(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 September 2002 (26.09.2002)

PCT

(10) International Publication Number
WO 02/075640 A2

(51) International Patent Classification⁷: G06K 9/36, G06F 15/16, 17/00, H03M 7/00

(21) International Application Number: PCT/US02/08783

(22) International Filing Date: 19 March 2002 (19.03.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/277,609 19 March 2001 (19.03.2001) US
60/365,203 15 March 2002 (15.03.2002) US

(71) Applicant *(for all designated States except US)*: **SOUNDPIX, INC.** [US/US]; 689 Bridger Court, Incline Village, NV 89451 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **OSBORNE, John** [US/US]; 793 McCourry Boulevard, Incline Village, NV 89451 (US). **RUSSELL, David, W.** [US/US]; 575 Sugarpine Drive, Incline Village, NV 89450 (US).

(74) Agents: **TONG, Rolando, J.** et al.; Ian F. Burns & Associates, P.O. Box 20038, Reno, NV 89515-0038 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Published:**
— *without international search report and to be republished upon receipt of that report*

WO 02/075640 A2

# (54) Title: SYSTEM AND METHOD OF STORING DATA IN JPEG FILES

(57) **Abstract:** The present invention comprises a system for communicating image data and other data. The system comprises a JPEG file having a JPEG data, at least one non-JPEG data, and a computer. The computer is programmed to read a predetermined number of sequential bytes of the non-JPEG data. The computer is also programmed to determine position of each byte in the sequential bytes that contain a JPEG marker and determine a locator byte, which is capable of indicating the position of any bytes in the sequential bytes that contain a JPEG marker. The computer is further programmed to write the locator byte in the JPEG file and write encoded bytes in the JPEG file. The computer writes the encoded bytes in the JPEG file in a manner wherein, for each byte in the sequential bytes, if the byte does not contain a JPEG marker, each byte is replaced with a bit, preferably written as 0, in the order of the byte's occurrence in the sequential bytes. If the byte does contain a JPEG marker, the byte containing a JPEG marker is replaced with a place holder bit, preferably written as 1 and in the order of the byte's occurrence in the sequential bytes.

# SYSTEM AND METHOD OF STORING DATA IN JPEG FILES

## [01] CROSS REFERENCES TO RELATED APPLICATIONS

This application claims the benefit of and incorporates by reference, U.S. provisional application number ________, filed on March 15, 2002, and U.S. Provisional application number 60/277,609, filed March 19, 2001.

## [02] BACKGROUND OF THE INVENTION

**[03] Field of Invention**

**[04]** The present invention relates to a system and method for storing data in JPEG files.

**[05] Description of Related Art**

**[06]** JPEG Files

**[07]** JPEG is a file format standard used for compressing, storing, and transmitting digital image data. As used herein, the term "JPEG" refers to all versions, revisions, and releases of the format, including "JPEG" and "JPEG 2000." The format was developed by the Joint Photographic Experts Group and it is currently the most popular and widely used image format. One of JPEG's advantages is that it provides the greatest compression of any bitmap format in common use, allowing devices and users to store image files using the leastamount of storage space. This compression capability is also one of the features that have made JPEG the most commonly used format for sending pictures over the Internet. Users can transmit and receive images in JPEG more quickly with a higher degree of image quality than in many other formats.

**[08]** The JPEG format uses markers to break files into segments. Markers comprise a first byte having the value 0xFF. JPEG markers can be grouped into two general types; stand-alone and data. A stand-alone maker has no data other than the two bytes of the marker. A data marker is immediately followed by two bytes that indicate the number of bytes of data that follow the marker. For example, a data marker may appear as follows:

**[09]** 0xFF 0xNN 0xCH 0xCL

**[10]** In this example, 0xFF indicates the beginning of a JPEG segment or field, 0xNN denotes the type of field, and 0xCH and 0xCL together indicate the length of the field. The marker may be followed by data or instructions.

**[11]** In addition to data for the subject image, JPEG allows non-JPEG data or application data to be stored within a JPEG file. The application data may comprise data for applications other than an application that uses image data. For example, data may be inserted into a JPEG file that indicates the setting of the camera that captured the image or the identity of the artist who created the image. In this way, one of the advantages of the JPEG format is that data other than the subject image data can be stored and transmitted in the JPEG file itself.

**[12]** Embedding random data into a JPEG file offers many practical uses. For example, still images or pictures may be transmitted over the Internet with an accompanying audio file, which may be a person's voice describing the picture. This allows families to share pictures with a family member's voice, realtors to present pictures of a property with a voice description of the property, medical professionals to present diagnostic images with a verbal description or diagnosis, and many other uses.

**[13]** However, one of the problems associated with storing application data in a JPEG file is the application data cannot contain marker codes. In some instances, decoders can read 0xFF in application data as a JPEG marker and cause irrecoverable failure of the decoding process. If application data in a JPEG file contains a marker code, the marker may cause the JPEG decoder to crash and the file may not be useable.

**[14]** One solution to this problem is to remove all instances of 0xFF or other marker code from the application data. This would allow the data to be stored and transported in a JPEG file without the danger of crashing the JPEG decoder. However, if one or more marker codes are removed from the application data, the data is usually significantly altered and it cannot be used by its intended application program.

**[15]** Therefore, a need exists for a system and method for removing marker codes from application data while the data is stored in a JPEG file and for returning the data to its original state for use with an application program.

**[16]** Association of Audio and Image Files

**[17]** Techniques exist for associating random data with JPEG files. For example, users wishing to create digital images with sound can attach a separate audio, e.g., .wav, file to their image file. However, .wav files break down often in that they are easily corrupted or detached during transmission and storage. Another problem with .wav files is that they require special software to be saved and used, such as Microsoft® Media software.

**[18]** Other techniques for associating random data with JPEG files include techniques disclosed in U.S. patent numbers 5,032,918, issued to Ota et al., and 5,815,201, issued to Hashimoto et al., wherein audio and image files are recorded separately and associated with each other. One of the disadvantages with the invention disclosed in Ota et al. is the necessity of maintaining two separate files. This can be very burdensome and vulnerable to loss of association, especially when high volumes of files are involved or when files are archived for long periods of time.

**[19]** U.S. patent number 6,102,505 issued to McIntyre et al. creates permanent associations between image and audio data by using invisible ink. U.S. patent number 5,644,557 issued to Akamine et al. and European Patent Application number EP 0 964 304 A1 issued to Nozaki et al. disclose magnetic recording on a film stock and recording data as optical pixels on a film stock, respectively. One of the problems associated with Akamine et al. and Nozaki et al. is the requirement of a physical film stock for storing a hard copy of the audio data, which is not well suited for digital image data.

Other storage techniques for audio and image data include the invention disclosed in U.S. patent number 5,276,866, issued to Paolini, whereinaudio data is stored as image frames in a video stream, which is then compressed. U.S. patent number 5,440,677, issued to Case et al., discloses storing audio files and image files on CD-ROM with additional files to enable associated playback. The problem with Paolini and Case et al., as well as with the above patents, is they require custom programs for both sending and receiving the data before any of the data can be retrieved, displayed, or played.

**[20]** In sum, a need exists for eliminating file maintenance requirements inherent in associating application data with image data. A need also exists for creating permanent associations with application data and image data in a manner that allows efficient and convenient transmission through a variety of mediums, such as the Internet. A further need exists for efficiently and conveniently combining application data and image data into a single file. A need further exists for storing application data with image data without requiring custom or non-widely used applications for storing, retrieving, displaying, or using application and image data. The present invention provides a system and method that fulfills these needs by approaching the problem in a way that is not suggested by the prior art.

