分类号_____

UDC_____ 学 号_____

重庆大学

硕士学位论文

论文题目：基于 DSC21 的数码相机研究

论文作者：付 丽

指导教师姓名、职称、工作单位：罗 钧 副教授 重庆大学

申请学位级别：硕士 专业名称：仪器科学与技术

论文提交日期：2005 年 5 月 30 日 答辩日期：2005 年 5 月 31 日

学位授予单位：重庆大学 授位日期：2005 年 5 月 31 日

答辩委员会主席：黄尚廉 教授 院士（中国工程院院士）

论文评阅人：唐晓初 副教授 吴晓波 副教授

2005 年 5 月 30 日

重庆大学硕士学位论文

基于 DSC21 的数码相机研究

硕士研究生：付丽

指导教师：罗钧　副　教　授

学科专业：仪器科学与技术

重庆大学光电工程学院

二 OO 五年五月

MS. D. Dissertation of Chongqing University

Digital Camera Study Based on DSC21

Ph.D. Candidate: Fu Li

Supervisor: Associate Prof. Luo Jun

Major: Instrument science and technology

College of optical and electrical engineering

Chongqing University

May 2005

摘　要

本文作为重庆市信息产业局科技攻关项目“数码相机整机及关键技术研究”的一部分，主要开展了基于 TMS320DSC21（简称 DSC21）的数码相机整机系统集成研究，重点进行了 CF 卡文件系统开发和 JPEG 编解码算法及其在 DSC21 平台上的实现。

基于DSC21数码相机硬件系统研究、软件程序设计是本论文的主要内容之一。该数码相机系统以DSC21为主控芯片，DSC21是双核（DSP 5409+ARM7）结构，这种结构是当今数码相机的主流图像处理引擎架构，它容易扩展外设，可以和CF卡、SM卡等存储器无缝连接。在本文研究的系统中，集成了丰富的外设：存储器（SDRAM、Flash、CF卡），音频接口、红外接口（IrDA）、USB接口等。程序在CCS环境下调试，ARM主要完成系统级的初始化、系统配置、用户接口、用户命令的执行、连接功能以及整个系统的控制。DSP主要负责图像处理，DSC21中两个协处理器iMX和VLC加速了图像处理过程。在该数码相机平台上调通了图像采集的应用程序，实现了对OV2610 CMOS传感器的编程控制。

本论文另一主要内容是 CF 卡文件系统即 FAT 文件系统的开发。其中 FAT16 是论文研究的重点。作者在基于 DSC21 数码相机平台上实现了 CF 卡文件系统的读写。文中详细阐述了 FAT 文件系统结构以及读写文件实现的方法。

在DSC21平台上实现了JPEG编解码，JPEG编码利用了DSC21中的协处理器VLC和iMX，VLC具有优化量化计算和霍夫曼（Huffman）编码功能，iMX作为专用的图像加速器，可以实现一维和二维数字滤波以及矩阵乘法等数学运算，使整个编码过程快速实现。解码过程是完全由软件编程实现的。为了验证JPEG编解码正确性，捕捉一帧图像后，在DSP中进行JPEG编码，编码后图像数据存放到SDRAM，并把图像数据写成JPEG文件，再把该文件从SDRAM转移到CF卡，并通过USB接口传送到PC机，在PC机中通过编码后的图像和原图像比较，验证了编码的正确。

另外，作者还进行了基于单片机的 USB 开发，这部分研究是为了实现通过 USB 接口，PC 机读写 CF 卡文件。USB 接口控制芯片采用 Philips 公司的 PDIUSBD12，它完全符合 USB1.1 标准。PC 机通过 USB 接口向单片机发送读写 CF 卡命令，在 USB 的中断服务程序中，单片机根据命令类型对 CF 卡进行相应的操作，然后 CF 卡上的数据经单片机与 PC 机进行通讯。这些工作积累都有助于在 DSC21 平台上开发 CF 卡文件系统。

关键词： DSC21；FAT；USB；JPEG

Abstract

This thesis is one component of the study on the digital camera integrity and key technologies which is the key project of Chongqing Information and Industry Administration. The focus is to study the digital camera integrity based on DSC21 and CF card file system development concerned with conformity, JPEG code and decode arithmetic as well as its realization on the platform of DSC21.

The study on the system of digital cameral hardware based on TMS320DSC21 and the software are main contents of this paper. The system used DCS21 as the controlling chip with two-core structure(DSP 5409+ARM7), which is the mainstream structure of image processing on the digital camera. This is easy to extend and can be connected seamlessly with CF card and SM card. In this paper, there are plenty of peripheral equipments, such as memory(SDRAM,FLASH,CF card), audio interface, infraed interface(IrDA) and USB. All the programs are debugged on the platform of CCS. And ARM is aimed at system initialization, system setup, customer interface ,the execution of users' command, connecting performance and the control of whole system. DSP is aimed at processing image, with the two assistant processors in DSC21 accelerating the processing of image processing. Application programme about image collection has been debugged on the digital camera platform, and programmer controlling OV2610 CMOS sensor had been realized.

Another content of the thesis is to develop the CF card file system, namely FAT file system, with FAT16 as the focus of this paper. The digital camera platform based on microprocessor or DSC21 can realize write and read of CF card file system. The paper gives a detail introduction in FAT file system structure and the method to realize write and read files.

The JPEG coding and decoding on the platform of DSC21 is completed. JPEG coding uses the assistant processor VLC and iMX in the DSC21. VLC can optimize and quantize computing and Huffman coding. And iMX is used as the special image accelerator, which can realize one-dimension and two-dimension digital filter and matrix product. The two assistant processors can make the coding processing easy. And the decoding is realized by software. In order to validate the correctness of JPEG coding, first catch a frame of picture , and make JPEG coding in DSP, put the image data to SDRAM, write a JPEG file about the image data, then transfer the JPEG file data to CF

card. The file transfer to PC through USB interface. Through comparing the image after being coding with the image being collected, the image being coding right is validated .

In the addition, the study developing USB on the base of microprocessor is to realize PC reads and writes CF card file through USB interface. USB interface uses PDIUSBD12 made by Philips as the controlling chip, which conforms to the USB1.1 standard. PC reads or writes CF card command from or to microcomputer through USB. In the interrupt program of USB, the microcomputer takes the corresponding operation in CF card according to command types and the data in CF card is transported to PC through the microcomputer. The study help to develop CF card file on the platform of DSC21.

Key words: DSC21, FAT, USB, JPEG

目录

1 绪论

1.1 引言

多媒体是综合性的信息资源，是文本(Text)、图形(Graphics)、声音(Sound)、动画(Animation)、视频(Video)等媒体元素的统称。当今，人们已在生活中实实在在感受到它的影响。多媒体技术的迅猛发展，已经广泛渗透到相关领域。其中数字视频技术应用广泛：可视通信、数字电视、多媒体家用电器等，数码相机就是数字视频技术的一种应用。

1.2 数码相机与传统相机比较

所谓数码相机[1]，是一种能够通过内部处理把拍摄到的景物转换成数字格式图像的特殊照相机。数码相机并不使用胶卷，而是使用固定的或者是可拆卸的半导体存储器来保存获取的图像，还可以直接将数字格式的图像输出到计算机、电视机或者打印机上。由于图像是内部处理的，所以使用者可以马上检查图像是否正确，而且可以将图像立刻打印出来或是通过电子邮件传送出去。数码相机只是将相机数字化了；就其本质而言，数码相机依然是相机，你完全可以将数码相机当作一部普通的相机来使用。数码相机与传统相机相比，主要在信号的捕捉、存储、处理与输出方面有着根本的不同[37]，具体体现在以下几个方面：

(1)数码相机与传统胶片相机捕捉信号的前端设备是相同的。数码相机也是使用镜头、光圈、快门来聚焦图像，这与传统胶片相机并无区别。但是，传统相机通过光透镜将图像聚集到感光银盐胶片上，胶片感光，将影像以光学模拟信号的形式记录下来；数码相机则将景物聚焦到 CCD 或 CMOS 图像传感器上，通过扫描电子模拟信号，然后经过 A/D 模数转换形成数字信号，最后以数字文件的形式保存信息。

(2)传统相机的信号存储媒体以胶卷(光敏卤化银胶片)为主；数码相机所存储的照片不再是实际的影像而是一个个数字文件，其信号存储体也不是底片，而是数字化存储器件。这使摄影突破了原来用胶片记录实际影像的传统观念，实现数字的直接生成，为摄影手段的丰富和发展提供了新的途径。

(3)数码相机的最大优势在于其信息的数字化，数字信息可直接倒入计算机，甚至借助遍及全球的因特网及时传送。这比对传统照片进行数字化处理减少了扫描的环节；数字化信息的另一优势为易处理性。传统影像的暗房工艺无法与强大高效的数码按时技术相比，数码相机的图像可在计算机上任意加工。高性能的微处理器和功能超强的图像处理软件更使得数据量庞大的高质图像处理变得方便。

有人将数码相机形象地比喻为三维物体的立体扫描仪。

(4)数码相机的成像质量很大程度上取决于相机中的图像传感器。由于图像传感器价格较高，因此数码相机的价格比同档次传统相机的要高。

与传统相机比较，数码相机的主要优点之一就是直接生成数码照片，而数码照片与传统照片的区别在于可以方便地对数码照片进行编辑和加工，其中有许多特殊的加工技巧是传统照片所难以实现的。因此，使用数码相机进行拍摄，可以使用更少的耗材，数码照片也可以保存得更长久。

1.3 数码相机工作原理

数码相机是光、机、电一体化产品[2]，通常数码相机的主要组成部分（如图1.1）及其功能如下：

. 镜头：将光线汇聚到感光元件CCD或CMOS上；

. CCD（或CMOS）：将光信号转变为电信号的感光元件；

. ADC（模数转换器）：将连续的模拟电信号转换为离散的数字信号；

. DSP（数字信号处理器）：经过高速运算处理，把数字信号转化为图像；

. 图像存储器：用于保存图像，固定式的内置存储器或是活动式的外置存储卡；

. MCU（主控程序芯片）：指挥数码相机各部分协同工作；

. LCD（液晶显示器）：通过它来取景或是查看拍摄到的图像；

. 输出接口：把拍摄好的图像输出给计算机、电视机、打印机或其它设备；

. 电源：为数码相机提供电能的电池或稳压电源；

. 闪光灯：与传统相机的功能完全一样，用来加强曝光量。

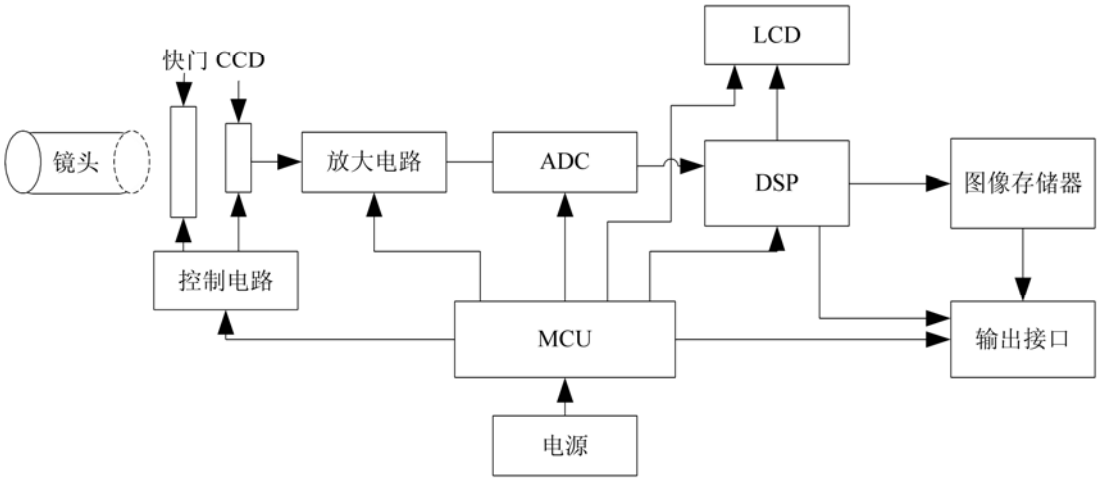

图1.1 数码相机的组成框图

Fig1.1 Digital Camera structure frame

数码相机用光学凸透镜作为镜头，将所要拍摄的主体的反射光聚焦到CCD（或CMOS）上，按下快门，感光成像器件CCD感受不同的光的强度，耦合出不同的电流，从而记录被摄景物上反射的光，并按红、蓝、绿三原色通过译码电路来形成不同的电流。电流经过放大电路放大后，通过A/D转换元件转换成数字信号。然后这些数字信号传送到DSP，把数字信号转化为图像文件的格式，这样就生成了数字格式的图像。图像被保存到数码相机的图存储器中，这时也可以使用LCD液晶显示器来观看照片的效果。最后可以把存储器中的图像通过电缆下载到电脑中。

1.4 图像传感器

图像传感器是数码相机中的核心部件，目前数码相机的图像传感器分为两类：CCD（Charge Coupled Device）和CMOS（Complementary Metal-Oxide Semiconductor），CCD是电荷耦合器，CMOS是互补型金属氧化物半导体。CCD或CMOS作为光电转换器件，将被摄景物以数字信号方式记录在存储介质(存储器、存储卡或软盘)中。

CCD是在大规模集成电路技术发展基础上产生的一种新型器件。CCD能对光照作出反应并把反应的强度转换成相应的数值。当光从红、蓝、绿滤镜中穿过时，就可以得到每种色光的反应值。CCD的突出特点是以电荷为信号，而其他的大多数器件是以电流或电压作为信号，CCD的基本功能是电荷存储和电荷转移。CCD的工作过程主要包括信号电荷的产生、存储、传输和检测。

CMOS技术自从推出以来，已表现出势头极强的发展趋势。CCD与CMOS生产工艺和器件结构不同，使它们在能力和性能上迥然不同。CMOS图像传感器也是把光信号转换成电信号的装置，即把入射到传感器光敏面上按空间分布的光强信息，转换为按时序串行输出的电信号—视频信号，该视频信号能再现入射的光学图像。

CCD 与 CMOS 区别主要在于光电转换后信息传送的方式不同。CMOS 具有信息读取方式简单、输出信息速率快、耗电少、体积小、重量轻、集成度高、价格低等特点，是数码相机理想的成像芯片，但目前 CMOS 的成像质量仍达不到 CCD。CMOS 与 CCD 相比，很容易与 A/D 电路、数字信号电路等集成在一起，CMOS 芯片生产成本低，成品率高，并且明显降低了功耗。此外，CCD 只能单一地锁存落到成千上万的采样点上的光线的状态，而 CMOS 则可以完成其他许多功能，如模/数转换、负载信号处理、白平衡处理及相机控制等，还有可能在不会大幅度提高成本的前提下增加 CMOS 的密度和位深度。随着 CCD 与 CMOS 传感器技术的进步，两者的差异有逐渐缩小的态势，例如，CCD 传感器一直在功耗上作改进，以应用于移动通信市场(这方面的代表业者为 Sanyo)；CMOS 传感器则在改善分辨率与灵敏度方面的不足，以应用于更高端的图像产品，

1.5 数码相机的发展现状

数码影像技术，最初由美国柯达公司、日本富士公司、尼康公司、佳能公司、索尼公司和荷兰的飞利浦公司六家在感光材料领域、照相机领域和家用领域卓有成效的世界级大公司共同发展起来的，并在光学、电子、和计算机领域进行了联合研制开发。

当今是高度数字化的时代，生活在现代社会的我们已经清晰地感受到了数字化所带来的快捷和方便。越来越多的数字化设备取代了传统设备，数码电视，数码摄像机，手机，数码相机等等。数码摄影技术之所以越来越受大众的喜爱，也是得益于数字化生存所带来的优点、从个人计算机的普及，到国际互联网的风靡，再到超大规模集成电路技术的高速发展，都无一例外地为数码摄影的普及应用开拓了广阔的空间。数码相机在多媒体演示、视频取证、情报及新闻采集，图像处理、资料管理、网络通信、印刷包装业、广告设计业、新闻出版业、医疗卫生业、家庭摄影、企业管理、现代教育技术，邮政影像快递、婚纱摄影及日常生活均占有一席之地，它还可以在一些传统相机无法拍照的地方作业，这一点已被广泛地运用于科技，医疗，探险等许多方面。因此，数码相机技术、应用和产业的发展对于推动我国信息化革命将起到重要的作用。

前很多市场分析家都认为，未来几年数码相机市场将呈现爆炸性增长态势。他们预测，2007 年数码相机市场需求量将超过 7 千万台，与此同时，传统胶片式相机的销售将不可避免地呈现下滑态势。据专家分析，未来的数码相机发展趋势是高像素化、小型化、多功能化、时尚化。

目前数码相机图像处理引擎主要采用 DSP+MCU 双内核架构，MCU 用于完成整个数码相机的控制功能，而 DSP 则用完成图像信号处理及一些附加功能的实现。图像处理引擎对最终数码相机输出的图片质量及其它各方面的性能表现起着至关重要的作用。由于未来数码相机系统所支持的像素数越来越高，以及类似 MPEG-4 视频编/解码、MP3 播放、录音等多种功能将逐步融入数码相机，这必然给数码相机系统设计带来巨大的挑战。而且，数码相机达到一定级别的像素之后，就会更注重相片本身的图像品质，例如对图像的防噪处理、色彩的还原性、亮度，数码相机的曝光和对焦，这些都要求采用处理能力更强大的专用 DSP(如 TI 的 TMS320DSC/TMS320DM)，并需要采用更好的图像处理算法。

1.6 嵌入式系统的介绍

嵌入式系统(Embedded System)，根据英国电机工程师协会的定义所作的翻译：“嵌入式系统为控制、监视或辅助设备、机器或工厂操作的设备”。它是把计算机

直接嵌入到应用系统之中，融合了计算机软/硬件技术、通信技术和半导体微电子技术，是信息技术的最终产品，它具备下列四项的特性：

①通常执行特定功能；

②以微电脑与外围构成核心；

③严格的时序与稳定性要求；

④全自动操作循环；

嵌入式系统是电脑软件与硬件的综合体，亦可涵盖机械或其他的附属装置。整个综合体设计的目的在于满足某种特殊功能。嵌入式系统的架构可分为五个部分：处理器、内存、输入与输出、操作系统与应用软件。

数码相机系统也属于典型的嵌入式系统，本论文研究的数码相机就是基于OMAP平台的嵌入式系统。

1.7 OMAP 平台

OMAP（Open Multimedia Applications Platform）平台是由 TI 公司 1998 年推出的可扩展的开放式平台，堪称是无线世界发展的里程碑。它是指开放式多媒体平台，采用一种独特的双核结构，把控制性能较强的 ARM 处理器与高性能低功耗的 DSP 核结合，是一种开放式的、可编程的基于 DSP 的体系结构，OMAP 系列处理器一般拥有双核（DSP 和 ARM）结构，具有很强的运算能力、极低的功耗和丰富的外围接口。广泛应用于数码相机、手机、PDA、Web 记事本、远程通信。本论文数码相机系统采用的核心芯片 TMS320DSC21 正是这种 OMAP 平台。

1.8 本论文的工作

本论文的主要工作，是分析基于TI公司的TMS320DSC21为核心的数码相机的研究，重点进行数码相机相关的技术研究：USB开发、CF卡FAT文件系统的开发以及JPEG编解码在DSC21平台上的实现。

2 基于 DSC21 的数码相机系统硬件设计与实现

系统硬件采用了TI公司的210万数码相机EVM板，在该系统中作者还扩展了液晶接口电路。

2.1 系统硬件结构

本系统采用TI推出的数字相机专用芯片TMS320DSC21（以下简称DSC21）作为数码相机的核心部分，它负责系统控制，图像处理。同时该系统还包括以下功能模块，图2.1给出了系统框图，各模块的功能如下所述：

. CCD模块：CCD时序控制，自动增益控制，AD转换；

. LCD模块：根据DSC21输出的信号驱动LCD；

. 电源模块：提供系统所需要的电源；

. Flash/SRAM模块：DSC21程序存储；

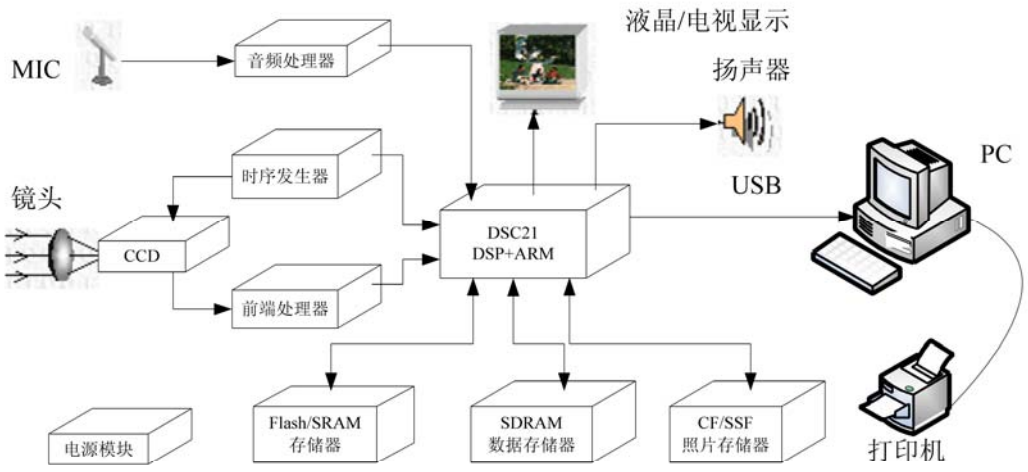

图2.1　基于DSC21数码相机系统框图

Fig2.1　Digital Camera frame diagram based on DSC21

2.2 DSC21 功能模块

DSC21作为系统的核心，内含两个处理器内核：TMS320C5409和ARM7 TDMI RISC MCU。ARM7用作整个系统的主控制器，可编程的5409 DSP 核心处理图像的编码与解码。

按照DSC21的功能可把DSC21分为以下九个模块：

. ARM子系统
. DSP子系统
. ARM与DSP通讯
. SDRAM控制器
. 预览引擎
. 突发模式压缩（解压缩）模块
. OSD屏幕显示模块与图像加速器
. JTAG接口
. I/O模块

