```
**************************
* THE ATARI JPEG DECODER *
**************************
```

Developer's Guide

Atari and Brainstorm has developed the fastest JPEG decoder, specifically designed for Falcon030. It uses both DSP and 68030 simultaneously. JPEG is a compression/decompression format based on a DCT which allows very high compression rates. Note that it is a lossy compression which means that there is some loss between the original picture and the compressed picture. But this loss may not be visible...

I) Introduction

Currently, the Atari JPEG decoder can decompress a 24 bits 320x200 picture in less than one second, which allows use of JPEG in games for example. This decoder is faster on the Falcon030 than the one we have tested on PC 486 DX2 66MHz.

This JPEG decoder is currently provided by Atari as an AUTO program which installs a specific cookie and JPEG decoding functions. In the future, it might be integrated in the ROMS and MultiTOS. Of course the cookie will remain the same.

When you are programming a software you first have to check if the JPEG cookie is installed. If NO, you will have to disable JPEG loading or to use your own routines (the C source code of JPEG is free ware). If YES, use it and your picture loading will be amazingly fast.

For the moment, only the JPEG decoder exists...

II) The JPD Cookie

When launched, the JPEG driver install a cookie in the Cookie Jar. It's name is '_JPD'. Following the cookie magic is a pointer to the cookie structure.

This structure is:

```
1 long:     'CookieVersion' (Currently $00000001)
1 long:     'JPGDOpenDriver' routine pointer
1 long:     'JPGDCloseDriver' routine pointer
1 long:     'JPGDGetStructSize' routine pointer
1 long:     'JPGDGetImageInfo' routine pointer
1 long:     'JPGDGetImageSize' routine pointer
1 long:     'JPGDDecodeImage' routine pointer
```

Warning:
The driver should be called in user mode. At that point, it works also in super mode, but future versions may not. Note that all user routines are called in the same mode as the driver.

III) What Calls to Decode?

The following calls must be performed in order to decode a JPEG file:

1: Call 'JPGDGetStructSize'. No input parameters are required. This function return the size (in bytes)of the JPEG decoder structure in register D0.

2: Allocate the structure, and CLEAR IT.

3: Call 'JPGDOpenDriver', with the JPEG structure in register A0. This function locks the DSP, and perform some internal initializations.

4: Call 'JPGDGetImageInfo', with the JPEG structure in register A0. This function return the image size and some other useful informations on the JPEG structure.

5: Set some internal information in the JPEG structure: The output image colorspace (Luma, RGB), and the size of the output image pixels (1 byte for Luma, 2 ,3 or 4 bytes for RGB).

5: Call 'JPGDGetImageSize', with the JPEG structure in register A0. This function return the size of the decoded image in the JPEG structure. If you only want to uncompress the image to disk, you don't have to take care of this size but: YOU ALWAYS HAVE TO CALL THIS FUNCTION.

6: If you want to uncompress the image in RAM, allocate a buffer of the size returned by the previous function call. NEVER CALCULATE THE OUTPUT BUFFER SIZE YOURSELF, since it's not exactly 'Width*Height*bytesPerPixel' of the image.

7: Call 'JPGDDecodeImage', with the JPEG structure in register A0. The image is now decoded in the buffer, or in the disk.

8: Call 'JPGDClosedriver'. The DSP is unlocked, and some internal buffers are freed.

NOTE:
You should always check the 'CookieVersion' number. A number other than 1 means that the decoder structure has changed. In this case, you should warn the user that calling the decoder may cause a crash. Of course, we will try to always keep this structure unchanged.