## [21] SUMMARY OF INVENTION

### [22] Advantages of the Invention

**[23]** An advantage of the present invention is that it allows reliable storage of application data with image data.

**[24]** Another advantage of the present invention is that it allows application data that may conflict with image decoders to be stored with image data.

**[25]** Another advantage of the present invention is that it eliminates maintenance requirements inherent in associating application data files with image data files.

[26] Another advantage of the present invention is that it provides a system and method where image data and application data are combined into a single file in a manner that allows application programs to detect the existence of the application.

[27] Another advantage of the present invention is that it prevents irrecoverable failure of JPEG decoders when the decoders read application data that contain 0xFF data.

[28] Another advantage of the present invention is that it removes instances of marker codes while at the same time providing data containing order, security, and identification data to enable reliable reconstruction of the original application data.

[29] Another advantage of the present invention is that it provides a system and method for storing application data with image data without requiring custom or non-widely used applications for storing, retrieving, displaying, or using application and image data.

[30] Another advantage of the present invention is that it provides a decoding mechanism wherein application data may be reconstructed.

[31] Another advantage of the present invention is that it provides an encoding mechanism that inserts additional data records at the beginning of each JPEG application record.

[32] Another advantage of the present invention is that it allows multiple non-application data to be permanently associated with at least one image file.

[33] Another advantage of the present invention is that it encodes original data without losing data.

[34] Another advantage of the present invention is that it preserves original data, including additional information, such as playback parameters, language identification, multiple track information, timing, and the like, during encoding of original data.

[35] Another advantage of the present invention is that allows computers to access image data even if the receiving system ignores application data.

[36] Another advantage of the present invention is that it provides digital images with sound.

[37] Another advantage of the present invention is that it provides a means for storing application in an image file in a manner that provides security against modified application data.

[38] Another advantage of the present invention is that it provides a marker in an image file that allows reliable reconstruction of application data.

[39] Another advantage of the present invention is that it provides a means for storing data in an image file that is larger than a maximum field length for the image file.

[40] Another advantage of the present invention is that it provides a means for storing data in an image file usable by playback applications.

[41] These and other advantages of the present invention may be realized by reference to the remaining portions of the specification, claims, and abstract.

## [42] **Brief Description of the Invention**

[43] The present invention comprises a system for communicating image data and other data. The system comprises a JPEG file having a JPEG data, at least one non-JPEG data, and a computer. The computer is programmed to read a predetermined number of sequential bytes of the non-JPEG data. The computer is also programmed to determine position of each byte in the sequential bytes that contain a JPEG marker and determine a locator byte, which is capable of indicating the position of any bytes in the sequential bytes that contain a JPEG marker. The computer is further programmed to write the locator byte in the JPEG fileand write encoded bytes in the JPEG file. The computer writes the encoded bytes in the JPEG file in a manner wherein, for each byte in the sequential bytes, if the byte does not contain a JPEG marker, each byte is replaced with a bit, preferably written as 0, in the order of the byte's

occurrence in the sequential bytes. If the byte does contain a JPEG marker, the byte containing a JPEG marker is replaced with a place holder bit, preferably written as 1 and in the order of the byte's occurrence in the sequential bytes.

**[44]** The above description sets forth, rather broadly, the more important features of the present invention so that the detailed description of the preferred embodiment that follows may be better understood and contributions of the present invention to the art may be better appreciated. There are, of course, additional features of the invention that will be described below and will form the subject matter of claims. In this respect, before explaining at least one preferred embodiment of the invention in detail, it is to be understood that the invention is not limited in its application to the details of the construction and to the arrangement of the components set forth in the following description or as illustrated in the drawings. The invention is capable of other embodiments and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein are for the purpose of description and should not be regarded as limiting.

## [45] BRIEF DESCRIPTION OF THE DRAWINGS

**[46]** Figure 1 is substantially a schematic diagram of the system and method of the present invention.

**[47]** Figure 2 is substantially a schematic diagram showing creation of a JPEG file containing image and application data.

**[48]** Figure 3 is substantially an example showing a portion of the encoding process of the present invention.

**[49]** Figure 4 is substantially an example showing a portion of the encoding process of the present invention.

**[50]** Figure 5 is substantially a flow chart showing 0xFF detection process of the present invention.

[51] Figure 6 is substantially an example of a JPEG file in hexadecimal format.

[52] Figure 7 is substantially an example of a JPEG file with encoded application data according to the present invention.

[53] Figure 8 is substantially an example of a JPEG file with multiple encoded application segments according to the present invention.

[54] Figure 9 is substantially a portion of the decoding process of the present invention.

## [55] DESCRIPTION OF THE PREFERRED EMBODIMENT

[56] In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings, which form a part of this application. The drawings show, by way of illustration, specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made with out departing from the scope of the present invention.

[57] The present invention comprises a system and method for storing application data with image data, generally indicated by reference number 10. The method generally comprises gathering image and application data 22, creating a single a single JPEG file 24, and presenting and/or using the image data and application data 26.

### [58] Gathering input data

[59] Input data comprises image data 100 and application data 104. In one use of the present invention, application data 104 comprises audio files. Other application data may include, text, word processing documents, email messages, spreadsheets, personnel files, fingerprints, etc. The application data may be in any digital format known in the art.

[60] In the example shown on the top of figure 1, the device used to gather the image data and application data is a digital camera 102. Image file input 100 is preferably stored in

JPEG compression format. Camera 102 comprises equipment for recording sounds in digital data, such as a microphone, an analog-to-digital converter, processor, and memory.

[61] Once image file 100 and audio file inputs 104 are obtained, input files are converted to a single JPEG file 32.

## [62] Application Data Encoding

[63] As shown in figure 2, combining image file 100 and application data 104 to a single JPEG file 32, encoding the application data 28, parsing the data 29, if necessary, combining the encoded application data with the image data 31 to form a single JPEG file with embedded encoded application data 32. The step of encoding the application data 28 generally comprises determining the location of any bytes in the un-encoded application data that contain a 0xFF byte, determining at least one locator byte that indicates the location of any 0xFF bytes, and removing the 0xFF bytes from the data.

[64] Referring to figure 3, the encoding process of the present invention begins by evaluating a predetermined number of sequential bytes of the data. In the preferred embodiment, seven bytes 112 of sequential data are evaluated from the application data file 104. A locator byte 115 is added to seven sequential bytes 112. To simplify decoding, the locator byte precedes the seven sequential bytes in file storage. Locator byte 115 is used to indicate the position within the seven sequential bytes before encoding that comprise a 0xFF byte. Locator byte 115 and the seven sequential bytes are combined to form an eight byte packet 131.

[65] Figures 4 and 5 illustrate one method by which the value of the locator byte 115 may be determined. In this method, a mask value 116 is assigned to each byte position in the seven sequential bytes. Most significant bit 113 of mask value 116 is prefaably zero to ensure that locator byte 115 does not become 0xFF. The algorithm first tests the first byte to determine if it comprises a 0xFF byte 120. If it does not contain this byte value, the test

result is false and the pointer is shifted to the next byte position 122. In the example shown in figures 3 and 4, the first byte is 0x01. Therefore, the test is false and the pointer is moved to the second byte and the result is 0000 0000. Since the second byte is 0x22, the pointer is shifted to the third byte and the result remains 0000 0000.