DSC21引擎系统结构如图2.2所示[6]。

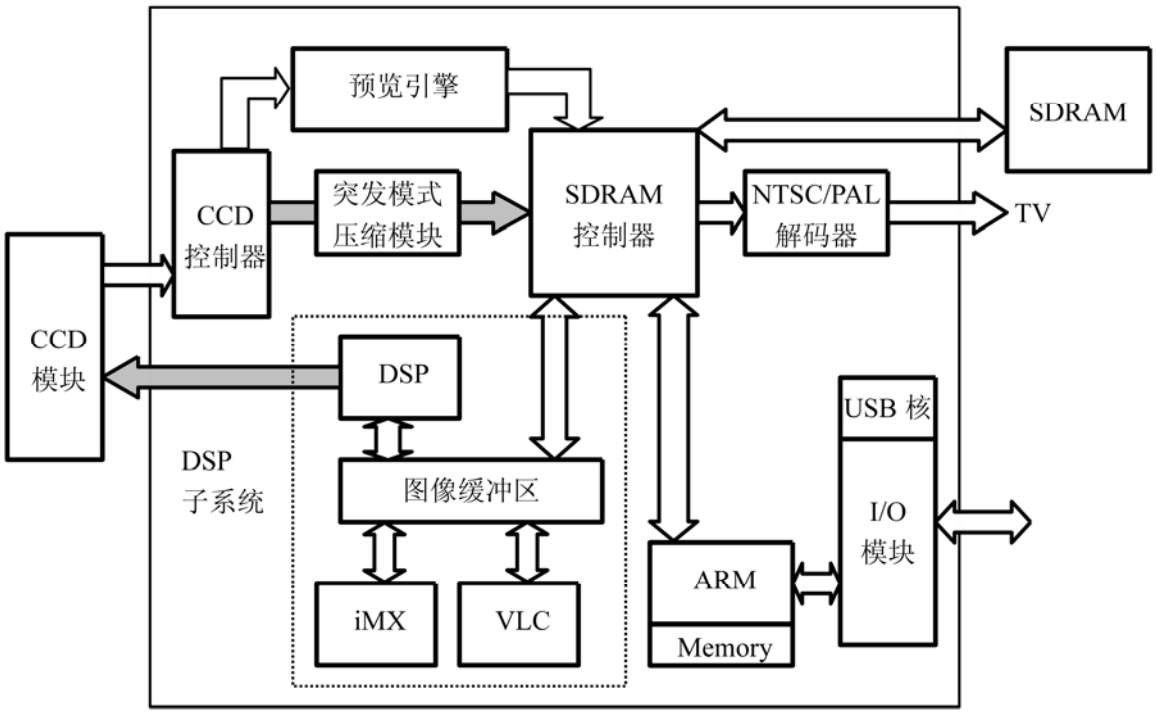

图2.2 DSC21引擎系统结构

Fig 2.2 The DSC engine system architecture

2.2.1 ARM 子系统[7]

ARM是32位嵌入式微处理器，最高工作频率可达47.5MHZ，主要完成系统级的初始化、系统配置、用户接口、用户命令的执行、连接功能以及整个系统的控制。ARM有较大的存储器空间，内部有32K静态RAM，有较好的接口转换能力，因此适合复杂、多任务和一般的处理操作。

ARM连接了所有的DSC21的外设，如图2.3所示，即：CCD控制器、TV解码器、预览引擎、红外接口（IrDA）、USB、 CF卡或SM（Smart media）卡、UART等，由ARM处理器来对这些外设进行控制。ARM处理器还负责CCD原始数据的管理以及CCD数据到SDRAM和LCD的控制。数码相机四种操作模式：预览模式、捕捉模式、回放模式、突发模式，都是由ARM产生请求，请求完成后，在有些情况下，ARM将管理完成这些操作后的数据。

系统复位后，在相机进行任何操作之前，ARM必须完成以下几个任务：初始化，即BOOT操作，不仅要要初始化IO设备和外设到已知状态，而且要准备装载DSP程序和开始DSP。ARM从Flash读DSP的boot代码，装载DSP代码到存储器，并且从DSP的HOLD状态释放DSP。

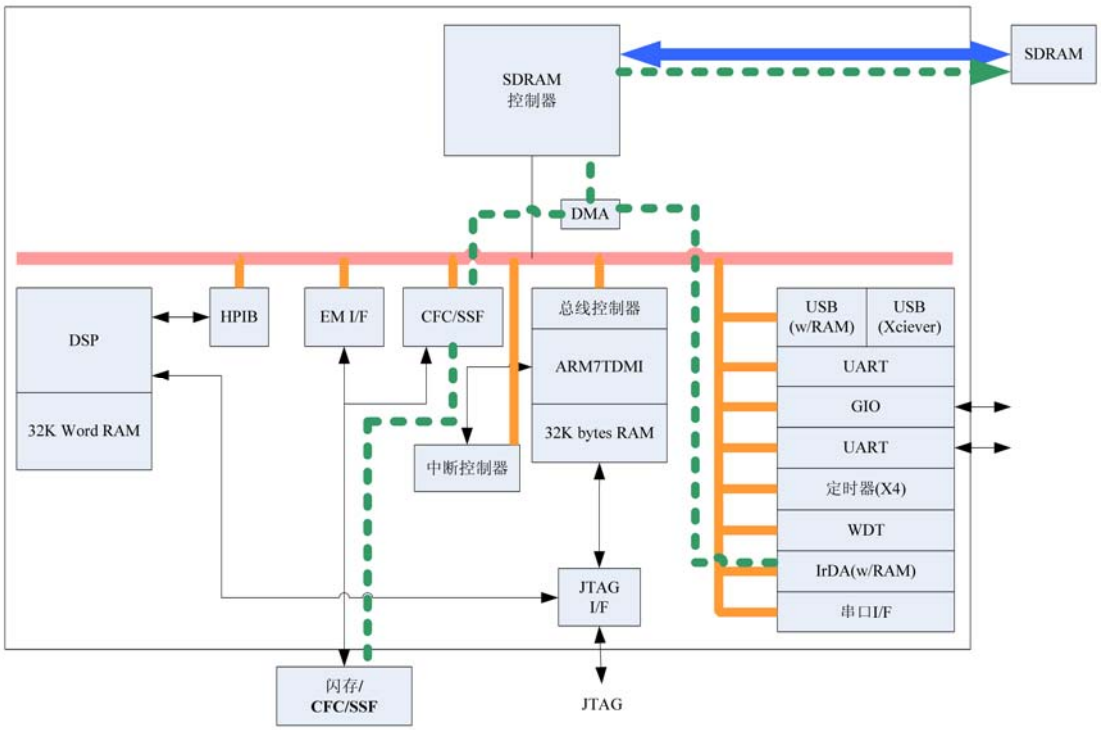

图2.3 ARM子系统与DSP/SDRAM接口

Fig2.3 ARM subsystem and DSP/SDRAM interfaces

①ARM与SDRAM接口

ARM通过SDRAM控制器与SDRAM接口，ARM有两种方式访问SDRAM：一

是通过SDRAM缓冲区突发方式读写SDRAM；二是直接读写SDRAM。

②ARM与外部存储器接口

ARM通过外部存储器接口模块与外部存储器连接，如：CF卡、SM卡。同时ARM与外部Flash（或SRAM）的连接也是通过这个接口。ARM与这些存储器的数据转移可以通过连接它们之间的DMA块进行。

2.2.2 DSP 子系统

DSP子系统由C5409 DSP、iMX（Imaging Extension）、VLC（Variable Length Coding）协处理器、图像缓冲区（image buffer）组成，C5409DSP是定点16位DSP，高性能、低功耗[3]。先进的修正哈佛结构，有一条程序总线，三条数据总线，程序空间和数据空间分开，可同时进行数据访问和指令读写，提高了数据吞吐率和程序运算速度，仅一个机器周期可完成两次读操作和一次写操作。片上具有一个可编程的锁相环。但集成在DSC21中的5409 DSP与TMS320VC5409相比，有些功能不再使用。以下表2.1中作了详细的描述。

表2.1 TMS320VC5409与TMS320DSC21中5409 DSP区别

Table 2.1 TMS320VC5409 and DSP 5409 in DSC21

功能区别	**TMS320VC5409**	**TMS320DSC21**	说 明
串 口 McBSP 数	3	1（仅串口 0）	限制 DSC21 引脚数
扩展地址	完全支持	仅能使用内部 32K 地址	因为地址总线不用作扩展地址
MP 模式	可以使用	不能使用	因为 DSP 外部 IO 总线（XIO）不在内部引脚中
内部 ROM	用户可以编程	BOOT ROM 固定，用户不能编程	由 ARM 实现对 5409 DSP 地标注控制
重叠模式	重叠、非重叠模式都是可能的	仅重叠模式可能	写数据和程序到内部 DARAM
DROM 位	当 DROM＝1 时，内部 ROM 可映射到数据空间	DROM 只能为 1	内部 ROM 对用户未公开
DMA	完全支持	仅 McBSP 自动缓冲功能可用	因为 DSP 外部 IO 总线（XIO）不在内部引脚中
内部 SWWSR 可以编程 PLL	完全支持	仅旁路模式可用	在 DSC21 中 DSP 有独立的 DPLL
GPIO	可用	不能使用	HPI8 的 HD[7:0]被使用
EMU0，EMU1 功能	完全支持	不能使用	DSC21 中没有这两个引脚

DSP执行自动曝光(AE)、自动对焦（AF）、自动白平衡（AWB）以及图像处理的部分任务。同时可进行SDRAM数据转移和驱动加速器实现图像处理任务和图像压缩。

DSP子系统的两个协处理器用来加速DSP的图像处理，其中iMX，是一个并行MAC引擎，具有4个MAC（乘加器）单元，具有灵活的控制和存储器接口，它与 DSP通讯是通过它们共享的存储器空间和两个存储器映射寄存器（命令开始寄存器，完成状态寄存器），iMX用来加速以下图像处理：

. CFA插补；
. 颜色空间转化；
. 色度下采样；
. 边缘增强；
. 色彩抑制；
. 离散余弦变换和反离散余弦变换；
. 颜色查找表。

VLC加速器是一个对JPEG图像压缩和MPEG压缩中的量化和Huffman解码进行优化的协处理器。

图像缓冲区介绍详见第三章。

2.2.3 ARM 与 DSP 接口[6][7]

HPI口是DSP与MCU常用的接口，DSC21中DSP与ARM连接也是使用这个接口（如图2.4）。ARM与DSP通讯通过HPIB（Host Port Interface Bridge），ARM访问DSP存储器通过编程HPIB来实现，HPIB物理上连接DSP 5409端口和ARM的总线控制器（BUSC）。HPIB包括五个子模块：接口、时间发生器、DSP控制寄存器、中断保持部分。接口部分接收和存储BUSC的数据并且转移DSP内部数据。接口可以是8位或16位；时间发生器负责HBIL，HDS以及检测HRDY信号。HBIL是HPI口DSP的字节定义信号，在HPI8模式下，HBIL为“0”时，表明第一次访问，为“1”时，表明第二次访问，此信号对HPI16模式无意义。HDS是数据选通信号。HRDY是DSP准备好信号；中断保持模块用来检测HPI口中断即检测HINT信号电平，并给INTC（中断控制器）一个脉冲。

除了5409DSP存储器映射寄存器0h~80h外，ARM均可直接访问DSP的内部DARAM区域数据。DSP也可访问这部分区域，当两者同时访问时，ARM优先访问，DSP中插入等待。

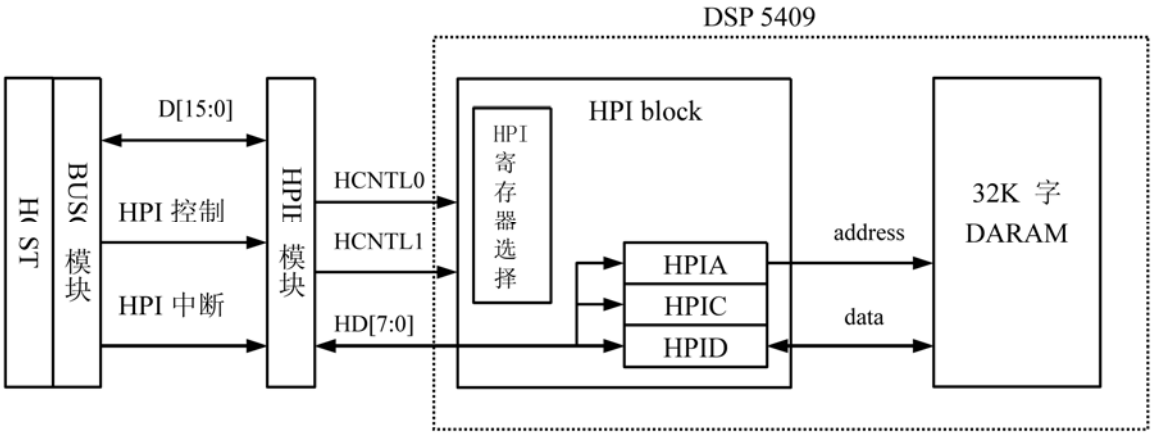

图 2.4 DSP 与 ARM 通讯接口

Fig 2.4 Communication interface between DSP and ARM

2.2.4 SDRAM 控制器

SDRAM控制器是SDRAM与所有功能模块的主要接口，这些功能模块是：ARM，DSP，CCD控制器，TV解码器，预览引擎等。支持最高80MHZ，32bit SDRAM，可以对SDRAM的访问单元区分优先级，支持CCD实时数据流和TV输出显示。管理SDRAM与处理模块或外设模块之间的数据流（如图2.5所示）：

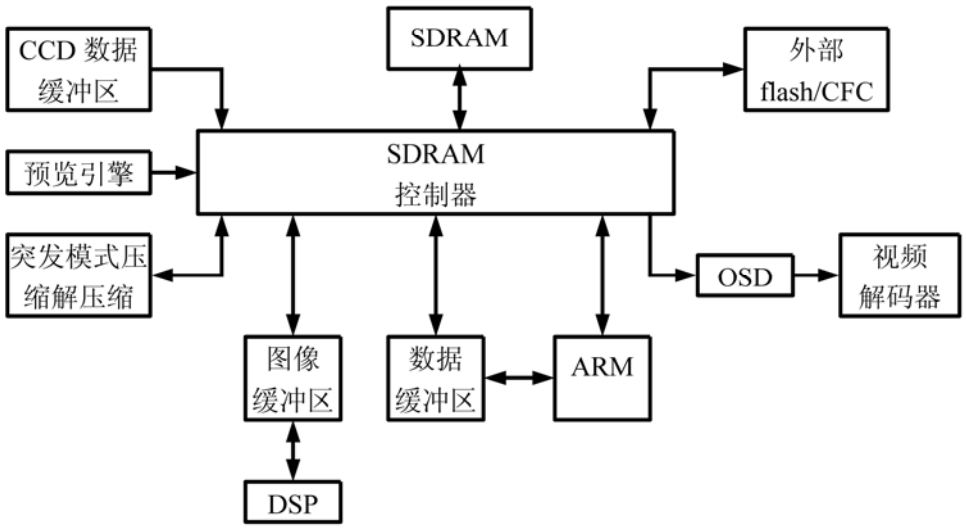

图2.5 SDRAM控制器数据流程

Fig 2.5 SDRAM controller data flow

. CCD数据缓冲区到SDRAM；

. 预览引擎到SDRAM；

. 突发压缩模块与SDRAM之间的数据；

．通过OSD(On Screen Display) 屏幕显示模块到视频解码器的从SDRAM传送的OSD数据和图像数据；

．DSP图像数据缓冲区与SDRAM之间的数据流；

．外部Flash或CF卡与SDRAM之间的数据流；

2.2.5 预览引擎

当需要TV显示CCD/CMOS采集实时图像时，必须使能预览引擎模块，ARM可以编程该引擎实时硬件处理：白平衡；CFA插补；颜色转换（RGB到YUV）；等。该引擎的原理框图如图2.6所示。

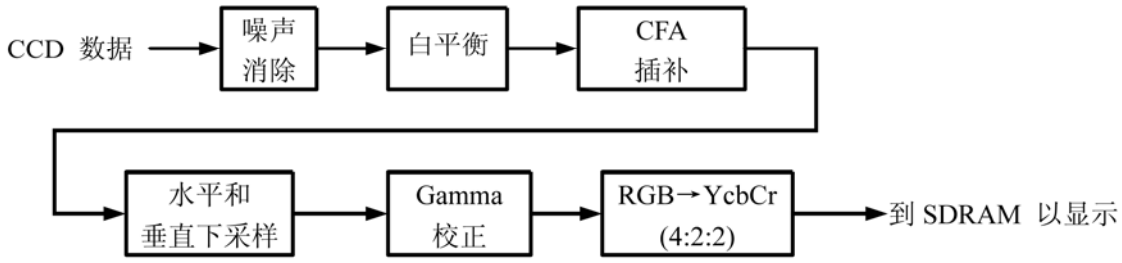

图2. 6 预览引擎框图

Fig 2.6 Preview Engine block diagram

① 微型滤色片阵列（CFA）

为了实现彩色摄影，在数码相机系统中通常采用给图像传感器器件表面加以CFA（Color Filter Array，彩色滤镜阵列）或使用分光系统将光线分为红、绿、蓝三色，用三片图像传感器接收的方法。

CFA（color filter arrange）排列如图2.7：

R	G	R	G
G	B	G	B
R	G	R	G
G	B	G	B

(a) RGB bayer

Ye	Cy	Ye	Cy
G	Mg	G	Mg
Ye	Cy	Ye	Cy
G	Mg	G	Mg

(b) Complementary

图 2.7 预览引擎支持的 CFA 排列图

Fig 2.7 Arrangements supported by the Preview Engine

②白平衡：每个颜色分量的增益可由ARM来编程，其精度是8位，如图2.8所示，

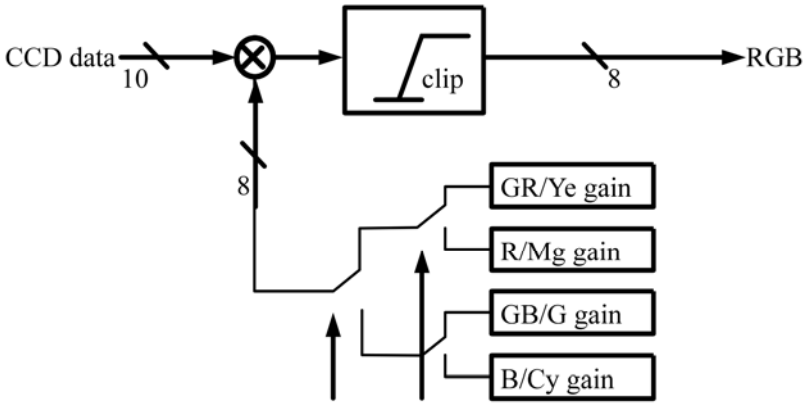

图 2.8 预览引擎中的白平衡

Fig 2.8 White Balancing in the Preview Engine

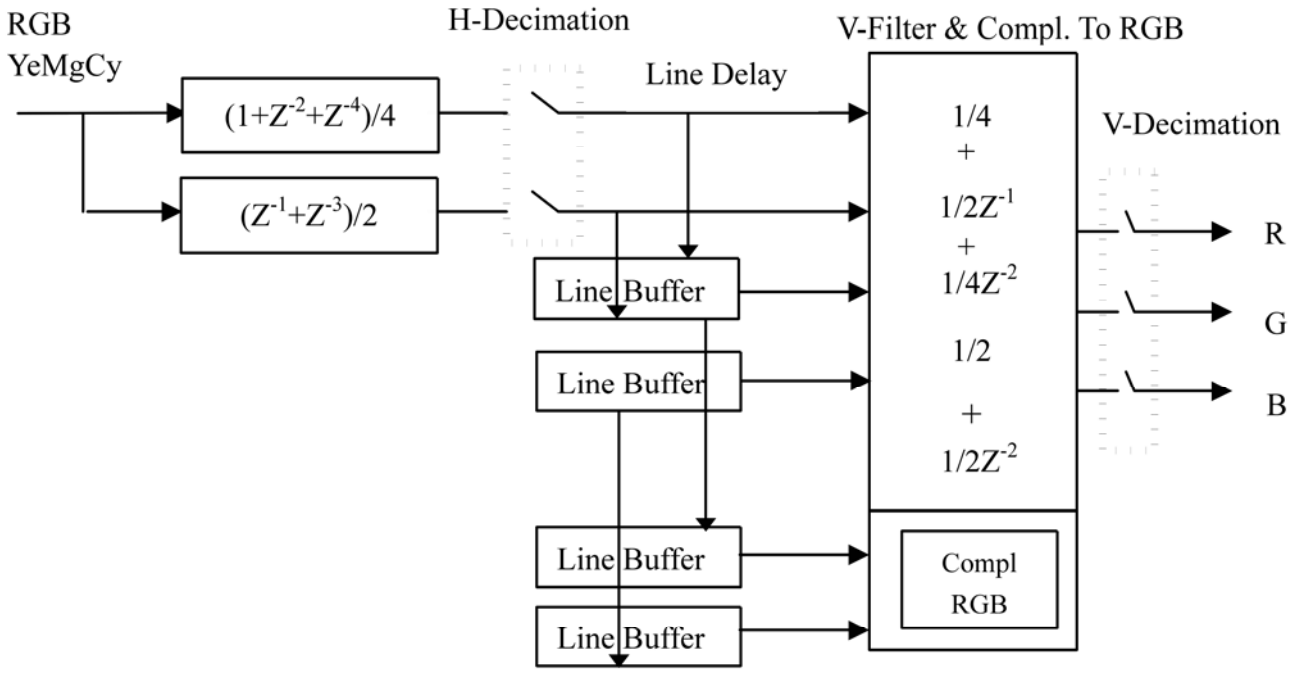

图 2.9 预览引擎中的 CFA 插补

Fig 2.9 CFA interpolation in the Preview Engine

③CFA插补与下采样逻辑图如图2.9所示。线性缓冲区(Line Buffer)为720×8bit大小，H/V下采样比例是N/M（N=1到64，N=1到M）

④RGB到YUV的转化：转化表达式如下：

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} A_0 & A_1 & A_2 \\ A_3 & A_4 & A_5 \\ A_6 & A_7 & A_8 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.1}$$

