，确定时钟除数因子*/
  rWTDAT=0xffff;
  rWTCNT=0xffff;
  rWTCON=((MCLK/1000000-1)<<8)|(divider<<3)|(1<<5);  /*启动看门狗定时器*/
}

int Timer_Stop()
{
  int k;
  rWTCON=((MCLK/1000000-1)<<8); /*禁止看门狗定时器*/
  k=(0xffff-rWTCNT);
  return k;
}
```

由于系统时钟频率MCLK=60MHz,看门狗计数寄存器rWTCON倒计数一次的时间：t_wachdog=1/(MCLK/(预分频值+1)/除数因子)；而预分频值为rWTCON寄存器的位[15：8]的值，除数因子为1/64；所以t_wachdog=64 μ s；Timer_Stop()运行结果与t_wachdog的乘积就是解码时间。

对图像重建质量可以用PSNR(峰值信噪比)(Peak Signal to Noise Ratio)来评估。峰值信噪比越高则图像质量越好，反之图像质量越差。一般说来，当PSNR值在28以上时，图像质量差异不太显著，当高于35～40时，则肉眼分辨不出差异[44]。峰值信噪比的定义如下[45]：给定一幅大小为 $M*N$ 的数字化图像 $f(x,y)$ 和参考图像 $f_0(x,y)$，则图像 f 的 $PSNR$ 为：

$$PSNR = 10 \cdot lg \frac{f_{max}^2}{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f(x,y)-f_0(x,y)]^2} \tag{5.7}$$

引入另一常用质量评价指标均方误差 MSE ,其中 MSE 的表达式为：

$$MSE = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{n-1}[f(x,y)-f_0(x,y)]^2}{MN} \tag{5.8}$$

则：

$$PSNR = 10 \cdot lg \frac{f_{max}^2}{MSE} \tag{5.9}$$

其中 f_{max} 是函数 $f(x,y)$ 的最大灰度值，例如，在常用的8bit的灰度图像中，f_{max} 的最大值为255。

而在彩色数字图像中，由于图像的颜色用*RGB* 3基色的组合表示,每个颜色分量需用一个字节表示，于是相应的峰值信噪比可以表示为：

$$\overline{PSNR} = (PSNR_R + PSNR_G + PSNR_B)/3 \tag{5.10}$$

其中 $PSNR_R$、$PSNR_G$、$PSNR_B$ 分别为图像的*R*、*G*、*B* 帧的峰值信噪比。

图5-5是一幅640*240像素的压缩前的BMP格式的彩色图像，图5-6是它对应的YUV 4:1:1、压缩率为9.11的JPEG图像的解码位图。图5-7是它对应的YUV 4:2:2、压缩率为8.04的JPEG图像的解码位图。图5-8是它对应的YUV 4:4:4、压缩率为6.92的JPEG图像的解码位图。

表5.2给出了图5-5对应的不同YUV比例的JPEG图像的解码时间和峰值信噪比。通过对大量的图片的测试，解码一幅640*240像素图像的时间都在2秒以内，峰值信噪比都大于30分贝，所以图像重建质量相当高，同时解码器可以满足实时解码显示的要求。

表5.2　图5-5对应的不同YUV比例的JPEG图像的解码时间和峰值信噪比

YUV比例	压缩率	解码时间(s)	峰值信噪比(dB)
4:1:1	9.11	1.67392	30.615
4:2:2	8.04	1.95546	30.750
4:4:4	6.92	2.47053	30.687

图5-5　640*240大小的压缩前的BMP格式位图

图5-6 图5-5对应的YUV 4:1:1、压缩率为9.18的JPEG图像的解码位图

图5-7 图5-5对应的YUV 4:2:2、压缩率为8.49的JPG图像的解码位图

图5-8 图5-5对应的YUV 4:4:4 、压缩率为7.55的JPEG图像的解码位图

图5-9是图5-5对应的YUV4:1:1的LCD显示器显示的图像照片。由图像可以看出，虽然是256彩色显示，存在失真，但是总体而言显示效果还是不错的。

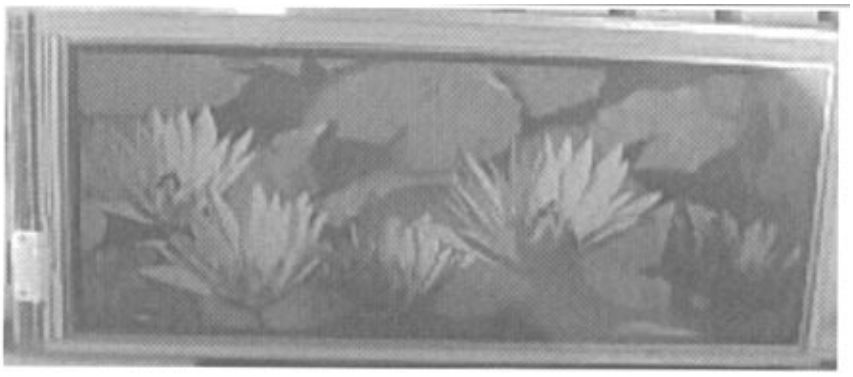