[66] This test and shift process is repeated until either a byte tests true or all seven bytes have been tested 128. In the example show in figures 3 and 4, the pointer would be shifted to the fifth position before the test result is true. When the test is true, the algorithm sets a flag bit to 1 126 in a position indicated by mask 116. Therefore, the result 117 is 0000 0100. At the sixth byte position, the test is false and the result does not change. At the seventh byte position, the test is true, and the result becomes 0000 0101. This result is used as the value of the locator byte 115, which, in this example, corresponds to 0x05 in hexadecimal format. If none of the seven sequential bytes contained a 0xFF byte, the value of the locator byte would be 0x00.

[67] Referring to figure 3, once the value of locator byte 115 is determined, the locator byte is combined with the sequential bytes 112 of application data. However, any bytes that comprise 0xFF are converted to 0x00, a predetermined place holder byte 128. Any bytes that do not comprise a 0xFF, remain the same 130. Together, these eight bytes form an encoded packet of application data 131.

[68] Having removed 0xFFs and having provided a locator byte, file corruption due to presence of 0xFFs is avoided and, at the same time, integrity of original data is preserved. Of course, different numbering systems may be used or different algorithms may be used to determine the value of locator byte 115. Such modifications are within the scope of the present invention.

[69] With regards to image data 100, the image data may be encoded using the JPEG methods, such as Discrete Cosine Transform (DCT) and Huffman encoding.

**[70] Insertion of Encoded Packet into JPEG file**

[71] After encoding, encoded packet 131 is inserted in a location in the image file to create a single JPEG file. Referring to figure 6, JPEG records follow the format where the first byte is an FF (132), the second is a header type byte (134), the third (136) and fourth bytes (138) are combined to form a 16-bit length of record including length bytes. Encoded packet 131 may be inserted between existing JPEG packets 140, 144. As seen in figure 7, A new JPEG packet 143 comprising of JPEG header 132, application header type 0xEA 134, a length of ten bytes 136, 138, and eight byte encoded application data packet 131 is inserted between two existing JPEG packets 140 (see figure 6), 144.

[72] A person skilled in the art would appreciate that a mechanism must be implemented whereby the data may be reconstructed in the proper order if it exceeds a 65,535-byte segment length limit imposed by the JPEG format. A person skilled in the art would also appreciate that systems reading the application data must be able to reliably detect the existence of the data even if other applications use the same header code. This is true especially considering JPEG allows sixteen application codes to be used, but JPEG2000 allows only one.

[73] The present invention addresses these concerns by inserting additional data records at the beginning of each JPEG application record. These headers contain order, security, and identification data to enable reliable reconstruction of the original application file. Additional header information further serves to enable detection of the embedded data file, provide traceability of any modification to the original data, and provide information usable by playback applications. Additional headers may be in form of data storage structures, hereinafter referred to as blocks. Block types are preferably removed of FF codes at least after the first 12 bytes to ensure data remains compliant with JPEG specification and does not corrupt JPEG data. Designation of bytes and number of bytes reserved for each block may of

course be varied according to programming objective. Referring now to table 1, an index block is shown.

[74] **Table 1**

| **Byte** | **Description** | **Values** |
|---|---|---|
| 1-4 | JPG Header, Header Type, and Size (this is JPG Specific information) | |
| 5-12 | ASCII value of the word "SOUNDPIX" | SOUNDPIX |
| 13-14 | Number of data blocks to follow | |
| 15-16 | Version. This is the version of encoder | Byte 15 is an 8-bit major version number. Byte 16 is an 8-bit minor version number. |
| 17 | Watermark flag | 0 = none; 1 = Digimarc; 2 = Signum; 3 = SoundPix. |
| 18 | Encryption Flag | 0 = none; 1 = RSA. |
| 19 | Registered Flag | If TRUE (1), signifies that this file was created with a registered version of an Encoder Tool |
| 20-21 | Profile Information (Reserved for later use) | |
| 22-24 | Reserved for later use | |
| 25-N | Extended File Information | |

[75] Index block contains global information that is applicable for the entire file. Bytes 1-12 allow automatic detection of the inserted data. Placing the ASCII string within the record (Bytes 5-12) makes it easy for playback applications to identify an audio/image file when viewing the file in an editor. If the playback application does not find the identification record, the image is displayed normally. If the audio ID is detected, then the audio data is read from the header and reconstructed as a simple audio file. The image is displayed without removal of the audio data. Because the 0xFF's have been removed, all current decompression systems can identify the audio packets as application data and skip them without affecting the image display.

[76] Security and digital watermark may be added to the packets (Bytes 17 and 18) such

that any alteration of the combined image and data document would be flagged as an error. This feature allows use of data as an encryption key, which may be practical for creating authentic police reports, investigative work, detective work, or using photos for evidentiary purposes.

[77] Bytes 20-21 and 22-24 are optional and may be reserved to accomplish future programming objectives. Bytes 25-N represent extended information that allow specific applications, which are capable of understanding the information, to process the information. The flag byte for extended file information is preferably a 0xFE to signify beginning of extended information. Second byte is preferably a length field and does not include 0xFE or 0xFF. Extended information preferably contains length indicator to allow applications, which cannot understand information, to skip data as well as an identifier to indicate type of information to follow. Table 2 shows various extended file information type identifiers. All identifiers are preferably preceded by the 0xFE extended information tag and a length indicator.

[78] **Table 2**

| ID | Description | Parameters |
|---|---|---|
| 01 | Audio Language Formats | 1 byte following the ID, giving the count of language formats present. |
| 02 | Text Language Formats | 1 byte following the ID, giving the count of language formats present. |
| 03 | Audio Language Format | 1-byte followed by the RFC-1766 Language code. This EFI Type is used to indicate all languages present in a given File. If there are multiple audio languages present, the file should contain a 03 record for each language. |
| 04 | Text Language Format | 1-byte followed by the RFC-1766 Language code. This EFI Type is used to indicate all languages present in a givenFile. If there are multiple text languages present, the file should contain a 04 record for each language. |

[79] For example, assume a multiple language file is involved. An EFI Type ID of 01 would indicate multiple languages embedded in the file. A type of 0x01 would indicate following byte describes the number of different language formats present in the file. Thus, Extended File Information area for a file with 5 languages would look like: 0xFE 0x02 0x01 0x05. Thus, using extended file information gives the file format the flexibility and extensibility to deal with future additions.

[80] Additional header further comprises header blocks. Header blocks will contain information about any file data that is to be included in the JPEG file. This data include, as examples, audio data and image data for slide shows. Data may also include word processing files, spreadsheets, and other random data. Base 14 bytes for Header Blocks are shown in table 3 below.