转化系数 A_0 ~ A_8 都可由ARM编程，系数的精度为8位。

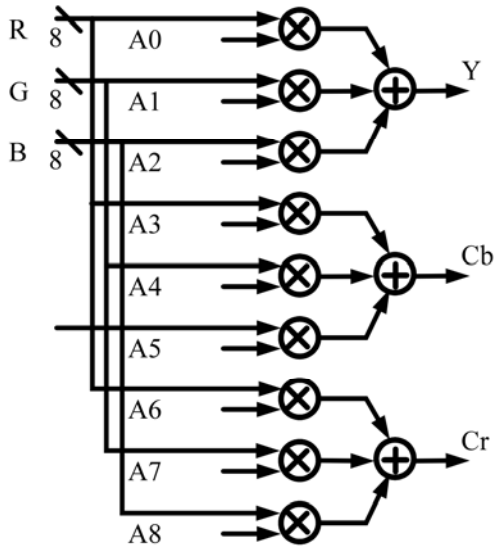

图 2.10 预览引擎 RGB 到 YUV 转化
Fig2.10 RGB to YUV Conversion in the Preview Engine

⑤伽玛校正

由物理设备产生的亮度信号与要显示的信号通常不是线性关系的。一个传统的CRT显示器，在显示图像时产生的亮度信号大致相当于要显示信号电压值的2.5次幂的大小。这种幂次关系中的指数2.5就是我们通常说的伽马值了。这种非线性性必须通过幕次关系来进行线性化，以便得到正确的亮度值。

视频系统中，红、绿、蓝线性性光线的亮度将会被伽马校正模块转换成非线性性的信号，在数码相机系统中也一样。首先，我们得知道人对亮度的感知反映L(linghtness),这反映了人类视觉感知对亮度的非线性性，其大小可以用下式来量度：

$$L = 116(\frac{Y}{Y_n})^{\frac{1}{3}} - 16 \qquad 0.008856 < \frac{Y}{Y_n} \tag{2.2}$$

其中，Yn是白点参考的亮度值，Y是目标点的亮度值。显示设备的显示信号与L的关系在某些区域可能是线性的，在另一些区域也可能是非线性的。例如HDTV的视频标准中显示信号电压与L的关系就是分段而异的。

预览引擎中Gamma纠正如图2.11，采用的伽玛曲线如下图(图2.12)，增益和区域都可由ARM编程。

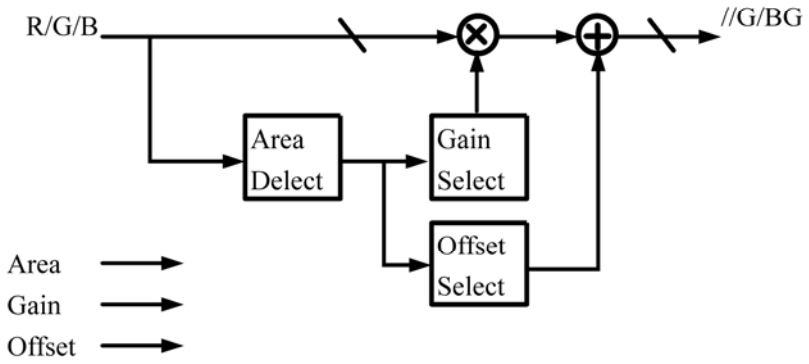

图2. 11 预览引擎中Gamma纠正

Fig2.11 Gamma Correction in the Preview Engine

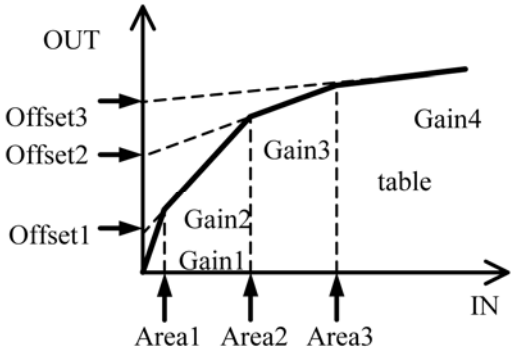

图 2.12 预览引擎中可编程 Gamma 曲线

Fig2.12 Programmable Gamma Curve in the Preview Engine

2.2.6 突发模式压缩（解压缩）模块

DSC引擎含有一个改进的突发捕捉功能，该模块具有硬件压缩和解压缩功能。CCD原始数据首先通过该模块存入SDRAM中，然后，可从SDRAM收回数据该模块进行处理，处理后的数据以JPEG文件形式存到SDRAM中。在动态的回放模式下可以显示这些JPEG文件。

2.2.7 OSD 模块和图像加速器

OSD是On Screen Display的简称。在视频窗口顶部，它可以使能窗口重叠显示，该窗口称OSD窗口。它支持三个窗口显示，窗口0是YUV格式，窗口1，2是位图格式。OSD模块负责管理OSD窗口的数据，从SDRAM中读OSD的数据，并把数据输出到DSC21中的PAL/NTSC解码器。由ARM配置好OSD并且使能OSD后，OSD模块读OSD数据，ARM CPU负责打开和关闭OSD的操作。OSD数据大小是可变的，在OSD的位图窗口中，每个像素可以是1，2，4或8位宽度。在YcrCb4:2:2窗口中，每个成分是8位，颜色分量排列顺序是CbYcrY…格式。如果RGB格式的图像数据需要OSD显示，则首先需要把RGB转换成YcrCb格式。

2.2.8 JTAG 接口

DSC21集成了 ARM 7TDMI和TMS320C54X两个处理器，因此需要多处理器调试和开发支持，它们是通过一个JTAG连接来实现对目标板的仿真和程序下载，JTAG接口采用IEEE标准1149. 1检测逻辑电路，该逻辑为界面器件提供边界检查。同时，它也可以象执行器件内的操作一样用于检测管脚的连续性。IEEE标准1149. 1检测逻辑区别于访问所有片内资源的内部逻辑检测电路，通过JTAG接口提供与目标板的连接。

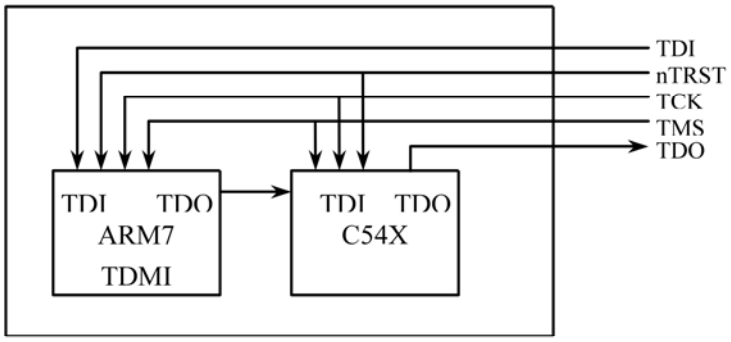

图 2.13　JTAG 连接图

Fig 2.13　Interface JTAG and PC

2.2.9 I/O 模块

DSC21的IO模块提供了对不同的DSC外设的接口。

① TV解码器

TV解码器把SDRAM中的CCD原始数据解码成NTSC/PAL制式的RGB数据，以供TV或LCD显示。

② CCD/CMOS控制器

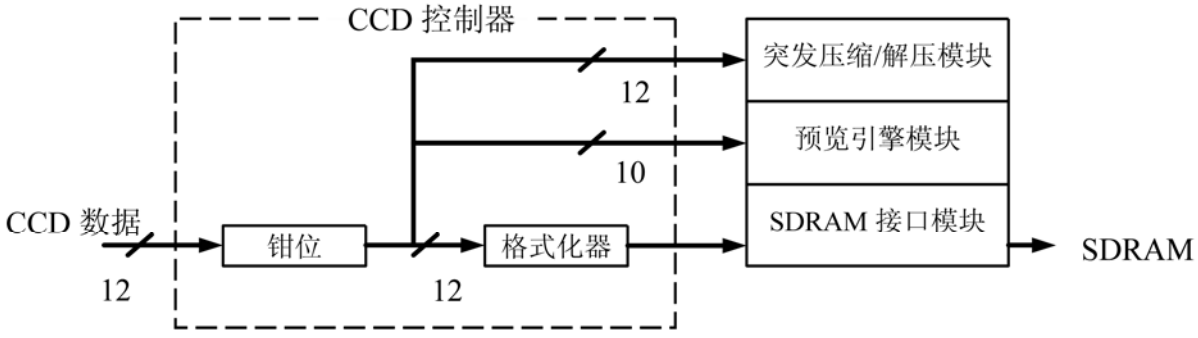

图 2.14 CCD 控制器数据流程

Fig 2.14 CCD Controller Data Flow

CCD或CMOS控制器作为与CCD/CMOS的接口，把CCD原始数据转移到

SDRAM，具有数字钳位电路、格式化器（Formatter）。它支持隔行或非隔行CCD。

CCD/CMOS图像数据通过钳位处理后，格式器格式化数据，然后转移到SDRAM，钳位处理后的12位数据进入突发压缩/解压缩模块，高10位图像数据进入预览引擎模块，如图2.14所示。

1）CCD接口[7]

CCD与DSC21连接信号如下，对每一个信号都有相应的寄存器：

CCD_DATA0- CCD_DATA11： CCD/CMOS图像数据位,最多12位,本论文采用高8位与CCD/CMOS连接，极性可由寄存器设置.

VD：垂直同步信号，极性和方向可由CCD控制器寄存器设置；

HD：水平同步信号, 极性和方向可由CCD控制器寄存器设置；

Field：场识别信号, 极性和方向可由CCD控制器寄存器设置；

WEN：写使能信号，是否使用可由CCD控制器寄存器设置；

Clock：时钟信号；

CCD type：CCD类型，隔行或非隔行。

2）钳位电路

CCD数据流入CCD控制器后，可以利用钳位电路进行钳位处理，这时CCD控制器寄存器CLEN位应使能。

3）格式化器

格式化器由三阶低通滤波电路、A律表数据压缩、数据选择三个模块构成。是否由它们进行处理都可由CCD控制器寄存器进行设置。

③ USB

USB模块有三个主要部分构成：FIFO控制器、UDC(USB Device Controller)控制器、UDC核。有6个FIFO，每个FIFO除了方向和缓冲区大小有区别外，结构都相同，都和每个端点（Endpoint）相联系。ARM可以访问FIFO读写端口寄存器以实现读写FIFO。

④ UART

DSC21支持两个串行UART接口，发送和接收端都有32字节的FIFO， 这减少了ARM在软件方面的负载。ARM控制UART模块编程，可以访问以下7个16位的寄存器：

. 数据发送/接收寄存器（FIFO）

. 波特率寄存器

. 模式寄存器

. 接收数据的FIFO控制寄存器

. 发送数据的FIFO控制寄存器

. Line控制寄存器

. 状态寄存器

⑤ CF卡/SM卡接口

DSC21有CF卡/SM卡控制器，两个卡的引脚是重叠的，CF卡/SM卡用来存储图像数据，以下着重介绍CF卡。

CF卡所有寄存器都映射到ARM存储器空间，CF卡控制器负责管理与CF卡接口的相关控制信号。CF卡接口支持两种操作模式：存储器映射模式和IO模式，不支持IDE模式。

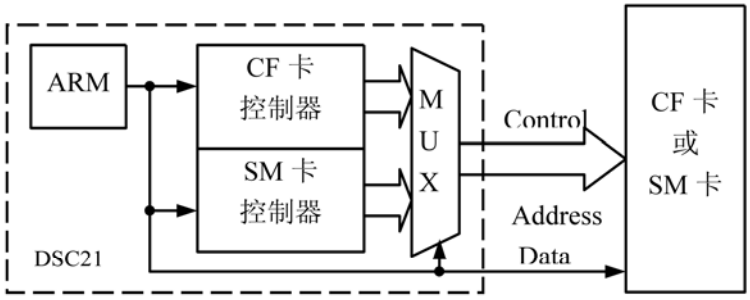

图 2.15 CF 卡与 SM 卡接口

Fig 2.15 Interface Smart media and Compact Flash card

⑥ 音频输入/输出接口

经过DSP的缓冲串行口McBsp可输入输出音频数据。

⑦ IrDA接口

DSC21集成了红外接口的Fast FIR（Fast InfraRed）核。

⑧ GIO

GIO是通用IO口，可以支持不同的客户的外设控制。可进行以下控制：

. CCD/CMOS快门控制和AGC增益控制；

. RTC控制

. LED控制

. 电源管理控制

. CCD/CMOS对焦的镜头电机控制

系统中，LED与部分GIO口连接， GIO状态可由这些LED来指示，该功能可用于测试GIO。

2.3 CCD/CMOS 模块

该模块具有 AD 转换电路，转换 CCD/CMOS 图像采集的模拟信号，并且控制

CCD/CMOD 的时序。论文选用了两个 CMOS 模块进行开发，它们分别是 OV7620 CMOS 模块与 OV2610 CMOS 模块。

2.4 电源模块

DSC21 的内核电压为 1.8V，其余芯片电源为 3.3V 或 5V，电压电源芯片采用了具有双电压输出的 TPS73HD318，输出电压为 1.8V 和 3.3V。

2.5 Flash/SRAM 模块

同步 Flash 可以与 DSC21 直接接口，Flash 可以是 16 位宽度或 8 位宽度，若 FLSH_SIZE 为“1”时，16 位宽度的 Flash 被选择；FLSH_SIZE 为“0”时，8 位宽度的 Flash 被选择。SRAM 与 Flash 有相同的接口。但当用 16 位宽度的 SRAM 时，不能以 8 位方式写 SRAM。它们都映射到 ARM 存储器空间 0x2000000~0x27FFFFF 可作为 ARM 的程序存储空间。

系统板上 Flash 与 SRAM 的片选通过跳线来选择。它们的地址与数据线分别重合的。

2.6 CF 卡模块

CF 卡可以直接和 DSC21 接口，用来存储 JPEG 图像压缩文件，系统板支持 II 型 CF 卡。

2.7 SDRAM

系统采用了两片 ISSI 公司的 IS42S16400 芯片。IS42S16400，高速、同步、动态存储器，LVTTL 接口，可直接连接到 DSC21 中的 SDRAM 控制器的数据、地址以及控制线。时钟频率支持：166M，133M，100MHZ，在数码相机系统中，SDRAM 都映射到 ARM 的存储器空间 0x8000000 到 0x9FFFFFFF 共 128M。

2.8 Audio 模块

DSC21 中 DSP 串口与音频编解码芯片 TMS320AIC23 连接，支持音频输入、输出，Microphone 信号输入。AIC23 是 TI 推出的一款高性能的立体声音频 Codec 芯片，内置耳机输出放大器，支持 MIC 和 LINE IN 两种输入方式（二选一），且对输入和输出都具有可编程增益调节。AIC23 的模数转换（ADCs）和数模转换（DACs）部件高度集成在芯片内部，采用了先进的 Sigma-delta 过采样技术，可以在 8K 到 96K 的频率范围内提供 16bit、20bit、24bit 和 32bit 的采样，ADC 和 DAC 的输出信噪比分别可以达到 90dB 和 100dB[34]。

2.9 RS232 与 USB 模块

数码相机照出的照片最终还是要传输到 PC 中才可以得到更好的欣赏，所以和 PC 之间就必须有个数据传输的接口。在系统板中，RS232 与 USB 可以和 PC 机之间进行通讯，DSC21 内含 USB 控制器，支持 USB1.1 全速方式。

2.10 DSC21 系统板与外部液晶接口

为了使CMOS采集的数字图像能实时被观察到，论文在DSC21系统板上扩展了液晶接口电路。SDRAM是高速存储器，读写数据要比SRAM快，但由于SDRAM的读写时序较难控制，因此采用以下方法做液晶扩展：存储在SDRAM的图像数据可以通过DMA转移到DSC21的外部存储器SRAM，数据转移完后，使GPIO为高电平（初始化为低电平），CPLD查询到该IO口为低后，FPGA控制SRAM数据在液晶上显示，其通过SRAM和DSC21的GPIO外接液晶扩展板。其原理框图如图2.16所示。

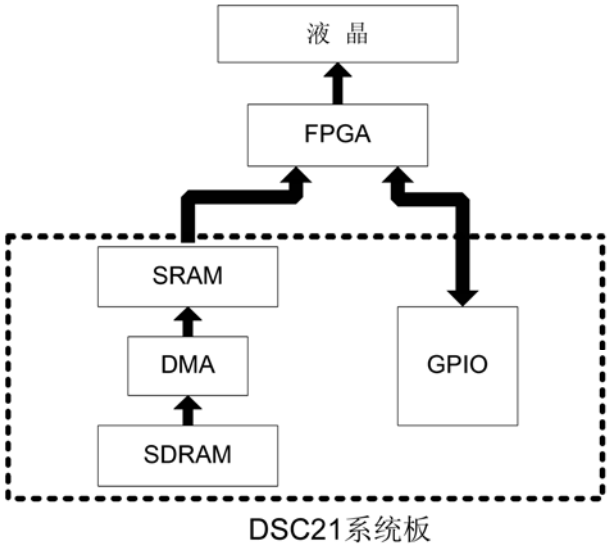

图2.16 DSC21与液晶接口

Fig2.16 Interface DSC21 and LCD

2.11 硬件调试的几点体会

作者在调试电路板的过程中遇到与JTAG接口有关的一些问题而导致仿真无法进行，现归纳如下：

- 仿真器是否支持OMAP芯片？
- 是否选择了OMAP驱动程序？
- 仿真器同目标系统是否连接正确？
- 是否设置了正确的芯片(processor)名？
- 仿真器的I/0口地址是否选择正确？检查是否I/0地址同其它的硬件冲突。如果冲突，修改相应的硬件设置。

3 基于 DSC2 的数码相机系统软件设计与实现

3.1 软件开发流程

整个平台是在 CCS 环境下仿真、调试，本章重点介绍系统的软件设计。CCS（Code Composer Studio）是一个完整的 DSP 集成开发环境。同时它还支持 OMAP 平台的开发。

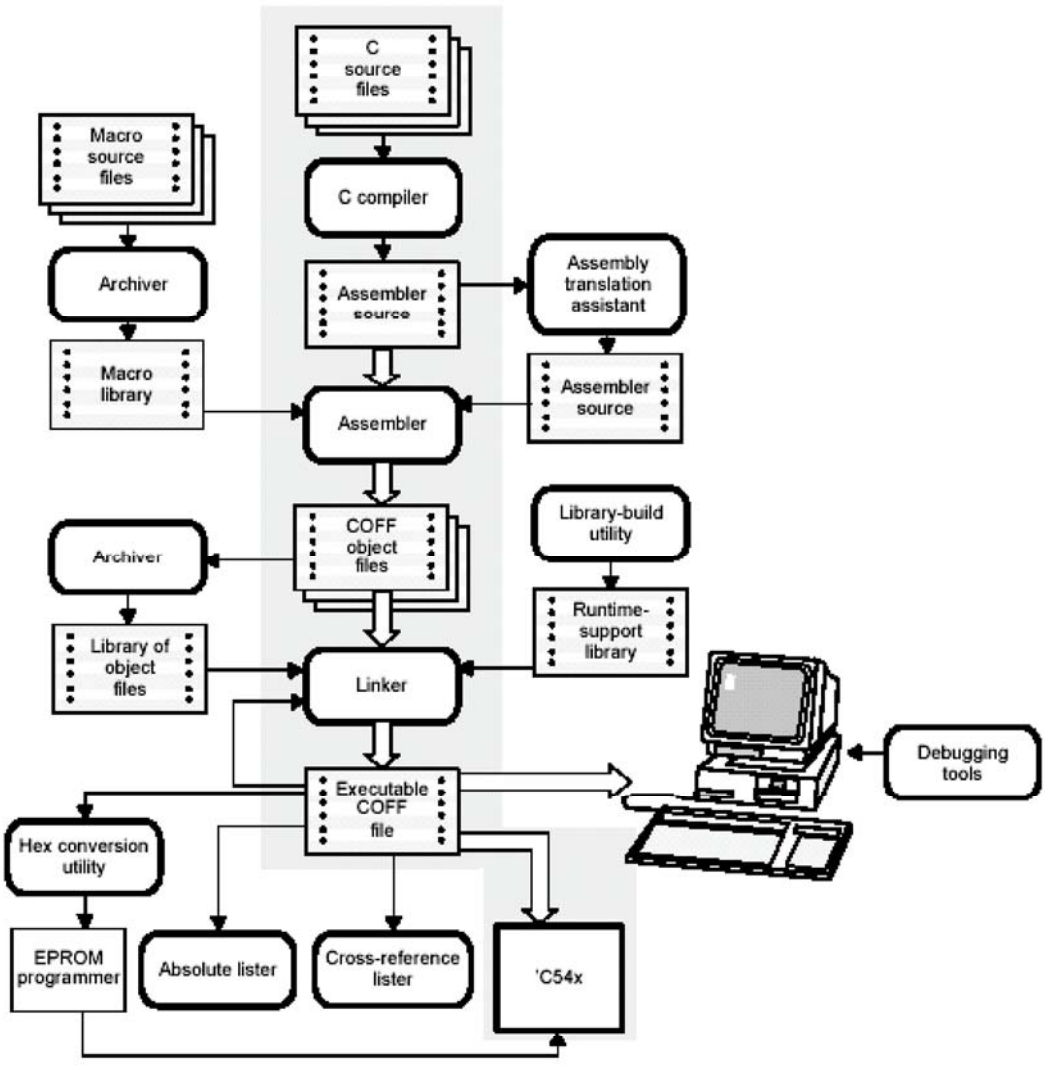

图 3.1 软件开发流程

Fig 3.1 Software design flow

软件开发流程如图 3.1[8]。其中阴影部分是最常用的软件开发路径，也是本论文所采用的，其余部分是任选的。C 编译器（Ccompiler）把用户所编写的 C 语言源程序自动编译为汇编语言源程序[9]；汇编器（Assembler）把汇编程序汇编成机

器语言，即为 COFF(Common Object File Format)格式的目标文件；链接器（Linker）把汇编生成的可重定位的目标文件组合成可执行的 COFF 目标模块；文档管理器（Archiver）提供了方便地管理一组文档的方法，将一组文件（源文件或目标文件）集中为一个文档文件库；建库工具（Library-built utility）可建立用户优化的 C 语言库或者运行支持库；16 进制转换工具可把 COFF 格式的目标文件转换为普通的 EPROM 编程器能识别的文件格式；绝对地址列表器(Absolute Lister)可列出已链接目标文件中各标号的绝对地址；交叉列表器（Absolute Lister）可列出多个目标文件之间标号的相互引用及其定义；调试工具（Dobugging Tools）用于目标代码的除错和优化，常用的调试工具有软件模拟器（Software Simulator，简称 SIM）和评估模块（Evaluation Module,简称 EVM）。

3.2 ARM 程序设计

3.2.1 程序流程

整个程序流程如图 3.2 所示。首先 ARM 对各模块进行初始化，若初始化不成功，则 ARM 发送错误信息到超级终端提示，则结束程序的执行；若初始化通过则 ARM 等待超级终端发送命令，根据命令代号执行相应的操作。