图5-9　图5-5对应的YUV4:1:1的LCD显示器显示的图像照片

5.7 本章小结

本章首先给出了解码芯片总体方案的设计，阐述了JPEG解码在PC机上实现的详细方法步骤，其次讲解了软件到硬件平台上的移植，然后从软件和硬件角度讲解了优化，最后对解码时间和图像重建质量进行了讨论，给出了测试结果。实验结果表明：本平台可以实现实时解码、显示，图像重建质量可以达到预期的效果。

第六章 总结与展望

本文主要研究基于ARM的图像解压缩技术，设计并成功实现了图像解压缩系统和显示系统。

本论文主要工作有：

(1) 阐述了软件系统的基础构建与设计

软件系统的基础构建主要做了以下工作：修改了开发商提供的Bootloader代码，对目标工作没用的代码予以裁减，以便满足自己工作的需要；对ARM参数进行初始化；编写开发板外围设备的驱动程序，如UART、LCD等。

(2) 编程实现了JPEG图像解压缩

认真研究图像编码原理及JPEG编解码标准，在Windows平台下用C语言实现了图像解码器。在软件设计中着重解决了以下问题：考虑到算法实现的复杂程度和嵌入式系统实时性的要求，选用了比较成熟高效的算法。

(3) 完成了JPEG图像解压缩程序调试、编译、优化以及程序到ARM平台的移植

由于PC机和ARM内部结构的不同，移植到ARM平台上程序效率低下，因此必须针对ARM的硬件特点，对程序进行了部分优化，使代码运行的速度有所提高。

实验结果表明：本设计实现的JPEG解码器能够正确完成JPEG图像的解码，解码速度较快，解码效果良好，可以应用于处理静止图像的场合(数码相机、数字手持设备等)，也可以对研究M-JPEG、MPEG、MPEG-II、H.26x有一定的参考价值。

通过本课题工作使我对JPEG标准、ARM ADS开发平台、LCD驱动模块和程序优化的方法有了更加深入的理解，并是我掌握了程序优化思想以及嵌入式系统软件开发流程等。这些为我今后的学习和研究工作打下了良好的基础，在硕士研究生学习阶段收获的知识也将会受用终生。

由于该课题的实践性较强，受限于具体应用开发环境和个人的知识背景以及时间等关系，本课题可以从以下几个方面进一步发展和完善：

1、虽然本文所设计的解码器速度基本能满足实时解码的要求，但针对ARM特性对程序可再做进一步优化，充分发挥ARM的潜能也是十分必要的。另外，虽然ADS集成环境在编译时会自动对代码进行相应的优化，但对运算耗时比较多的IDCT部分，可以用内嵌汇编来实现，那样软件运行的效率将会有所提高。

2、由于文中所用LCD是触摸屏，所以可以充分利用硬件资源，开发GUI(Graphics User Interface)图形用户界面。

3、由于开发板的FLASH只有2M字节，在实际应用中容量可能比较小，为此，可以进一步利用开发板的USB接口和IDE硬盘接口。

4、如果为了进一步提高解码速度，可以更换主频更高的ARM平台(如ARM9)，我们只需修改底层程序，就可以将解码程序移植过去，从而缩减开发周期。

当今网络与移动多媒体技术的飞速发展，对诸如图像、视频、音频的实时处理需求不断增强，而嵌入式也是未来的发展主流，因此本系统未来的应用会有更加广阔的前景。

致 谢

光阴荏苒，三年的硕士研究生学习即将结束。在攻读硕士学位期间，我得到了许多老师、同学和朋友的帮助，使我能顺利地完成学业。在这里，让我向他们表达深深的敬意和衷心的感谢。

首先，我要感谢我的导师屈百达教授。在我攻读硕士学位期间，屈老师的悉心指导和亲切关怀给了我很大帮助。屈老师思维敏捷，学识渊博，令人钦佩。他不仅教给我知识，更多的是启发我如何去学习和思考。他那严谨的治学态度和孜孜不倦的工作作风，使我受益终身。而他的谦虚和真诚依然令人感动。屈老师的培养使我能圆满地完成研究生阶段的学习和实践工作，在此我要向屈老师致以衷心的感谢。祝福屈老师工作顺利、事业顺心，能够创造更多的辉煌。

此外，李金宝、张鸿恺、肖连军、郭宏民还有同实验室的其他师弟师妹们，对我的研究课题提出了很多有价值的参考意见。在此，对他们表示诚挚的感谢。

感谢挚友邓学辉对我的大力支持和鼓励。愿他前程似锦。

感谢我的妻子杜喆和儿子对我的无私支持；特别感谢岳父岳母对妻子和儿子的悉心照顾和对我的理解；感谢父母对我的全力支持；感谢我的几位兄长和嫂子对我的全力支持；感谢姐姐的支持。祝愿他们快乐幸福！

最后，感谢所有曾经关心、帮助过我的人！