**[81] Table 3**

| Byte | Description | Values |
|---|---|---|
| 1-4 | JPG Header, Header Type, and Size (this is JPG specific information) | |
| 5-12 | Data Format Tag (Similar to the SOUNDPIX tag above, but related to the actual type of data stored). For audio information this will be SPXAUDIO, for image information, SPXIMAGE, SPXTRADE, and SPXTHUMB, for text, SPXTEXT, etc. | SPXIMAGE = Image data; SPXTRADE = Trademark Data; SPXTHUMB = Thumbnail Image; SPXAUDIO = Audio data; SPXTEXT = Text data; SPXEXTRA = File data. |
| 13-14 | ID. | This ID may be used to uniquely identify and distinguish audio data headers. It may also used to match header records with their associated data blocks. ID allows for reconstruction of data. |
| 15 | Reserved for later use | |
| 16-34 | Language | Language format codes as specified in RFC-1766 |
| 35 | Audio file format flag | 0 = wav;<br>1 = au;<br>2 = AIFF;<br>3 = MP3 |

| | | |
|---|---|---|
| 36-38 | Audio sample rate flag | Unsigned 24-bit sample rate in Hz. |
| 39 | Audio bits/sample | Unsigned 8-bit bits/sample |
| 40 | Display preference flag. This defines such things as looping, auto play or other effects. | 8 bit field: 0x01 = mute; 0x02 = auto play; 0x04 = on click; 0x08 = loop; 0x10 = sync (Reserved for later use). |
| 41 | Loop Count | If byte 40 is set to 08, this byte will determine how many times the audio will loop. If set to 0, the audio will loop continuously. |
| 42-43 | Start time of audio file. | Unsigned 16-bit start delay in deciseconds. Also used for time delay between replays. |
| 44-45 | Data Block Count | This field tells the number of data block that make up the included audio file. |
| 46-57 | Password | Encoded password that requires verification for the audio to be heard. |

The Data Format Tag (bytes 5-12) preferably uses an 8-byte text field to define the type of data to follow. Audio data are preferably represented as SPXAUDIO. Image data are preferably represented as SPXIMAGE, SPXTRADE, or SPXTHUMB. Text data are preferably represented as SPXTEXT. Any unsupported file information are preferably represented as SPXEXTRA and placed in a File Data block. The data format tag also allows for embedding more data types and allows playback systems to know what to do with a given data type. For example, a word document could be embedded into the file with a Data Format Tag of MSWORD®. This would allow the player of the same version or higher to recognize the MSWORD® format tag, and send the data directly to Microsoft Word.

[82] Additional header further comprises image data header shown in table 4 below.

**[83] Table 4**

| Byte | Description | |
|---|---|---|
| 13-14 | ID | The ID Field for this type, as well as the types to follow, serves to |

| | | |
|---|---|---|
| | | uniquely identify and distinguish audio data headers. It may also used to match header records with their associated data blocks. ID allows for reconstruction of data. Preferably, all Ids that are embedded start at 1. 0 will be reserved for the main JPG that contains the data. |
| 15 | Image Format | 0 = Jpeg; 1 = BMP; 2 = TIFF; 3 = GIF. |
| 16 | Image Effect. Image Effects describe different actions that the image can perform or be a part of. It also defines whether or not the image has thumbnail or trademark data associated with it. If the Image effect has the Thumbnail of Trademark bits set, the corresponding ID listed under bytes 23 and 24 for thumbnails and bytes 25 and 26 for trademarks tells where those thumbnail and trademark images may be located. Image format further defines whether the image is part of a slideshow, of should be activated based on a user interaction, such as mouse-over. No Effect must be selected for all SPXTRADE and SPXTHUMB types. | 0x00 = No Effect;<br>0x01 = Slideshow;<br>0x02 = Mouse over;<br>0x04 = Thumbnail;<br>0x08 = Trademark; |
| 17-18 | Start Time. Used with slideshows to tell when image is to be displayed | Unsigned 16-bit start delay in deciseconds. Also used for time delay between replays. Not used for SPXTRADE and SPXTHUMB types |
| 19-20 | Data Block Count | This field tells the number of data block that make up the included audio file. |
| 21-22 | Associated Image | If the Data Format Tag is set to SPXTRADE or SPXTHUMB, this value represents the ID of Image that the thumbnail or trademark is associated with. If the Data Format Tag is set to SPXIMAGE, this field is not used |
| 23-24 | Thumbnail ID | Only used if the Data Format Tag is set to SPXIMAGE and Image Effect has Thumbnail set. This is the ID of a Thumbnail associated with this image. |
| 25-26 | Trademark ID | Only used if the Data Format Tag is set to SPXIMAGE and Image Effect has Trademark set. This is |

| | | |
|---|---|---|
| | | the ID of a Trademark associated with this image. |

**[84]** Additional header further comprises File Data Header shown and described in table 5 below.

**[85] Table 5**

| Byte | Description | |
|---|---|---|
| 13-14 | ID | |
| 15 | Mime Type Length (L). Mime Type length defines the length of the Mime Type field. These values are standardized and widely accepted and allow for the most flexibility. Once we have the length, then we can extract the actual mime type and read the file data. | |
| 16 to 16+L | Mime Type | |
| 17+L to 18+L | Data Block Count | This field tells the number of data block that make up the included audio file. |
| 19+L | File Name Length (FL) | Length in bytes of file name |
| 20+L to 20+L+FL | File Name | Name of the file that this header references. |

**[86]** Additional header also comprises text data header shown and described in table 6 below.

**[87] Table 6**

| Byte | Description | |
|---|---|---|
| 13-14 | ID | |
| 15-16 | Associated Image ID. The Associated Image ID in the Text block allows assignment of text to several images that might be contained within a file. An example of this is in the case of a slide show, where multiple images will be displayed at different times. If the text is to be assigned to only the main image, then its Associated ID is preferably 0. | |
| 17-35 | Language | Language format codes as specified in RFC-1766 |
| 36-37 | Data Block Count | This field tells the number of data |

| | | |
|---|---|---|
| | | block that make up the included audio file. |
| 38 | Effect | 0x00 = No Effect;<br>0x01 = Mouse Over;<br>0x02 = Append to Image;<br>0x04 = Overlay Image;<br>0x08 = URL |
| 39-40 | Horizontal Offset for Overlay | If Overlay Image is set, this sets the horizontal starting point for the overlay. |
| 41-42 | Vertical Offset for Overlay | If Overlay Image is set, this sets the vertical starting point for the image. |

[88] Additional headers further comprises user data block shown and described in table 7 below. User Data Block allows users to store various types of information. Specific uses for this field type are envisioned to be user preference information and Meta data, such as image titles. The Description field allows developers to assign a meaningful name to the data stored within this area. The Data field preferably stores the information. All information stored within the Data field is preferably XML formatted. This allows for multiple parameters to be stored within this area and gives users added flexibility for the data being stored. For example, a user storing a title that was "Summer 2001" would have an entry in the data area that was: <Title>Summer 2001</Title>. Using this feature along with the Description field would allow a user to group related data together into one data block and keep it together.

[89] **Table 7**

| **Byte** | **Description** | |
|---|---|---|
| 1-4 | JPG Header, Header Type, and Size (this is JPG specific information) | |
| 5-12 | Data Format Tag | SPXUSER |
| 13-14 | ID. This identifier is currently used only as a consistency measure for other blocks. Currently there is no associated data block for user data. This allows for that possibility in future releases | |
| 15 | Description Length (L) | |

| | | |
|---|---|---|
| 16 to 16+L | Description | Text field describing the type of data stored |
| 17+L to N | Data | |

[90] Additional header further comprises data block. Data Block stores data described in the Header Blocks. As shown and described in table 8 below, using the ID fieli, in conjunction with the Sequence field, an application can successfully decode the data and know what action to take, based on the information provided by its associated Header Block.