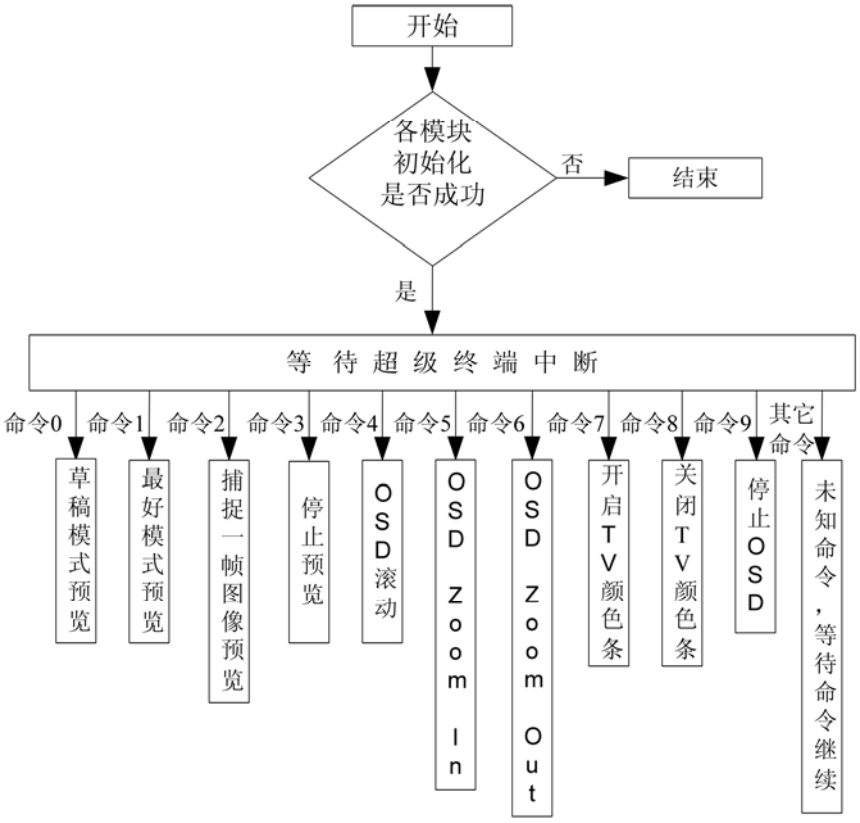

图3.2 ARM程序流程图

Fig3.2 The flow frame of ARM programme

3.2.2 系统的初始化

在系统需要完成某项任务前，必须对系统各个模块初始化，进行相关的设置。初始化程序在 ARM 程序中占有很重要的作用，需要注意的是若 ARM 中程序用到 SDRAM 数据空间，则必须首先对 SDRAM 进行初始化，否则程序链接后生成的目标文件从 SDRAM 开始地址处不能下载。SDRAM 初始化过程如下[7]：

①上电等待时间设置 CAS： 设置 SDRAM 控制寄存器中模式寄存器（地址为 0X000309A6 的寄存器位 11）

Bank 选择：设置 SDRAM 控制寄存器中模式寄存器位 10；

存储器类型：设置 SDRAM 控制寄存器中模式寄存器位 9－8；

②所有 Bank 充电：设置 SDRAM 控制寄存器中模式寄存器位 0～5：设为 2；

③自动刷新：设置 SDRAM 控制寄存器中刷新寄存器，至少刷新 8 次；

④模式寄存器设置位 0～5。

具体方法是首先在 CCS 下建立一工程文件，完成对 SDRAM 初始化操作，cmd 存储器配置文件不能用 SDRAM 空间即，0x8000000 到 0x9fffffff，初始化 SDRAM 程序运行后，才能用这段空间进行数据的存放。

SDRAM 控制器地址为 0x00030980，因此 SDRAM 控制器寄存器可定义如下[10]：

```
static struct SDRAMC
{
volatile short   sdbuf_d[16];  // 15:0   SDRAM R/W data 0..F
short   sdbuf_ad0;          // 8  address auto increment (1: enable)
                            // 5:0     SDRAM address (upper) (/8)
short   sdbuf_ad1;          // 15:0 SDRAM address (lower)
short   sdbuf_ctl;      // 3  buffer clear (1:clear)
                        // 2  write to SDRAM only modified (1:write)
                        // 1  write SDRAM all (1:write)
                        // 0  read from SDRAM (1:read)
short   sdram_mode;     // 12  DMA selection (0:External memory, 1:IrDA)
                        // 11       cas latency (0: 2cycle, 1: 3cycle)
                        // 10       bank number (0: 2bank, 1:4bank)
                  // 9:8  memory type (00:2k*256,01:4k*256, 10:4k*512, 11:8k*512)
                     // 5:0   SDRAM command
short refresh_control;  // 8:     enable refresh (1:enable)
                        // 7:0 refresh cycle (REFNUM) - 8xREFNUM SDRAM clocks
```

```
} *sdramc= (struct SDRAMC *)0x00030980;
```

SDRAM 初始化程序如下：

```
unsigned short    Mode;
Mode= (0<< 11) | (1<< 10) | (1<< 8) | 0x00;
sdramc->sdram_mode=Mode;
dramc->refresh_control= (1 << 8) |0x40;

// SDRAM  命令初始化 SDRAM
sdramc->sdram_mode= Mode | 0x02;
for (i= 0; i < 8; i++)
{
sdramc->sdram_mode= Mode | 0x04;            // 至少刷新 8 次，自动 refresh
}
sdramc->sdram_mode=Mode | 0x01;                  // 模式寄存器设置
```

其它模块初始化过程如下：

①时钟控制器初始化

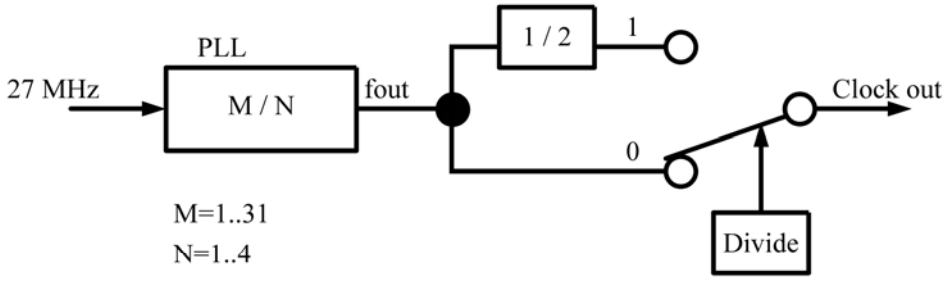

图 3.3 锁相环框图

Fig 3.3 PLL block diagram

ARM，DSP和SDRAM都有各自独立的锁相环(PLL)，它们的工作频率都可由时钟控制器设置。锁相环框图如下图3.3所示。

系统用 27M 时钟源，M 可为 1～31，N 为 1～4，因为锁相环最大晶振频率是 160MHZ，因此不能随意设置 M/N。对 ARM，DSP 和 SDRAM 都存在最大工作频率，ARM 最大工作频率为 40MHZ，DSP 为 90 MHZ，SDRAM 接口为 77 MHZ。以下介绍 ARM 的锁相环设置方法。

ARM 的 PLL 寄存器（地址为 0x00030A80）位 10～14 设置 M 值，位 8～9 设置 N 值，是否 1/2 分频由位 7 设置。例如若 ARM 的工作频率为 27.000MHZ*（11/4）

/2＝37.125MHZ，则：

M=11　N=4　1/2 分频，ARM 的 PLL 寄存器设为 0x2f80。

对于DSP和SDRAM设置方法类似。

②CMOS 初始化[11]

该平台用图像传感器 CMOS7620（30 万像素）和 CMOS2610（采用 OV2610 CMOS 芯片）均调试成功，以下以 OV2610 为例介绍 CMOS 初始化方法。

OV2610 是美国 OmniVision 技术公司推出的彩色 CMOS 图像传感器，内置 10 位 A/D，图像分辨率可为：1600*1200（UXGA 模式），800*600（SVGA 模式），CMOS2610 模块上可由跳线来设置。输出数据格式为 RGB 或 YUV。可编程 / 自动曝光增益控制、可编程 / 自动白平衡控制；控制接口为 SCCB（Serial Camera Control Bus）从接口，降功耗模式。

OV2610 具有 10 位数字视频接口，MSB 与 LSB 由寄存器设置可交换，OV2610 与 DSC21 中 CCD 控制器采用默认的 8 位连接，即连接 OV2610 的 D2~D9 到 CCD 控制器的 D0~D7。OV2610 有两种工作方式：主模式和从模式。主模式下，OV2610 则作为从设备，此时 XCLK 不能与外部晶体震荡器相接，而是直接受主设备时钟的控制。从模式下，OV2610 作为主设备，此时 XCLK 上由外部晶体震荡器输入，并且经过内部分频后可得到像素时钟（PCLK）信号。该系统中，OV2610 以主模式工作。

系统上电后，应首先对 CMOS 图像采集芯片进行初始化，以确定采集图像的开窗位置、窗口大小和彩色工作模式等。这些参数均受 OV2610 内部相应寄存器值的控制。内部寄存器的值可以通过 OV2610 芯片上提供的 SCCB 串行控制总线接口来存取。SCCB 接口是采用一种简单、双向二线制的同步串行总线 I2C 总线，配置的具体方法可采用三相写数据的方式，即在写寄存器过程中先发送 OV2610 的 ID 地址，然后发送写数据的目地寄存器地址，接着是要写的数据。I²C 总线写数据和读数据流程如图 3.4 所示。

在电路中，I2C 总线的串行总线 SDA（串行数据）、SCK（串行时钟）连接到 ARM 的通用 I/O 引脚 GPIO。以下介绍 I2C 总线数据传输。

I2C 总线的信号：

I2C 总线上每传输一个数据位必须产生一个时钟脉冲。SDA 线上的数据必须在时钟线 SCL 的高电平周期保持稳定（如图 3.5 所示），数据线的电平状态只有在 SCL 线的时钟信号是低电平时才能改变，在标准模式下，高低电平宽度必须不小于 4.7us. I2C 总线的起始条件流程如下：

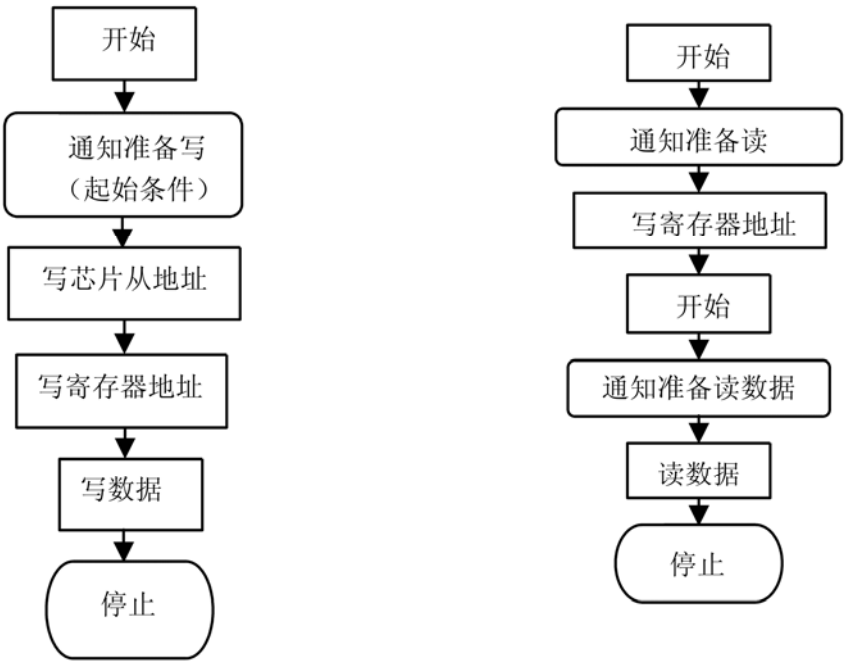

图 3.4 I²C 总线写数据(左)和读数据(右)流程

Fig3.4 Read and write data in I^2C flow frame

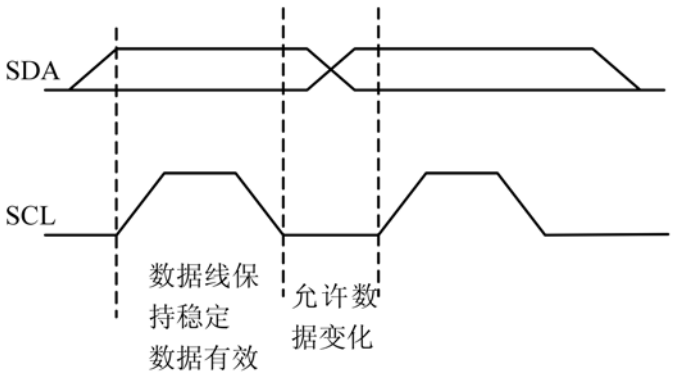

图3.5 I²C总线的位传输

Fig3.5 I²C bus bit flow

I2C总线的起始和停止条件：在SCL线是高电平时，SDA线从高电平向低电平切换。停止条件是在SCL线是高电平时，SDA线从低电平向高电平切换。

首先读OV2610寄存器PIDL（地址为0x0B），该寄存器值为0x41或0x40，在这两种情况下，需要设置的控制寄存器及寄存器值是有区别的，该论文所用的OV2610芯片PIDL寄存器值为0x41，则在UXGA格式下，需要设置的控制寄存器及相应的值如下表3.1所示。

表 3.1 UXGA 寄存器设置

Table3.1 UXGA Register Settings

Register	Address	Value	Description
COMH	0x12	0x80	复位寄存器
COMI	0x13	0x00	不使用自动功能
CLKRC	0x11	0x80	10fps UXGA
COMH	0x12	0x20	UXGA 格式
COMC	0x09	0x01	减小 FPN
COME	0x0D	0x00	减小 FPN
CHLF	0x33	0x0C	设置改进图像的参考电流
VBLM	0x35	0x90	调整内部 reference 和 anti-blooming
VCHG	0x36	0x37	调整内部 reference

③视频解码器初始化

视频解码器把CCD的数据转换为NTSC或PAL格式。初始化操作主要设置以下相关的寄存器：

- 转换的格式，NTSC或PAL格式；
- 扫描模式，隔行扫描或非隔行扫描；
- 输出模式，RGB或合成输出；
- 系统时钟选择，27MHZ或28.6MHZ

④外部总线控制器初始化

在DSC21系统板上，外部总线控制器接口设备为：Flash存储器、CF卡、Smart Media卡，初始化需要进行以下设置：

- 通过CFCRST引脚复位外部总线控制器；
- 选择外部接口设备是CF卡还是Smart Media卡；
- 若选择了CF卡则选择CF卡工作模式是IO模式还是存储器映射模式；
- 访问接口设备的等待时间；
- 写CF或Smart Media卡的时间。

⑤超级终端初始化

DSC21支持两个通道串行UART接口，UART发送和接收端均有32字节的FIFO缓冲区，初始化进行以下设置：波特率（计算公式如下），极性，传送字节大小，使能接收中断。

Bit rate set value=CLK frequency/（bit rate×16）-1

⑥通用IO口（GIO）初始化

DSC21含有32个通用IO口，每个IO口可设为输入或输出，需要注意的是GIO0

到GIO13不是作输入输出用，而是有特殊功能，如中断、串行接口时钟、CCD快门等。IO口初始化设置IO口方向以及初始值即可。

⑦CCD控制器初始化

CCD控制器需要进行以下设置：

- 工作模式设置，包括：场模式设置、行场同步信号极性和方向；
- HD（水平同步信号）宽度，VD（垂直同步信号）宽度；
- 图像数据存入SDRAM的首地址；
- VD1，VD2中断发生在一帧图像中的位置。

⑧OSD初始化

OSD模块初始化设置主要是：OSD三个窗口的位置、视频格式、各个位图颜色查找表的地址，OSD管理的三个窗口都可直接由相应的寄存器控制是否激活。

3.3 DSP 程序设计 [15][16]

DSP处于DSP子系统的核心控制地位，负责与ARM协调工作，并控制其外围资源完成所需的图像处理任务。DSP编程主要集中于图像处理：图像压缩与解压、3A控制（自动白平衡、自动曝光、自动对焦），图像的噪声去除等。

当系统上电后，ARM会对DSP的片上存储器RAM进行测试，然后下载可执行代码到DSP的RAM中（必要时，ARM会将部分代码下载到SDRAM中），这样DSP便可以独立的工作，一般处于低功耗的等待模式，即等待ARM的命令然后完成相应的任务。在执行任务的过程中，DSP可以利用的存储资源包括32K字长的片上RAM以及扩展的外部存储资源，即两个图像缓冲区：iMX系数存储区、iMX命令存储区、VLC量化表存储区以及VLC霍夫曼编码表存储区等。需要指出的是，DSP还可以通过图像缓冲区访问SDRAM，这项功能极大的扩展了DSP的可用资源。更重要的一点是，DSP可以通过与iMX协处理器共享iMX命令存储区实现对iMX的配置，动态的控制iMX完成所需的计算任务，而使自身解放出来做其他工作，实现并行处理；DSP对VLC的控制则是通过直接将外部存储空间映射到VLC的寄存器实现的，其控制特点也是具有并行处理的特点，即只需完成相应的准备工作，VLC和DSP便可以各自独立的工作。这两个协处理器可以并行运行，但这时它们所用的存储器缓冲区必须分开，由DSP来控制这两个加速器的缓冲器的分配。DSP可以通过查询或中断的方式判断iMX或VLC的工作状态。DSP处理后的数据都是转移到SDRAM。DSP不能直接控制DSC21的外设，如CCD控制器、CF卡等[12]。

DSP程序由C语言和汇编混合编程实现。整个程序有以下类型的文件：

.c　C语言程序模块

.asm 汇编程序模块

.inc 汇编程序头文件

.lib 库文件

.cmd 链接文件，描述存储器分配

.gel CCS集成环境的配置文件

3.3.1 编程 DSP 外设

DSP可以访问或者说可以和以下外设通讯：图像缓冲区、SDRAM、iMX加速器、VLC引擎、ARM与DSP的通讯[6]。

① 图像缓冲区

DSC21中存在两个缓冲区块，即图像缓冲区A（Image Buffer A）与图像缓冲区B（Image Buffer B）、DSP、加速器引擎与SDRAM控制器是共享这两个缓冲区的。每个缓冲区为2K字大小，都映射到DSP的数据空间page 1。对DSP来说，它们是外部存储器，不允许双口存储器访问。iMX和VLC引擎都含有命令和系数缓冲区，它们可以被DSP和加速器访问。下表3.2是这些缓冲区的分配情况：

表 3.2 缓冲区分配表
Table3.2 Image buffer map table

缓冲区 **ID**	大小	位宽	地址		可访问模块			
			Start	**End**	**DSP**	**iMX**	**VLC**	**SD-DMA**
IMGBUFA	2k	16 bit	0x8000	0x87FF	x	x	x	x
IMGBUFB	2k	16 bit	0x8800	0x8FFF	x	x	x	x
iMXCOEFFBUF	5k	12 bit	0x9000	0xA3FF	x	x		
iMXCMDBUF	0.5k	16 bit	0xB000	0xB1FF	x	x		
VLCQBUF	0.5k	16 bit	0xB200	0xB3FF	x		x	
VLCHFFBUF	2k	16 bit	0xB800	0xBFFF	x		x	

以下是图像缓冲区的定义：

```
#define     IMGBUFA              0x8000
#define     IMGBUFB              0x8800
#define     iMXCOEFFBUF          0x9000
#define     iMXCMDBUF            0xB000
#define     VLCQBUF              0xB200
#define     VLCHUFFBUF           0xB800
#define     IMGBUFASIZE          0x0800
#define     IMGBUFBSIZE          0x0800
#define     iMXCOEFFBUFSIZE      0x0F00
#define     iMXCMDBUFSIZE        0x0200
#define     VLCQBUFSIZE          0x0200
#define     VLCHUFFBUFSIZE       0x0800
```

TI公司为DSC21开发提供有专门的API函数SwitchBuffer（bufferselector，

bufferID）可以方便地实现外设之间转换缓冲区，例如：

SwitchBuffer（SELIMGBUFA， iMXCOEFFBUFDSP）可以转换IMGBUFA到到DSP。

②SDRAM控制器

图像缓冲区A与图像缓冲区B与SDRAM控制器相连接，可以读写与SDRAM的数据，对DSP来说，虽然直接访问存储器最多16K，但这种接口使得DSP可以处理大量数据，其方法如下：

- DSP写寄存器配置SD-DMA。
- DSP发送SD-DMA数据转移请求。
- SD-DMA进行数据转移，DSP可以进行其它任务。
- SD-DMA数据转移完成后，给DSP产生中断3。

为了简化编程，DSC库提供了两个函数以使SDRAM与图像缓冲区A、图像缓冲区B之间进行数据转移。这两个函数是：SDRAMRdFast与SDRAMWrFast。

③iMX加速器

iMX 作为专用的图像加速器，可以很方便的通过与 DSP 共享存储区域完成配置，实现一维和二维数字滤波以及矩阵乘法等数学运算。根据运算的需要，iMX 有专用的输入数据存储区（端口）、系数存储区（端口）和运算结果输出区（端口）。在 DSC21 中 iMX 可以直接使用的存储资源有两个 2K 字长的图像数据缓冲区、iMX 系数存储区（5K 字长）以及 iMX 命令存储区（0.5K 字长）。iMX 工作与否可以通过 DSP 控制，为了提高开发速度，DSC21 开发板提供有专门的 API 函数供 DSP 调用。

④VLC 引擎

VLC 加速器是为了优化量化计算和霍夫曼（Huffman）编码而设计的，DSP 通过共享存储区为 VLC 提供量化矩阵和霍夫曼编码表，同时控制 VLC 的工作模式和工作状态。VLC 的基本功能包括量化、“之”字扫描、8 位精度的基于 DCT 变换的 JPEG 编码，以及 MPEG-1 视频编码。

3.3.2 DSP 软件的功能结构

DSP程序按照程序的功能可以分为以下五层：顶层、应用层、应用支持层、功能层、系统层（图3.6所示）。顶层位于主程序main函数中，系统复位后开始执行，基本的操作是：一是系统初始化，包括初始化DSP、SDRAM接口编程变量和中断服务程序中的内部变量。二是初始化内部变量；三是执行死循环等待ARM命令，没有命令时DSP处于idle状态。应用层是执行图像的预览、捕捉、回放。应用支持层执行JPEG编码以及图像的3A处理。功能层是执行管道的处理功能以及IMX信号处理功能。系统层是完成ARM与DSP的通讯、SDRAM接口和IMX接口的控制。

