Subject: Re: jpeg text field
Posted by Rick Towler on Tue, 22 Aug 2006 16:42:29 GMT
View Forum Message <> Reply to Message

In JPEG files metadata is stored as Exchangeable Image File Format aka
EXtraInFormation aka EXIF, or in IPTC, or adobe's XMP. There are a
number of libraries and programs to manipulate EXIF data so you could
either write a .dlm or use SPAWN to insert the data after creating the file.

-Rick

greg michael wrote:
> Does anyone know how to embed a text field into a jpeg file? Digital
> cameras seem to be able to do it, but in IDL I can only find this kind
> of facility for tif-files.
>
> kind regards,
> Greg
>

Subject: Re: jpeg text field
Posted by Andrew Cool on Tue, 22 Aug 2006 23:43:32 GMT
View Forum Message <> Reply to Message

greg michael wrote:
> Does anyone know how to embed a text field into a jpeg file? Digital
> cameras seem to be able to do it, but in IDL I can only find this kind
> of facility for tif-files.
>
> kind regards,
> Greg

Greg,

The code below from the JHUAPL library reads Exif data, and goes into
some
detail on the format. You might be able to use this info to write out
your
own Exif structure? Perhaps by reading your JPEg file as one large
bytarr, and then
overwriting the necessary Exif section?

