Exiv2 - Feature #1219
Adding Camera Temperature

31 Aug 2016 12:11 - Raphael Attie

Status:	Assigned	**Start date:**	31 Aug 2016
Priority:	Normal	**Due date:**	
Assignee:	Raphael Attie	**% Done:**	30%
Category:	makernote	**Estimated time:**	30.00 hours
Target version:	0.28		

Description

Modern DSLRs released the last years have reached a degree of quality that has shown great potential for high quality astrophotography. Canon, Nikon and Sony are the preferred ones in this realm. Most of their camera have digital thermometer giving the temperature either of the inner electronics, or of the sensor itself. This parameter, can be retrieved by Exiftool "Camera Temperature" but not known yet by Exiv2. Knowing the camera temperature is essential to perform a sensible calibration. So far, astrophotographer take dark files using the outside temperature as reference, but without certainty of the temperature of the sensor. The end result, after subtracting the dark files, sometimes show that noise have been added instead of being removed, which happens less with CCD cameras dedicated for astronomy which are temperature controlled/regulated.
As a professional astronomer, but also as an amateur astronomer, and developing my own software (see Lightdrops in Github), I encourage the support of this feature with Exiv2, and would like (at least to try) to contribute to this support.

Related issues:

Related to Exiv2 - Feature # 894: Canon Highlight Tone Priority	**Assigned**	**17 Mar 2013**
Related to Exiv2 - Bug # 1206: In specific Canon makernote tags, certain valu...	**Assigned**	**13 Aug 2016**
Related to Exiv2 - Bug # 1203: Exif.CanonCs.FocusContinuous reported as Short	**Assigned**	**11 Aug 2016**

History

#1 - 31 Aug 2016 12:23 - Robin Mills

- Status changed from New to Assigned

- Target version set to 0.28

Thanks for logging this request and for volunteering to work on this.

#2 - 31 Aug 2016 12:23 - Robin Mills

- Category changed from api to makernote

#3 - 31 Aug 2016 14:44 - Raphael Attie

Looking at an updated Makernote tag list from the Exiftool website

It appears the Camera Temperature, for Canon EOS Makernote, is found within
Tag ID: 0x0004 (Exiftool tagname "CanonShotInfo") , at location 12 with byte offset 2 (so called "index2" in Exiftool). It is still unclear how to translate this within Exiv2 API. Exiv2 documentation gives an Exif.Photo.MakerNote , yet a makernote is also mentioned as Exiv2::Internal::CanonMakerNote in the API documentation
Are they the same and only makernote?

#4 - 31 Aug 2016 14:58 - Robin Mills

Raphael

I don't know how it works either. I can step the code with the debugger and get to the bottom of this. However, not today. I'm trying very hard to finish v0.26 and have had about 4 hours of interruptions today already.

Sorry ... I don't have time to work on this at the moment.

Robin

#5 - 31 Aug 2016 15:00 - Raphael Attie

No pb. Just reporting my findings as requested. No rush.

#6 - 01 Sep 2016 05:33 - Robin Mills

Raphael

I have copied your test file to: http://exiv2.dyndns.org:8080/userContent/testfiles/1219/ This is a publicly accessible server. If you are concerned about privacy/copyright I will delete the file.

#7 - 01 Sep 2016 06:05 - Andreas Huggel

Raphael, see http://www.exiv2.org/tags-canon.html, is this tag Exif.CanonSi.0x0006? If that is the case, all that is needed to support it would be to give it a name here and probably add some logic (a "pretty-print function") to make the values readable. See plenty of other examples in the same file for how to do that.

#8 - 01 Sep 2016 08:16 - Raphael Attie

Exif.CanonSi.0x0006 is the 6th tag in there. Exiftool extracts it at the 12th, which would be Exif.CanonSi.0x000c , also unknown. Can be?

#9 - 01 Sep 2016 10:56 - Raphael Attie

I just realised that for the 5D, Camera temperature has a different address than other model, it's in the Canon CameraInfo5DmkIII Tags at
http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/Canon.html#CameraInfo5DmkIII
So, ignore my last comment, the address is not right.

I'm trying to confirm this by looking at the values with Hexa editor. This way i'll understand also a bit more the specificity of the 5D raw files.

Robin, no pb for the file in the public server.

#10 - 01 Sep 2016 11:32 - Robin Mills

Good news all round. This teamwork stuff really works! Thanks for the test file.

#11 - 01 Sep 2016 12:33 - Andreas Huggel

Yes, in Exiftool's Canon tags there is a CameraTemperature in the ShotInfo array:

12	CameraTemperature	int16s	(newer EOS models only)

It may be in other places for different camera models. This one we already support in Exiv2 as Exif.CanonSi.0x000c as you said, we just don't have a name for it yet. So you can compare the exiftool value with that from exiv2. The changes needed to support it properly are as advised before. I don't think we support the CameraInfo5DmkIII tags yet, that would be more work.

#12 - 01 Sep 2016 14:46 - Raphael Attie

Guys, I'm puzzled with all these "unsignedShort" that I see in CanonMakerNote::tagInfoSi_[] = { blah... } , in canonmn.cpp.
Most of them are meant to be signed short, not unsigned.
I think that's the reason I couldn't match your temperature values with Exiftool.