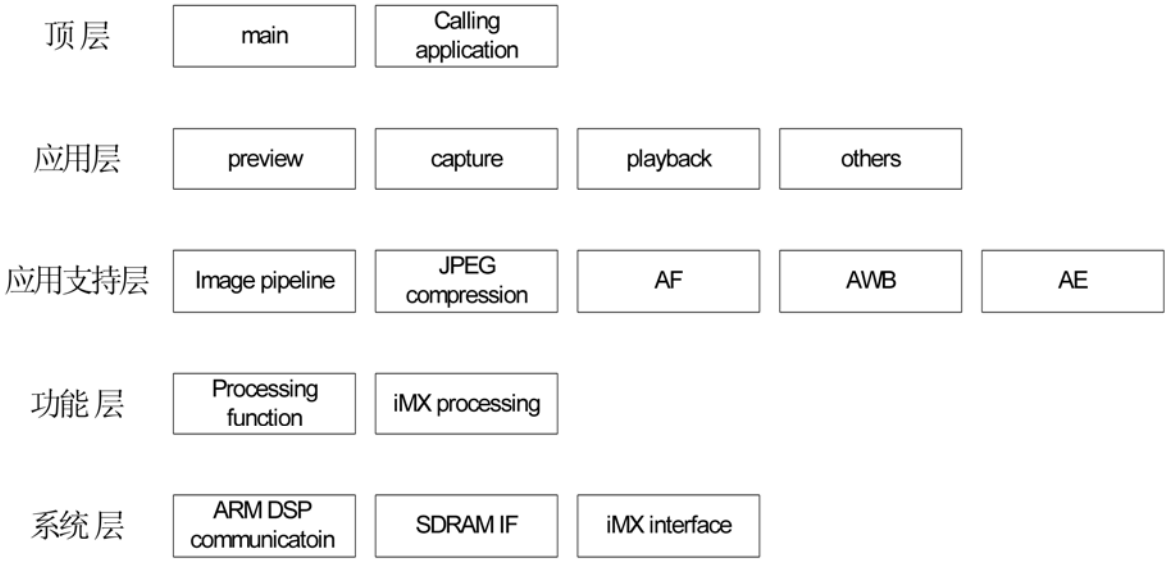

图3.6 DSP软件功能层次图

Fig3.6 Functional level of the software

3.4 ARM 与 DSP 通讯程序设计

ARM与DSP之间通讯是通过把DSP数据空间当作命令寄存器来实现的。ARM通过HPI口获得对这些寄存器的访问。DSP把命令ID号写到ARM命令寄存器（ARMCR），并且发中断给HPI。同样，ARM写命令到DSP命令寄存器（DSPCR），并且发HPI中断，DSP在中断服务程序中判断命令类型并执行相应的操作。命令序列是有继的，DSP发送命令给ARM命令完成后ARM才有可能继续发一个新命令。

有两个寄存器对来建立ARM与DSP之间的异步通讯，一个寄存器对是ARM使用，另一个是DSP使用。每个寄存器对有命令寄存器（ARMCR，DSPCR）和命令完成寄存器（ARMCCR，DSPCCR），命令完成寄存器是存放命令执行完后返回的状态[13]。

3.4.1 DSP 执行 ARM 的命令

DSP执行ARM的命令，首先是由ARM发送命令，以下是ARM发命令给DSP的通讯过程[13]：

- ARM发命令ID给DSP命令寄存器（DSPCR）；
- ARM发中断给HPI；
- DSP在中断服务程序中读命令ID并把该值赋给变量modectl；
- DSP开始执行命令；
- DSP完成命令后写命令完成代码（命令ID+状态代码）；
- DSP清除DSP命令寄存器DSPCR；

• DSP给ARM发HPI中断。

DSP程序中定义由ARM产生的HPI中断向量入口地址用汇编实现，另外，当SDRAM数据与DSP图像缓冲区DMA传送完成时，SD-DMA给DSP发中断，该中断为DSP外部中断3。

```
.ref      c_int00、c_int24、c_int25;
.sect     "vectors"
  B       _c_int00        ;复位中断
.space    4*23*16+32
  B         _c_int24      ;外部中断3
.space    32
  B       _c_int25        ;HPI中断
```

3.4.2 ARM 读写 DSP 存储器

ARM读写DSP存储器数据可以8位或16位模式，即HPI8或HPI16模式。

①HPI8模式

在HPI8模式下，ARM访问DSP内部DARAM利用8位HPI总线HD[7:0]与多个控制线。在DSP 5409内部，有3个16位寄存器：HPIA，HPID，HPIC，它们分别是地址寄存器、数据寄存器、控制寄存器，访问这三个寄存器是由HCNTL1和HCNTL0两位来决定，如表3.3所示。ARM的映射空间0x100000h到0x1FFFFF可以和DSP交换数据，在这种模式下，在此空间内，任何一个地址空间数据都是相同的，例如，当HCNTL1-HCNTL0=00时，读HPIC控制寄存器，HPIC的值从ARM映射空间0x100000h-0x1FFFFF任何一个地址读出都是相同的。因为DSP利用HD[7:0]，DSC21内部HPIB时钟自动产生2个8位访问。其时序如图3.7所示[7]。

表 3.3 HCNTL1-0 与 HPI 寄存器

Table3.3 HCNTL1-0 and HPI registers

HCNTL1	HCNTL0	HPI 寄存器
0	0	HPIC
0	1	HPID 自动增量模式
1	0	HPIA
1	1	HPID

HPIC控制HPI8模式功能，产生和清除在ARM与DSP之间通讯的中断，ARM和DSP都可访问HPIC，但它们有些位的定义是不同的(见表3.4)，HPIC在DSP的存储器映射寄存器地址是0x2c。

说明：RSV：保留位

HINT：主机中断，当 DSP 写该位为“1”时，则给 ARM 引起一个中断；当 ARM 写该位“1”时，则清除中断。

DSPINT：DSP 中断，当 ARM 写该位“1”时，则给 DSP 引起一个中断，该位只能写。

BOB：字节顺序位，当为“1”时，16 位数据的高字节和低字节交换。

HPIEN：HPI 使能位，表示 HPIEN 信号的状态，仅仅 DSP 能读该位。

表 3.4 HPIC 位定义区别

Table3.4 HPIC registers bit in DSP and ARM

位	11	10	8	7	2	0
DSP	HINT	DSPINT	BOB	RSV	DSPINT	BOB
ARM	RSV	RSV	RSV	HPIEN	RSV	RSV

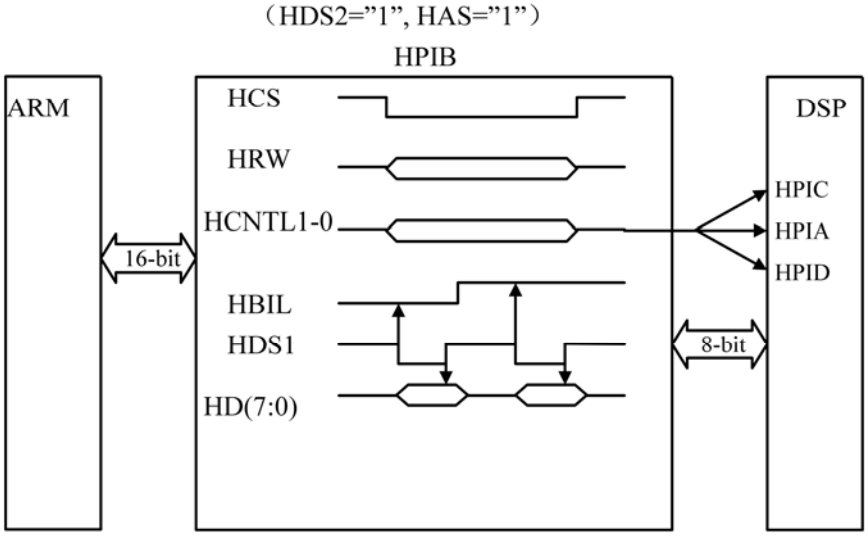

图 3.7 HPI8 时序图

Fig3.7 HPI8 timing diagram

DSP的HPIA寄存器是表示DSP内部DARAM的地址，当HCNTL1-HCNTL0=00或11时，ARM内部0x100000h-0x1FFFFF空间由HPIA指定的地址16位数据被转移，当HCNTL1-HCNTL0=11时，HPIA寄存器值在数据转移后不会变化，但当HCNTL1-HCNTL0=01时，在数据转移后HPIA寄存器值自动增加，当读数据时，HPIA寄存器值自动后增，当写数据时，自动前增。

以下ARMRdDSPData函数实现从DSP内部RAM读数据到ARM，函数参数说明：

*DataARMAddr：读数据存放的目的地址

DataCount：数据个数

DSPAddr：读数据DSP源地址

dspc->hpib_ctrl：DSP的HPI控制寄存器

函数中的常数定义：

```
#define HINT        0x0808    // DSP的HPI中断
#define HCNTL0      0x1000    // HPI-8时HPI寄存器控制
#define HCNTL1      0x2000    // HPI-8时HPI寄存器控制
```

函数定义：

```
void ARMRdDSPData(unsigned short *DataARMAddr,
unsigned short DataCount, unsigned short DSPAddr)
{
    unsigned short * pSource;
    unsigned short  nCtrl;
    int i;
    nCtrl= dspc->hpib_ctrl & ~(HCNTL0 | HCNTL1);
    dspc->hpib_ctrl= nCtrl;                          // "00" HPIC控制寄存器
    *(volatile short *)DSPRAM_START= HINT;           // 清中断clear INT
    dspc->hpib_ctrl= nCtrl | HCNTL1;                 // "10"HPIA地址寄存器
    *(volatile short *)DSPRAM_START= DSPAddr;        // DSP地址
    dspc->hpib_ctrl= nCtrl | HCNTL0;                 // "01"HPID数据寄存器
    pSource= ((unsigned short *)DSPRAM_BASE) + DSPAddr;
    for (i= DataCount; i > 0; --i)
    {
       *DataARMAddr++= *pSource++;
     }
    }
```

② HPI16模式

当HPIBCTL寄存器位HPIM为“1”时，则ARM与DSP通讯以HPI16模式进行，对该模式，ARM可以直接读写DSP内部DARAM数据，ARM存储器空间0x100000h-0x1FFFFF与DSP存储器空间是共享的。除了DSP存储器映射寄存器空间0x00－0x80h外，其余DSP DARAM空间都可以被ARM访问。

HPI16模式下寄存器不能象HPI8模式下访问，HPIA、HPIC、HPID都没有HPI8模式的功能，访问的数据是以16位（字）为单元，不能以字节方式访问。ARM与DSP之间的通讯不能通过中断来实现。

需要注意的是，该模式下，由于DSP的外部IO引脚XIO被利用，因此，连接到XIO的以下外设不能使用：图像缓冲区（Image Bufer）、IMX、VLC。

3.4.3 ARM 执行 DSP 的命令

DSP 发给 ARM 的命令只有一个，即下载 DSP 程序代码，DSP 的程序在数据空间 Page 1，通过 HPI 口下载该空间的程序代码给 ARM。

3.5 程序代码优化和混合编程

3.5.1 C 程序代码优化

C语言的开发效率较高，代码的可读性和可移植性好，深受广大开发者喜爱，大多数算法的开发主要集中在C代码的开发，因此C代码的优化相当重要，直接影响到开发工作的难度和进程。一般地，C代码的优化方法主要包括以下几种[17]:

- 对算法实现结构进行优化；
- 利用编译器编译选项进行优化；
- 采用内联函数和宏定义，对一些使用频率较高的函数，尤其在循环中反复大量调用的小函数，采用内联函数或者宏定义，可以减少函数调用带来的额外开销，提高执行效率；
- 使用实时库函数来实现DSP运算函数。

3.5.2 汇编程序代码优化

本论文大部分程序代码都是用C编写的，为了提高程序执行效率，也用了汇编编写一部分程序，汇编程序优化常用以下几种方法：

- 使用乘加单元，使用MAC指令，在DSP 5409中，提供了一个乘加单元，在一个指令周期内可实现一次乘加运算。
- 使用32位数据，DSP 5409每条数据总线可以存取16位数据，也可以将CB和DB联合使用，存取32位数据。这样加倍CPU总线的使用效率，大大节约指令周期。但要注意32位数据地址必须偶对齐。
- 使用循环指令：如RPT、RPTB指令。

3.5.3 C 和汇编混合编程

混合编程是指整个程序的实现既有C语言，又包含汇编语言，最后通过开发工具形成一个应用程序。它是一种既能提高软件开发效率，又能增加软件执行效率的行之有效的编程方法，在软件开发中经常被使用。混合编程通常以C语言编程为主，以汇编语言编程为辅[36]。在C54x开发环境下，实现汇编程序与C程序的相互结合通常表现在以下三个方面：

①使用单独的汇编语言模块，即分开编译或汇编形成各自的目标代码模块，链接器将C模块和汇编模块链接[18]。这种方法灵活性较大，但要注意的是传递的参

数在堆栈中的偏移量。若有返回值，返回值必须放在寄存器A中。

②在C语言代码中使用汇编程序中定义的变量和函数；

③在C语言代码中嵌入单行汇编代码。此方法可在C程序中实现C语言无法实现的一些硬件控制功能，如：修改中断控制器，中断标志寄存器等。

3.6 CCS 下软件调试的几点体会

作者在 CCS 下调试程序，遇到过以下一些问题在此总结：

程序跑飞现象：C 程序编译后运行时 PC 指针跑飞一般是由于访问不存在的存储区造成的。这时可以首先要检查程序链接后生成的.MAP 文件与存储器映射图对比，看是否超出范围或者是否用到了存储器映射寄存器作为段的分配。如果在有中断的程序中跑飞，应重点查在中断程序中是否对所用到的寄存器进行了压栈保护。如果在中断程序中调用了 C 程序，则要查汇编后的 C 程序中是否用到了没有被保护的寄存器并提供保护(在 C 程序的编译中是不对 A、B 等寄存器进行保护的)。

移位问题：在 C 语言中把变量设为 char 型时，它是 8 位的，但在 DSP 汇编中此变量仍被作为 16 位处理。所以会出现在 C 程序中的移位结果与汇编程序移位结果不同的问题。解决的办法是在 C 程序中，把移位结果再用 0x00FF 相“与”即可。

cmd 文件段的分配区域：.cmd 文件编译链接器默认 page 0 是 ROM 区，page 1 是 RAM 区，有些段必须放在 ROM 区，它们是：

.text

.const

.cinit

.switch

3.7 结论

为了验证 ARM 程序设计的正确性，我们采用了两种方法：

一是通过 CMOS 采集，在程序运行在预览模式或捕捉模式下时，数码相机系统合成视频输出连接到 PC 机视频卡，通过 PC 机显示器观察的图像(图 3.8 示)。

另外，合成视频是模拟视频，为了得到数字视频，目标板上的存储器的数据可以以数据文件形式保存到PC机，作者在CCS下从JTAG口读取目标板上SDRAM中图像数据生成 dat 文件，然后用 VC 编写的 JPEG 编码程序读取该文件图像数据，生成 JPEG 文件，图 3.9 示 JPEG 文件图像。

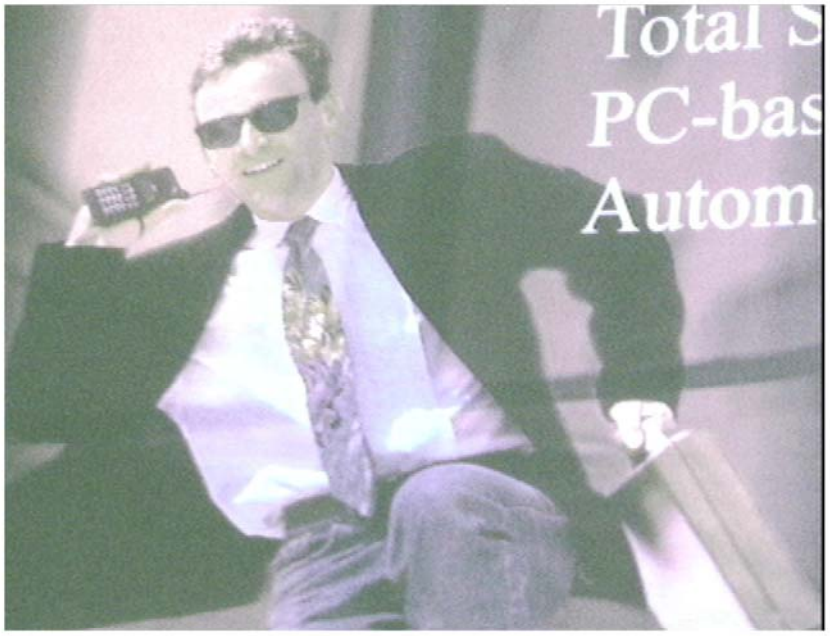

图 3.8 合成视频图像
Fig3.8 Compositive vide image

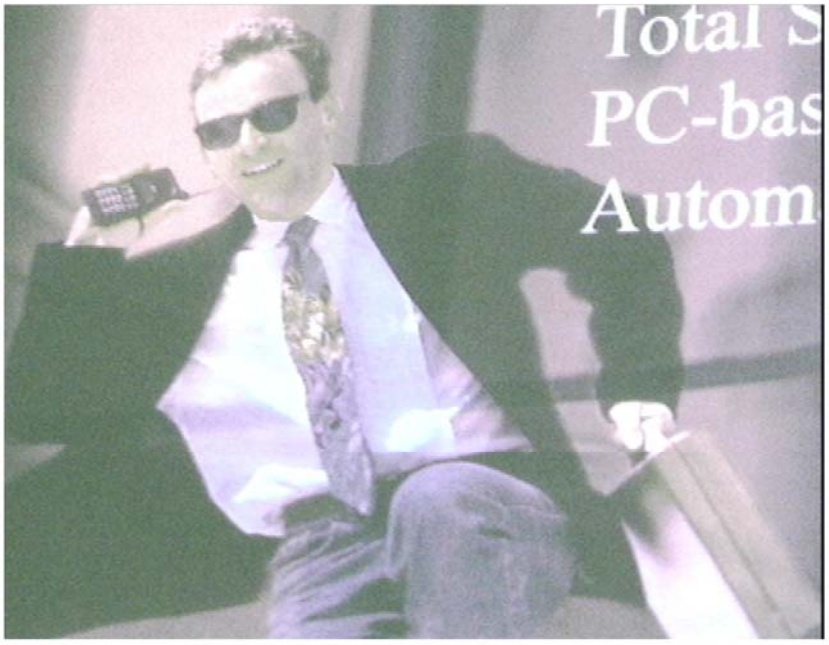

图 3.9 JPEG 编码图像
Fig3.9 JPEG Coding image

4 基于 CF 卡的文件读写程序设计

本章首先介绍了常用的数码相机存储介质以及它们的优缺点，然后对FAT文件系统结构进行阐述，本论文实现了对CF卡文件系统的读写。

4.1 数码相机的存储介质

目前市场上数码相机所用的存储介质可谓五花八门，它们基本上可以分为闪存记忆体(如 SM 卡、CF 卡等)、磁记录体(如 1.44MB 软盘、ZIP 和 ClIk 等)和相变记录体(如 CD－R/CD－RW 等)，容量从几 MB 到几百 MB 不等，有的甚至达到了 1G。虽然这些存储介质都用在数码相机上，但都有各自的优点和缺点。SM 卡和 CF 卡是在数码相机中应用得最广泛的两种存储卡。

SM 卡全称为 SmartMedia，也称“聪明卡”，是最早应用在数码相机上的存储介质之一。它的优点是体积小巧，重量轻，所以采用它的数码相机也能做得比较小巧，而且它也应用在其他一些数码产品上，如 MP3 播放器等。由于体积小，它的缺点也是挺明显的。它的容量不能做得更大，大部分 SM 卡的容量都在 64MB 以内，最高的也不过 128MB，目前只有富士、奥林巴斯全系列的数码相机支持 SM 卡。

CF 卡全称 CompactFlash，也称“微型闪存卡”，是目前广泛应用在数码相机上的存储介质。它的体积比 SM 卡大，所以容量也能做得更大一些。它又分为 I 型和 II 型接口，II 型接口兼容 IBM 的微型硬盘。不过，它目前的容量和价格与 SM 卡相比并没有优势，64MB 的 CF 卡和 SM 卡价格基本相当。佳能、尼康和柯达的大部分数码相机都采用 CF 卡。

4.2 CF 卡结构与工作模式[19]

Compact Flash 卡目前有 8MB、16MB、32MB、64MB、128MB 和 256MB 等不同容量的产品。论文采用 Compact Flash Card 作为数码相机的存贮介质，CF 卡的内部结构原理如图 4.1 所示。

CF 卡内部含有一个 MCU、一个 DMA 控制器、2k×8×2 的双口 RAM 缓冲区及 Flash 媒体(其容量从数兆 byte 至数十兆 byte 不等)。D0～D15 为数据线，A0～A9 为地址线，CE1，CE2 为卡选择及数据宽度选择。其引脚图见图 4.2。CF 卡有三种工作模式，分别为 Common Memory Mode； I/O Transfer Mode；Ture IDE Mode，上电时若 OE(PIN9)为“0”则 CF 卡自动进入 True IDE 模式，若 OE=“1”则进入 Common Memory 模式。还有一点需要注意的是：Reset 信号在 True IDE Mode

时是低电平有效，而其它模式时是高电平有效[22]。CF 卡是以地址来区分命令和数据的，不论以何种模式工作，CF 卡均以扇区(512bytes)为基本单位进行操作，一次可以读写一个或多个扇区，具体情况由命令(Command)来决定，若要对 CF 卡进行读写操作，首先要指定读写哪个(或哪几个)扇区，即先指定 SecCnt(offset=2，总扇区数，一个扇区还是多个扇区)，SecNum(offset=3，起始扇区号)，Cylinder(offset=4、5，柱面号)，C/D/H(offset=6，LBA 模式/驱动器号/磁头号)。