==========
IV) A look at the JPEG decoder structure
==========

Now, take a look at the JPEG decoder structure:

```
1 long: InPointer         ;JPEG Image Pointer
1 long: OutPointer        ;Output Buffer/Filename Pointer
                          (see OutFlag)
1 long: InSize            ;JPEG Image Size (Bytes)
1 long: OutSize           ;Output Image Size (Bytes)
1 word: InComponents      ;JPEG Image Components Number (1->4)
1 word: OutComponents     ;Output Components Number (1->4)
1 word: OutPixelSize      ;Output Pixel Size (1->4)
1 word: OutFlag           ;0 (RAM Output) / -1 (Disk Output)
1 word: XLoopCounter      ;Number of MCUs per Row
1 word: YLoopCounter      ;Number of MCUs per Column
1 long: CreateOutputPtr   ;Pointer to User Routine / 0
1 long: WriteOutputPtr    ;Pointer to User Routine / 0
1 long: CloseOutputPtr    ;Pointer to User Routine / 0
1 long: SigTermOutputPtr  ;Pointer to User Routine / 0
1 long: Comp1GammaPtr     ;Component 1 Gamma Table / 0
```

```
1 long: Comp2GammaPtr     ;Component 2 Gamma Table / 0
1 long: Comp3GammaPtr     ;Component 3 Gamma Table / 0
1 long: Comp4GammaPtr     ;Component 4 Gamma Table / 0
1 long: UserRoutinePtr    ;Pointer to User Routine
                          (Called during Decompression)/0
1 long: OutTmpPointer     ;Current OutPointer / Temporary Disk
                           Buffer Pointer (see OutFlag)
1 word: MCUsCounter       ;Number of MCUs not Decoded
1 word: OutTmpHeight      ;Number of Lines in OutTmpPointer
1 long: UserLong1         ;Free, Available for User
1 long: UserLong2         ;Free, Available for User
1 word: OutHandle         ;0 / Output File Handle (see OutFlag)
1 long: MFDBAddress
1 word: MFDBPixelWidth
1 word: MFDBPixelHeight
1 word: MFDBWordSize
1 word: MFDBFormatFlag
1 word: MFDBBitPlanes
1 word: MFDBReserved,3
```

Now let us explains the exact meaning of each element of the structure:

- InPointer: Pointer to the JPEG datas.

- OutPointer: Pointer to the output buffer (RAM output), or to the filename of the output file (Disk output). In case of disk output, if OutPointer is 0, the decoder will save a file called OUTPUT.TGA in the current directory.

- InSize: JPEG datas size in bytes.

- OutSize: Output image size.

- InComponents: Number of components in the JPEG datas (usually 1 for Y datas or 3 for YCbCr datas).

- OutComponents: Number components in the output image (usually 1 for Y image or 3 for RGB image).

- OutPixelSize: Size (in bytes) of a pixel in the output image. It can be:
    1: Y output
    2: RGB 15 bits format (ATARI true color)
    3: RGB 24 bits format (24 bits true color)
    4: RGB 32 bits format (32 bits true color,
                           MATRIX Gmbh format)

- OutFlag: 0 means RAM output, -1 means Disk output.

- XLoopCounter: Number of MCU per row in the image. MCU width is 16 pixels in most JPEG files.

- YLoopCounter: Number of MCU per column in the image: MCU height

is 16 pixels in most JPEG files.

- CreateOutputPtr: Pointer to a routine called when creating the output file, in case of disk output.

- WriteOutputPtr: Pointer to a routine called when writing the output file, in case of disk output.

- CloseOutputPtr: Pointer to a routine called when closing the output file, in case of disk output.

- SigTermOutputPtr: Pointer to a routine called when an error is found during JPEG decompression and Disk output is specified.

NOTE: If you specify 'DiskOutput' (OutFlag = -1) and the above 4 pointers are 0, the driver writes the file in uncompressed TARGA format.

- Comp1GammaPtr: pointer to a 256 bytes table, used by the decoder as a color correction table for the first image component.

- Comp2GammaPtr: pointer to a 256 bytes table, used by the decoder as a color correction table for the second image component.

- Comp3GammaPtr: pointer to a 256 bytes table, used by the decoder as a color correction table for the third image component.

- Comp4GammaPtr: pointer to a 256 bytes table, used by the decoder as a color correction table for the fourth image component.

NOTE: If one or more of the above 4 pointers is 0, the decoder uses the following linear table 0, 1, 2, 3, ..., 254, 255 for the component(s).

- UserRoutinePtr: Pointer to a routine called during decompression, after decoding one row of the image (XLoopCounter MCUs). You can use it for displaying something on the screen during decompression.

- OutTmpPointer: Current OutPointer (if RAM output), or temporary disk buffer pointer (if disk output).

- MCUsCounter: Number of MCUs not decoded.

- OutTmpHeight: Number of lines in OutTmpPointer.

- UserLong1: Free, Available for the user.

- UserLong2: Free, Available for the user.

- OutHandle: Output file handle (if OutFlag=-1).

