基于 LPC2292 的手持 JPEG 图像显示器设计

摘要：文章介绍了基于 32 位 ARM7 芯片 LPC2292 实现的软件 JPEG 解码算法及应用了此算法的手持 JPEG 图像显示器的设计。文章从 JPEG 的编码原理入手，针对 LPC2292 的资源和性能特点，进行了可行性分析。重点论述了 Huffman 解码和 IDCT 解码的优化算法的实现，很大程度上提高了解码算法的效率，使得算法在 ARM7 芯片上的移植成为可能。
关键字:JPEG 解码;Huffman 解码;IDCT 解码

1 概述

在数字图像处理过程中，JPEG 编码是一种经常运用的压缩算法，目的是有效地压缩数字图像大小，便于存储和传输。其具有压缩比例高、失真小等特点。目前对应的 JPEG 解码方式有两种：硬件解码和软件解码。硬件解码是利用硬件解码芯片来完成对 JPEG 数据的解码；软件解码则是通过编写程序来完成对数据的还原。由于软件解码需要大量的浮点运算，目前较多的由 PC 及 DSP 来实现。

本文主要介绍了一种基于 Philips 公司的 ARM 芯片 LPC2292 来实现 JPEG 图像软件解码的方法及基于此算法实现的手持式 JPEG 图像显示器。

2 JPEG 解码可行性分析

2.1 LPC2292 的特点

LPC2292 是一款基于 32 位 ARM7TDMI-S，并支持实时仿真和跟踪的 CPU，带有 256 k 字节(kB)嵌入的高速 Flash 存储器。128 位宽度的存储器接口和独特的加速结构使 32 位代码能够在最大时钟速率下运行。对代码规模有严格控制的应用可使用 16 位 Thumb 模式将代码 规模降低超过 30%，而性能的损失却很小。

2.2 存储器容量计算

JPEG 解码涉及到大量的数据计算，因此需要大量的数据缓存器。手持 JPEG 显示器的缓存主要包括原始 JPEG 图像数据缓存、JPEG 解码缓存以及 RGB 数据显示缓存等。原始 JPEG 图像数据缓存用于存放需要解码的 JPEG 图像数据，根据 JPEG 图像的复杂度及压缩比不同，一般一帧 320×240 的彩色 JPEG 图像的大小在 2k～20k 字节。JPEG 解码缓存主要用于存放一些 JPEG 解码过程中的中间数据，主要包括 MCU 解码中 Huffman 解码的临时结果，IDCT 解码的临时结果等。RGB 数据显示缓存用来存放 JPEG 解码完后的 RGB 原始数据，用于 LCD 液晶显示，一帧 320×240 的彩色 16 位 RGB 图像，共需要 320×240×16÷8＝57600 字节。

LPC2292 内部带有 16k 的 SRAM。如果采用整张图像缓存、解码完再显示的话，根据上面的分析，仅显示缓存区一项就不能满足需要。因此，设计采用一边读取数据、一边解码、一边显示的策略，来减少对存储器的需求。

