Application Report
SPRA664 - May 2000

JPEG for Digital Panel

Lim Hong Swee *Digital Signal Processing Solutions*

ABSTRACT

JPEG, an acronym for Joint Photographic Experts Group, is an ISO/CCITT-backed international standards group that defined an image-compression specification (also referred to as JPEG) for still images. The following are mandatory in JPEG specification: algorithms must operate at or near the state of the art in image-compression rates; algorithms must allow for software only implementations on a wide variety of computer systems; compression ratios need to be user variable; compressed images must be good to excellent in quality; compression must be generally applicable to all sorts of continuous-tone (photographic type) images and interoperability must exist between implementations from different vendors.

JPEG is intended to be used in different applications, so few absolute requirements are written into the specification to cater to different implementation. In terms of interoperability, the JPEG specification was so vague that a group of vendors established the JPEG File Interchange Format (JFIF), a defacto file format for encapsulating JPEG compressed images so that files can be shared between applications from different manufacturers.

It is also important to distinguish between JPEG's image-processing algorithms, which comprise JPEG image compression, and the file format in which JPEG images are typically stored. In this application note, I will focus on JPEG algorithms, including discrete cosine transforms, quantization, and entropy encoding. I will also discuss how the JPEG header parser is implemented.

Contents

List of Figures

1 Introduction

The JPEG specification defines four different modes of operation:

1. Sequential encoding mode, in which the image is encoded in a left-to-right, top-to-bottom fashion.

2. Progressive encoding mode, in which images are compressed on decoding that paint an image that is successively refined for each decoded scan. This is similar to interleaved GIF images.

3. Lossless encoding method, where a decoded image is guaranteed to be a bit-for-bit copy of the original image.

4. Hierarchical encoding method, where multiple copies of image are encoded together with each copy being a different resolution. This allows an application to decode an image of the specific resolution it needs without having to handle an image with too-high or too-low resolution.

In this application note, I will focus only on the baseline sequential mode of operation because implementing this mode alone is difficult enough and it is sufficient for many imaging applications.

2 JPEG Standard

JPEG achieves image compression by methodically throwing away visually insignificant image information. This information includes the high-frequency components of the image, which are less important to image content than the low-frequency components. When an image is compressed using JPEG, the discarded high-frequency components cannot be retrieved, so baseline sequential JPEG is considered lossy. When a JPEG compressed image is displayed, much of the high-frequency information is missing: lossy.

Figure 1 and Figure 2 show the JPEG encoding/decoding of an image. During encoding, an image is broken up into 8x8 pixel blocks that are processed individually, from left-to-right and top-to-bottom. Each block of image data is subjected to a forward discrete cosine transform (FDCT), which converts pixel values into their corresponding frequency components. A block of pixels is submitted to the FDCT, which returns a block of frequency components. The frequency-component values, or coefficients, are resequenced in order of increasing frequency.