参考文献

1. *G. K. Wallace. The JPEG Still Picture Compression Standard[S]. IEEE Trans. Consumer Electronics.*1992, 38(1), 18-34
2. 小野定康、铃木纯司著,叶明译. JPEG/MPEG2 技术[M]. 北京:科学出版社,2003,62-65
3. 钟玉琢. 多媒体技术(高级)[M] . 北京:清华大学出版社,1999,1-3
4. 周立功,ARM 嵌入式系统基础教程[M]. 北京航空航天大学出版社,2005,1-2
5. David Salomon 著,吴乐南译. 数据压缩原理与应用(第二版),北京: 电子工业出版社,2003,427-435
6. *Megumi Takezawa, Hirofumi Sanada, Kazuhisa Watanabe, etc. Quality Improvement Technique for JPEG Images with Fractal Image Coding [J]. IEEE International Symposium.* 2005, 6, 6320 -6323
7. *Subramania Sudharsanan. Shared Key EnUyption of JPEG Color Images [J]. IEEE Transactions on Consumer Electronics,* 2005, 51(4), 1204-1211
8. *Shinfeng D. Lin, Shih-Chieh Shie, and Chung-Chien Chou. A Suitable Image Hiding Scheme for JPEG Images [J]. IEEE ICIP,* 2004, 3, 1565-1568
9. *Kyeong-Yuk Min; Jong-Wha Chong. A real-time JPEG encoder for 1.3 mega pixel CMOS image sensor SoC. Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE,* 2004, 3, 2633-2636
10. 车载低端图像数据采集压缩存储及传输系统的实现[EB/OL]. http:/ www.ca800.com/apply/ html/2007-1-12/n8172.html, 2005-9-30
11. 日经 BP 社报道.松下综合评价 UniPhier 现状,古池专务笑容满面[EB/OL]. http:/ china.nikkeibp.co.jp/china/news/elec/200510/pr_elec200510170120.html, 2005-10-17
12. 罗凤武. 基于 MCF5275 的 JPEG 解码算法的设计与实现[D].电子科技大学学位论文, 2003,
13. 魏忠义,朱磊. 基于 DSP 的 JPEG 图像解码算法的实现[J]. 现代电子技术,2005, 12,66-68
14. 金燕波，朗锐，罗发根等. 利用 TMS320C6201 DSP 芯片进行图像压缩[J]. 电子技术应用, 2004, 1(1): 63-66
15. 章承科.多核处理器架构的高速 JPEG 解码算法[J].单片机与嵌入式系统应用,2006,1,44-47
16. 魏璞. JPEG 解码算法在多 CPU 嵌入式系统中的实现及性能优化[D].电子科技大学学位论文,2006
17. 汪宇飞. JPEG 高速编码芯片的设计及性能优化[D].西北工业大学学位论文,2006
18. 尹伟. 基于 FPGA 的编解码芯片的设计[D].大连理工大学学位论文,2004

19. 李岩,荣盘祥. 基于S3C44B0X嵌入式μCLinux系统原理及应用[M].北京:清华大学出版社,2005,19-24,109-117。
20. 陈赜.ARM嵌入式实践教程[M].北京:北京航空航天大学出版社,2005, 142-149
21. 吕凤军.数字图像处理编程入门[M]. 北京:清华大学出版社,1999, 105-109
22. 魏本杰,刘明业, 章晓莉. 适用于嵌入式系统的二维DCT算法[J].计算机应用, 2005, 25(4),772-774.