3 具体实现

3.1 硬件实现

JPEG 图像显示器的硬件实现框图如下图所示：

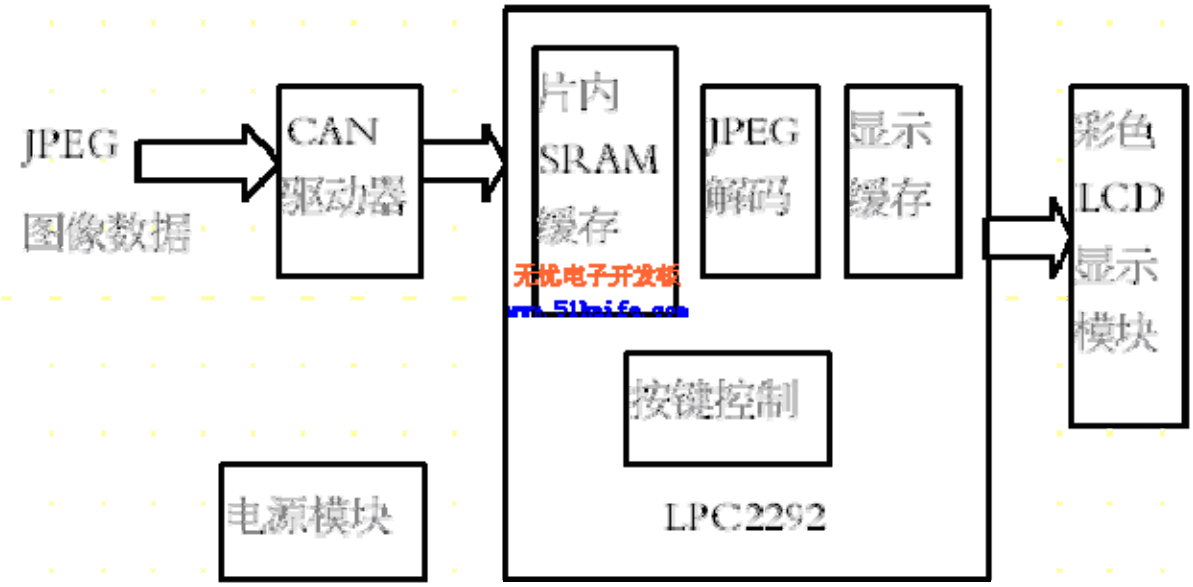

图 1　JPEG 图像显示器硬件框图

系统以 LPC2292 为控制核心，采用 CAN 总线来从外界读取原始 JPEG 数据，彩色 LCD 显示模块则选用 TFT6758 彩色 LCD 液晶显示模块。LPC2292 内部自带的 CAN 接口及 32 位宽度的数据总线，可以很方便的与 CAN 总线及 LCD 液晶显示模块实现连接。

3.2 软件实现：

JPEG 编码技术主要包括颜色转换、DCT 变换、量化、熵编码及 Huffman 编码等部分，其数据格式在文献[1]中进行了详尽的阐述。

JPEG 解码主要包括 Huffman 解码、反量化及 IDCT 变换、色彩变换等模块，其流程框图如下：

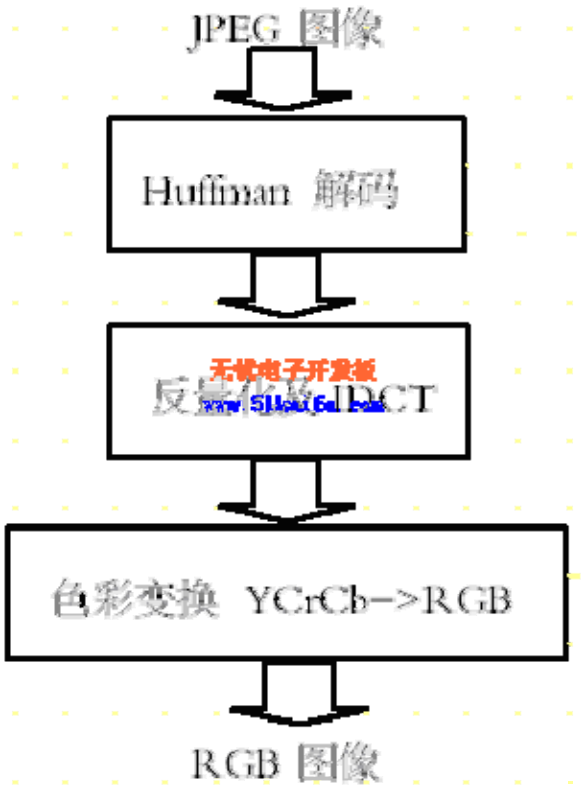

图 2　JPEG 图像解码软件流程框图

3.2.1 Hffman 解码的实现：

Huffman 解码主要通过反向查找 Huffman 表来实现。由于查找过程需要大量的移位及循环，效率很低。为了提高效率，软件事先构造了一张预查找表，该表的维数由宏定义 HUFF_LOOKAHEAD 来决定。位数低于 HUFF_LOOKAHEAD 的 Huffman 编码，解码结果都可以从该预查找表中直接读出，而对于位数大于 HUFF_LOOKAHEAD 的则再通过移位循环来遍历 Huffman 表实现解码。可以看出，预查找表维数越高，则整个 Huffman 解码所需的循环移位次数越少，解码效率越高，但同时，需要存放预查找表的空间也越高，其占用字节数为 2 的 HUFF_LOOKAHEAD 次方字节。

预查表分两部分，其构造思想如下：

假设某个亮度的 DC 系数的 Huffman 编码表如下，HUFF_LOOKAHEAD 的值为 4：

表 1　亮度 DC Huffman 编码表

真实值	码长	编码字
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

则构造 2 张维数为 24 的表，一张中的数据为码长，另一张为真实值。

表中第一行码字 00 代表的是真实数据 0，则构造的码长表中，序号为 00XX2（其中 X 表示 0 或 1，即 00XX2 为 0－310）的数据全为 2，真实值表中，序号为 00XX2（0－310）的数据全为 0；