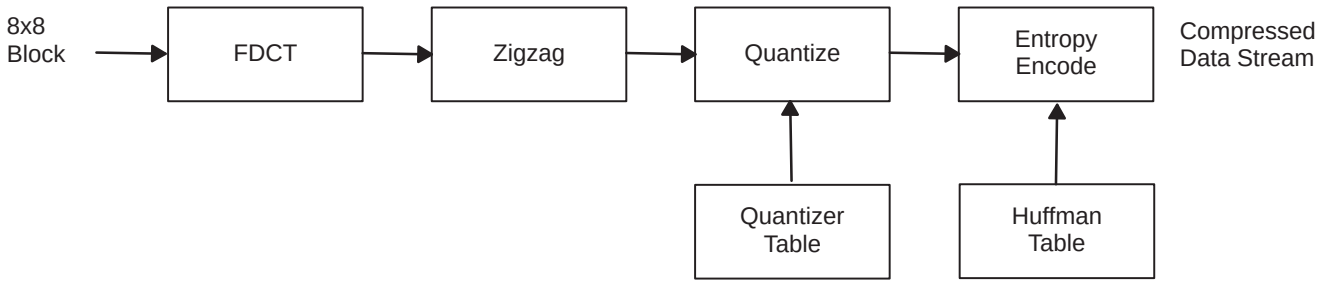

Figure 1. JPEG Encoder

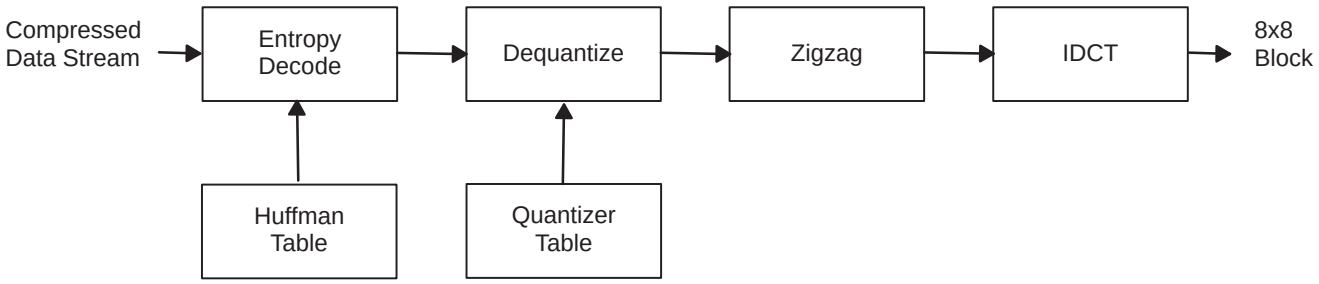

Figure 2. JPEG Decoder

The resultant frequency coefficients are then quantized, which causes components with near zero amplitudes to become zero. The level of frequency component quantization depends upon a user specified image quality factor. The more an image is compressed, the more frequency components in a block become zero.

The entropy encoding process performs two types of image compression on the block of frequency coefficients. First, it run-length encodes the number of zero coefficients in a block. Secondly, it bit encodes the coefficient values using statistically generated tables. The encoded bit stream is written to the output stream.

During decoding, the encoded bit stream is read from the input stream, and the quantized frequency coefficient for a block of pixel data are decoded. These frequency components are then dequantized by multiplying them by the quantization table with which they are produced.

The frequency coefficients are resequenced from frequency order back into pixel order. The block of frequency components is then subjected to an inverse discrete cosine transform (IDCT), which converts the frequency component values back into pixel values. The reconstructed pixel values are stored in the decoded image.

Discrete cosine transforms (DCTs) convert 2-D pixel-amplitude and spatial information into frequency content for subsequent manipulation. Basically, a DCT is used to calculate the frequency components of a given signal. The FDCT is applied to the 8x8 block of pixel data, resulting in a series of 64 frequency coefficients. In JPEG files, the first coefficient—the “DC coefficient”—is the average of all of the pixel values within the block. Coefficients 1–63—the “AC coefficients”— contain the spectrum of frequency components that make up the block of image data. As a result of the slowly changing nature of continuous tone images, many high frequency coefficients produced by the FDCT are zero or close to zero in values. This is exploited during the quantization process. Note that there is usually a strong correlation between DC coefficients of adjacent blocks of image data.

Once the block of image pixels is processed with the FDCT, the result is a frequency spectrum for the pixels. The coefficients that make up the frequency spectrum are not conveniently arranged in ascending order at the conclusion of FDCT processing. Instead, they are clustered with the DC coefficient in the upper left, surrounded by the lower frequency components. The higher frequency components are grouped towards the lower right. The application of the zigzag sequence after the FDCT converts 64 frequency coefficients into ascending order, the DC and low frequency coefficients are grouped together, followed by the high frequency coefficients.

The quantization step is where most image compression takes place and is therefore the principal source of loss in JPEG image compression. Quantization is performed by dividing each frequency coefficient by a corresponding quantization coefficient mandated by the JPEG specification. One quantization table is specified for the luminance portion of the image data, and another, for the chrominance. Quantization coefficients with values approaching one allow the corresponding frequency coefficient to pass through the quantization process unmodified. Large quantization coefficients force the corresponding frequency coefficients to approach zero in value. Thus visually insignificant, high frequency information is discarded.

The final step in the JPEG encoding process is entropy encoding. The JPEG specification allows for either arithmetic or Huffman encoding. Generally, Huffman encoding is utilized in most JPEG implementation. Huffman compression is based upon the statistical characteristics of the data to be compressed: Symbols that occur frequently in the data are assigned shorter Huffman codes; those that occur infrequently are assigned longer codes. Compression will occur as long as there is a large difference between the occurrence counts of the most common and the least common symbols. Note also that Huffman coding is bit oriented, not byte oriented. The Huffman codes assigned to the various symbols are bit packed together into the tightest possible configuration of bytes. This makes the code for Huffman encoding/decoding difficult to write and debug because the data stream has to be examined at the bit level. Convenient byte boundaries do not exit at the lowest level.

Two additional forms of data compression occur in the entropy-coding step: delta coding of the DC coefficients of adjacent blocks of image; and run-length encoding of zero-valued frequency coefficients. These ancillary compression mechanisms contribute to the overall compression achieved for JPEG-compressed images.