23. 杜相文, 陈贺新, 赵岩. 基于查表的无乘法DCT快速算法[J].计算机工程,2004, 30 (20), 159-160.
24. *Feig E., Winograd S . . Fast algorithms for the discrete cosine transform [J]. IEEE Transactions on Signal Processing,* 1992, 40(9), 2174-2193
25. *Leoffler C., Ligtenberg A., Moschytz G. S.. Practical fast 1D DCT algorithms with 11 multiplications [J]. In Proceedings of the IEEE International Conference Acoustics, Speech and Signal Processing (ICASSP) , Glasgow ,* 1989 , 2 ,988-991
26. *Arai Y, Agui T, Nakajima M.A fast DCT-SQ scheme for images [J]. Trans IEICE,* 1988, 71, 1095-1097
27. *Lee B. G.. A new algorithm to compute the discrete cosine transforms [J]. IEEE Transactions on Acoustics, Speech, and Sig2nal Processing,* 1984, 32 (6), 1243 - 1245
28. *Chen W Het al. . A fast computational algorithm for the discrete cosine transforms[J]. IEEE Transactions on Communications,* 1977, 25 (9), 1004-1009
29. *Yeonsik Jeong, Imgeun Lee, Taekhyun Yun, Gooman Park, Kyu Tae Park. A fast algorithm suitable for DCT implementation with integer multiplication [J]. IEEE TENCON. Digital Signal Processing Applications,* 1996, 2,784-787
30. *Chin-Liang Wang. High-throughput VLSI architectures for the 1-D and 2-D DCT [J].IEEE Transactions on Circuits and Systems for Video Technology,* 1995, 5(1), 31-40
31. *Z.Wang, Fast Algorithms for the Discrete W-Transform and for the Discrete Fourier Transform[J],IEEE Trans ASSP,* 1984,32(4),803-816
32. 林福宗.多媒体技术基础[M].北京:清华大学出版社,2000,83-89
33. 万永波,张根宝,田泽等.基于ARM的嵌入式系统Bootloader启动流程分析[J], 微计算机信息.2005,21(11),90-92
34. 徐宇清,黄彦平,夏耘.S3C44B0X的BootLoader 技术分析[J].上海理工大学学报, 2005, 27(4),369-372
35. 刘娅.基于 ARM 嵌入式系统的 Bootloader 的设计与实现 [J].现代电子技术,2006,7,142 -144
36. 李驹光.ARM 应用系统详解——基于 S3C4510B 的系统设计[M].北京:清华大学出版社, 2003,240-246
37. 孙昊,曹玉强, 杜秀芳.ARM 处理器启动代码的分析与编程[J].工业控制计算机,

2005, 18(11),54-55

38. *Samsung Electronics User's Manual S3C44B0X 32BitRISC Microprocessor [G]. 2003.*
39. 田泽.嵌入式系统开发与应用教程[M].北京航空航天大学出版社,2005, 418-427
40. Andrew N.Sloss,Dominic Symes,Chris Wright著,沈建华译.ARM嵌入式系统开发[M],北京航空航天大学出版社,2005,93-146
41. 杜春雷.ARM体系结构与编程[M],北京:清华大学出版社,2003,230-237
42. 金丽,包志华,陈海进.嵌入式系统的 C 程序优化设计方法[J].南通大学学报,2005, 5(3), 61-63
43. 费浙平.基于ARM 的嵌入式系统程序开发要点(六)[J].单片机与嵌入式系统应用,2004,1, 84-86