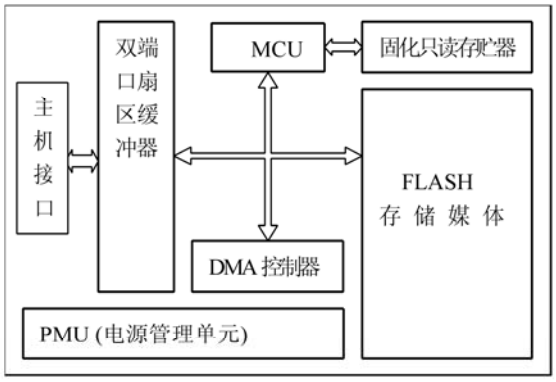

图 4.1 CF 卡内部结构框图

Fig4.1 CF Card inside structure frame

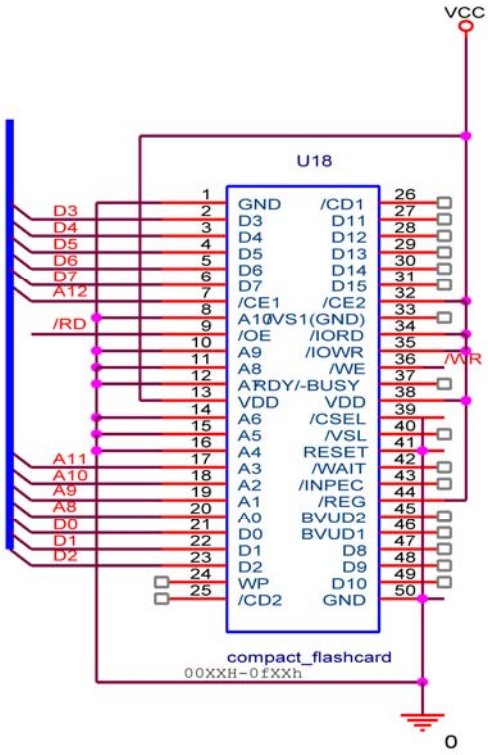

图 4.2 CF 卡引脚原理图

Fig4.2 CF card electronic scheme

4.3 文件

广义上讲，任何一个带有文字、数字、符号、图形等信息的纸片、卡片、磁盘等信息载体，都可以称之为文件[20]。但什算机中的文件显然不能包含那些无意义的信息集合，它必须是在逻辑上县有完整意义的信息集合。文件可以是系统程序、应用程序、数据库或其它信息，它们都是保存在存贮介质（如软盘或硬盘）上的。

4.3.1 文件名

各种系统的文件名规则略有不同，但所有操作系统都允许用一到八个字母组成的字符串作为合法文件名。有些文件系统区分大小写，如UNIX，LINUX，而有些操作系统则不加以区分，如MS-DOS，WINDOWS。许多操作系统支持两部分命名的文件名，中间为“.”号, 后部分称为文件扩展名。

4.3.2 文件系统

文件系统是操作系统中负责存取和管理辅助存储器上文件信息的软件机构。它既要负责对用户的私人专用存储器上信息的访问。也要负责为用户提供一种有控制的方式来访问共享的信息。它是由管理文件所需的数据结构（如目录表、文件控制块、存储分配表）和相应的管理软件以及访问文件的一组操作组成。

从系统角度看，文件系统是对文件存储器的存储空间进行组织、分配，负责文件的存储并对存入的文件进行保护、检索的系统。从用户角度看．文件系统主要是实现了“按名存取”。就是说，当用户要求系统保存一个已命名的文件时，文件系统根据一定的格式把他的文件存放到文件存储器中适当的地方；当用户要使用文件时，文件系统根据他给出的文件名，能够从文件存储器中找到所要的文件或文件中的某一个记录。因此，文件系统的用户只要知道他的文件名，就可以存取文件中的信息，而无需知道这些文件究竟存放在什么地方。

4.4 CF 卡文件系统[21]

CF卡文件系统与硬盘文件系统结构相同：按照顺序分以下四个区域：保留区、FAT区、目录区、数据区域，文件管理系统通过以上4个区域实现对CF卡上的文件进行有效的管理。文件管理系统将文件数据存放在文件数据区，将文件的属性存放在文件对应目录下的FDT表中，将文件的存放位置存放在FAT表中。因此文件管理系统通过FAT表和FDT表可以很方便地对文件数据区的文件进行管理。

FAT类型有：FAT12、FAT16、FAT32，类型的划分是根据数据区所占簇数（CountofClusters）来划分：

如果CountofClusters<4085，则FAT类型为FAT12；

如果CountofClusters<65525，则FAT类型为FAT16；

否则FAT类型为FAT32

CountofClusters由引导扇区的相关信息可以计算出。

FAT文件系统以簇作为空间管理单位，每一簇包括多个扇区。

4.4.1 FAT 文件系统引导信息

保留区的第一个扇区为BIOS参数块BPB，有时也称为引导扇区或第0扇区。它包含一些引导信息，例如：每扇区字节数、每簇扇区数、FAT大小等，引导扇区偏移量36开始，FAT12与FAT16结构相同，但它们与FAT32结构不相同。对DSC21 数码相机系统，作者用64M CF卡调试，用FAT16文件系统结构。在SDRAM中开辟一块512字节的缓冲区用来存放CF卡引导扇区数据，CFCBootBuffer为指向该区域的首地址，以下程序是CF卡引导数据存入SDRAM缓冲区以后计算CF卡的引导信息，各变量定义如下：

BPB_BytsPerSec：每扇区字节数

BPB_SecPerClus：每簇扇区数

BPB_ResvdSecCnt：保留扇区

BPB_NumFATs：FAT大小

BPB_RootEntCnt：根目录最大目录项数

BPB_FATSz16：每个FAT的扇区数

RootDirSectors：根目录占用的扇区数

FirstDataSector：数据区的起始扇区

FirstRootDirSecNum：根目录第一个扇区号

```
//以下是计算CF卡引导信息程序
BPB_BytsPerSec=*(CFCBootBuffer+11) +*(CFCBootBuffer+12)<<8;
BPB_SecPerClus=*(CFCBootBuffer+13);
BPB_ResvdSecCnt=
(unsigned int)(*(CFCBootBuffer+15)<<8)+*(CFCBootBuffer+14);
BPB_NumFATs=*(CFCBootBuffer+16);
BPB_RootEntCnt=*(CFCBootBuffer+17)+(unsigned int)(*(CFCBootBuffer+18)<<8);
BPB_FATSz16=*(CFCBootBuffer+22)+(unsigned int)(*(CFCBootBuffer+23)<<8);
RootDirSectors = ((BPB_RootEntCnt * 32) + (BPB_BytsPerSec - 1)) /
BPB_BytsPerSec;
FirstDataSector = BPB_ResvdSecCnt + (BPB_NumFATs * BPB_FATSz16) +
RootDirSectors;
FirstRootDirSecNum = BPB_ResvdSecCnt + BPB_NumFATs * BPB_FATSz16;
```

4.4.2 FAT 文件系统目录结构

对于 FAT12、FAT16，根目录位于固定的位置，紧位于最后一个 FAT 表，并且是固定大小。根目录区的第一个扇区号可以由引导区的数据计算出来。而 FAT32 的根目录区是大小可变的一个簇链。对任何类型的 FAT，在目录区，都是由多个表目构成，每个表目称为一个目录项。若是短目录，每个目录项占 32 字节，如果是长目录，则每个表项为 64 个字节，其中，前 32 个字节为长文件链接说明；后 32 个字节为文件属性说明，包括文件长度、起始地址、日期、时间等。

表4.1是短目录文件目录结构：

表4.1 文件目录项结构

Table4.1 FAT directory structure

名 称	偏移量	字节大小
文件名	0	11
属性	11	1
保留	12	1
文件建立时间的毫秒标记	13	1
目录或文件建立时间	14	2
建立日期	16	2
最后访问日期	18	2
起始簇号高16位	20	2
最后写时间	22	2
最后写日期	24	2
起始簇号低16位	26	2
文件大小	28	4

①文件名：文件名是两部分命名，包括8字节文件名及3字节扩展名，若文件名长度小于8个字节，则在地址高位以0x20填补，文件名超过11字节将只截取前8个字节，文件名第一个字节DIR NAME[0]说明：

.若DIR_NAME[0]＝0XE5，则在这个目录入口没有文件或目录名；

.若DIR_NAME[0]＝0X00，则从这个目录入口开始已经没有目录入口了，可作为文件检索结束的标志。

扩展名长度小于3字节，则在地址高位以0x20填补，若超过3字节，则超过部分舍弃。

③ 属性：高两位保留，其余位定义如下：

位 0：为 1 时，只读属性；

位 1：为 1 时，隐藏属性；

位 2：为 1 时，系统文件；

位 3：为 1 时，表示该目录项记录了这个 逻辑盘的卷名；

位 4：为 1 时，表示该目录项是子目录：

位 5：为 1 时，存档文件。

④ 时间和日期：文件建立或最后修改的时间和日期，时间格式：

位：15 11 10 5 4 0

小时（0-23）	分（0-59）	秒/2（0-29）

日期格式：

位：15 9 8 5 4 0

年（1980 为基准）	月（1-12）	日（1-31）

⑤ 文件起始簇簇号：目录入口偏移量 20 与 26 开始两字节分别是文件起始簇号的高 16 位和低 16 位，对 FAT12 和 FAT16 文件起始簇号的高 16 位总是为 0。

文件大小：以字节为单位

CF卡格式化后，目录区数据全部为0x00。从目录区我们可以创建或读出目录或文件名，以及文件的第一簇簇号，由文件的第一簇簇号用以下公式可以计算出文件的起始扇区号：

FirstSectorofCluster = ((N – 2) * BPB_SecPerClus) + FirstDataSector;

其中N为簇号；FirstSectorofCluster 为N簇的第一扇区；BPB_SecPerClus与FirstDataSector和5.2.1中定义相同。

4.4.3 FAT 表结构

FAT文件分配表位于引导区后，记录了文件区空间每个簇的使用情况，用于控制文件空间的管理与分配。

在FAT表中，每个表目对应于一个簇号，FAT12、FAT16与FAT32文件系统每个表目所占字节数各不相同，它们分别为：1.5字节、2字节、4字节。以下以FAT16为例，介绍FAT16文件分配表结构。

在文件分配表中，以下数据具有特定的含义：

0000h：表示空闲簇

FFF7h：坏簇

FFFFh：文件结束标志

FAT表的前三个簇号0xFFFFF8是保留簇，其中第一个字节F8h是磁盘介质标志。第二三字节0xFFFF是文件结束标志（EOC），从第三个簇号开始分配给文件，在FAT表中，文件簇号是一个链表，文件的第一个簇号在目录区中（4.2.1所述），

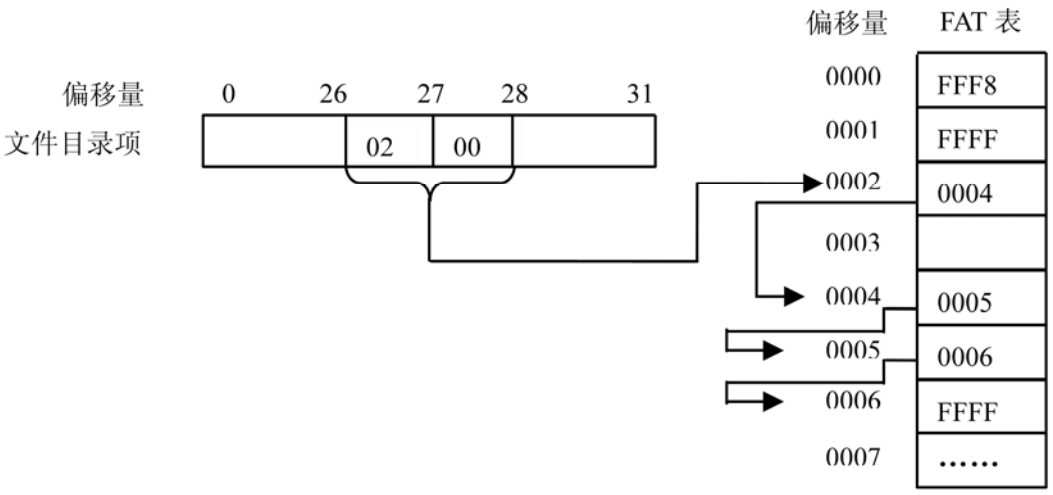

图4.3 文件分配表

Fig 4.3 FAT table

由这个簇号查找第二个簇号，第二个簇号在FAT表中的位置即是第一个簇号作为在FAT表中的偏移量。

假设一个文件在磁盘或CF卡上存储的第一个簇号是2，其余簇依次是：4，5，6，则该文件的数据所占用的簇号在FAT表中的排列可用如4.3图来表示。

4.4.4 CF 卡文件系统的读写

CF卡的扇区寻址有两种方式：物理寻址方式（CHS）和逻辑寻址方式（LBA）。

物理寻址方式使用柱面、磁头和扇区号表示一个特定的扇区，起始扇区是0磁道、0磁头、1扇区，接下来是2扇区，一直到EOF扇区；接下来是同一柱面1头、1扇区等。逻辑寻址方式将整个CF卡同一寻址，逻辑块地址和物理地址的关系为：

LBA地址=(柱面号×磁头数+磁头号)×扇区数+扇区数-1

作者分别在一单片机平台和DSC21数码相机平台上分别实现了对CF卡文件系统的读写，以下分别介绍两个平台的实现方法。

① 在单片机平台上实现

单片机控制CF卡的读写，该平台上，作者实现了USB与CF卡文件系统开发，其原理框图如图4.4，采用的开发环境是Keil C51。

实现过程：PC机与USB通讯。PC机首先测试USB通讯是否正确，可根据自定义的协议，采用异或效验，上位机发命令和数据，若下位机（单片机）通过USB控制器读的命令和数据正确，则表示USB通讯成功，这时才能进行以下的任务。通过USB发命令给单片机，单片机读写CF卡，由于读CF卡一个扇区数据为512字节，

因此需要选用内部RAM比较大的单片机，论文选用Winbord公司的W77E58芯片，其内部具有1K的SRAM，PC机在C++BUILDER6环境下与USB设备通讯，USB的开发在第五章详细介绍。以下以读CF卡文件为例进行讨论。

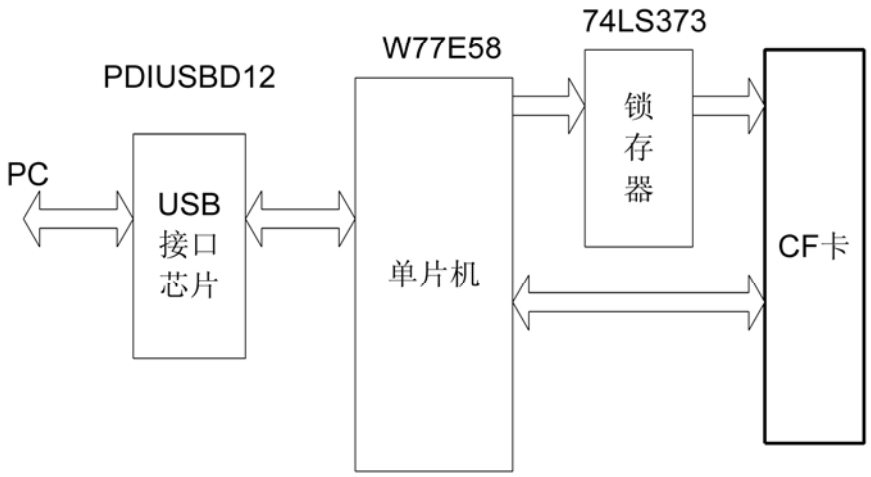

图4.4 单片机与CF卡连接

Fig4.4 MCU and Compact Flash connection

通过USB把CF卡的数据读到PC，则需要依次读CF卡的以下数据：引导区数据、目录结构、FAT文件分配表、文件数据，其中读出的文件簇号存储到上位机缓冲区，需要读文件时，计算出簇号所在的第一个扇区发送给下位机，下位机读CF卡该簇的数据，并传送到上位机，文件所在簇的所有数据都用这样的方式读到上位机，上位机把文件名和其文件数据写成文件，则PC机直接可以打开该CF卡的文件。下位机根据自定义的命令代码，在USB的中断服务程序中按照上位机发的命令读CF卡的相应数据。

②在DSC21平台上实现方法

在该平台上，CMOS图像数据在DSP处理器中经过JPEG压缩后存储到SDRAM，再从SDRAM写JPEG文件到CF卡，可以用PC机通过该平台上的USB接口读该JPEG文件。以下介绍写JPEG文件到CF卡实现过程：

CF卡接口支持8位、16位、32位数据转移。CF卡所有寄存器都映射到ARM存储器空间，CF卡的2K属性存储器（attribute memory）映射到ARM存储器0x04000000－0x040007FF，而2K的普通存储器（COMMON memory）映射到ARM存储器0x04000800－0x04000FFF，所以CF卡的寄存器定义如下：

```
#define    CFC_BaseAddr   0x04000800
#define   DATA_REG       (UCHAR *)(CFC_BaseAddr+0x00)    //数据寄存器
#define   ERROR_REG      (UCHAR*)(CFC_BaseAddr+0x01)    //错误寄存器
```

```
#define   FEATURES_REG   (UCHAR*)(CFC_BaseAddr+0x01)   //特征寄存器
#define   SEC_CNT_REG    (UCHAR*)(CFC_BaseAddr+0x02)   //扇区数寄存器
#define   SEC_NUM_REG    (UCHAR*)(CFC_BaseAddr+0x03)   //扇区号寄存器
#define   CYL_LO_REG     (UCHAR*)(CFC_BaseAddr+0x04)   //柱面低寄存器
#define   CYL_HI_REG     (UCHAR*)(CFC_BaseAddr+0x05)   //柱面高寄存器
#define   HEAD_REG       (UCHAR*)(CFC_BaseAddr+0x06)   //磁头寄存器
#define   STATUS_REG     (UCHAR*)(CFC_BaseAddr+0x07)   //状态寄存器
#define   COMMAND_REG    (UCHAR*)(CFC_BaseAddr+0x07)   //命令寄存器
```

CF卡与SDRAM之间的数据转移可以用两种方法：一是直接转移数据；二是通过DMA转移；DMA转移数据时，可以连续访问DMA控制器中所设置的CF卡普通存储器地址，DMA数据转移大小、方向等设置都是对DMA相应寄存器来说的，转移是否结束是查询DMA的状态寄存器位15是否为“0”，若为“0”，表明转移完成。

从CF卡上读出的引导区数据存放到SDRAM中，如4.2.1节所述。目录定义为如下结构体，该结构体也存储到SDRAM，起始地址为CFFDTDataBuffer，写文件时，直接把信息写到相应的结构体成员，然后把数据从SDRAM转移到CF卡上。

```
struct _FDT      {
    char       Name[11];                //短文件名主文件名
    unsigned char     Attr;             //文件属性
    unsigned char      NTRes;           //保留给NT
    unsigned char      CrtTimeTenth;    //建立时间（FAT16保留）
    unsigned short    CrtTime;          //建立时间（FAT16保留）
    unsigned short    CrtDate;          //建立日期（FAT16保留）
    unsigned short    LstAccDate;       //最后访问日期（FAT16保留）
    unsigned short    FstClusHI;        //起始簇号高两个字节（FAT16保留）
    unsigned short    WrtTime;          //最后写时间
    unsigned short    WrtDate;          //最后写日期
    unsigned short    FstClusLO;        //起始簇(cluster)号低两字节
    long int    FileSize;               //文件大小，32bit
} *FDT= (struct _FDT *)(CFFDTDataBuffer);
```

在CF卡上写入一个文件的过程是这样的，在CF卡初始化后（CF卡上电复位和统计剩余空间等工作已经完成），控制器ARM开始向CF卡的一些寄存器填写必要的信息，如向扇区号寄存器填写读写数据的起始扇区号（LBA地址）和扇区数寄存器填写读写数据所占的扇区个数等，然后向CF卡的命令寄存器写入CF卡操作的命令，如写操作则向CF卡的命令寄存器写入30H，读操作向CF卡的命令寄存器写

入20H等[35]。删除或者读文件的过程相似。

读CF卡函数编程如下：

```
void readCF (long int lba)
{
      unsigned int i;
      while((*STATUS_REG & 0xf0)!=0x50);
      lba = lba & 0x00FFFFFF;
      lba = lba | 0x0E0000000;

     *HEAD_REG=*(((char*)&lba)+3);//drive/head or drive/lba27-24;
     *CYL_HI_REG= *(((char*)&lba)+2);
     *CYL_LO_REG= *(((char*)&lba)+1);
     *SEC_NUM_REG= *(((char*)&lba)+0);
     *SEC  CNT  REG= 0x01;                //SECTOR COUNT REGISTER    0x01

     *COMMAND_REG= 0x20;
      while((*STATUS_REG & 0xf8)!=0x58);
        for (i=0;i<=511;i++)
             *(CFCReadDataBuffer+i)= *DATA_REG;
}
```

写CF卡程序与以上程序相似，只需把命令寄存器写入0x30命令，等待CF卡准备好后，把数据写到数据寄存器。

4.5 小结

本章详细阐述了FAT文件系统的内容、结构，介绍了数码相机图像的一种存储介质CF卡，并在两个平台上实现了FAT文件的读写。

5 PDIUSBD12 的 USB 开发

正如章节4.4.4所述，本论文在基于单片机基础上实现了PC机通过USB接口读写CF卡文件。本章论述基于PDIUSBD12芯片的USB接口开发。

USB（Universal Serial Bus）是通用串行总线的简称。USB 是目前在打印机、数字存储设备、输入/输出设备、数码像机、MP3播放器等其他周边设备中得到广泛应用的连接方式。USB设备具有使用方便，速度快，连接灵活，即插即用，总线供电等优点。

5.1 PDIUSBD12 芯片结构

PDIUSBD12（以下简称 D12） Philips 公司一种 USB 接口芯片采用的专用芯片。该芯片完全符合 USB1.1 规范，还符合大多数器件的分类规格：成像类、海量存储器件、通信器件、打印设备以及人机接口设备[22]。

D12集成了SIE、320B的多配置FIFO存储器、收发器、电压调整器、Soft Connect、GoodLink、可编程时钟输出、低频晶振和终端电阻等，支持双电压工作、完全自动DMA 操作、多中断模式，内部结构如图5.1所示。