- MFDBAddress: Output MFDB

- MFDBPixelWidth: Output MFDB

- MFDBPixelHeight: Output MFDB

- MFDBWordSize: Output MFDB

- MFDBFormatFlag: Output MFDB

- MFDBBitPlanes: Output MFDB

- MFDBReserved1: Output MFDB

- MFDBReserved2: Output MFDB

- MFDBReserved3: Output MFDB

NOTE:
The REAL JPEG decoder structure is longer than the OFFICIAL JPEG decoder structure: There are a lot of internal variables following the MFDB. That's why you should ALWAYS call 'JPGDGetStructSize' function before allocating the structure.

==========
V) Official calling protocol:
==========

You will now discover the list of parameters for each functions. In each case, Input parameters are passed using A0 register. Each function returns an error code via D0 register.

JPEGOpenDriver:

This function locks the DSP, performs internal initialisations, allocates internal buffers.

IN:
A0 = JPEG Structure Pointer

OUT:
D0 = 0 / Error

JPEGCloseDriver:

This function unlocks the DSP (so if your program is also using

the DSP you can reload your LOD file after this call), and frees the internal buffers.

IN:
A0 = JPEG Structure Pointer

OUT:
D0 = 0 / Error

JPEGGetStructSize:

This function has to be called. It returns the exact size of the JPEG decoder structure. Note that this structure is bigger than the normal JPEG structure. So don't forget to call it to allocate the r ight structure size.

IN:
Nothing

OUT:
D0 = JPEG Decoder Structure Size (Bytes)

JPEGGetImageInfo:

This functions returns A STRUCTURE which contains lot of useful informations. For more informations please read the chapter IV (a look at the Jpeg Structure)...

IN:
A0 = JPEG Structure Pointer

-> means 'Input Parameter'
<- means 'Output Parameter'

```
 ->     InPointer
        OutPointer
 ->     InSize
        OutSize
 <-     InComponents
        OutComponents
        OutPixelSize
   OutFlag
        XLoopCounter
        YLoopCounter
        CreateOutputPtr
        WriteOutputPtr
        CloseOutputPtr
        SigTermOutputPtr
        Comp1GammaPtr
        Comp2GammaPtr
```

```
        Comp3GammaPtr
        Comp4GammaPtr
        UserRoutinePtr
        OutTmpPointer
        MCUsCounter
        OutTmpHeight
        UserLong1
        UserLong2
        OutHandle
        MFDBAddress
  <-    MFDBPixelWidth
  <-    MFDBPixelHeight
        MFDBWordSize
        MFDBFormatFlag
        MFDBBitPlanes
        MFDBReserved,3

 OUT:
 D0 = 0 / Error


JPEGGetImageSize:
-----------------
ALWAYS CALL THIS FUNCTION!  NEVER FORGET TO CALL IT!  EVEN IF YOU
DON'T CARE ABOUT THE SIZE! (It returns the size of the picture in
bytes inside the "OutSize" element of the structure).

 IN:
 A0 = JPEG Structure Pointer

 -> means 'Input Parameter'
 <- means 'Output Parameter'

        InPointer
        OutPointer
        InSize
  <-    OutSize
        InComponents
  ->    OutComponents
  ->    OutPixelSize
        OutFlag
        XLoopCounter
        YLoopCounter
        CreateOutputPtr
        WriteOutputPtr
        CloseOutputPtr
        SigTermOutputPtr
        Comp1GammaPtr
        Comp2GammaPtr
        Comp3GammaPtr
        Comp4GammaPtr
        UserRoutinePtr
        OutTmpPointer
```

```
        MCUsCounter
        OutTmpHeight
        UserLong1
        UserLong2
        OutHandle
        MFDBAddress
        MFDBPixelWidth
        MFDBPixelHeight
 <-     MFDBWordSize
        MFDBFormatFlag
        MFDBBitPlanes
        MFDBReserved,3

 OUT:
 D0 = 0 / Error


JPEGDecodeImage:
----------------
Incredible but true!  This call decodes at an amazing speed  your
JPEG file...

 IN:
 A0 = JPEG Structure Pointer

 -> means 'Input Parameter'
 <- means 'Output Parameter'
 ( ) means 'Optional Parameter'

        InPointer
        OutPointer
        InSize
        OutSize
        InComponents
        OutComponents
        OutPixelSize
 ->     OutFlag
        XLoopCounter
        YLoopCounter
(->)    CreateOutputPtr
(->)    WriteOutputPtr
(->)    CloseOutputPtr
(->)    SigTermOutputPtr
(->)    Comp1GammaPtr
(->)    Comp2GammaPtr
(->)    Comp3GammaPtr
(->)    Comp4GammaPtr
(->)    UserRoutinePtr
        OutTmpPointer
        MCUsCounter
        OutTmpHeight
        UserLong1
        UserLong2
        OutHandle
```

```
  <-    MFDBAddress
  <-    MFDBPixelWidth
  <-    MFDBPixelHeight
  <-    MFDBWordSize
  <-    MFDBFormatFlag
  <-    MFDBBitPlanes
  <-    MFDBReserved,3

 OUT:
 D0 = 0 / Error
```