Cheers,
Andrew

```
 ;---------------------------------------------------------- --
```

```
;+
; NAME:
;       EXIF_READER50
; PURPOSE:
;       Read values from an Exif file (digital camera file).
; CATEGORY:
; CALLING SEQUENCE:
;       exif_reader50, file, values (IDL 5.0 version)
; INPUTS:
;       file = Name of digital camera image file.   in
; KEYWORD PARAMETERS:
;       Keywords:
;         /LIST means list values.
;         OUT=out Return values in a text array.
;         DESCRIPTION=desc Returned text array of descriptive text.
;         /TAGS means list in hex all tag values found.
;         ERROR=err Error flag: 0=valid EXIF file, 1=not Exif.
; OUTPUTS:
;       values = Returned structure with values.    out
; COMMON BLOCKS:
; NOTES:
;       Note: This version avoids using ulong, seems to work
;         but may have a problem at some point.
; MODIFICATION HISTORY:
;       R. Sterner, 2000 Dec 19
;       R. Sterner, 2001 Jul 12 --- Added ori, also more comments.
;       R. Sterner, 2001 Aug 19 --- Fixed for odd endian, added out.
;       R. Sterner, 2001 Nov 16 --- Dropped uint for backward
compatibility.
;       R. Sterner, 2002 Jan 02 --- IDL 5.0 version.
;
; Copyright (C) 2000, Johns Hopkins University/Applied Physics
Laboratory
; This software may be used, copied, or redistributed as long as it is
not
; sold and this copyright notice is reproduced on each copy made.  This
; routine is provided as is without any express or implied warranties
; whatsoever.  Other limitations apply as described in the file
disclaimer.txt.
;-
 ;------------------------------------------------------------- --
 pro exif_reader50, file, values, list=list, description=desc, $
   tags=tags, error=err, out=out, help=hlp

 if (n_params(0) lt 1) or keyword_set(hlp) then begin
   print,' Read values from an Exif file (digital camera file).'
   print,' exif_reader50, file, values (IDL 5.0 version)'
   print,'   file = Name of digital camera image file.   in'
```

```
  print,'   values = Returned structure with values.    out'
  print,' Keywords:'
  print,'   /LIST means list values.'
  print,'   OUT=out Return values in a text array.'
  print,'   DESCRIPTION=desc Returned text array of descriptive text.'
  print,'   /TAGS means list in hex all tag values found.'
  print,'   ERROR=err Error flag: 0=valid EXIF file, 1=not Exif.'
  print,' Note: This version avoids using ulong, seems to work'
  print,'   but may have a problem at some point.'
  return
endif

;------------  Init  ---------------------
hend = endian()   ; Host endian (current computer).
eflag = 1-hend   ; JPEG file values are big endian.
txtman = ''   ; Camera Manufacturer.
txtmod = ''   ; Camera Model.
txtver = ''   ; Camera firmware version.
txttim = ''   ; Date/Time.
shutter = 0.   ; Shutter speed (1/sec).
fnum = 0.   ; F-number as rational.
dt_tm = ''   ; Date/Time of original.
xprog_tab = ['<?','manual control','program normal',$
  'aperture priority','shutter priority',$
  'program creative (slow program)',$
  'program action (high-speed program)',$
  'portrait mode','landscape mode','>?']
expprog = ''   ; Exposure program (see table).
isospeed = 0   ; ISO Speed.
dist = 0.   ; Focus distance (m).
subsec = ''   ; Time subsecond.
maxap = 0.   ; Max Aperture.
foclen = 0.   ; Focal len (mm).
flash = -1   ; Flash?
x_pixels = 0   ; X size in pixels.
y_pixels = 0   ; Y size in pixels.
ori = 0    ; Orientation flag (1 to 8).
bitsperpix = 0   ; Est. Compression in bits/pixel,
cmt = ''   ; User comment.

;--------  Open Image file  ------------
openr,lun,file,/get_lun

 ;---------------------------------------------------------- ---
; Check for and read Exif data
; JPEG uses Markers to indicate pieces
; of data.  Every JPEg file starts with the
; Marker SOI (start of Image) which is the
; 2 bytes 255,216 = FF,D8
; Markers FFE0 to FFEF are "Application Markers" (=APPn)
; used by applications but not needed to decode JPEG.
; Digital Cameras use the EXIF data structure
; with marker FFE1=APP1 (older cameras used JFIF with
; marker FFE0=APP0).
; So the APP1 marker is FFE0.  The length in bytes
; of the APP1 data follows in the next 2 bytes (including
; the 2 length bytes).  Next the APP1 data itself follows.
; The APP1 data (EXIF data) may be thousands of bytes long.
 ;--------------------------------------------------------- ---
soi = bytarr(2)   ; Start of Image (JFIF).
readu,lun,soi   ; SOI = 255,216 = FF,D8.
pntr = 2L   ; Pointer into file.