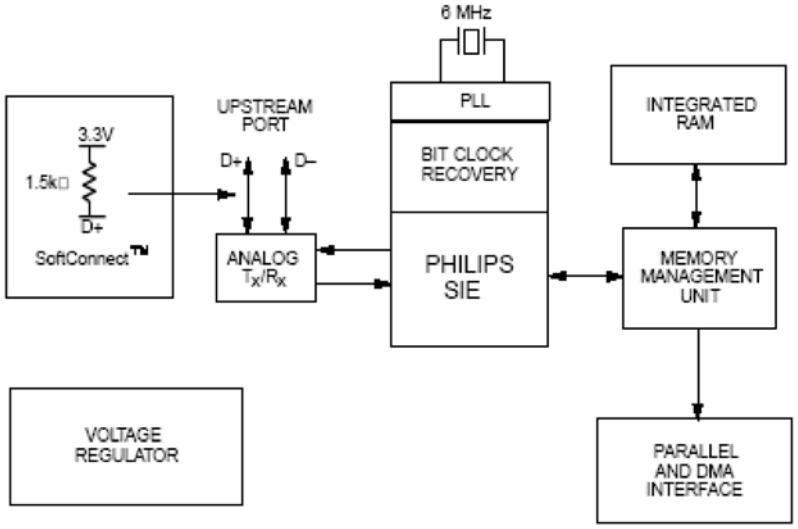

图5.1 PDIUSBD12内部结构图
Fig5.1 PDIUSBD12 inside frame

D12是一款性价比很高的USB器件。它可以用作微控制器系统中实现与微控制器进行通信的高速通用并行接口。D12与单片机连接如下图5.2所示。D12的电源电压可以采用5V或3.3V，若用3.3V电压，则VCC和VOUT同时接到3.3V。D12控制需要占用单片机的一个硬件中断（INT0或INT1）资源，用于响应D12发出的中断请

求。D12地址线A0接单片机P0.0口经锁存器74LS373后的对应输出，单片机访问D12时，当A0为0时，表示偶地址，则送往D12的是读写数据，当A0为1时，表示奇数地址，则送往D12的是写入一个命令。

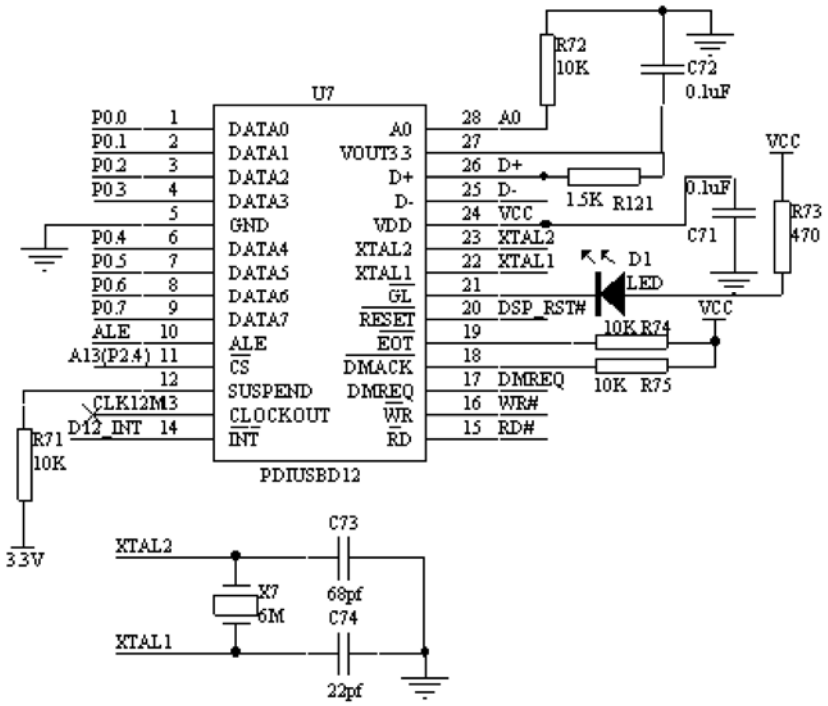

图5.2 PDIUSBD12与USB连接

Fig5.2 The connection between PDIUSBD12 and USB

5.2 USB 接口与 PC 接口

USB通过一个4线电缆来传输数据和提供电源，如图5.3所示。其中D+和D一是一对差分的信号线，而VBUS和GND则提供5V的电源。

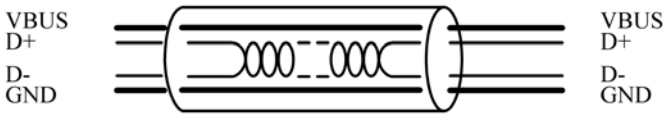

图 5.3 USB 与 PC 连接

Fig5.3 The connection between USB and PC

5.3 USB 接口程序设计[23]

USB 程序分为两个部分：USB 固件程序和 PC 机上位机程序。

USB 固件程序程序主要是对 D12 进行编程，程序在 Keil C51 环境下编写、调试，共由三部分组成：

①初始化单片机和所有的外围电路（包括 D12）；

②主循环部分，其任务是可以中断的；

③中断服务程序。

D12 和其它常见的接口芯片一样,也是通过向芯片写入控制字来操作的。D12 芯片提供了 3 个端点，每个端点都有输入和输出两个端点号，同时端点 2 还提供了 4 种方式的数据传输方法[18]。端点 0 包含的两个端点号为：端点号 0（控制输出），端点号 1（控制输入）。这是 USB 器件的两个基本端点号，主要用于与主机进行配置信息的交换和控制信息的接收，一般还可以作为厂商请求的传输。所以就不选择端点 0 来与主机进行用户的数据交换；选择端点 1 进行命令的传输和应答，而端点 2 进行数据传输。单片机的中断服务程序(ISR)就负责 USB 数据的传输，其流程图如图 5.4 所示。在 ISR 的入口调用 D12_ReadInterruptRegiste( ) 来判断中断源或者可以说 USB 事件标志类型（其定义如下），然后将进入相应的子程序进行处理。后台 ISR 中断服务程序和前台主程序循环之间的数据交换通过事件标志和数据缓冲区来实现，例如 D12 的批量输出端点可使用循环的数据缓冲区。当 D12 从 USB 收到一个数据包，那么就对 CPU 产生一个中断请求，CPU 立即响应中断，在 ISR 中将数据包从 PDIUSBD12 内部缓冲区移到循环数据缓冲区并在随后清零 PDIUSBD12 的内部缓冲区，以使能接收新的数据包。

USB 事件标志定义如下：

```
typedef union _epp_flags
{
    struct _flags
    {
    unsigned char timer                : 1;      //时间溢出
    unsigned char bus_reset            : 1;      //总线复位标志
    unsigned char suspend              : 1;      //挂起改变标志
    unsigned char setup_packet         : 1;      //收到 setup 包
    unsigned char remote_wakeup        : 1;      //远程唤醒标志（未使用）
    unsigned char in_isr               : 1;      //USB 中断服务标志
    unsigned char control_state        : 2;          //2 位，控制端点处理状态
                                                  //0：IDEL    空闲状态
                                                  //1：TRANSMIT 数据发送状态
                                                  //2：RECEIVE    数据接收状态
    unsigned char configuration        : 1;   //配置标志（0：未配置；1：已经配置）
    unsigned char command              : 1;          //未使用
```

```
    unsigned char ep1_rxdone        : 1;        //端点 1 收到数据标志
    unsigned char ep2_rxdone        : 1;        //端点 2 收到数据标志
    unsigned char ep1buf_full       : 1;        //端点 1 输出双缓冲区满标志
    unsigned char ep2buf_full       : 1;        //端点 2 输出双缓冲区满标志
    } bits;
    unsigned short value;
} EPPFLAGS;
```

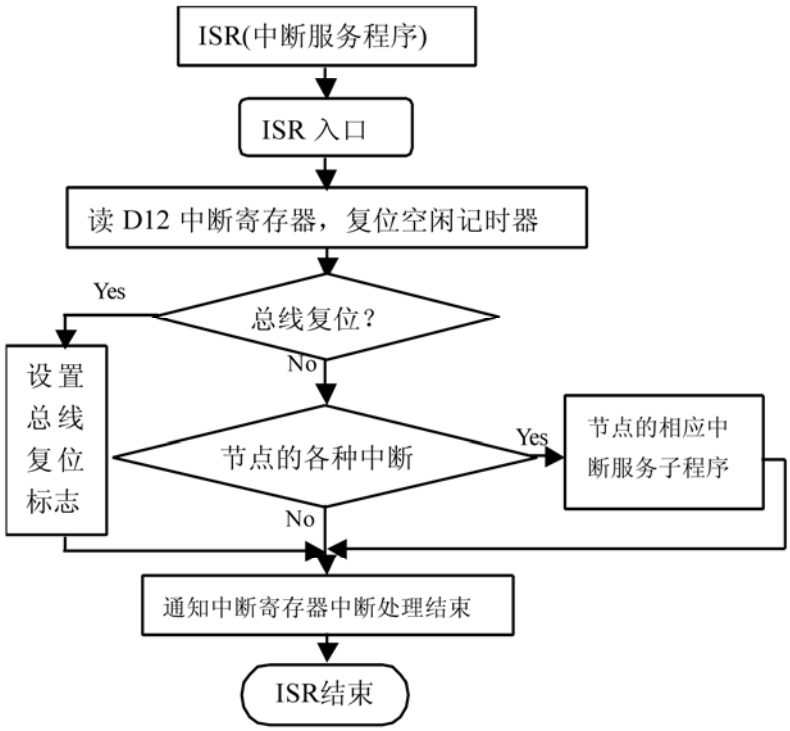

图 5.4 USB 中断服务程序流程图

Fig5.4 USB Interrupt flow chart

根据 USB 协议，任何传输都是由主机（Host）（这里是 PC 机）开始的。单片机作它的前台工作，等待中断。主机首先要发令牌包给 USB 设备（这里是 PDIUSBD12）， D12 接收到令牌包后就给单片机发中断。单片机进入中断服务程序，首先读 PDIUSBD12 的中断寄存器，判断 USB 令牌包的类型，然后执行相应的操作。

PC 机上位机程序用 C++Builder 6 编写，为快速完成这部分程序的开发，我们利用 PHILIPHS 公司提供的动态连接库，该库文件提供了 4 个函数给用户程序调用:

```
DWORD __stdcall ReadPort1 (BYTE* pData, size_t iLen);     //读 USB 端点 1
DWORD __stdcall WritePort1 (BYTE* pData, size_t iLen);    //写 USB 端点 1
```

DWORD __stdcall ReadPort2 (BYTE* pData, size_t iLen);　　//读 USB 端点 2
DWORD __stdcall WritePort2 (BYTE* pData, size_t iLen);　　//写 USB 端点 2
要注意的是端点 1 的读写字节长度不能超过 4 个字节，因此端点 1 我们作为发送接收命令字，端点 2 作为传送数据。

当上位机写端点时，D12 产生中断给单片机，单片机执行中断服务程序，并使 bEPPflags.bits.in_isr＝1，读 D12 的中断寄存器，并置位端点收到数据标志，主程序判断端点收到数据标志，若收到端点 1 数据，则首先判断接收是否正确，效对异或校验，若正确则判断命令字类型，根据命令字类型，单片机对 CF 卡进行相应的操作。命令字类型包括以下命令字：

判断 CF 卡是否准备好命令字；

读 CF 卡目录命令字；

写 CF 卡目录命令字；

读文件命令字；

写文件命令字。

6 DSC21 平台实现 JPEG 编解码

本章首先介绍了图像压缩的基本原理以及分类，同时对 JPEG 编解码原理进行了阐述。然后详细论述了在基于 DSC21 的数码相机平台上实现的方法。

6.1 图像压缩的原理

图像数据的压缩基于两点[24]:

(1) 图像信息存在着很大的冗余度，数据之间存在着相关性，如相邻像素之间色彩的相关性等。

(2)人眼是图像信息的接收端。因此，可利用人的视觉对于边缘急剧变化不敏感(视觉掩盖效应)，以及人眼对图像的亮度信息敏感、对颜色分辨率弱的特点实现高压缩比，而解压缩后的图像信号仍有着满意的主观质量。从信号系统的角度理解，数据的压缩就是对原来信号进行某种变换。借助这种变换，信号的表达更经济，存储传输更为方便。从信息论角度理解，信号本身的具体表达形式不过是其内在携带信息的外在表象，一定的信息可以用各种形式加以体现，每种表达形式的表达效率并不相同，存在着信息冗余。数据压缩的目的就是寻找在一定约束条件下最为高效的信息表达方式。从压缩技术的角度理解，数据压缩一般分为:建模、去相关、量化、编码四道工序。

由此发展出数据压缩的两类基本方法：无损压缩和有损压缩。

无损压缩是将相同的或相似的数据或数据特征归类，使用较少的数据量描述原始数据，达到减少数据量的目的。无损压缩又称信息保持编码，或叫做熵保持编码[25]。图像的无损压缩通常分为两步，即去相关和编码。去相关就是要去除图像冗余，降低信源熵[26]。常用的无损压缩算法有霍夫曼（Huffman）算法和 LZW（Lempel-Ziv & Welch）。

有损压缩是指使用压缩后的数据进行重构，重构后的数据与原来的数据完全不同，但不影响人对原始资料表达的信息造成误解[27]。

6.2 JPEG 编解码原理[27][28]

JPEG 全名为 Joint Photographic Experts Group（联合摄影专家组），它是一个在国际标准组织(ISO)下从事静态图像压缩标准制定的委员会。由它制定出了的 JPEG 标准是第一套国标静态图像压缩标准。JPEG 标准用于连续色调、多级灰度、彩色 / 单色静态图像压缩，由于 JPEG 优良的品质，使得它在短短的几年内就获得极大的成功。它在压缩过程中的失真程度很小，能适用于任何种类的连续色调的

图像，且长宽比不受限制，同时也不受限于景物内容、图像的复杂程度和统计特性等。

JPEG 压缩是有损压缩，它利用了人的视觉特性，使用量化和无损压缩相结合去掉视觉的冗余信息和数据本身的冗余信息。JPEG 算法框图如图 6.1 所示。

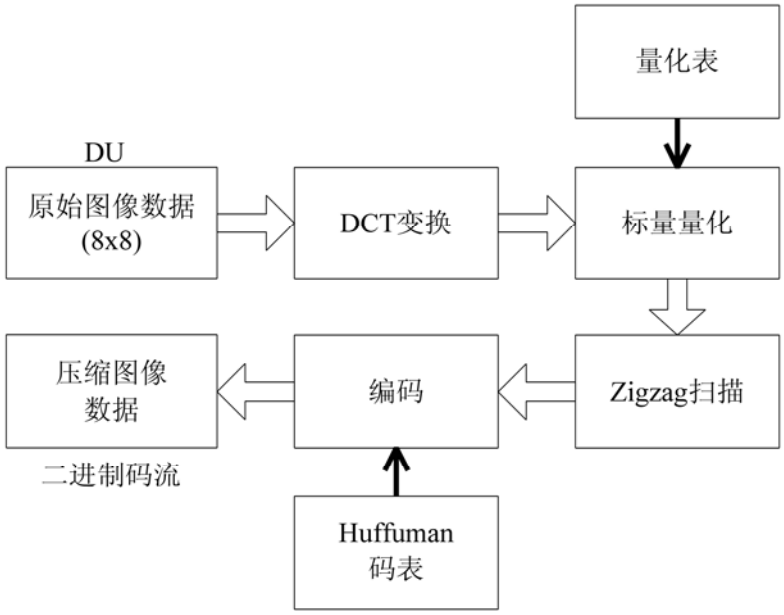

图 6.1 JPEG 编码示意图(编码器)

Fig6.1 JPEG coding frame

压缩编码大致以下分为三个步骤：

• 使用正向离散余弦变换（DCT）把空间域表示的图变换成频率域表示的图。

• 使用加权函数对 DCT 系数进行量化，这个加权函数对于人的视觉效果是最佳的。

• 使用霍夫曼可变字长编码器对量化系数进行编码。

译码或者叫做解压缩的过程与压缩编码过程正好相反。

6.3 JPEG 算法步骤

JPEG 算法主要计算步骤如下：

• 正向离散余弦变换（DCT）。

• 量化（quantization）。

• Z 字形编码（Zigzag Scan）。

• 使用差分脉冲编码调制（DPCM）对直流系数（DC）进行编码。

• 使用行程长度编码（run-length encoding，RLE）对交流系数（AC）进行编码。

• 熵编码（entropy coding）。

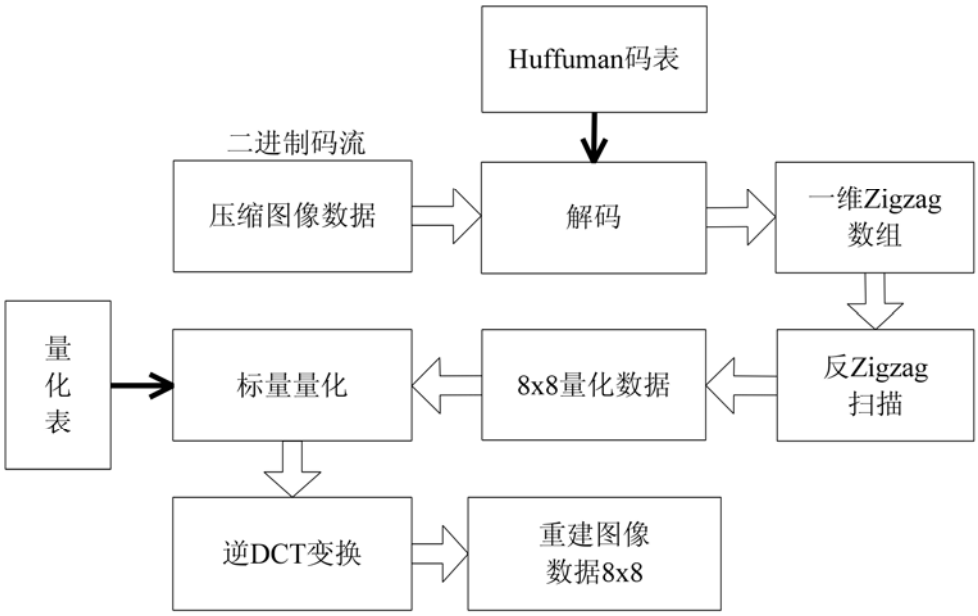

图 6.2　JPEG 解码示意图（解码器）

Fig6.2　JPEG decoding frame

6.3.1 DCT 和 IDCT 算法

离散余弦变换(Discrete cosine Transform)简称 DCT[33]。DCT 是先将整体图像分成 N×N 像素的块，然后对 N×N 像素的块逐一进行 DCT 变换。根据 JPEG 标准，取 N=8。

设原始图像数据矩阵为：$s=[S_{XY}]_{8\times8}$，X、Y 分别为原始数据域内的水平像素号和垂直像素号；

变换后的矩阵为：$S=[S_{UV}]_{8\times8}$，　U、V 分别为变换域内的水平像素号和垂直像素号。

那么，正变换公式如下：

$$S_{UV}=\frac{1}{4}CuCv\sum_{x=0}^{7}\sum_{y=0}^{7}S_{XY}COS\left(\frac{(2x+1)u\pi}{16}\right)COS\left(\frac{(2y+1)v\pi}{16}\right) \qquad (6.1)$$

式中，$Cu,Cv=\begin{cases}1/\sqrt{2} & u,v=0\\ 1 & u,v\neq0\end{cases}$

逆变换公式如下：

$$S_{XY}=\frac{1}{4}CxCy\sum_{u=0}^{7}\sum_{v=0}^{7}S_{UV}COS\left(\frac{(2u+1)x\pi}{16}\right)COS\left(\frac{(2v+1)y\pi}{16}\right) \qquad (6.2)$$

式中，Cx,Cy=$\begin{cases} 1/\sqrt{2} & x,y=0 \\ 1 & x,y\neq 0 \end{cases}$

考察式（6.1）和式（6.2）可以证明，如果定义矩阵：$A=\left(a_{ij}\right)_{8\times 8}$，$a_{ij}$ 由下式求得：

$$a_{ij}=\frac{1}{2}C_i COS\left(\frac{(2j+1)i\pi}{16}\right) \tag{6.3}$$

式中，$C_i=\begin{cases} 1/\sqrt{2} & i=0 \\ 1 & i\neq 0 \end{cases}$

用 A^T 表示矩阵 A 的转置，那么，

由式（6.1）可以得到 DCT 正变换的矩阵表示：

$$S=AsA^T \tag{6.4}$$

由式（6.2）可以得到 DCT 逆变换的矩阵表示：

$$s=A^T SA \tag{6.5}$$

对于 8×8 的二维数据块，经 DCT 变换后得到的 64 个系数具有明确的物理意义。例如，当 U=V=0 时，S_{00} 是原始图像 64 个样本值的平均，相当于 DC 分量，随着 U、V 值增加，相应系数分别代表逐步增加的水平空间频率分量和垂直空间频率分量的大小。如果删除一些离 DC 分量较远的值（即高频信息），就会损失一定量的信息。但是，在大部分情况下人眼能推知或感觉不到丢失的信息。

由式（6.4）（6.5）可以看到 DCT 算法是对称的，即是说：经由 DCT 变换压缩的图像，可以利用逆 DCT 变换进行解压缩。还要注意，正反 DCT 变换主要涉及到矩阵的乘法，运算特点为乘积运算和累加运算，考虑到 DSP 处理器有专门的乘法处理器和累加器，式（6.4）和式（6.5）将成为解决正反 DCT 变换编程实现的主要依据。

由式（6.3）可知，A 是一个常数矩阵，其组成元素为纯小数。为了提高程序的执行效率，保证数据的精度，可以求出的该常数矩阵，并用 Q15 格式表示。编程实现时，只需引用该常量表即可。我们已经计算出了该矩阵，并用 Q15 格式表示如下：

至此，我们已经解决了图像压缩所必须的数学处理问题。以下将重点讨论 JPEG 编码问题。

$$A=\begin{bmatrix}11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585\\ 16096 & 13623 & 9102 & 3196 & -3196 & -9102 & -13623 & -16096\\ 15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137\\ 13623 & -3196 & -16069 & -9102 & 9102 & 16069 & 3196 & -13623\\ 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585\\ 9102 & -16069 & 3196 & 13623 & -13623 & -3196 & 16069 & -9102\\ 6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270\\ 3196 & -9102 & 13623 & -16069 & 16069 & -13623 & 9102 & -3196\end{bmatrix}$$

6.3.2 量化和反量化

所谓量化，就是一种通过降低整数精度而减少存储整数所需的位数，进而达到压缩目的的过程。量化的目的是减小非“0”系数的幅度以及增加“0”值系数的数目。量化是图像质量下降的主要原因。