So, I ran all your unknown values of the Shot Info tags:

"Exif.CanonSi.0x0001" = "0"
"Exif.CanonSi.0x0006" = "0"
"Exif.CanonSi.0x0008" = "3"
"Exif.CanonSi.0x000a" = "8"
"Exif.CanonSi.0x000b" = "8"
"Exif.CanonSi.0x000c" = "155"
"Exif.CanonSi.0x000d" = "0"
"Exif.CanonSi.0x0010" = "0"
"Exif.CanonSi.0x0011" = "0"
"Exif.CanonSi.0x0012" = "3"
"Exif.CanonSi.0x0014" = "0"
"Exif.CanonSi.0x0018" = "0"
"Exif.CanonSi.0x0019" = "0"
"Exif.CanonSi.0x001a" = "248"
"Exif.CanonSi.0x001b" = "65535"
"Exif.CanonSi.0x001c" = "65535"
"Exif.CanonSi.0x001d" = "65535"
"Exif.CanonSi.0x001e" = "65535"
"Exif.CanonSi.0x001f" = "0"
"Exif.CanonSi.0x0020" = "0"
"Exif.CanonSi.0x0021" = "0"
Exif.CanonSi.ISOSpeed = "320"

Let's look at "Exif.CanonSi.0x000c" = "155" , as coming from an unsigned short reading.

Now, 155 - 128 = 27 C... Which is the value from Exiftool.

I took a different file, to see if a linear relations holds with this offset, Exiv2 gave me 149, Exiftool gave 21 C, and 149 - 128 = 21.
I did this test from files coming from cameras Canon 5D Mark iii with different Firmwares, and confirm that this holds for old firmwares 1.0.x and more recent 1.1.x.

Note also in my list of values above, the clipped value 65535 of unsigned short, can this be symptomatic of incorrect typing?... See here

Finally, for the 5D mark iii, at a different address (27 = 0x1b) with Exiftool, that value is typed as "int8u" by Exiftool (8-bit unsigned char?). I wonder if that's the same parameter you have at "Exif.CanonSi.0x001b" = "65535", whose value is the max of unsigned short. I'll leave it for maybe later discussions.

Raphael

#13 - 01 Sep 2016 15:18 - Robin Mills

Unless I'm mistaken I believe this is exactly what Sridhar is working on in #1203 (and possibly #1204 #1205 and #1206) !

It's amazing how 2 people can discover the same issue at the same instant even although this issue has been in the code for 10 years without anybody mentioning this.

I will leave Andreas to comment as he's working with Sridhar on this topic and Andreas knows this part of the code.

#14 - 01 Sep 2016 15:55 - Raphael Attie

- File CR2_Scrutinized.pdf added

Indeed, the timing is unbelievable!

I'm uploading a document i've set up to better understand the raw format. I've been using the notorious CR2 poster

and if a kind soul could take the time to tell me where I'm wrong in locating the Shot Info part of my raw file, using an Hex editor, that would be wonderful. So, see attached. Thanks!

#15 - 01 Sep 2016 17:13 - Robin Mills

Somebody has put a lot of work into the document. Here's my way of looking at your file with tiffinfo (part of libtiff) and exiv2:

```
1003 rmills@rmillsmbp:~/gnu/exiv2/trunk $ tiffinfo ~/Downloads/F36A7292.CR2 2>/dev/null | grep -i maker | head -1 | cut -d, -f-20
  MakerNote: 0x2a,0x0,0x1,0x0,0x3,0x0,0x31,0x0,0x0,0x0,0x0,0xe2,0x5,0x0,0x0,0x2,0x0,0x3,0x0,0x4,0x0
1004 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pR ~/Downloads/F36A7292.CR2  | grep -i maker
     640 | 0x927c MakerNote                | UNDEFINED |    68200 |       996 | *.....1.............D..... ...
1005 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

Looks to me as though there's a load of MakerNote data starting with a *\0 Gosh, 42. The meaning of life. Could it be a TIFF without the II?

```
1023 rmills@rmillsmbp:~/gnu/exiv2/trunk $ dd bs=1 count=68200 skip=996 if=~/Downloads/F36A7292.CR2 of=stuff.tiff
68200+0 records in
68200+0 records out
68200 bytes transferred in 0.225005 secs (303104 bytes/sec)
1024 rmills@rmillsmbp:~/gnu/exiv2/trunk $ (echo -n 'II' ; cat stuff.tiff ) > eh.tif
1025 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pS eh.tif
STRUCTURE OF TIFF FILE (II): eh.tif
END eh.tif
1026 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

Nope. I'm lost. Andreas will enlighten us.

#16 - 01 Sep 2016 17:55 - Raphael Attie

- File CR2_Scrutinized.pdf added

The 42 puzzled me too without the double I for endianness...

Nonetheless, my document misled you. I made a little mistake when locating the address (dec) 1620, I was pointing a few bits before. I'm attaching an updated document which shows the bits corresponding to the camera temperature.
It appears that unsigned short or signed short makes no difference. This 128 offset is probably coming from the other Camera Temperature makernote

(5d mark iii makernote) which is unsigned char, so ranging within [0-255]. That must come from the choice of scaling of the original data, which is a reasonable scale given the typical temperature range (unless you put your camera in the oven...). Then it is probably casted as signed short as-is. Exiftool probably took the initiative to convert it by offsetting by -128. That's just a theory.
Still, if someone can comment on the attached document, I still have things I don't fully understand, regarding e.g. the ISO. My ISO was set to 3200, so the Base ISO and Auto ISO aren't fully extracted. There one of them at 320, so, we're missing a factor 10. Note that Exiftool doc says that ISO = Base ISO * Auto ISO / 100.