44. 李红蕾,凌捷,徐少强.关于图像质量评价指标 PSNR 的注记[J].广东工业大学学报,2004, 21(3), 74-78
45. 王汇源.数字图像通讯原理与技术[M] . 北京:国防工业出版社,2000, 159-167

攻读硕士期间发表的论文清单

1. 蒙智明,屈百达,徐保国. 基于ARM的LCD控制及触摸屏接口设计.《微计算机信息》，已录用
2. 蒙智明,屈百达. 基于 TM320DM275 嵌入式系统的 JPEG 解码实现.《2007 中国控制与决策学术年会(19thCDC)》，已录用

附录：JPEG 编解码表

表1 JPEG编码分类表

数值范围	DC差分值分组	AC系数分组
0	0	不保存
-1,1	1	1
-3,-2,2,3	2	2
-7,…,-4,4,…,7	3	3
-15,…,-8,8,…,15	4	4
-31,…,-16,16,…,31	5	5
-63,…,-32,32,…,63	6	6
-127,…,-64,64,…,127	7	7
-255,…,-128,128,…,255	8	8
-511,…,-256,256,…,511	9	9
-1023,…,-512,512,…,1023	A	A
-2047,…,-1024,1024,…,2047	B	B
-4095,…,-2048,2048,…,4095	C	C
-8191,…,-4096,4096,…,8191	D	D
-16383,…,-8192,8192,…,16383	E	E
-32767,…,-16384,13684,…,32767	F	不保存

表2 直流DC差分值Huffman编码表

亮度DC差分值Huffman编码表			色度DC差分值Huffman编码表		
差值位数	编码位数	Huffman编码	差值位数	编码位数	Huffman编码
0	2	00	0	2	00
1	3	010	1	2	01
2	3	011	2	2	10
3	3	100	3	3	110
4	3	101	4	4	1110
5	3	110	5	5	11110
6	4	1110	6	6	111110
7	5	11110	7	7	1111110
8	6	111110	8	8	11111110
9	7	1111110	9	9	111111110
10	8	11111110	A	A	1111111110
11	9	111111110	B	B	11111111110

表3　AC系数 Huffman编码表(部分)

亮度AC系数 Huffman编码表			色度AC系数 Huffman编码表		
Run/Size	码长	码字(Value)	Run/Size	码长	码字(Value)
0/0(EOB)	4	1010	0/0(EOB)	2	00
0/1	2	00	0/1	2	01
0/2	2	01	0/2	3	100
0/3	3	100	0/3	4	1010
0/4	4	1011	0/4	5	11000
0/5	5	11010	0/5	5	11001
0/6	7	1111000	0/6	6	111000
0/7	8	11111000	0/7	7	1111000
0/8	10	1111110110	0/8	9	111110100
0/9	16	1111111110000010	0/9	10	1111110110
0/A	16	1111111110000011	0/A	12	111111110100
1/1	4	1100	1/1	4	1011
1/2	5	11011	1/2	6	111001
1/3	7	1111001	1/3	8	11110110
1/4	9	111110110	1/4	9	111110101
1/5	11	11111110110	1/5	11	11111110110
1/6	16	1111111110000100	1/6	12	111111110101
1/7	16	1111111110000101	1/7	16	1111111110001000
1/8	16	1111111110000110	1/8	16	1111111110001001
1/9	16	1111111110000111	1/9	16	1111111110001010
1/A	16	1111111110001000	1/A	16	1111111110001011
2/1	5	11100	2/1	5	11010
2/2	8	11111001	2/2	8	11110111
2/3	10	1111110111	2/3	10	1111110111
2/4	12	111111110100	2/4	12	111111110110
2/5	16	1111111110001001	2/5	15	111111111000010
2/6	16	1111111110001010	2/6	16	1111111110001100
2/7	16	1111111110001011	2/7	16	1111111110001101
2/8	16	1111111110001100	2/8	16	1111111110001110