对于有损压缩算法，JPEG 算法使用均匀量化器进行量化，量化步距示按照系数所在的位置和每种颜色分量的色调值来确定的。因为人眼对亮度信号比对色差信号敏感，因此使用了两种量化表：如表 6.1 所示的亮度量化值和表 6.2 所示的色差量化值。此外，由于人眼对低频分量的图像比对高频分量的图像更敏感，因此图中的左上角的量化步距要比右下角的量化步距小。

表 6.1 亮度量化值

Table 6.1 luminance quantization

$$\begin{bmatrix}17 & 18 & 24 & 47 & 99 & 99 & 99 & 99\\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99\\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99\\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99\\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99\\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99\\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99\\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99\end{bmatrix}$$

表 6.2 色度量化值

Table6.2 Chroma quantization table

$$\begin{bmatrix}16 & 11 & 10 & 16 & 24 & 40 & 51 & 61\\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55\\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56\\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62\\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77\\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92\\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101\\ 72 & 82 & 95 & 98 & 112 & 100 & 103 & 99\end{bmatrix}$$

6.3.3 Zigzag 扫描

经加权量化后的 DCT 系数矩阵构成一个稀疏矩阵：除了左上角的 DC 分量外，其它只有少数不为零。为了提高编码的效率，量化后的 DCT 系数需要重新编排，目的是为了增加连续的“0”系数的个数，就是“0”的游程长度。JPEG 规定从左上角开始按 Z 字形扫描（Zigzag Scan）方式将二维量化系数阵列重组为一个一维

数组 zig_outdata，使其尽量匹配量化系数的能量分布，基本上能按能量递减的方式排序。处理后，第一个为 DC 系数，其他为 AC 系数。Zigzag 扫描按如下顺序访问 8x8 数据矩阵：

$$\begin{bmatrix} 1 & 2 & 6 & 7 & 15 & 16 & 28 & 29 \\ 3 & 5 & 8 & 14 & 17 & 27 & 30 & 43 \\ 4 & 9 & 13 & 18 & 26 & 31 & 42 & 44 \\ 10 & 12 & 19 & 25 & 32 & 41 & 45 & 54 \\ 11 & 20 & 24 & 33 & 40 & 46 & 53 & 55 \\ 21 & 23 & 34 & 39 & 47 & 52 & 56 & 61 \\ 22 & 35 & 38 & 48 & 51 & 57 & 60 & 62 \\ 36 & 37 & 49 & 50 & 58 & 59 & 63 & 64 \end{bmatrix}$$

表 6.3 Zigzag 扫描矩阵

Table6.3 Zigzag scan matrix

6.3.4 编码

8×8 图像块经过 DCT 变换之后得到的直流系数有两个特点：一是系数的数值比较大，二是相邻 8×8 图像块的 DC 系数值变化不大。根据这个特点，JPEG 算法使用了差分脉冲调制编码（DPCM）技术，对相邻图像块之间量化 DC 系数的差值（Delta）进行编码，

$$\mathrm{Delta} = \mathrm{DC}(0,0)_{k} - \mathrm{DC}(0,0)_{k-1} \qquad (6.6)$$

编码是对量化后的 DCT 系数进行无损处理。JPEG 分别对 Y 分量、Cr 和 Cb 分量进行编码,并且对量化后矩阵中的 64 个元素也采用不同的编码方式，其中，DC 系数最重要，编码时采用差分的方法进行单独编码；其余 63 个 AC 系数则采用相同的编码方法。

DC 系数的编码算法：DC 编码是“前缀码（Huffman 码）+尾码（长度为 SSSS）”组合而成的二进制码。由于 DC 系数反映的是图像块的平均亮度，相临块的亮度比较接近，所以，编码时只对相临数据块 DC 系数的差值（DIFF）进行编码，并规定初始电平为 0。这样编码相当于只对第一个数据块的 DC 系数编码。解码时，则可以根据差值逐次求得各数据块的 DC 值。不同幅度差值对应于不同的 Huffman 码和不同长度的尾码，JPEG 中定义了相应的尾码长度赋值表，如表格 6.4 所示。编码时，根据 DIFF 的值，查得尾码对应的长度 SSSS 和相应的 Huffman 码，再由 SSSS 值确定尾码，具体方法是：当 DIFF≥0 时，直接截取 DIFF 的（自然二进制）低 SSSS 位作为尾码，显然，此时的尾码最高位是 1；当 DIFF<0 时，取 DIFF 的反码（二进制）的低 SSSS 位作为尾码，此时尾码的最高位是 0。可见，解码时，只需根据尾码高位的取值确定 DIFF 的正负。由于 Huffman 码具有唯一性，编解码可以形成单一映射，因此，编码是无损的。

表 6.4 原始图像分量为 8 位精度时的 DC 系数 Huffman 码表

Table6.4 Huffman code table of DC coefficient

SSSS	DIFF 的值	亮度码长	亮度码字	色度码长	色度码字
0	0	2	00	2	00
1	-1，1	3	010	2	01
2	-3，-2，2，3	3	011	2	10
3	-7，…，-4，4，…，7	3	100	3	110
4	-15，…，-8，8，…，15	3	101	4	1110
5	-31，…，-16，16，…31	3	110	5	11110
6	-63，…，-32，32，…，63	4	1110	6	111110
7	-127，…，-64，64，…，127	5	11110	7	1111110
8	-255，…，-128，128，…，255	6	111110	8	11111110
9	-511，…，-256，256，…，511	7	1111110	9	111111110
10	-1023，…，-512，512，…，1023	8	11111110	10	1111111110
11	-2047，…，-1024，1024，…，2047	9	111111110	11	11111111110

AC 系数的编码：编码的基本思想与 DC 编码是大体相同的，只是在编码时首先对 Zigzag 扫描后得到一维数组进行扫描，记录两个非零系数之间的零的个数（nnnn），即游程。接着，再由非零系数的幅度值，查 AC 系数尾码位数赋值表（见表 6.5）得到尾码位数 SSSS。然后，根据游程和尾码位数查 AC 系数 Huffman 码表得到编码的前缀。尾码部分的处理和 DC 系数编码一样。可见，AC 编码同样是无损的。要强调的是，为了提高编码效率，在编码时，当游程超过 16 时，先对游程进行单独编码（即游程编码），直到游程小于 16 后，再进行前述编码过程。如果，从某一非零系数开始，此后再无非零值，那么直接以 EOB 码结束本块的编码，开始下一块的编码。

表 6.5 AC 系数尾码位数赋值表

Table 6.5 AC code table of AC coefficient

SSSS	AC 系数幅度值
0	0
1	-1，1
2	-3，-2，2，3
3	-7，…，-4，4，…，7
4	-15，…，-8，8，…，15
5	-31，…，-16，16，…31
6	-63，…，-32，32，…，63
7	-127，…，-64，64，…，127
8	-255，…，-128，128，…，255
9	-511，…，-256，256，…，511
10	-1023，…，-512，512，…，1023

为了提高查表效率，在定义AC系数Huffman码表时（编码解码表的定义见附录1），其查表算法为：10*nnnn+ssss。并规定nnnn=15且ssss=0时，下标取为161，即数组中的末位存放游程Huffman码。合理定义Huffman码表，提高查表速度，是此次实践的一个创新点。

6.4 在 DSC21 数码相机平台上实现 JPEG 编解码

6.4.1 JPEG 编码

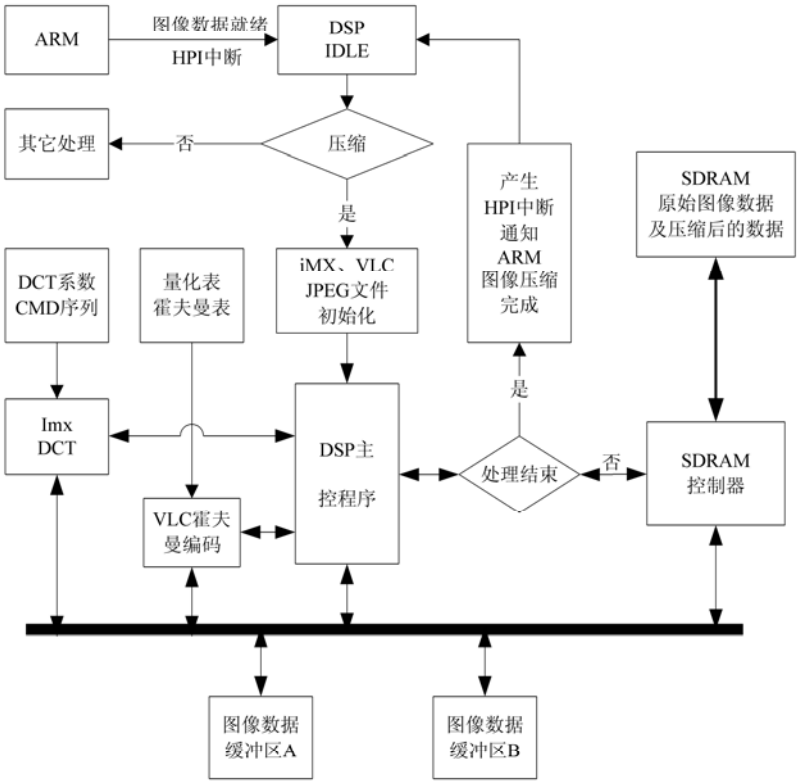

图6.3 编码流程图

Fig 6.3 JPEG code flow

正如本文VLC编程章节所述，VLC的基本功能包括量化、“之”字扫描、8位精度的基于DCT变换的JPEG编码[15]，在DSC21数码相机平台上JPEG编码都是通过VLC硬件实现的。

VLC的状态和控制寄存器都映射到DSP存储器空间。其基地址为0x0B440，VLC寄存器占有0x0B440-0x0B47F共64字大小，在DSC21数码相机平台上JPEG编码流程图如图6.3。

6.4.2 JPEG 解码

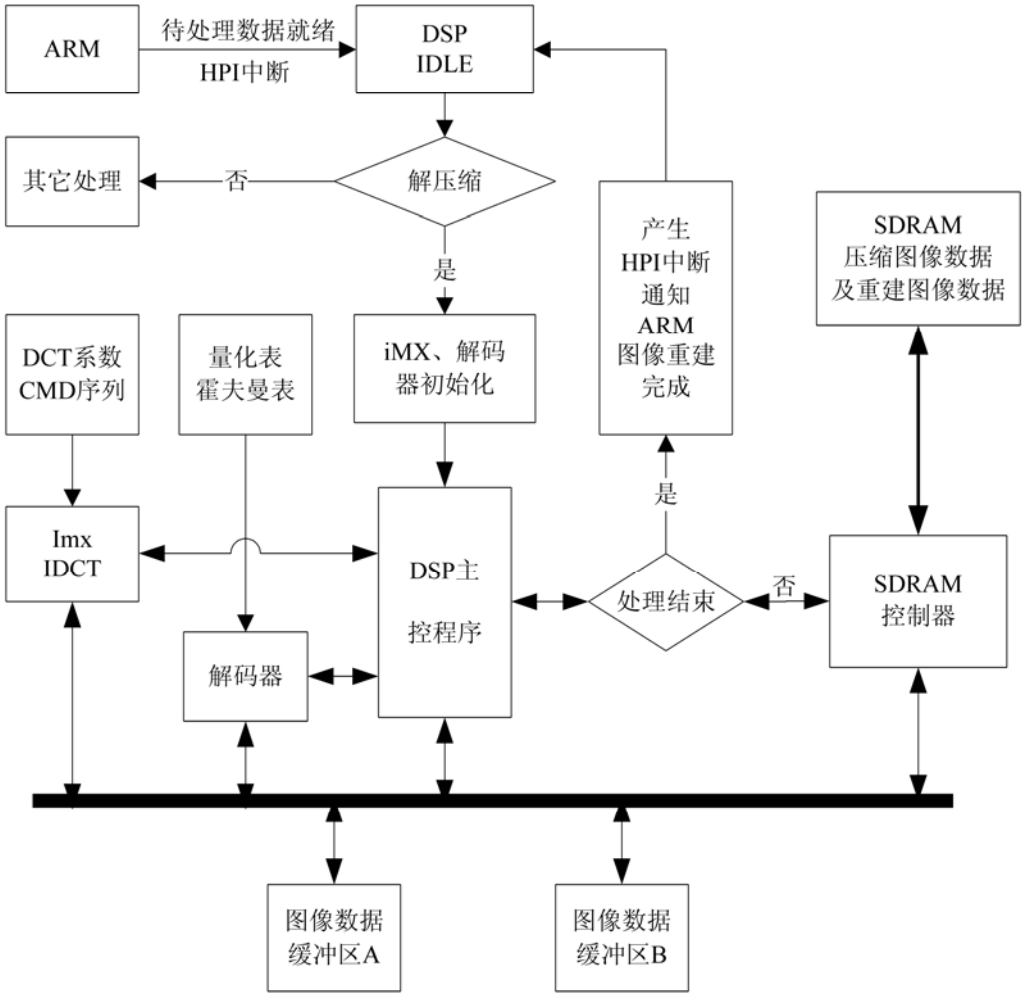

图6.4 JPEG解码流程图

Fig 6.4 JPEG decode flow

DSC21平台上，JPEG解码都是由DSP软件编程实现的。解码流程如图6.4。解码的主要目标是根据已知的码流，还原出原始的被编码的数据。JPEG压缩图像(Compressed Image Data)首先要经过熵解码器(Entropy Decoder)进行Huffman解码，结果送入反量化器(Dequantizer)中，此过程之前需要先得到存储于JPEG图像中的量化表和Huffman表；接着，进行逆向离散余弦变换(IDCT)，至此，就得到了一个8X8的块，依次处理得到每一个块，将各块拼接即可得到整幅图像。其实，解码器基本上是编码器的一个逆向操作。

解码过程需要频繁的访问Huffman码表，如何编制查表算法是提高程序执行效

率的关键。我们在编码的过程中发现，Huffman码的长度越短，出现的概率相对越高。有鉴于此，在设计解码算法时，按Huffman码的长短顺序作为访问Huffman码的主要依据，总是按从短到长的顺序访问Huffman码表。实验证明，这种处理有效的提高了查表效率。单次解码平均次数不超过10次。

6.5 结论

在DSC21平台上为了验证JPEG编解码正确性，捕捉一帧图像后，进行JPEG编码，编码后再进行解码，解码后数据存放到SDRAM中，使能OSD，则合成视频输出通过显示器可以观察到解码后的图像与原图（图6.5）看不出差别。另外，为了验证JPEG编码正确性，捕捉一帧图像后，在DSP中进行JPEG编码，编码后图像数据存放到SDRAM，并把图像数据写成JPEG文件，再把该文件从SDRAM转移到CF卡，并通过USB接口传送大批PC机，在PC机中通过编码后的图像（图6.6）和原图像比较，验证了编码的正确。

图6.5 拍照的原图像
Fig 6.5 Inward image

图6.6 JPEG编码后图像
Fig 6.6 Image after JPEG coding

7 结论

7.1 主要工作

本论文研究了基于 DSC21 数码相机整机的相关技术，其主要工作有以下几个方面：

① 基于 DSC21 的数码相机硬件研究以及软件设计，整个系统实现了数码相机的照相功能。

② 对数码相机的一种存储介质—CF 卡及 CF 卡的文件系统 FAT 的阐述，最终在 DSC21 的数码相机平台上和基于单片机的平台上实现了对 CF 卡上文件的读写。

③ 基于单片机的 USB 接口的开发，这部分是为了实现 PC 机能通过该接口存储卡 CF 卡上读取文件。为在 DSC21 平台上进行 CF 卡文件系统开发打下了基础。

④ 在 DSC21 数码相机平台上对图像进行 JPEG 编解码，在该系统中通过原始图像编码再进行解码后存储到 SDRAM，则通过 OSD 以及视频解码器输出合成视频可以观察到该图像与原图像看不出来有差别，并且作者通过了对原图像和编码后的图像比较验证，证实了编码的正确性。

7.2 工作展望

在基于 DSC21 的数码相机系统中，本文只讨论和实现了数码相机的一部分功能，在该系统中还可以实现 MP3 文件播放功能，让数码相机兼有 MP3 播放器功能，这是未来数码相机多功能化的发展发向。

为了使照片具有更好的质量，DSP 中，图像还可以进行其它处理，如：白平衡、Gamma 纠正等。另外，由于 CMOS 传感器在分辨率、动态范围和成像质量不如 CCD 好。所以可以通过更换 CMOS 传感器，使用 CCD 传感器做图像采集，可提高照出的图像质量。

致　谢

首先，衷心感谢我的导师罗钧副教授。导师严谨求实的治学作风、平易近人的态度、孜孜不倦的工作精神，为我树立了良好榜样，给我留下了极其深刻的印象，导师严格的要求、悉心的指导和富有启发性的建议，为该课题和论文的完成起到了关键性的作用。

感谢黄尚廉院士、陈伟民教授和重庆市光电工程中心对本课题的关怀、支持和方向把握，同时感谢重庆市信息产业局提供该课题项目。

感谢我的师弟师妹们，谢谢他们对我的关心，谢谢他们对此课题给予我的帮助和支持。

深深地感谢我的父母、丈夫和家人，是他们为我的学习提供了良好的保障，并不断给予我关怀和鼓励，使我得以顺利完成学业。

感谢所有帮助、关心和支持过我的人！

付丽

二零零五年五月二十五日

参考文献

[1] 怀石工作室. 数码相机完全手册[M], 北京：中国电力出版社,2000:1

[2] 俞宙，刘煜. 用好数码相机[M],北京：中国水利水电出版社,2001.10:11-12

[3] 张雄伟，曹铁勇. DSP芯片的原理与开发应用[M]. 第二版。北京：电子工业出版社。2000。

[4] 赵文伯,刘俊刚. CMOS 图像传感器发展现状半导体光电 No. 1 1999

[5] Eric R.Fossum,CMOS Image Sensor: Eletronic Camera-On-Chip.IEEE Transactions on Eletron Devices,Vo1.44 ,No.10,1997.10

[6] Texas Instruments, DSC Phase 2 Architecture， 2000.6, Pages:5, 17-28

[7] Texas Instruments, DSC Phase 2 ARM Peripherals User Manual，Jan.2001

[8] Texas Instruments, TMS320C54x C Source Debugger User's Guide, Literature Number: SPRU099C,Apr.1998

[9] 戴明桢，周建江。TMS430C54X DSP 结构、原理及应用[M]。北京：北京航空航天大学出版社。2004。

[10] Texas Instruments, TMS320DSC21 Register Manual Rev2.0.1, Oct.1999.

[11] OmniVision, OV2610 Color UXGA CameraChip Implementation Guide. Sep.2003

[12] Texas Instruments, DSC Phase 2 DSP Programmers Guide, Dec.1999

[13] Texas Instruments, DSC21 DSP and ARM Communications, Feb.1999

[14] Texas Instruments, DSC Phase 2 iMX API Specification，Apr. 2000.

[15] Texas Instruments,DSC Phase 2 VLC (Variable Length Coding Accelerator).Mar.2000

[16] TMS320C54X DSP Programmer's Guide, Literature Number: SPRU538 July 2001

[17] Texas Instruments, TMS320C54x Optimizing C Compiler User's Guide, October 1998.

[18] 郑红，吴冠。TMS430C54X DSP应用系统设计[M]。北京：北京航空航天大学出版社。2002 :123

[19] CompactFlash Association, CF+ and CompactFlash Specification Revision 1.4. Jul.1999

[20] 张锐昕. 计算机文件系统及应用[M],吉林大学出版社.

[21] Microsoft Corporation ,Microsoft Extensible Firmware Initiative FAT32 File System Specification,2000

[22] Philips Semiconductors. USB interface device with parallel bus PDIUSBD12 Data Sheet.1999

[23] Philips Semiconductors. Firmware Programming Guide for PDIUSBD12，23 September 1998

[24] 冯勇。数码相机图像压缩的研究（硕士论文），哈尔滨工业大学，2003.7.1：1-3

[25] 刘玮，王红星。图像的无损压缩编码方法及 JPEG 标准模式.现代电子技术 2002, 5:7-10

[26] C. E. SHANNON. A Mathematical Theory of Communication. Bell System Technical Journal.

1948,27:379-423

[27] 林福宗。多媒体技术基础[M]，清华大学出版社，2002，12：45，76-80

[28] JPEG File Interchange Format Version 1.02,September 1,1992

[29] Texas Instruments, TMS320VC5409 FIXED-POINT DIGITAL SIGNAL PROCESSOR ，SPRS082A – APRIL 1999 – REVISED JUNE 1999

[30] ISSI，IS42S16400 1 Meg Bits x 16 Bits x 4 Banks (64-MBIT) SYNCHRONOUS DYNAMIC RAM，MAY 2001

[31] SST, SST39VF800A datasheet,2003

[32] Dolle, M.; Jhand, S.; Lehner, W.; Muller, O.; Schlett, M.A 32-b RISC/DSP microprocessor with reduced complexity。Solid-State Circuits, IEEE Journal ofVolume 32, Issue 7, July 1997 Page(s):1056 - 1066

[33] K. Okamoto et al., A DSP for DCT-based and wavelet-based video codecs for consumer applications. IEEE J. Solid-State Circuits, Mar. 1997.vol. 32, Page(s): 460–467.

[34] Texas Instruments. TLV320AIC23 StereoAudio CODEC data manual. Jul.2001

[35] Y. Qin, R. Zhou. Y. Yang. etc. Design of Enibcdded CF Card Controllcr in Digital Camera System. Journal of Fudan University. Vol. 42. No.1. Feb 2003. 114-117

[36] Redmond, S.M. DSP algorithms to optimised ASICs-an automated route. , IEE Colloquium on 1993, Page(s):2/1 - 2/4

[37] 丁启芬. 数字相机实用技术白问[M], 北京:人民邮电出版社, 1999

附 录：攻读硕士学位期间发表的论文

[1] 罗钧，付丽，基于 DSP 的 MP3 解码系统设计，重庆大学学报，2005 年 1 期

[2] 罗钧，刘京诚，付丽，廖红华，创新实验 培养创新人才，重庆大学学报，2004 年增

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。据我所知，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，也不包含为获得 重庆大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：[signature]　　签字日期：2005 年 6月 3日

学位论文版权使用授权书

本学位论文作者完全了解 重庆大学 有关保留、使用学位论文的规定，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。本人授权 重庆大学 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。

保密（　），在____年解密后适用本授权书。

本学位论文属于

不保密（√）。

（请只在上述一个括号内打“√”）

学位论文作者签名：[signature]　　导师签名：[signature]

签字日期：2005 年 6月 3日　　签字日期：2005年 6月 3日