Raphael

#17 - 01 Sep 2016 18:41 - Robin Mills

I've added Andreas as a watcher of this thread so he's aware of this discussion.

I like to keep track in an issue of the time that's been spent. With a thing like this, I'd probably put in Estimate = 10 hours 10%, and when I update it after a couple of hours effort, I adjust the % and estimate as I go along. First time into unstudied code is usually time consuming.

#18 - 01 Sep 2016 20:30 - Raphael Attie

- File CR2_Scrutinized_updated.pdf added

A little update to my file to show and confirm the value and addresses for the Camera Temperature value at two different "expected" locations.
For my .CR2 file, Camera Temperature value is indeed 155 also in the "CanonCameraInfo Tag" with type unsigned char, which is equal to the value in unsigned or signed short in the Shot Info block (named CanonSi in Exiv2).
So this supports that the values in Exiftool are obtained by subtracting 128, to get the Camera Temperature value in degrees Celsius. I've only concluded that with a few samples. In the future I will make more tests with the same camera, putting it in the fridge (cold), and in warm temperatures, and see if the supposed Exiftool's conversion roughly holds.

And let's call it a day...

#19 - 01 Sep 2016 20:33 - Robin Mills

- % Done changed from 0 to 60
- Estimated time set to 10.00

You've worked really hard at this. Sure - take the rest of the day off!

In fixing the JPEG/ICC support today I've introduced a new bug #1220. I'm also going to call it a day!

#20 - 01 Sep 2016 20:41 - Raphael Attie

Thank you Robin. I'm glad I've crossed your road, and I thank you also for accepting my modest contribution. I've already learned a lot thanks to this issue. I hope I'll be able to help more in the future. This project, like Exiftool, Libraw, etc... are indeed really important to my dev project, hence my implication.

Have a good rest!

#21 - 02 Sep 2016 12:20 - Robin Mills

I hope you're having a nice day. I'm having a miserable one with #1220, so I've decided to have lunch and try again this afternoon. While eating, I had a thought about your comment concerning ***Exiftool ... converts it by offsetting by -128***.

Exiv2 reports metadata two different ways ***value*** or ***translated***. ***value*** is the vanilla data in the file. ***translated*** is a human representation of the data. The difference is demonstrated here: 1290 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pv --grep LensIDNumber
http://clanmills.com/Stonehenge.jpg
0x000c NikonLd3 LensIDNumber Byte 1 146
1291 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pt --grep LensIDNumber http://clanmills.com/Stonehenge.jpg
Exif.NikonLd3.LensIDNumber Byte 1 Sigma 18-250mm F3.5-6.3 DC OS Macro HSM
1292 rmills@rmillsmbp:~/gnu/exiv2/trunk $
The LensIDNumber in the Nikon Maker Note is a single byte 0..255. Exiv2 has lens tables which identify 146 as 'Sigma .....'. How can Nikon represent 1000s of different lenses with a single byte? Let's not start that discussion!

If you are able to complete your investigation and provide a patch, please respect the difference between "value" and "translated". So when you report "value", don't adjust it. When you report "translated", please adjust.

#22 - 02 Sep 2016 12:51 - Raphael Attie

Hi Robin,

I'm having a very relaxed day, in fact. Thank you for the clarification. I did notice all the codes for the various lenses with the canon file, but haven't looked at other files, Nikon files always scared me off. I acknowledge the difference between value and translated. I more or less work things the same way with my scientific data and FITS files, having keywords that stores the necessary calibration values for converting raw uncalibrated units to physical units.

As far as i'm concerned with this issue my investigation is completed on the "data analysis" on the Canon side for now, which is the brand I needed to support first. I will have a look at Nikon but I would like to complete the work for Canon first, and then i'll iterate with Nikon, and Sony, provided I have access to enough documentation to understand their raw file.
I don't mind writing an updated code, but how far does the patch need to go? I've seen your canonmn.cpp file, but don't see anything related to value/translated. Don't you have a schematic of your the structure of the files to help me see where I shall provide new input to support the addition of the Camera Temperature (or any future partially supported "unknown" tag)? I'm rather free this week end, so I'll use it to make more progress on discovering your code. But I'm getting married in mid-September (true, TRUE!!!!) so for several days starting end of next week I might be less available... so no promise but i'll try to finish before then, at least for Canon files.

Raphael

#23 - 02 Sep 2016 14:11 - Raphael Attie

Maybe i should target my questions to get me started without you explaining me everything since the Big Bang...

Can someone just tell me how this TagInfo from the Canon Shot Info Tag is structured?

```
TagInfo(0x0003, "MeasuredEV", N_("Measured EV"), N_("Measured EV"), canonSiId, makerTags, unsignedShort, 1, printSi0x0003)
```

I see different TagInfo declarations in "tags_int.hpp" but can't figure out to which one the above example corresponds:
(from tags_int.hpp)

```
//! Return the tag list for \em ifdId
const TagInfo* tagList(IfdId ifdId);
```

```
//! Return the tag info for \em tag and \em ifdId
const TagInfo* tagInfo(uint16_t tag, IfdId ifdId);
//! Return the tag info for \em tagName and \em ifdId
const TagInfo* tagInfo(const std::string& tagName, IfdId ifdId);
```