表中第二行码字 010 代表的是真实数据 1，同理码长表中，序号为 010X2（4－510）的数据全为 3，真实值表中，序号为 010X2（4－510）的数据全为 1；

依次类推，直到填满，序号为 11112（1510）的码字在 Huffman 表中没有出现，则在 2 张预查表中，最后一个数据均为 0，构造的预查表如下：

码长表

{2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 0 }

真实值表

{0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 0}

预查找解码时，收到一串 2 进制数据，我们先将前 4 个 bit（HUFF_LOOKAHEAD = 4）的数据读入移位缓存，假设为 01012（510），则我们在 2 张预查表中分别读取序号为 5 的数据（第 6 个数据），得到码长为 3，真实值为 1，因此解码得到第一个真实数据为 1，丢弃缓存中前 3 个 bit，从接收到的数据流中重新移入 3 个 bit 的数据，与之前的最后一位 1 组成一个新的 4bit 数据，进行下一次预查找解码。

3.2.2 反量化及 IDCT 变换的实现

在进行 IDCT 变换前，先要将数据进行反量化及重排列处理。将 8×8 块中的数据对应的乘上量化表中的系数，再将 64 个数据进行 Z 字型的重新排列，用于后面的 IDCT 变换。

IDCT 可以用下面的计算公式来实现：

其中 x，y＝0...7　当 u＝v＝0 时，C(u)=C(v)=0.5，其他情况 C(u)=C(v)=1

由于 ARM7 芯片在数据处理方面性能远不及 DSP，如果简单按照这个公式来计算 IDCT 变换，则会因为运算量太大而严重影响解码速度。因此，文献[2]中提到了 IDCT 变换的另一种实现算法。该算法的思想是将二维 8×8 数组的 IDCT 转换为行和列的 2 个一维数组的 IDCT，进而再转换成一系列简单的加减法及与常数相乘的乘法运算，大大减小了运算量。

3.2.3 色彩转换的实现：

经过 IDCT 变换后，JPEG 数据被解码成 YCrCb 信号。图像数据要在液晶显示器上显示，还需要进行最后一步变换，即将 YCrCb 信号转换成 RGB 信号，两者对应的转换公式如下：

R = Y + 1.402 ×(Cr-128)

G = Y - 0.34414×(Cb-128) - 0.71414×(Cr-128)

B = Y + 1.772 ×(Cb-128)

其中，R、G、B、Y、Cr、Cb 都是 0~255 的无符号数据。

公式中有浮点数的乘法运算，为了提高性能，软件事先将 1.402×（0~255）、0.34414×（0~255）、0.71414×（0~255）以及 1.772×（0~255）的结果制成表格，计算时只要查表即可得到结果，免去了由于浮点运算而占用的大量处理时间。

4 结论

由于采用了 Huffman 预解码查找表，用简单的查表语句代替了 Huffman 解码过程中大量繁琐的循环和移位操作；优化的 IDCT 算法及色彩转换算法，减少了大量复杂的乘法运算；提高了解码的效率，大大压缩了解码所需的时间，使得原本复杂的解码算法在数据处理能力并不出色的 ARM7 芯片上得以很好的实现。

设计完成的手持 JPEG 图像显示器如图 3 所示，其已经在自行设计的电缆排管疏通监视机器人项目中取得了很好的效果，在 320×240 的液晶屏上显示的 JPEG 解码图像效果如图 4 所示：

图 3　手持 JPEG 图像显示器

图 4　JPEG 显示器图像显示效果

本文作者创新点：将计算量庞大的 JPEG 解码算法成功地在数据处理性能并不出众的 ARM7

系统上实现了移植，并由此设计完成了手持式彩色图像显示器。

参考文献:

[1]邢赛鹏,平西建,詹杰勇.JPEG 图像数据格式简明分析[J].微计算机信息,2005,12-3:166-168

[2]魏忠义，朱磊，基于 DSP 的 JPEG 图像解码算法的实现 [J]，陕西 西安，2005 年

[3]Information Technology Digital Compression And Coding Of Continuous-Tone Still Images Requirements And Guidelines CCITT Recommendation T.81