**[91] Table 8**

| **Byte** | **Description** | |
|---|---|---|
| 1-4 | JPG Header, Header Type, and Size (this is JPG specific information) | |
| 5-12 | Data Format Tag | SPXDATA |
| 13-14 | ID. This ID will match a Header record so the data can be stored in the proper format. | |
| 15-16 | Sequence. The sequence in the original file that this data block falls into | |
| 17-N | Data. | |

[92] Referring now to figure 8, additional headers described in tables 1-8 are inserted into a JPEG file beginning with JPEG application marker 132. In the embodiment shown, additional header is in the form of index header 146 and data format tag 148.

[93] **Presenting files of different types as output**

[94] Once a single embedded JPEG file exists, it may be transferred to any playback system 26, preferably a device capable of at least displaying image data such as a personal computer, a personal digital assistant (PDA), a cell phone, a specialized application player such as a talking picture frame, and the like. The file may also be transferred to any camera, display device, or application computer through any electronic media such as disk, e-mail, or other telecommunications medium.

[95] During decoding, playback system 26 detects embedded data, separates it from the

JPEG image data, decode it back to original data format, and presents two files as output. Figure 9 shows decoding method of the present invention. Playback system 26 begins reading encoded data by testing locator byte 115 (144). If locator byte 115 is 0x00, then playback system 26 transfers, without changes, the seven bytes that follow 146. Playback system 26 proceeds to read the rest of data stream. If locator byte 115 is not 0x00, then playback system 26 tests each bit of the locator byte 115 (148). If thebit is 1, then corresponding byte is set to 0xFF (150). If bit is not 1, then playback system 26 proceeds to test the remaining bits until all seven bits are tested (152,154).

**[96] Applications of Data Embedded into JPEG**

[97] Listed below are just some of the practical applications for random data embedded into JPEG. When random data embedded into JPEG is an audio file, still pictures may be displayed with playable audio files on devices such as computers, PDAs, or cell phones. With still pictures and audio files combined into a single JPEG file, viewers can click on an image to hear description of what is being portrayed by the image. Audio description may be in a form of an advertisement for products from internet auction or catalogue sites, advertisement for a real estate, investigative work, family activity, research, or newsletters. Researchers or investigators can attach audio files to image files thereby allowing their colleagues to listen to their verbal observations or analyses while looking at their work. Images may be in a form of X-ray scans, ultrasound scans, image files generated from specialized devices, any other types of digital imagesknown in the art. Voice and speaker recognition software to identify the speaker may also be included with the present invention to provide a system that allows searching and categorization of images through their audio content. The system of the present invention may be configured to allow a user select between opening a stored thumbnail or stored reduced image and the original image, while also allowing the user to select between playing the original audio, or a reduced version of the

audio.

[98] When non-JPEG data comprises multiple sound files, such as multiple speeches in different languages, the present invention allows users to store and play sound in a particular language specified by either the stored profile of the user, the site providing access, or profile based on the most popular language users use. Next, when random data comprises time-related data, multiple images may be stored and played at specific times to coincide with the audio playback. Alternatively, multiple audio files may be played with specific start times. This is most practical in creating slideshows.

[99] Random data such as confidential documents, ID photos, security data, text, other images, personnel files, spreadsheets, or fingerprint biometrics may be combined in one image file to build a paperless record or database.

[100] When random data comprises three-dimensional solid modeling information scanned by laser or other methodology, such random data may be combined with the visual image data stored in the image thereby allowing mapping of the visual image onto the solid model, resulting in more realistic models.

## [101] CONCLUSION

[102] The present invention solves many of the problems associated with the prior art. The present invention further provides novel and unique JPEG related applications. The present invention allows reliable storage of random data with an image file. It eliminates maintenance requirements inherent in associating application data with image files. It also prevents irrecoverable failure when computer reads application data containing 0xFF data entries while at the same time provides a header that enables detection of embedded data, allows reliable reconstruction of the original application file, provides information usable by playback applications, and provides traceability of any modifications to the data.

**[103]** Although the description above contains many specifications, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of presently preferred embodiments of this invention. Thus, the scope of the invention should be determined by the appended claims and their legal equivalents rather than by the examples given.

## CLAIMS

What is claimed is:

1. A method of storing non-JPEG data in a JPEG file, comprising:
   (A) reading a predetermined number of sequential bytes of the non-JPEG data;
   (B) determining a position in the sequential bytes of each byte that contains a JPEG marker;
   (C) determining a locator byte, wherein the locator byte is capable of indicating the position of each byte in the sequential bytes that contains a JPEG marker; and
   (D) determining encoded bytes, the encoded bytes comprising:
      (a) the locator byte;
      (b) each byte of the sequential bytes that does not contain a JPEG marker; and
      (c) a place holder byte for each byte of the sequential bytes that comprises a JPEG marker.

2. The method of claim 1, further comprising the step of writing the encoded bytes in a predetermined order.

3. The method of claim 2, wherein the locator byte is written in the most significant position.

4. The method of claim 2, wherein the sequential bytes are written in order of their occurrence in the sequential bytes and the place holder bytes are written in the order of the bytes they replace.

5. The method of claim 2, wherein the encoded bytes are written in the JPEG file

6. The method of claim 5, further comprising writing an application data marker in the JPEG file before the encoded bytes.

7. The method of claim 1, wherein the predetermined number of sequential bytes is seven.

8. The method of claim 1, wherein the JPEG marker comprises 0xFF.

9. The method of claim 1, wherein the place holder byte comprises 0x00.

10. The method of claim 1, wherein the locator byte is between 0x00 to 0x7F.

11. The method of claim 1, wherein a locator byte of 0x00 indicates that none of the sequential bytes contain a JPEG marker.

12. The method of claim 1, wherein the non-JPEG data comprises sound data.

13. The method of claim 1, further comprising:

(A) locating an application data marker in the JPEG file;

(B) writing the encoded bytes into the JPEG file after the application data marker.

14. The method of claim 1, further comprising:

(A) repeating steps A through D of claim 1 for all of the non-JPEG data;

(B) determining a total number of encoded bytes;

(C) if the total number of encoded bytes exceeds a maximum JPEG field length:

(a) dividing the encoded file into segments, the number of bytes in each segment being less than the maximum JPEG field length; and

(b) writing the segments into the JPEG file.

15. The method of claim 13, further comprising recording the JPEG file with the encoded bytes.

16. The method of claim 13, further comprising transmitting the JPEG file with the locator byte and the encoded bytes.

17. The method of claim 13, further comprising:

(A) reading the encoded bytes in the JPEG file;

(B) determining the locator byte;

(C) determining the location of any place holder bytes in the encoded bytes from the locator byte; and

(D) changing each place holder byte to a byte containing JPEG markers, thereby decoding the encoded bytes and obtaining the sequential bytes.

18. The method of claim 17, further comprising using the sequential bytes in an application program.

19. The method of claim 17, further comprising recording the sequential bytes.

20. The method of claim 13, further comprising writing control data into the JPEG file, the control data being relevant to the non-JPEG data.

21. A system for communicating image data and other data, comprising:

(A) a JPEG file, the JPEG file comprising JPEG data;

(B) non-JPEG data;

(C) a computer, the computer being programmed to perform the following:

(a) reading a predetermined number of sequential bytes of the non-JPEG data;

(b) determining a position in the sequential bytes of each byte that contains a JPEG marker;

(c) determining a locator byte, wherein the locator byte is capable of indicating the position of each byte in the sequential bytes that contains a JPEG marker;

(d) determining encoded bytes, the encoded bytes comprising:

(i) the locator byte;

(ii) each byte of the sequential bytes that does not contain a JPEG marker; and

(iii) a place holder byte for each byte of the sequential bytes that comprises a JPEG marker; and

(e) writing encoded bytes in the JPEG file, the encoded bytes being written in a predetermined order.