#24 - 02 Sep 2016 14:19 - Raphael Attie

or does it rather correspond to these definitions that I didn't notice when grepping TagInfo earlier...?

```
include/exiv2/tags.hpp:    struct TagInfo;
include/exiv2/tags.hpp:    typedef const TagInfo* (*TagListFct)();
include/exiv2/tags.hpp:    struct EXIV2API TagInfo {
include/exiv2/tags.hpp:        TagInfo(
include/exiv2/tags.hpp:    }; // struct TagInfo
include/exiv2/tags.hpp:        static const TagInfo* tagList(const std::string& groupName);
include/exiv2/tags.hpp:          @brief Constructor to create an Exif key from a TagInfo instance.
include/exiv2/tags.hpp:          @param ti The TagInfo instance
include/exiv2/tags.hpp:        ExifKey(const TagInfo& ti);
include/exiv2/tags.hpp:    //! Output operator for TagInfo
```

#25 - 02 Sep 2016 14:53 - Raphael Attie

Got it. For some reason my grep command with "TagInfo" did not grep the text "TagInfo::TagInfo" in the tags.cpp file at L2949:

```
TagInfo::TagInfo(
        uint16_t tag,
        const char* name,
        const char* title,
        const char* desc,
        int ifdId,
        int sectionId,
        TypeId typeId,
        int16_t count,
        PrintFct printFct
    )
        : tag_(tag), name_(name), title_(title), desc_(desc), ifdId_(ifdId),
          sectionId_(sectionId), typeId_(typeId), count_(count), printFct_(printFct)
    {
    }
```

I should be fine from here, but it will always save some time if you have some internal documentation on the inner design of the library that is not explained in the official API documentation. Otherwise, don't worry about it.

#26 - 02 Sep 2016 15:36 - Robin Mills

Apologies. The power has been off for a couple of hours. I wrote this 2 hours ago and couldn't post it. I think you've already discovered most of this for yourself!

Yes. You have indeed completed the research for Canon Cameras.

Congrats on your forth-coming wedding. I've been happily married to Alison for 42 years. The three happiest days of our lives were the days on which our 2 sons (now 39 and 40) were born and the day of our wedding. http://dev.exiv2.org/projects/exiv2/wiki/_Robin_Mills_

Here's how the value/translated feature is implemented. Have a look at the class Exiv2::Metadatum http://www.exiv2.org/doc/classExiv2_1_1Metadatum.html A Metadatum is a container for different types of data as defined in the EXIF specification on page 20. http://www.exiv2.org/Exif2-2.PDF It can be an array of bytes, unsigned, signed, rational and other data. The method "toString()" is called to obtain the "translated" value. When you add a record for a tag in xxxxxmn.cpp (a TagInfo record), you define a function to be called by toString(). The default **printValue** is usually sufficient. In your case you'll require a special function. Here's an example from canonmn.cpp

```
TagInfo(0x0009,
    "OwnerName", N_("Owner Name"), N_("Owner Name"), canonId, makerTags, asciiString, -1, printValue),
    TagInfo(0x000c, "SerialNumber", N_("Serial Number"), N_("Camera serial number"), canonId, makerTags, unsignedLong, -1, print0x000c),
```

You'll probably find your function useful for other tags, so give it meaningful name such as **printByteMinus128**.

How does the MakerNote decoder engage with the TagInfo records? I don't know because I've never had to investigate it. It's likely that you know lots about this from your work yesterday. Having identified the location in memory of the ShotInfo record, you're almost there. I believe Andreas implemented this and can help you, and Niels is knowledgable about this part of the code.

Nikon Files aren't difficult because the MakerNote is a TIFF. And Nikon's raw format .NEF is also a TIFF. You can see the structure by running my (beautiful and ingenious) command:

```
$ exiv2 -pR http://clanmills.com/Stonehenge.jpg
```

Down in the murky depths of the MakerNote lies a Nikon Tag which provides the information you want. Thanks to our good buddy Phil, a lot of this is documented on the ExifTool web site.

lensID conversion is a misery because we get weekly requests for new lenses. I wish the lensID "translated" feature was not part of Exiv2. There is an important new feature in v0.26 to enable the user to define their own "translated" value for a given manufacturer/lensID combination: http://dev.exiv2.org/projects/exiv2/wiki/Lens_Recognition_in_Exiv2_v026_(and_later)/

#27 - 02 Sep 2016 16:03 - Robin Mills

All of our documentation is on the project web-site. We have documentation about Manufacturers' MakerNotes. As you've seen, we have extensive class documentation which is generated by doxygen from comments in the code. We have numerous articles in the Wiki about the build, test suite, and the structure of image files.

The code is in good shape and well commented. And we have a good set of sample programs. We have a robust test suite. And we have a build server and publish the daily build for 15 platforms.

And we have Redmine. When I'm not asleep, or on vacation, I'll usually answer questions within a couple of hours. There is a search box in Redmine which will search the issue database, articles and forum discussions.

So ... all in all ... I think Exiv2 is a well organised project with good sources of information. Our documentation is accurate and up to date. The Exiv2 docs are easily better that what I found working in Silicon Valley for 14 years. The Silicon Valley way is to have a Wiki with a huge amount of out-of-date and factually wrong information. And to make matters worse, the Engineers are so pushed and stressed by the bosses that they seldom have time to talk to each other.

Ultimately the source of understanding and knowledge of the code comes from working on it and stepping with the debugger. To deal with "intellectual property" you need "intelligent people". I have no doubt that you are a smart guy and will come up to speed quickly on the structure of Exiv2.