2/9	16	1111111110001101	2/9	16	1111111110001111
2/A	16	1111111110001110	2/A	16	1111111110010000
3/1	6	111010	3/1	5	11011
3/2	9	111110111	3/2	8	11111000
3/3	12	111111110101	3/3	10	1111111000
3/4	16	1111111110001111	3/4	12	111111110111
3/5	16	1111111110010000	3/5	16	1111111110010001
…	…	…	…	…	…

基于S3C44B0X的JPEG图像解码及LCD显示的实现

作者： 蒙智明
学位授予单位： 江南大学

相似文献(10条)

1.学位论文 林薇 基于JPEG2000和ROI技术的医学图像压缩编码研究 2005

医学影像在临床诊断和手术导航中有很重要的意义，各种影像技术产生了大量的图像数据，为了存储与传输的目的，必须对医学图像进行有效压缩。

新一代图像压缩标准JPEG2000采用了小波变换，EBCOT及算术编码等算法，不仅具有优越的压缩性能，还支持感兴趣区域(RegionOfInterest：ROI)编码、码流率控制和渐进传输等新功能，它可为医学图像压缩提供优越的解决方案。

该文分析、总结了医学图像编码方案的四大特点，即：支持无损压缩；渐进编码特性；支持感兴趣区域编码；压缩过程的时间限制。

基于医学图像压缩的特点和JPEG2000标准的优越性能，研究在JPEG2000核心编码系统内，医学图像压缩编码的实现及所能达到的最佳压缩性能。对医学图像进行JPEG2000下的无损压缩和适当的有损压缩，通过计算峰值信噪比(PSNR)，对编码后的图像进行率失真评价；并对JPEG2000和JPEG标准在医学图像中的应用效果进行对比。实验结论为验证JPEG2000技术在医学图像压缩中的可行性和优越性提供了客观、可靠的证明。

引进感兴趣区域编码方案，很好地解决了医学图像压缩领域一直存在的压缩比与重建图像质量之间的矛盾。ROI编码处理能够满足人们在低比特率条件下对重要图像信息实现高质量恢复的要求。该文基于ISO/ITU-T公布的JPEG2000Part Ⅰ标准，将感兴趣区域编码应用于医学图像压缩，重点研究和分析ROI编码中各编码参数对医学图像压缩效率的影响，并对ROI编码和无ROI编码进行了率失真性能方面的对比。

实现了对矩形和任意形状的感兴趣区域的编码。任意形状的ROI编码主要针对图像中的组织区域进行，首先对图像进行分割处理，提取对诊断有重要意义的组织区域，再用这部分信息构建二进制掩模，然后采用Maxshift算法实现感兴趣区域编码。通过对大量图像数据的处理，结果显示：在ROI编码中，ROI区域的大小和数量、编码块大小和目标码率这三个因素对编码性能的影响最大，并且ROI编码技术的采用可在压缩比和率失真性能方面提高医学图像的压缩效率。实验结论丰富和拓展了感兴趣区域编码技术在医学图像压缩中的具体应用。

该项研究采用JJ2000软件实现了JPEG2000及ROI编码技术在医学图像压缩中的应用。

2.会议论文 段鹏.刘钊 3D DCT视频图像压缩技术的改进 2008

本文介绍了对3D DCT视频压缩方法的两方面改进。首先,用在文中被称作“背景复用”的技术,对判决为背景的视频立方体进行复用。从而对在多个相邻帧中变化都不大的块,只用一个标志位便可编码。另外,对JPEG标准建议的Huffman编码方案进行了改进,使其更适合于3DDCT视频编码。这种改进是用“换码”的方式对多于15个的连续零用另一套Huffman码表进行编码,而非采用JPEG标准中的“0xF/0”符号。基于上述两方面的改进,在视觉质量、编码器复杂度和存储空间消耗都较令人满意的情况下,提高了压缩比。

3.期刊论文 范为菊.姜培刚.宋均思.詹勇.FAN Wei-ju.JIANG Pei-gang.SONG Jun-si.ZHAN Yong 一种新的静态图像压缩编码算法的研究 -自动化技术与应用2010,29(1)

在静态图像传输中,针对含有噪声的数字图像去除相关性差的问题,借助数字图像的噪声模型,对图像作去噪处理后用静态图像压缩标准JPEG基本系统对图像压缩编码,采用VC++6.0实现该算法.实验结果表明,该方案具有较高压缩比、信噪比等良好性能,具有广泛的应用价值.

4.期刊论文 杨进.刘建波.Yang Jin.Liu Jianbo 基于区域特征加权的JPEG改进算法 -计算机应用与软件2007,24(11)

JPEG标准在对彩色图像编码时,为了提高图像的压缩率,需要将图像从RGB颜色空间转换到YCbCr颜色空间并对颜色分量Cb与Cr进行重采样.颜色空间转换与重采样会引起还原图像光谱畸变现象.在研究图像融合技术的基础上,提出一种基于区域特征的动态加权色彩空间变化方法.根据主观目视和客观评价指标,对还原图像进行了比较和分析.仿真试验表明,提出的方法可以提高还原图像的光谱分辨率,同时很好保持了原图像的空间分辨率.