22. The system of claim 21, further comprising a transmission device in communication with the computer, the transmission device being configured to transmit the JPEG file with the encoded bytes.

23. The system of claim 21, further comprising an image data input device in

communication with the computer, the image data input device being adapted to provide JPEG data.

24. The system of claim 23, wherein the image data input device comprises digital camera.

25. The system of claim 21, further comprising a non-JPEG data input devicein communication with the computer, the non-JPEG data input device being configured to provide non-JPEG data.

26. The system of claim 21, wherein the non-JPEG data input device comprises a sound recording device.

27. The system of claim 21, wherein the computer is further programmed to write the encoded bytes in the JPEG file after an application data marker.

28. The system of claim 21, wherein the computer is further programmed to write control data in the JPEG file, the control data being relevant to the non-JPEG data.

29. The system of claim 21, further comprising a second computer in communication with the computer, the second computer being configured to perform the following:

(A) read the encoded bytes in the JPEG file;

(B) determine the locator byte;

(C) determine the location of any place holder bytes in the encoded bytes from the locator byte; and

(D) change each place holder byte to a byte containing JPEG markers, thereby decoding the encoded bytes and obtaining the sequential bytes.

30. The method of claim 26, wherein the computer is adapted to use the non-JPEG data

31. A method of storing non-JPEG data in a JPEG file, comprising:

(A) reading the non-JPEG data;

(B) determining which bytes of the non-JPEG data, if any, comprise a predetermined JPEG marker;

(C) determining encoded bytes, the encoded bytes comprising:

(a) each of the bytes of the non-JPEG data that do not comprise a JPEG marker;

(b) a place holder byte for each byte of the non-JPEG data that comprises the predetermined JPEG marker;

(c) at least one locator byte, the locator byte being capable of indicating which of the encoded bytes are place holder bytes.

32. The method of claim 31, further comprising the step of writing the encoded bytes in a predetermined order.

33. The method of claim 32, wherein the locator byte is written in the most significant position.

34. The method of claim 32, wherein the encoded bytes are written in the JPEG file.

35. The method of claim 34, further comprising writing an application data marker in the JPEG file before the encoded bytes.

36. The method of claim 31, wherein the JPEG marker comprises 0xFF.

37. The method of claim 31, wherein the place holder byte comprises 0x00.

38. The method of claim 31, wherein a locator byte of 0x00 indicates that none of the bytes of non-JPEG data contain a JPEG marker.

39. The method of claim 31, wherein the non-JPEG data comprises sound data.

40. The method of claim 31, further comprising:

(A) locating an application data marker in the JPEG file;

(B) writing the encoded bytes into the JPEG file after the application data marker.

41. The method of claim 31, further comprising:

(A) determining a total number of encoded bytes;

(B) if the total number of encoded bytes exceeds a maximum JPEG field length:

(a) dividing the encoded bytes into segments, the number of bytes in each segment being less than the maximum JPEG field length; and

(b) writing the segments into the JPEG file.

42. The method of claim 34, further comprising recording the JPEG file with the encoded bytes.

43. The method of claim 34, further comprising transmitting the JPEG file with the locator byte and the encoded bytes.

44. The method of claim 34, further comprising:

(A) reading the encoded bytes in the JPEG file;

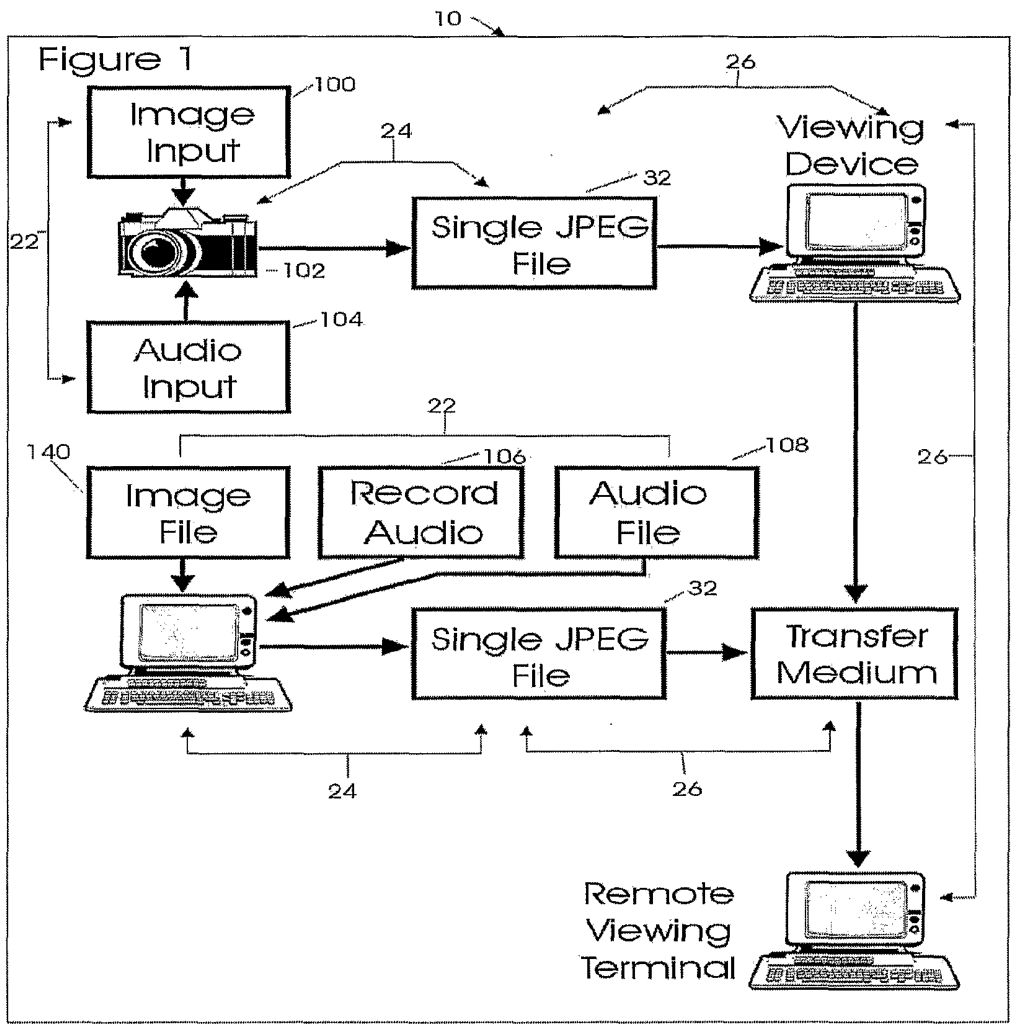
(B) determining the locator byte;

(C) determining the location of any place holder bytes in the encoded bytes from the locator byte; and

(D) changing each place holder byte to a byte containing JPEG markers, thereby decoding the encoded bytes and obtaining the sequential bytes.

45. The method of claim 34, further comprising using the non-JPEG bytes in an application program.

46. The method of claim 34, further comprising writing control data into the JPEG file, the control data being relevant to the non-JPEG data.