#28 - 02 Sep 2016 19:42 - Raphael Attie

Robin, thank you for the nice words on both the wedding, sharing your experience and on the dev part. I had a close look at your wiki, I finally "meet" you! Well, I need to tell you that I fell in love with Scotland a few years ago when I was working in Glasgow for a few months in the astronomy dept. at the University. Believe it or not, I still have kilt in my cupboard offered to me by one of my colleague there. I was born on your national day too, January 25th, Burns' day! That made me famous in the whole department when I told them!! I'm originally from France, currently working in Brussels, but live in Lille (France, near the belgian border), I will move to the US for my new job at Nasa/GSFC in January where I'll carry on my research project on solar physics (solar flares, space weather, etc...), and I read that you live in the US too? That's great! From your great CV, you might become my new guru in software development!!

I'm gonna dive into the nicely structured documentation and codes of Exiv2 in the next days. I might pester you for clarifications on some aspects, but don't feel obligated to address everything I may ask. I can be stuck for days on something, then by asking someone else something unlocks in my head and after a few minutes I find the answer, happens all the time in fact! So I intend to learn the hard way anyway, walking my own way into the nice and inspiring piece of code that you guys wrote! Let's hope that all this is just the beginning of a great team work!

#29 - 02 Sep 2016 20:59 - Robin Mills

I did Astronomy 101 at Glasgow University in 1972. It was mostly taught by Archie Roy. Professor Sweet was the Regius Professor. I believe the Regius Professor today is John Brown and I think he was a grad student when I was an Engineering Undergraduate. Remarkably, I found spherical trig very useful when working on GPS systems in Silicon Valley in 2008-2012.

Alison and I are proud to be US Citizens. It's unlikely that we'll ever live in the States again as we're happy to be home and close to our family in England. We've been to all 50 States. We lived near NASA/Ames in Mountain View, CA and been several times to JSC/Mission Control in Houston, TX. Never been to GSFC. I'm sure you'll have a wonderful time. Nasa Engineering has a reputation for being very bureaucratic. I suspect Nasa Science will have some of the smartest people on the planet.

When I'm really stuck with something, I often write an email to somebody that I believe can help. I don't send the email. I have a cup of coffee. I pretend I am the person receiving the email and try to reply! Remarkably, I often can. The process of explaining the issue in words clears the brain. The debugger can be your worst enemy. All that information from stepping code can prevent you from asking the simple question "what is the issue?".

However, when I'm team leader, I have a rule for team members. "If your wheels are spinning for more than 2 hours, I want to know about it.". In a company, it's important to keep everybody moving. If an engineer is stuck, I want to get them moving again as quickly as possible.

So, I leave it to you to decide when to ask for help. If I can't help, you'll have to dig in and solve it yourself. However, some of my time will help you at lot at the beginning.

#30 - 03 Sep 2016 10:03 - Raphael Attie

Since this morning I've started diving into the documentation, cross-reading with the codes. I prefer too ask that question now:
Many times you refer to "component(s)" or "value components" or the "n-th component", although I don't recall such a term in general C++ programming and so I wonder if it's more local jargon to Exiv2? Or even more general than C++? (general programming maybe?) So what are you referring to by "component"? Does this relate to each of the bytes that can define a number or is this something else? Since these "components" and Exiv2::Value Class are one of the quanta around which many of the base classes are made (Metadatum and their parent, ExifDatum, etc...), it would be wise to be enlightened on this wording: "component(s)" and "n-th component". In addition, n-th component refers to some positional criterion and is that read from the "left" or from the "right", and thus, is this affected by endianness?

Thanks

#31 - 03 Sep 2016 10:11 - Raphael Attie

And just a self-rectification, I meant "Metadatum and its children ExifDatum, etc..." instead of "Metadatum and their parent (...)" (the latter made no sense).

#32 - 03 Sep 2016 11:48 - Robin Mills

I think you're referring to http://www.exiv2.org/doc/classExiv2_1_1Exifdatum.html and this documentation: TypeId typeId () const

Return the type id of the value.

const char * typeName () const

Return the name of the type.

long typeSize () const

Return the size in bytes of one component of this type.

long count () const

Return the number of components in the value.

long size () const

Return the size of the value in bytes.

I believe this is Exiv2 speak. This was in place before I joined the project 8 years ago. Have a look at the Exif spec page 20. It defines Exif data as a structure with a type field, count and data. You can see these being reported with this command: 594 rmills@rmillsmbp:~/gnu/exiv2/trunk $