VI) User definable functions:

The developer can enhance the functionalities of the JPEG Decoder by adding his own routines to force the decoder to save the picture in a specific format for example. If you have yet created a program with a specific way to manage screen in the memory, it is very easy to implement the JPEG cookie in it without modifying anything, thanks to the user definable routines.

UserRoutinePtr:

It is called by the JPEG Decoder each time one row has been decoded. It allows you to perform some operations on the row or to display something during compression (a status bar for example).

```
  IN:
     A0 = JPEG Structure pointer
 OUT:
     D0 = 0 (continue decoding) / -1 (abort decoding)
```

CreateOutputPtr:

If you have decided to decompress the JPEG image into a file, you can use this function to create your own file format. It is called by the JPEG decoder when it creates the file.

```
 IN:
    A0 = JPEG Structure pointer
 OUT:
   D0 = 0 (continue decoding) / error
```

WriteOutputPtr:

If you have decided to decompress the JPEG image into a file, you can use this function to create your own file format. It is called by the JPEG decoder when it writes datas.

```
 IN:
     A0 = JPEG Structure pointer
 OUT:
     D0 = 0 (continue decoding) / error
```

CloseOutputPtr:

If you have decided to decompress the JPEG image into a file, you can use this function to create your own file format. It is called by the JPEG decoder when it closes the file.

```
 IN:
     A0 = JPEG Structure pointer
 OUT:
     D0 = 0 (continue decoding) / error
```

SigTermOutputPtr:

This function is called by the decoder if a JPEG error happens during file saving.

```
 IN:
     A0 = JPEG Structure pointer
```

VII) JPEG decoder return codes:

Here is a complete list with meanings of the Error Codes returned by the functions of the JPEG Decoder. Don't forget to manage them in your program... Note that the decoder can return a negative code. It's a GEMDOS error.

```
 0=NOERROR           (File correctly uncompressed)
 1=UNKNOWNFORMAT     (Error, File is not JFIF compatible)
 2=INVALIDMARKER     (Error, Reserved CCITT Marker Found)
 3=SOF1MARKER        (Error, Marker not handled by the decoder)
 4=SOF2MARKER        (Error, Marker not handled by the decoder)
 5=SOF3MARKER        (Error, Marker not handled by the decoder)
 6=SOF5MARKER        (Error, Marker not handled by the decoder)
 7=SOF6MARKER        (Error, Marker not handled by the decoder)
 8=SOF7MARKER        (Error, Marker not handled by the decoder)
 9=SOF9MARKER        (Error, Marker not handled by the decoder)
10=SOF10MARKER       (Error, Marker not handled by the decoder)
11=SOF11MARKER       (Error, Marker not handled by the decoder)
12=SOF13MARKER       (Error, Marker not handled by the decoder)
13=SOF14MARKER       (Error, Marker not handled by the decoder)
14=SOF15MARKER       (Error, Marker not handled by the decoder)
15=RSTmMARKER        (Error, Unexpected RSTm Marker)
16=BADDHTMARKER      (Error, Buggy DHT Marker)
```

```
17=DACMARKER          (Error, Marker not handled by the decoder)
18=BADDQTMARKER       (Error, Buggy DQT Marker)
19=BADDNLMARKER       (Error, Unexpected/Invalid DNL marker)
20=BADDRIMARKER       (Error Invalid DRI Marker Size)
21=DHPMARKER          (Error, Marker not handled by the decoder)
22=EXPMARKER          (Error, Marker not handled by the decoder)
23=BADSUBSAMPLING     (Error, Invalid components subsampling)
24=NOTENOUGHMEMORY    (Error, Not enough memory...)
25=DECODERBUSY        (Error, Decoder is busy)
26=DSPBUSY            (Error, the DSP is locked)
27=BADSOFnMARKER      (Error, Buggy SOFn marker)
28=BADSOSMARKER       (Error, Buggy SOS marker)
29=HUFFMANERROR       (Error, Buggy Huffman Stream)
30=BADPIXELFORMAT     (Error, Invalid Output Pixel Format)
31=DISKFULL           (Error, Hard/Floppy Disk Full)
32=MISSINGMARKER      (Error, Marker expected but not found)
33=IMAGETRUNCATED     (Error, More bytes Needed)
34=EXTRABYTES         (Warning, Dummy Bytes after EOI Marker)
35=USERABORT          (User Routine signaled 'Abort')
36=DSPMEMORYERROR     (Error, Not Enough DSP RAM available)
37=NORSTmMARKER       (Error, RSTm Marker expected but not found)
38=BADRSTmMARKER      (Error, Invalid RSTm Marker Number)
39=DRIVERCLOSED       (Error, Driver is Already Closed.)
```