3 Header Parser

JPEG by itself is a compression standard and does not specify a file format. JPEG images may come in many file formats, such as JFIF (JPEG File Exchange Format), TIFF/JPEG (Tagged Image File Format), and PICT/JPEG (for Macintosh). The JPEG decoder that we have built for the Digital Panel project can only recognize JFIF in addition to being able to decode JPEG-compressed images.

3.1 High-Level Syntax

The generic JPEG/JFIF file format is shown is Figure 3.

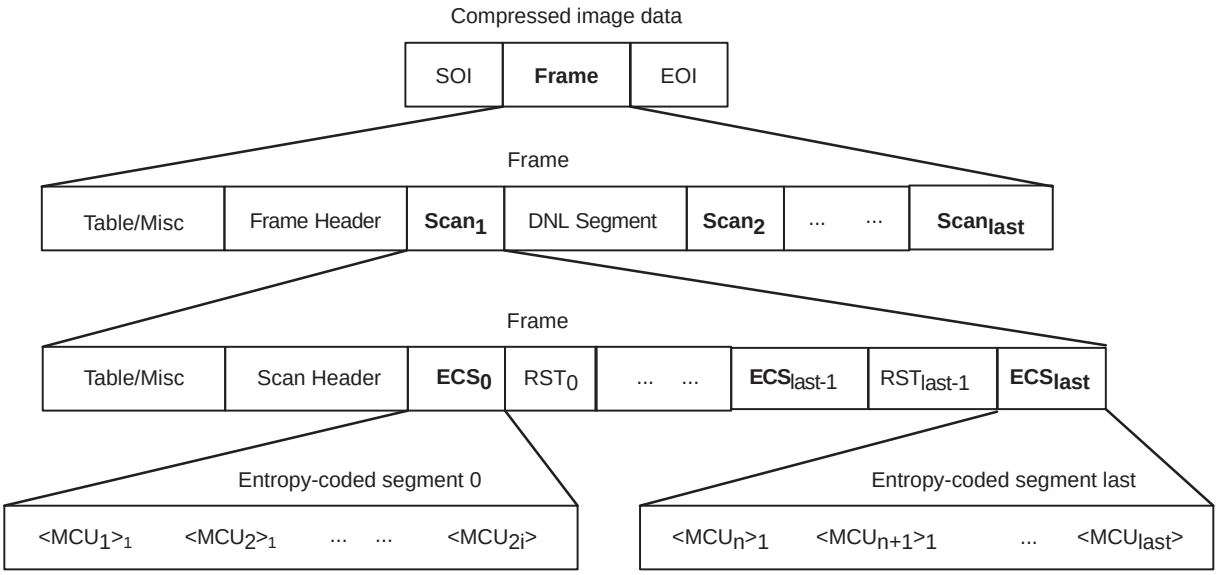

Figure 3. Syntax for JFIF

The three markers shown in Figure 3 are defined as follows:

- SOI (Start Of Image marker: 0xFFD8)– Marks the start of a compressed image represented in the interchange format or abbreviated format.
- EOI (End Of Image marker: 0xFFD9)– Marks the end of a compressed image represented in the interchange format or abbreviated format.
- RST_m (Restart Marker)– A conditional marker that is placed between entropy-coded segments only if restart is enabled. There are 8 unique restart markers (m=0–7) which repeat in sequence from 0 to 7, starting with zero for each scan, to provide a modulo 8 restart interval count.

The JPEG specification states that the JPEG file is composed of segments. A segment is a stream of bytes with length less than 65535. The segment beginning is specified with a marker 0xFF and an ending with a byte different by 0 to 0xFF. The second byte of the marker specifics what that marker does.

3.2 Frame Header Syntax

The frame header shall be present at the start of a frame.