app = bytarr(2)   ; App marker.
readu,lun,app   ; APP1 = 255,225 = FF,E1.
pntr = pntr + 2L
len = bytarr(2)   ; APP1 data length (2 bytes).
readu,lun,len
pntr = pntr + 2L
len = fix(len,0)  ; Convert 2 bytes to int.
if eflag then len=swap_endian(len)
;----------------------------------------------
; Catch endian problem here
;----------------------------------------------
if len lt 0 then begin
  eflag = 1-eflag
  len=swap_endian(len)
  print,' Unexpected endian.'
endif
;----------------------------------------------
buff = bytarr(len-2)  ; App data buffer size.
readu,lun,buff   ; All the EXIF data, many bytes.
free_lun, lun
id = string(buff)  ; Better say 'Exif'
if id ne 'Exif' then begin
  print,' Not an Exif file.'
  err = 1
  return
endif

 ;--------------------------------------------------------- ---
; This is an Exif file
; EXIF data starts with the word Exif, then 0,0 = 6 bytes.
; EXIF uses TIFF format to store data, this starts after
; the first 6 bytes.
; The first 2 bytes give the byte alignment used in the
; TIFF data, II or MM.
```

```
  ;---------------------------------------------------------- ---
 buff = buff(6:*)  ; All offsets are from this start.
 balign = string(buff(0:1)) ; Find target endian.
 if balign eq 'II' then tend=0 else tend=1
; if hend ne tend then print,' Must endian swap TIFF values.'
 ifd_off = long(buff(4:7),0) ; Offset to first Image File Directory.
 if hend ne tend then ifd_off=swap_endian(ifd_off)

  ;---------------------------------------------------------- ------
 ;  First Image File Directory (IFD0 = Main Image)
 ;  Thumbnail image has IFD1 but we don't care about that.
 ;  IFD0 has a link at end to IFD1, we want link to SubIFD
 ;  which is given by the value of IFD0 tag 8769 (hex).
 ;  IFD0 only has a few interesting items, like Camera
 ;  manufacturer and model number.  All the other good stuff
 ;  is in SubIFD.
 ;
 ;  Each item is 12 bytes.  The format of an item entry is:
 ;    Tag_number (2 bytes)
 ;    Format (2 bytes)
 ;    Number of components (4 bytes)
 ;    Data or offset to data (4 bytes)
  ;---------------------------------------------------------- ------
 subifd_off = 0
 num = fix(buff(ifd_off:ifd_off+1),0) ; # entries in IFD0.
 if hend ne tend then num=swap_endian(num)
;print,' IFD0 = Main Image'
;help,num
 if keyword_set(tags) then print,' IFD0 Tags:'
 for i=0,num-1 do begin   ; Loop through entries.
   lo = 12*i+ifd_off+2   ; Each entry is 12 bytes.
   hi = lo+11
   t = buff(lo:hi)   ; Grab bytes for entry.
; tag = uint(t(0:1),0)   ; Get tag number.
   tag = fix(long([t(0:1),0B,0B],0))
   if hend ne tend then tag=swap_endian(tag)
   if tag lt 0 then tag=tag+2L^16
   form = fix(t(2:3),0)   ; Get format.
   if hend ne tend then form=swap_endian(form)
   nn = long(t(4:7),0)   ; Get # items.
   if hend ne tend then nn=swap_endian(nn)
   dd = long(t(8:11),0)   ; Get data or pointer to data.
   if hend ne tend then dd=swap_endian(dd)
   xtag = basecon(to=16,tag) ; Want tag in hex.
   if keyword_set(tags) then print,' Tag = '+xtag
   ;---------  Process selected tags  -------------
   if xtag eq '10F' then txtman = string(buff(dd:dd+nn-1))
   if xtag eq '110' then txtmod = string(buff(dd:dd+nn-1))
   if xtag eq '131' then txtver = string(buff(dd:dd+nn-1))
   if xtag eq '132' then txttim = string(buff(dd:dd+nn-1))
   if xtag eq '112' then begin
; ori = uint(t(8:11),0)
     ori = fix(long([t(8:11),0B,0B],0))
     if hend ne tend then ori=swap_endian(ori)
     if ori lt 0 then ori=ori+2L^16
   endif
   if xtag eq '8769' then subifd_off = dd
 endfor

  ;---------------------------------------------------------- ------
 ;  SubIFD
  ;---------------------------------------------------------- ------
 ifd_off = subifd_off   ; Offset to SubIFD.
 num = fix(buff(ifd_off:ifd_off+1),0) ; # entries in SubIFD.
 if hend ne tend then num=swap_endian(num)
;print,' SubIFD: num = ',num
 if keyword_set(tags) then print,' '
 if keyword_set(tags) then print,' SubIFD Tags:'
 for i=0,num-1 do begin   ; Loop through entries.
   lo = 12*i+ifd_off+2   ; Each entry is 12 bytes.
   hi = lo+11
   t = buff(lo:hi)
; tag = uint(t(0:1),0)
   tag = fix(long([t(0:1),0B,0B],0))
   if hend ne tend then tag=swap_endian(tag)
   if tag lt 0 then tag=tag+2L^16
   form = fix(t(2:3),0)
   if hend ne tend then form=swap_endian(form)
   nn = long(t(4:7),0)
   if hend ne tend then nn=swap_endian(nn)
   dd = long(t(8:11),0)
   dds = fix(t(8:11),0)
   if hend ne tend then dd=swap_endian(dd)