exiv2 -pa --grep Date/i --grep Comment/i --grep GPS http://clanmills.com/Stonehenge.jpg

```
Exif.Image.DateTime                  Ascii      20  2015:07:16 20:25:28
Exif.Photo.DateTimeOriginal          Ascii      20  2015:07:16 15:38:54
Exif.Photo.DateTimeDigitized         Ascii      20  2015:07:16 15:38:54
Exif.NikonWt.DateDisplayFormat       Byte        1  Y/M/D
Exif.Photo.UserComment               Undefined  44
Exif.Image.GPSTag                    Long        1  4060
Exif.GPSInfo.GPSVersionID            Byte        4  2.3.0.0
Exif.GPSInfo.GPSLatitudeRef          Ascii       2  North
Exif.GPSInfo.GPSLatitude             Rational    3  51deg 10.69690'
Exif.GPSInfo.GPSLongitudeRef         Ascii       2  West
Exif.GPSInfo.GPSLongitude            Rational    3  1deg 49.59840'
Exif.GPSInfo.GPSAltitudeRef          Byte        1  Above sea level
Exif.GPSInfo.GPSAltitude             Rational    1  97 m
Exif.GPSInfo.GPSTimeStamp            Rational    3  14:38:55.9
Exif.GPSInfo.GPSSatellites           Ascii       3  09
Exif.GPSInfo.GPSMapDatum             Ascii      17  WGS-84
Exif.GPSInfo.GPSDateStamp            Ascii      11  2015:07:16
Xmp.xmp.ModifyDate                   XmpText    25  2015-07-16T20:25:28+01:00
595 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

The land of metadata is populated with different creatures (types) who roam in collection of 1 or more (count) The types are: 1 = BYTE An 8-bit unsigned integer

2 = ASCII An 8-bit byte containing one 7-bit ASCII code. The final byte is terminated with NULL

3 = SHORT A 16-bit (2-byte) unsigned integer

4 = LONG A 32-bit (4-byte) unsigned integer

5 = RATIONAL Two LONGs. The first LONG is the numerator and the second LONG expresses the denominator

7 = UNDEFINED An 8-bit byte that can take any value depending on the field definition

9 = SLONG A 32-bit (4-byte) signed integer (2's complement notation)

10 = SRATIONAL Two SLONGs. The first SLONG is the numerator and the second SLONG is the denominator

Exiv2 has extended the types when support was added for Xmp. "Undefined", "Byte" and "Ascii" are effectively the same, however "Ascii" is intended for printing and "Byte" is intended for binary. MakerNotes are often "Undefined".

DateTime "Ascii" of count 20. Correct! **2016-09-03 12:06:43** is 20 ascii bytes including the \0 terminator. Regrettably Ascii dates do not include time-zone information. *(how did the Exif committee cook up that nonsense?)*

Exif does not have a floating point number. It uses Rational which are a pair of integers: numerator/denominator. So GPSLatitude is expressed by three Rationals (degrees, minutes, seconds). You can see the 'vanilla' values with this command: 595 rmills@rmillsmbp:~/gnu/exiv2/trunk $

```
exiv2 -pv --grep Latitude http://clanmills.com/Stonehenge.jpg
0x0001 GPSInfo      GPSLatitudeRef              Ascii       2  N
0x0002 GPSInfo      GPSLatitude                 Rational    3  51/1 106969/10000 0/1
596 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

LatitudeRef is a two byte ascii field. Null terminated as you can see with this command: 596 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -ph --grep Latitude http://clanmills.com/Stonehenge.jpg

```
0x0001 GPSInfo      GPSLatitudeRef              Ascii       2   2
 0000  4e 00                                   N.

0x0002 GPSInfo      GPSLatitude                 Rational    3  24
 0000  33 00 00 00 01 00 00 00 d9 a1 01 00 10 27 00 00  3............'..
 0010  00 00 00 00 01 00 00 00                          ........

597 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

Now you answer the question: What does count(), type() etc tell you?

////////////////////////////

Notice that different types are used by different tags. GPSTimeStamp is represented by 3 Rationals and not 20 ascii characters. Look's like UTC to me. So, if your camera has a GPS (and my Nikon 5300 does), you can deduce the time-zone: 609 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pa -g TimeStamp -g DateStamp -g Original http://clanmills.com/Stonehenge.jpg

```
Exif.Photo.DateTimeOriginal                 Ascii      20  2015:07:16 15:38:54
Exif.Photo.SubSecTimeOriginal               Ascii       3  00
Exif.GPSInfo.GPSTimeStamp                   Rational    3  14:38:55.9
Exif.GPSInfo.GPSDateStamp                   Ascii      11  2015:07:16
610 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

None the less, you can see the GPS people and the DateTime people have a different idea about how to represent time. Confusing? It's software. It's always confusing.

// More discussion translated values:

And of course you can see the difference between the 'value' of GPSLatititude and 'translated'. Look in src/tags.cpp TagInfo(0x0002,

```
"GPSLatitude", N_("GPS Latitude"),
        N_("Indicates the latitude. The latitude is expressed as three "
        "RATIONAL values giving the degrees, minutes, and seconds, "
        "respectively. When degrees, minutes and seconds are expressed, "
        "the format is dd/1,mm/1,ss/1. When degrees and minutes are used "
        "and, for example, fractions of minutes are given up to two "
        "decimal places, the format is dd/1,mmmm/100,0/1."),
        gpsId, gpsTags, unsignedRational, 3, printDegrees),
```

And the magic is to translate this is: std::ostream& printDegrees(std::ostream& os, const Value& value, const ExifData*)

```
{
    std::ios::fmtflags f( os.flags() );
    if (value.count() == 3) {
        std::ostringstream oss;
        oss.copyfmt(os);
        static const char* unit[] = { "deg", "'", "\"" };
```

```
        static const int prec[] = { 7, 5, 3 };
        int n;
        for (n = 2; n > 0; --n) {
            if (value.toRational(n).first != 0) break;
        }
        for (int i = 0; i < n + 1; ++i) {
            const uint32_t z = (uint32_t) value.toRational(i).first;
            const uint32_t d = (uint32_t) value.toRational(i).second;
            if (d == 0)
            {
                os << "(" << value << ")";
                os.flags(f);
                return os;
            }
            // Hack: Need Value::toDouble
            double b = static_cast<double>(z)/d;
            const int p = z % d == 0 ? 0 : prec[i];
            os << std::fixed << std::setprecision(p) << b
               << unit[i] << " ";
        }
        os.copyfmt(oss);
    }
    else {
        os << value;
    }
    os.flags(f);
    return os;
} // printDegrees
```

#33 - 04 Sep 2016 13:36 - Raphael Attie

I've studied in more depth the Exif PDF. Regarding my question above, I'd like to ask again clarification on the wording regarding "component". Indeed, it is used in Exifdatum documentation in:

> long typeSize () const
> Return the size in bytes of one component of this type.
>
> long count () const
> Return the number of components in the value.