10
Figure 1
26
100
Image Input
24
Viewing Device
32
Single JPEG File
22
102
104
Audio Input
22
140
106
108
Image File
Record Audio
Audio File
26
32
Single JPEG File
Transfer Medium
24
26
Remote Viewing Terminal

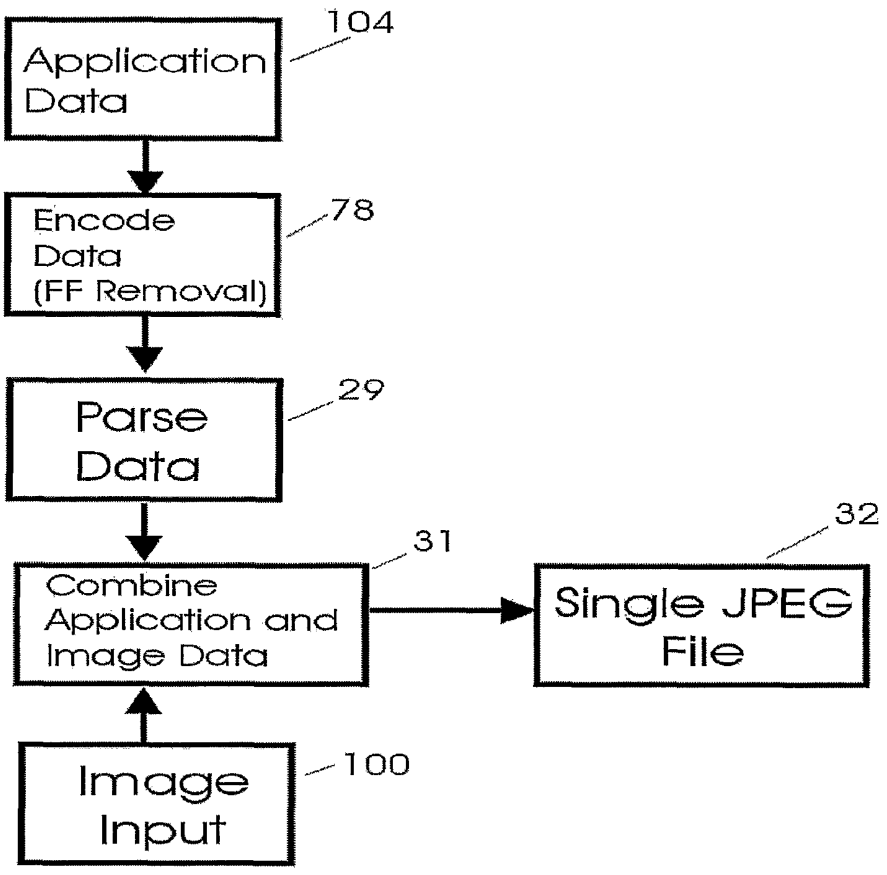

Application Data
104
Encode Data (FF Removal)
78
Parse Data
29
Combine Application and Image Data
31
Single JPEG File
32
Image Input
100

Figure 2

Figure 3

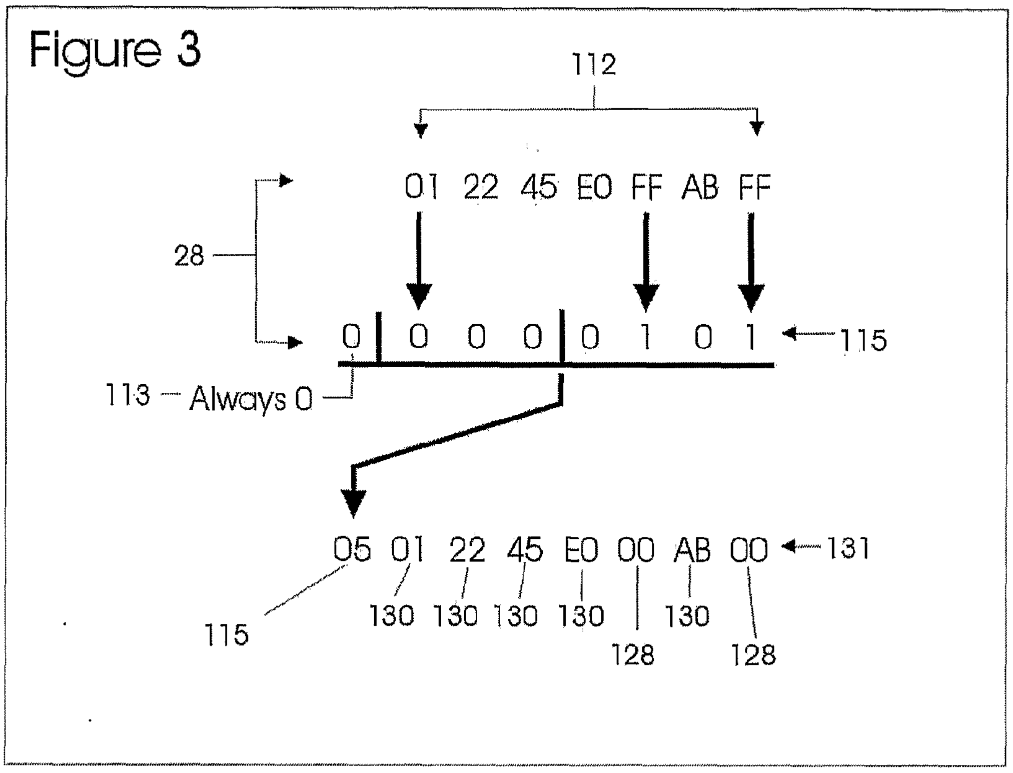
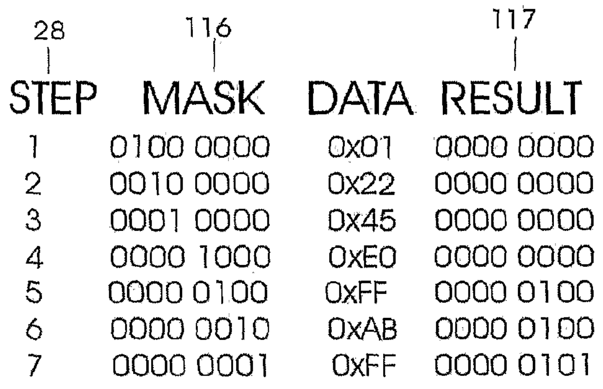

112
28
01 22 45 E0 FF AB FF
0 | 0 0 0 | 0 1 0 1
115
113 — Always 0
05 01 22 45 E0 00 AB 00
131
115
130 130 130 130 130
128 128

Figure 4



| STEP | MASK | DATA | RESULT |
|---|---|---|---|
| 1 | 0100 0000 | 0x01 | 0000 0000 |
| 2 | 0010 0000 | 0x22 | 0000 0000 |
| 3 | 0001 0000 | 0x45 | 0000 0000 |
| 4 | 0000 1000 | 0xE0 | 0000 0000 |
| 5 | 0000 0100 | 0xFF | 0000 0100 |
| 6 | 0000 0010 | 0xAB | 0000 0100 |
| 7 | 0000 0001 | 0xFF | 0000 0101 |