5.学位论文 刘成文 基于多平台图像编解码研究与技术实现 2008

本文以北京市教委的科研课题——基于DSP的嵌入式图像WEB服务器的构建为应用背景，作者展开了在PC和DSP两个平台上对于数字图像编解码的研究以及技术实现。

本论文首先介绍了图像编码的意义、图像编码原理及图像编码国际标准。然后重点介绍了6000系列DSP的特性及软件开发环境CCS。认真研究了JPEG标准之后，在PC机上用C语言实现了JPEG编码器的软件原形。但因其目的是用硬件实现图像编码器，所以必须把PC机程序修改之后移植到DSP上，主要是把文件操作等改成针对DSP的内存操作。使用C6711DSP和RCM3200设计并构建一个嵌入式图像WEB服务器。移植PC机的JPEG程序到DSP，反复研究实验来优化代码，图像压缩速度达到分辨率为576*720 YUV2：1：1的图像每秒压缩五帧。在RCM3200平台上开发基于CGI功能的主动刷新WEB服务器。

在该图像处理系统上做实验，改变质量参数Q，获得不同压缩比的图像。从实验结果可看出：质量参数从100到30变化，图片质量没有明显下降，但质量参数降到25以后，图像开始明显失真。在实际应用中一般质量参数控制在50左右，能完全满足要求，既可以使用高压缩比又能保证图像质量。

6.学位论文 潘健 基于提升小波的图像压缩与检索技术 2008

随着经济的发展、科技的进步，图像作为一种信息传播方式越来越被人们所重视。然而由于图像存储和传输占用太多的空间和带宽，因此，寻求有效的图像压缩编码方法具有重大意义。近20多年来，多种图像压缩技术和标准被提出：如基于离散余弦变换（DCT）的JPEG标准能够在不降低图像质量的基础上成倍降低图像存储空间，后又提出的基于离散小波变换（DWT）的JPEG2000标准比JPEG效果更好。但DWT算法复杂，耗时较长，一直是个待解决的问题。提升小波算法因其计算简单，克服了传统小波卷积运算的复杂性而被提出，并已成为图像编码的重要技术之一。

本文先对基于邻域预测、基于线性预测和基于双线性预测的提升算法，进行了研究，并结合SPIHT（set partitioning in hierarchical trees，分层树中分配样本图像编码）算法，对图像进行嵌入式编码。实验结果表明，在同样的比特率和同样分解级数的情况下，对于纹理丰富的图像，基于双线性预测提升的小波图像编码算法，具有更好的图像主客观质量，最差的是基于邻域的预测方法，而线性预测方法则介于两者之间。

图像存储量降低带来了图像存储和传播的便利，网络带宽提升带来了多媒体网络的发展，但如何从浩瀚的多媒体信息中找出需要的图像信息已成为研究人员所面临的难题，为此，图像检索技术应运而生。其中基于小波的图像检索技术是目前研究的热点之一。

本文利用提升小波算法，对图像检索进行了研究，一方面我们对提升小波压缩域的图像检索进行了分析，在算法中采用了两种特征提取的方法来进行检索，一种是基于能量和、代数和的方法，一种是基于直方图的方法。另一方面，我们对提升小波变换后的小波系数进行统计分析，分别求解提升小波系数的能量、均值和方差，以此作为特征向量进行检索，并提出了一种基于小波系数二进制直方图的图像检索方法，取得了较好的检索效果。

7.期刊论文 杜伟娜.孙军 新一代静止图像编码系统-JPEG2000 -电路与系统学报2002,7(3)

JPEG2000是最新制定的静止图像编码国际标准,JPEG2000不仅提供的率失真性能和主观图像质量优于原JPEG标准,而且在支持渐进图像传输、感兴趣区图像编码和抗误码性能上也优于传统的JPEG标准和其它编码方法.本文对JPEG2000系统的结构、特性以及其编码算法进行了分析,并给出JPEG2000与原JPEG标准的性能比较.