But in the Exif specifications, we speak of "pixel components" or "number of components per pixel". See p.24 (= p.18 in the PDF footer).

Does that word describe the same thing in both cases?

Thanks

#34 - 04 Sep 2016 16:40 - Robin Mills

The documentation is correct. A metadatum is arranged as follows:

```
+--------+-------+-------+-----+-----+...
|  Tag  | Type | Count |  v1 |  v2 |...
+--------+-------+-------+-----+-----+...
```

v1, v2 can be 1,2,4 or 8 bytes. 1 byte for ASCII, UNDEFINED, BYTE, 2 bytes for SHORT, 4 bytes for LONG, SLONG and 8 bytes for RATIONAL, SRATIONAL.

//////////

On page 24/-18- the spec is discussing SamplesPerPixel (Tag == 0x0115) and says the default is:

```
+--------+-------+-------+-----+-----+-----+
| 0x0115 |   3   |   3   | 8 | 8 | 8 |
+--------+-------+-------+-----+-----+-----+
```

Type SHORT=3. Count = 3. v1=8, v2=8,v3=8

The word *component* is regrettably overloaded. In the metadatum structure *component* refers to individual parts of the value such as v1 or v2.

The expression *The number of components per pixel.* is referring the number of bits per pixel in the image. The default is 3 colours of 8 bits / pixel.

Exiv2 reports data found in an image. So when a default is implied, but not actually present in the file, it is not reported. You won't find *SamplesPerPixel* very often because the default has been chosen for RGB images with 24 bits/pixel used in most commercial cameras.

#35 - 06 Sep 2016 15:55 - Raphael Attie

- File Exiv2_Explained_V1.pdf added

Don't mind this for now. I'm uploading a document in-progress for understanding the subset of the framework starting from file input and that will go down to the reading and output of the Camera Temperature in the case of Canon CR2 files. I might refer to this later if I have questions.

#36 - 08 Sep 2016 15:22 - Raphael Attie

- File Exiv2_Explained_V2.pdf added

Updating the "study-slides".

#37 - 09 Sep 2016 15:21 - Raphael Attie

Robin Mills wrote:

> *Here's how the value/translated feature is implemented. Have a look at the class Exiv2::Metadatum http://www.exiv2.org/doc/classExiv2_1_1Metadatum.html A Metadatum is a container for different types of data as defined in the EXIF specification on page 20. http://www.exiv2.org/Exif2-2.PDF It can be an array of bytes, unsigned, signed, rational and other data. The method "toString()" is called to obtain the "translated" value. When you add a record for a tag in xxxxxmn.cpp (a TagInfo record), you define a function to be called by toString(). The default* ***printValue*** *is usually sufficient. In your case you'll require a special function. Here's an example from canonmn.cpp[...]You'll probably find your function useful for other tags, so give it meaningful name such as* ***printByteMinus128****./*

Exifdatum::toString() does not seem to use the aforementioned **translated** value.

```
TagInfo(0x000c, "CameraTemperature", N_("Sensor temperature"), N_("Temperature of electronics around sensor"), canonSiId, makerTags,
unsignedShort, 1, printValueMinus128)


//! Print the tag value minus 128
  std::ostream& CanonMakerNote::printValueMinus128(std::ostream& os, const Value& value, const ExifData*)
  {
    return os << value.toLong() - 128;
  }
```

The Exiv2 command line app does output the translated value when used in the command line:

```
Exif.CanonSi.CameraTemperature              Short       1  27
```

But still prints the non-translated value when called from Exifdatum:

```
Exiv2::Exifdatum data = exifData["Exif.CanonSi.CameraTemperature"];
QString dataValue = QString::fromStdString(data.toString());
qDebug() << "Exif.CanonSi.CameraTemperature = " << dataValue;


Exif.CanonSi.CameraTemperature =  "155"
```

#38 - 09 Sep 2016 16:03 - Robin Mills

Well, I'm obviously wrong. Step into the toString() code with the debugger and find out what's going on!

#39 - 09 Sep 2016 19:01 - Raphael Attie

- File Exiv2_Explained_V3.pdf added

Nailed it. But stepping only confirms hat toString() is indeed not related to any of this. So I pulled out my last study-slide (see update, V3) , slide 3, bottom.
I think you meant to use Exifdatum::print(). That's the one that's sniffing into TagInfo to get whatever printing function was defined:

```
std::ostream& Exifdatum::write(std::ostream& os, const ExifData* pMetadata) const
  {
    if (value().count() == 0) return os;
    PrintFct fct = printValue;
    const TagInfo* ti = Internal::tagInfo(tag(), static_cast<IfdId>(ifdId()));
    if (ti != 0) fct = ti->printFct_;
    return fct(os, value(), pMetadata);
  }
```

Using the value() also does not help. So Exifdatum::print is the (only?) way to go:

```
Exiv2::Exifdatum data = exifData["Exif.CanonSi.CameraTemperature"];
QString dataValue = QString::fromStdString(data.toString());
QString dataValue2 = QString::fromStdString(data.value().toString());
QString dataValue3 = QString::fromStdString(data.print());
qDebug() << "Exif.CanonSi.CameraTemperature = " << dataValue;
qDebug() << "Exif.CanonSi.CameraTemperature = " << dataValue2;
qDebug() << "Exif.CanonSi.CameraTemperature = " << dataValue3;
```

```
Exif.CanonSi.CameraTemperature =  "155"
Exif.CanonSi.CameraTemperature =  "155"
Exif.CanonSi.CameraTemperature =  "27"
```

So I think we're finally getting to the bottom of that issue.
I'll post the updated .cpp and .hpp soon.