In future versions of the decoder, some new return codes will be probably added.

VIII) Notes:

The current version of the JPEG decoder (0.85) is completely compatible with MultiTOS. It allows you to decode up to 32 files at the same time. Note that decoding two files at the same time is slower than decoding the two files one after the other because some DSP context switch is needed.
If you are working under TOS, the decoder can be runned from the desktop, or from the AUTO folder. If you are working under MultiTOS, it's better to launch the decoder from the AUTO folder, AFTER MINT.PRG (it's then impossible to kill it). Currently, killing the decoder is a very bad idea, specially while de coding a file. This will be fixed on future versions.

When processing a file, the decoder tests if the end of the file has been reached or not, in order to avoid memory violation problems when decomperssing a truncated JPEG file. But, in the Huffman decoder, this test is not performed, for speed reasons. So, you you should always allocate JPEG file size +10, and fill the last ten bytes with: $ff,$ff,$ff,$ff,$ff,$ff,$ff,$ff,$00,$00. This sequence will generate a Huffman error, and memory violation will be avoided.

Note that the 'InSize' value in the JPEG structure should contain the real JPEG file size, not the size +10.

IX) Questions & Answers:

- How to decode just one channel (i.e. RED)?

There are several ways to do it. The easier (and the worst) is simply to decode the complete image, and then to discard the GREEN and BLUE information. This is a bad way to do it since a lot of memory will be required, .i.e. for a 320x200 image the decoder will need 320x200x3 bytes, and the application itself will need 320x200 bytes (a total of 256000 bytes!). A better approach is to specify disk output. The decoder then allocates just one row of MCUs (typically 16 lines of pixels). Then, you install your own disk routines:

The create routine just performs a RTS.

The write routine copy the current MCU row into the application's internal buffer, copying only the RED components.

The close routine just performs a RTS.

A third way to do it is to use the gamma pointers. On GREEN and BLUE gamma pointers (second and third pointers), put a pointer to an empty array of 256 bytes. In this case, the decoder will output a RGB image with all GREEN and BLUE components null.

The best way to decode files in a format not handled by the decoder (i.e. CMY) is to patch the standard disk output routines, and put a conversion routine in the write output routine. In this case, don't forget to patch the create, close, and sigterm routines. They can point to a RTS instructions. You should always keep in mind this rule: Use the user routine for perform some graphic effect, like an animated mouse or a bar, and the disk output routines for everything else.

- Is this driver 100% JPEG compatible?

No. Currently, we have never seen a 100% JPEG compatible program. But, decoding files created by the JPSRC package from the Independant JPEG Group, the PVRG JPEG codec, Alchemy, ... should work with NO problem.

What is currently not implemented is:

- Multiscan images (non interleaved input stream).

- Some strange and not useful subsampling modes.
- Fractional subsampling.

Note that Apple Quicktime JPEG files are not standard JPEG files. If you want to decode such files on the Falcon, you will have first to convert them into standard JPEG files using an utility program called PictPixie, written by Apple inc.