8.学位论文 王战盟 基于ARM的嵌入式静态图像显示系统的研究与实现 2007

JPEG 标准是当前在静态图像编解码领域中应用最为广泛的一种图像编解码技术，被广泛应用于图像的存储和传输。随着嵌入式技术的发展，JPEG 图像编解码技术被用到了许多嵌入式系统中，产生了如数码相机、多媒体手机、掌上电脑等许多嵌入式产品。目前 JPEG 编解码多采用专用集成电路 ASIC来实现，这种方法集成度较高，实现起来相对容易，但成本较高，升级拓展困难。一种更加灵活的方案是基于高性能的处理器，如ARM、DSP等，通

过采用各种优化算法实现JPEG编解码功能，该方案使系统功能拓展和升级可以通过软件升级来实现，具有高度的灵活性和适应性，而且通过软件来实现系统功能的扩展和升级成本也较硬件升级低。本文的研究题目：“基于ARM的静态图像显示系统的研究与实现”，研究了在ARM硬件平台上采用软件编程的方法来实现 JPEG 图像解码和显示。

本文首先介绍了JPEG 静态图像编码标准的基本原理，然后基于SAMSUNG公司的 ARM 处理器S3C4480X设计出用于静态图像解码和显示的硬件平台，并根据该硬件平台设计了相关的引导程序、驱动程序和显示模块的显示程序，为最终实现了一个嵌入式的图像处理系统做好了软硬件基础。接着进一步分析和研究了JPEG 基本系统的解码过程，TPEG解码过程主要由头文件信息处理、熵解码、反量化、反离散余弦变换和色度空间转换五个部分组成。在对解码过程中运算量最大，耗时最多的反离散余弦变换 IDCT 部分，结合S3C44BOX的流水线操作与并行操作特征和反离散余弦变换原理，将二维8×8矩阵的反离散余弦变换运算转换成1 6次的一维8点反离散余弦变换运算，并对一维8点的反离散余弦变换采用一种快速算法，高效快速地实现了反离散余弦变换。最后实现了基于ARM平台的 JPEG 静态图像的解码显示系统，并且从S3C4480X的硬件特性和 C 程序结构方面对解码程序做了的优化。通过从解码的图像质量和速度两个方面进行测试，系统能够达到图像重构的要求和实时解压显示的目标。

9.学位论文 邵华 基于DSP的高分辨率动态图象采集与处理 2006

随着科学技术的发展和社会生活的进步，人类对信息的需求量越来越大，人们希望无论何时何地都能够方便、快捷、灵活地通过语音、数据、图像与视频等多种方式进行快速通信，其中对视觉信息的需求又占据了整个通信需求大约60％以上。因此，高分辨率实时视频的高速传输受到人们越来越广泛的关注。本系统研究的目的即希望能够研发出满足目前现实需求的高分辨率与处理系统。

本文主要研究了对视频图像进行采集、压缩、传输的软硬件具体实现方法，设计了以TI的TMS320DM642 为核心的视频图像压缩系统。该系统是一个独立的视频图像压缩和传输设备，它能直接对视频信号进行数字化和压缩编码，并通过USB2.0接口将压缩的图像数据发送到计算机中。

首先，论文详细介绍了该系统压缩算法实现过程：在充分借鉴目前主流图像压缩标准的基础上，对传统的JPEG压缩算法加以改进，将小波变换图像处理、逐步量化算法与传统的JPEG图像编码标准等结合起来处理帧内压缩，并针对摄像头特点提出图像采集关键帧、非关键帧概念及相应帧间压缩处理方式；同时，还具体论述了如何将该算法移植到DM642芯片上实现，以及通过采用乘法量化、移位量化替代除法量化，使用线性汇编语言改写关键代码等进行有针对性的算法优化。

其次，论文也着重对图像采集与处理系统的硬件电路设计进行了比较系统的研究。该图像采集与处理系统的硬件电路设计主要包括视频图像采集电路、DSP应用系统核心板电路、USB接口相关电路等，论文详细介绍了各个部分的原理图构建、印刷电路板制作、固件程序编写等的具体过程及关键部分。

最后，本系统介绍了编写USB接口的微型设备驱动程序，通过微型设备驱动程序向用户提供标准的接口，用户可以利用其进行主机应用程序开发。

10.会议论文 姜恩华.韩朝军.王利娟.王璞 小波变换在JPEG2000图像编码中的应用分析 2003

随着多媒体技术的发展,对图像压缩技术的要求也越来越高,于是,JPEG标准委员会推出了新一代静止图像压缩标准:JPEG2000,它是基于小波变换的图像压缩标准.本文首先论述了小波变换的思想和Mallat算法,然后对JPEG2000的图像压缩编码过程及其核心编码算法EBCOT作了详细的分析.