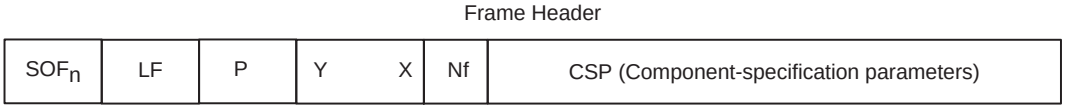

Figure 4. Frame Header Syntax

SOF0 = 0xFFC0 ; Start of Frame when it is baseline JPEG
SOF1 = 0xFFC1 ; Extended sequential DCT, not supported
SOF2 = 0xFFC2 ; Progressive DCT, not supported
SOF3 = 0xFFC3 ; Lossless, not supported
SOF5 = 0xFFC5 ; Not supported
SOF6 = 0xFFC6 ; Not supported
SOF7 = 0xFFC7 ; Not supported
SOF9 = 0xFFC9 ; Extended sequential DCT, not supported
SOF10 = 0xFFCA ; Progressive DCT, not supported
SOF11 = 0xFFCB ; Lossless, not supported
SOF13 = 0xFFCC ; Differential sequential DCT, not supported
SOF14 = 0xFFCD ; Differential progressive DCT, not supported
SOF15 = 0xFFCE ; Differential lossless, not supported
Lf = Frame header length given by 8 + Nf * 3 (high byte, low byte)
P = Data precision (1 byte)
Y = Image height (2 bytes)
X = Image width (2 bytes)
Nf = Number of image components in frame (1 byte)
CSP = For each components (3 bytes)
- Component ID (1=Y, 2=Cb, 3=Cr, 4=I, 5=Q)
- Sampling factors (bits 0–3: vert., bits 4–7: hor.)
- Quantization table number

3.3 Scan Header Syntax

The scan header shall be present at the start of a scan.

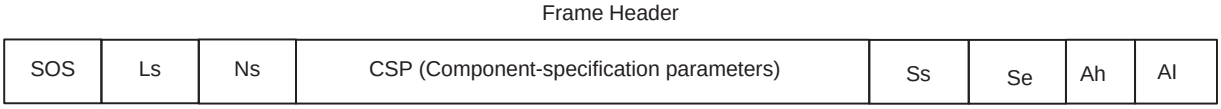

Figure 5. Scan Header Syntax

SOS = 0xFFDA ; Start of Scan
Ls = Scan header length given by 6 + Ns * 2 (high byte, low byte)
Ns = Number of image components in scan (1 byte)
CSP = For each components (3 bytes)
- Component ID (1=Y, 2=Cb, 3=Cr, 4=I, 5=Q)
- Huffman table to use (bits 0–3: AC table, bits 4–7: DC table.)
- Quantization table number

Ss = Start of spectral selection (1 byte); ignore
Se = End of spectral selection (1 byte) ; ignore
Ah = Successive approximation bit position high (1/2 byte) ; ignore
Al = Successive approximation bit position low (1/2 byte) ; ignore

3.4 Table Specification and Miscellaneous Marker Segment Syntax

Any of the table-specification segments or miscellaneous marker segments may be present in any order and with no limit on the number of segments.

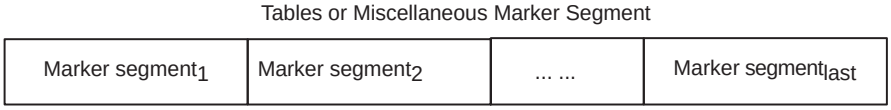

Figure 6. Miscellaneous Marker Segment Syntax

Marker segment

```
DHT    =    0xFFC4    ; Define Huffman Table
DAC    =    0xFFCC    ; Define Arithmetic Table, usually unsupported
DQT    =    0xFFDB    ; Define Quantization Table
DNL    =    0xFFDC    ; Usually unsupported
DRI    =    0xFFDD    ; Define Restart Interval
DHP    =    0xFFDE    ; Define hiearchical progressive, not support
EXP    =    0xFFDF    ; Ignore
APP0   =    0xFFE0    ; JFIF segment marker
APP15  =    0xFFEF    ; Ignore
JPG0   =    0xFFF0    ; Reserved for JPEG extension
JPG13  =    0xFFFD    ; Reserved for JPEG extension
JPG    =    0xFFC8    ; Reserved for JPEG extensions
COM    =    0xFFFE    ; Comment
*RST0  =    0xFFD0    ; RSTn are used for resync, ignored in our implementation
*RST1  =    0xFFD1
*RST2  =    0xFFD2
*RST3  =    0xFFD3
*RST4  =    0xFFD4
*RST5  =    0xFFD5
*RST6  =    0xFFD6
*RST7  =    0xFFD7
```

A JFIF-standard file will start with the four bytes (hex) FF D8 FF E0, followed by two variable bytes (often hex 00 10), followed by ’JFIF’. For detail information for the rest of the Miscellaneous marker, please refer to T.81 specification.

4 Conclusion

The image data following the header is stored in sequential, Huffman-encoded, minimum coded unit (MCU) blocks. Each MCU of the image is compressed separately and padded to a byte boundary with bits. Six blocks of image data make up an MCU for true-color. The number of blocks per MCU is contained in the file header. In order to fulfill the customer’s requirement and be able to extract the above information, a generic JPEG decoder was successfully built using code from the Digital Still Camera (DSC) group and implementing aheader parser on the C54x platform.

5 References

1. CCITT T.81, Information Technology– Digital Compression and Coding of Continuous-tone Still Images– Requirements and Guidelines.

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.