Figure 5

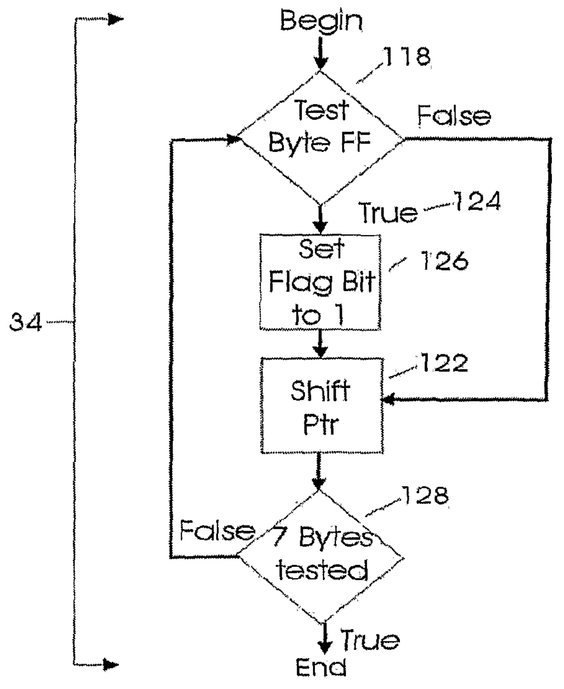
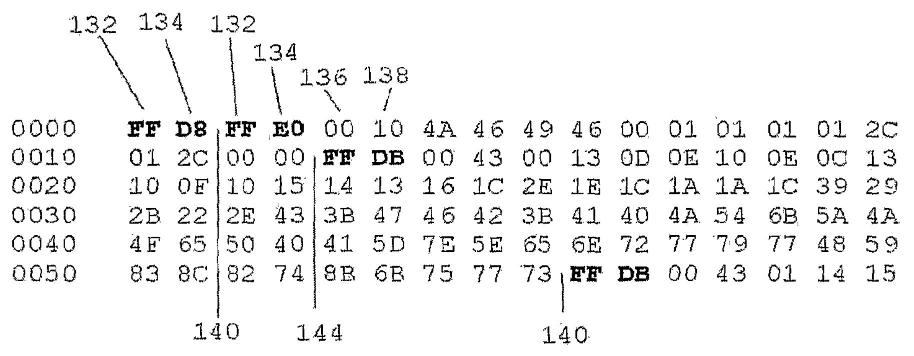
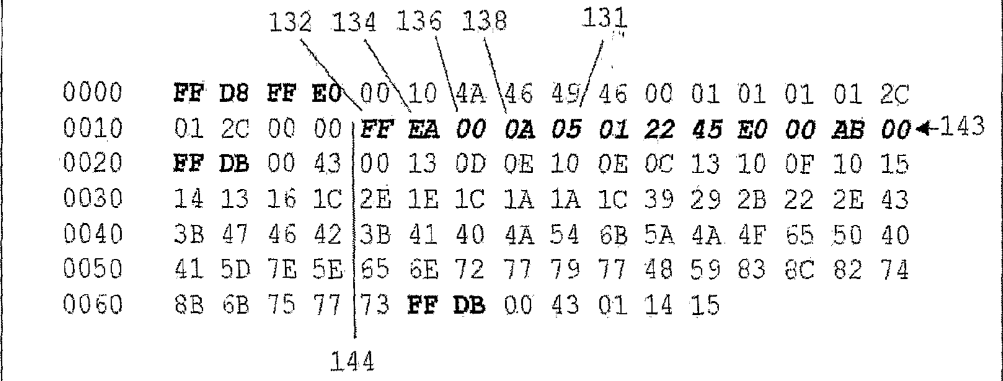

Begin
118
Test Byte FF
False
True
124
Set Flag Bit to 1
126
34
Shift Ptr
122
128
False
7 Bytes tested
True
End

FIGURE 6



```
0000   FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C
0010   01 2C 00 00 FF DB 00 43 00 13 0D 0E 10 0E 0C 13
0020   10 0F 10 15 14 13 16 1C 2E 1E 1C 1A 1A 1C 39 29
0030   2B 22 2E 43 3B 47 46 42 3B 41 40 4A 54 6B 5A 4A
0040   4F 65 50 40 41 5D 7E 5E 65 6E 72 77 79 77 48 59
0050   83 8C 82 74 8B 6B 75 77 73 FF DB 00 43 01 14 15
```

FIGURE 7

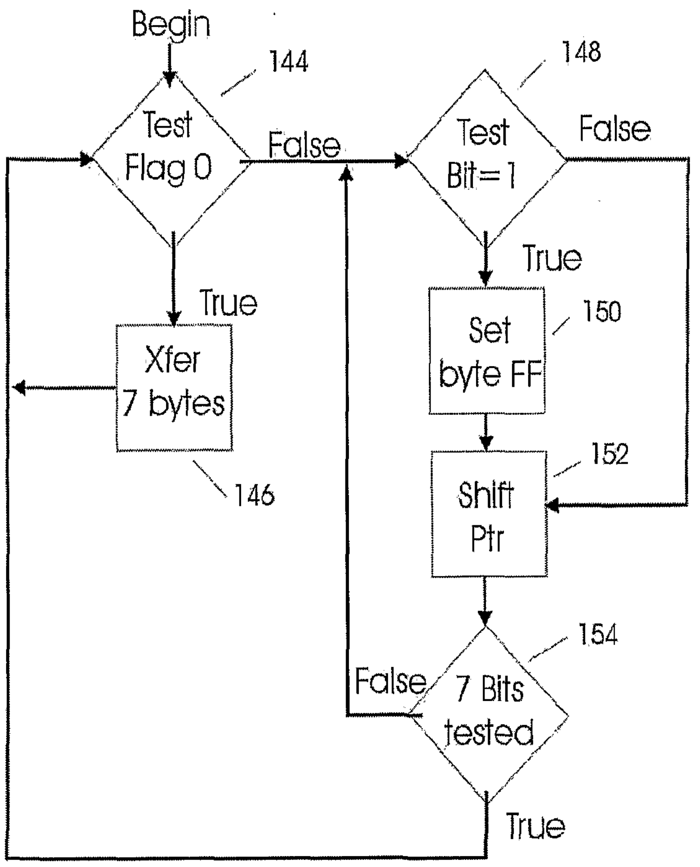


```
0000    FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C
0010    01 2C 00 00 FF EA 00 0A 05 01 22 45 E0 00 AB 00
0020    FF DB 00 43 00 13 0D 0E 10 0E 0C 13 10 0F 10 15
0030    14 13 16 1C 2E 1E 1C 1A 1A 1C 39 29 2B 22 2E 43
0040    3B 47 46 42 3B 41 40 4A 54 6B 5A 4A 4F 65 50 40
0050    41 5D 7E 5E 65 6E 72 77 79 77 48 59 83 8C 82 74
0060    8B 6B 75 77 73 FF DB 00 43 01 14 15
```

FIGURE 8

```
      148                     132          146                       132
0000   FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C
0010   01 2C 00 00 FF EA 00 18 53 4F 55 4E 44 50 49 58
0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF EA
0030   00 3E 53 50 58 41 55 44 49 4F 00 00 01 00 00 00
0040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  132
0050   00 00 00 00 00 00 56 22 08 02 00 00 00 00 00 00
0060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF EA
0070   7C 2F 53 50 58 44 41 54 41 00 00 00 01 00 01 52
0080   49 46 00 46 94 6C 00 00 57 41 00 56 45 66 6D 74
          148
```

Figure 9


Begin
144
Test Flag 0
False
True
Xfer 7 bytes
146
148
Test Bit=1
False
True
Set byte FF
150
Shift Ptr
152
154
False
7 Bits tested
True