   if hend ne tend then dds=swap_endian(dds)
   xtag = basecon(to=16,tag)
;help,xtag,dd,dds
   if keyword_set(tags) then print,' Tag = '+xtag
   ;---------  Process selected tags  -------------
   if xtag eq '9003' then dt_tm = string(buff(dd:dd+nn-1))
   if xtag eq '8822' then expprog = (xprog_tab[[dds]])(0)
   if xtag eq '8827' then isospeed = dd
   if xtag eq '9291' then subsec = string(buff(dd:dd+nn-1))
   if xtag eq '9209' then flash = dd
;   if xtag eq 'A002' then x_pixels = dd ; dd and dds same for Nikon.??
;   if xtag eq 'A003' then y_pixels = dd
   if xtag eq 'A002' then x_pixels = dds
   if xtag eq 'A003' then y_pixels = dds
   if xtag eq '9102' then begin
     tmp = long(buff,dd,2)
     bitsperpix = (float(tmp[0])/tmp[1])
   endif
   if xtag eq '9206' then begin
     tmp = long(buff,dd,2)
     if tmp[1] ne 0 then dist=(float(tmp[0])/tmp[1])
   endif
   if xtag eq '829A' then begin
     tmp = long(buff,dd,2)
     shutter = (float(tmp[1])/tmp[0])
   endif
   if xtag eq '829D' then begin
     tmp = long(buff,dd,2)
     fnum = (float(tmp[0])/tmp[1])
   endif
   if xtag eq '9205' then begin
     tmp = long(buff,dd,2)
     maxap = sqrt(2)^(float(tmp[0])/tmp[1])
   endif
   if xtag eq '920A' then begin
     tmp = long(buff,dd,2)
     foclen = (float(tmp[0])/tmp[1])
   endif
   if xtag eq '9286' then begin  ; User comment.
     cmt = buff(dd:dd+nn-1)
     w = where(cmt eq 0, cnt)  ; Replace leading 0s with spc.
     if cnt gt 0 then cmt(w) = 32B
     cmt = strtrim(string(cmt),2)
   endif
 endfor

 ;------  Finish up some values  ------------
 if subsec ne '' then subsec='.'+subsec
 dt_tm = dt_tm + subsec
 flash = (['NO','YES'])(flash)
 yy = getwrd(dt_tm,0,del=':')+0
 nn = getwrd(dt_tm,1,del=':')+0
 dd = getwrd(dt_tm,2,del=':')+0
 ss = secstr(getwrd(dt_tm,1))
 js = ymds2js(yy,nn,dd,ss)
 if subsec eq '' then begin
   date = dt_tm_fromjs(js,form='Y$ n$ 0d$ h$:m$:s$ w$')
 endif else begin
   date = dt_tm_fromjs(js,form='Y$ n$ 0d$ h$:m$:s$f$ w$')
 endelse
```

```
 tprint,/init
 tprint,' '
 tprint,' File: '+file
 tprint,' Image shot at ',date
 tprint,' Comment: '+cmt
 tprint,' Picture size: '+strtrim(x_pixels,2)+' x '+strtrim(y_pixels,2)
 tprint,' Shutter speed (1/sec): '+strtrim(shutter,2)
 tprint,' F-number: '+strtrim(fnum,2)+' (max aperture: '+ $
   strtrim(maxap,2)+')'
 tprint,' Focal length used (mm): '+strtrim(foclen,2)
 tprint,' ISO Speed: '+strtrim(isospeed,2)
 if dist gt 0. then txt=strtrim(dist,2) else txt='unavailable'
 tprint,' Focus distance (m): '+txt
 tprint,' Flash: ',flash
 tprint,' Exposure program: ',expprog
 tprint,' Estimated average Bits/pixel: '+strtrim(bitsperpix,2)
 tprint,' Orientation flag: '+strtrim(ori,2)
 tprint,' Camera manufacturer: '+txtman
 tprint,' Camera model: '+txtmod
 tprint,' Camera firmware version: '+txtver

 ;---------  List  -------------------
 if keyword_set(list) then begin
   tprint,/print
 endif

 tprint,out=out

 ;------------  Pack into a returned structure  ---------
 values = {file:file, manufacturer:txtman, model:txtmod, $
  firmware_vers:txtver, image_time:dt_tm, date:date, $
         date_js:js, shutter:shutter, $
  fnum:fnum, exp_prog:expprog, ISO_speed:isospeed, $
  focus_dist:dist, max_aper:maxap, focal_len:foclen, $
  flash:flash, x_pixels:x_pixels, y_pixels:y_pixels, $
  ori:ori, ave_bits:bitsperpix, comment:cmt}

 ;---------  Description  ----------------------------
 if not arg_present(desc) then return
 desc = ['Image_time is when image was shot: YYYY:MM:DD
HH:MM:SS.FFFFF',$
  'Shutter is shutter speed in 1/seconds',$
  'fnum is F/number', 'ISO_speed is the film speed equivalent',$
  'focus_dist is focus distance in meters (if available)',$
  'max_aper is the camera maximum aperture as an F/number',$
  'focal_len is lens focal length in mm (for the shot?)',$
  'flash is flash used? NO/YES',$
  'ori is orientation flag (1-8, 0=none)',$
  'ave_bits is the estimated average bits/pixel (compression)']

 err = 0

 end
```

Subject: Re: jpeg text field
Posted by greg michael on Wed, 23 Aug 2006 10:24:00 GMT
View Forum Message <> Reply to Message

Thanks, Rick, Andrew. I guess I should be able to figure out how it
works from that code. I'm writing an ION program, so I think everything
would work more cleanly if I could avoid an external solution. I'm
afraid, though, that the EXIF fields appear to be variable-length, so
it may cause an awful mess with pointers in the file if I try to insert
something. I'll let you know if I have any luck...

regards,
Greg