The rest is about digging into the other Nikon and Sony raw files to check, like I did with Canon files, to get the temperature tag, and figure out the conversion.

#40 - 09 Sep 2016 21:10 - Raphael Attie

- File canonmn_int.hpp added
- File canonmn.cpp added

Attaching updated canonmn.cpp and header canonmp_int.hpp supporting Camera Temperature of Canon files.

#41 - 09 Sep 2016 21:26 - Robin Mills

- Target version changed from 0.28 to 0.26

This is really great work, Dr Attie. Well done. And yes, you've unpuzzled my confusion about print() and toString().

Here's the way forward. Do you have a test file? Run your test file with/without your change to show the effect. When you submit the change, can you also update the test harness. Ben fixed #825 earlier this week and I added the update to the test harness. Take a look at what was done. This is similar.

I've discovered a problem in the CRW support down inside the enchanted decoding/encoding garden #1124. I've asked Andreas for code-review as I don't know anything about that code. One day, you'll be our makernote/tiff expert.

Great Work, Raphael. So happy that you're contributing to Exiv2.

#42 - 11 Sep 2016 09:16 - Robin Mills

Raphael

You have the right idea here. I don't think you've totally solved this. Here's some output using files in test/data. Revision 4478 536

```
rmills@rmillsmm:~/gnu/exiv2/t4478 $ exiv2 -pa --quiet --grep temp/i test/data/*canon*jpg
test/data/exiv2-canon-eos-20d.jpg  Exif.CanonPr.ColorTemperature          SShort      1  5200
```

537 rmills@rmillsmm:~/gnu/exiv2/t4478 $

With your patch: 541 rmills@rmillsmm:~/gnu/exiv2/t4478 $ exiv2 -pv --quiet --grep temp/i test/data/*canon*jpg

test/data/exiv2-canon-eos-20d.jpg 0x000c CanonSi CameraTemperature Short 1 0

test/data/exiv2-canon-eos-20d.jpg 0x0009 CanonPr ColorTemperature SShort 1 5200

test/data/exiv2-canon-eos-300d.jpg 0x000c CanonSi CameraTemperature Short 1 0

test/data/exiv2-canon-eos-d30.jpg 0x000c CanonSi CameraTemperature Short 1 0

test/data/exiv2-canon-powershot-a520.jpg 0x000c CanonSi CameraTemperature Short 1 0

test/data/exiv2-canon-powershot-s40.jpg 0x000c CanonSi CameraTemperature Short 1 0

542 rmills@rmillsmm:~/gnu/exiv2/t4478 $

542 rmills@rmillsmm:~/gnu/exiv2/t4478 $ exiv2 -pa --quiet --grep temp/i test/data/*canon*jpg

test/data/exiv2-canon-eos-20d.jpg Exif.CanonSi.CameraTemperature Short 1 -128

test/data/exiv2-canon-eos-20d.jpg Exif.CanonPr.ColorTemperature SShort 1 5200

test/data/exiv2-canon-eos-300d.jpg Exif.CanonSi.CameraTemperature Short 1 -128

test/data/exiv2-canon-eos-d30.jpg Exif.CanonSi.CameraTemperature Short 1 -128

test/data/exiv2-canon-powershot-a520.jpg Exif.CanonSi.CameraTemperature Short 1 -128

test/data/exiv2-canon-powershot-s40.jpg Exif.CanonSi.CameraTemperature Short 1 -128

543 rmills@rmillsmm:~/gnu/exiv2/t4478 $

As you can see, you are "picking up" Exif.CanonSi.CameraTemperature". It's reporting -128 when the data in the file is a Short == 0. I think a value of "-128" is an OK default as, unless the camera is on board a space craft, it's unlikely that would be the correct temperature. The existing behaviour to ignore the key in that state is preferable.

I suggest we do a couple of things:
1) Do you have a test file with "good" temperature data that we can add to the test suite?
2) Discuss this with Andreas. I'm sure it's possible to prevent the key being added to ExifData during metadata decoding. By the time printValueMinus128() is called, ExifData is "const" and cannot be modified. More importantly, it must not be modified by a print function.

In Phil's documentation he says of that tag: ***newer EOS models only***
http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/Canon.html

Somewhere in the pipeline between jpgimage/readMetadata() and your printValueMinus128() the code can decide "Yes: CameraTemperature is valid" and add it to exifData, or "No: CameraTemperature is not valid so ignore it".

As we have discussed, I don't know anything about the TiffVisitor and MakerNote classes. Ask Andreas to help you. And we have to ask Andreas to grant you write access to the SVN repository. If we don't hear from Andreas, I will submit your patch later this week.

#43 - 11 Sep 2016 14:03 - Raphael Attie

Indeed, my text in the description on the current issue only talks about "Modern DSLRs released the last years". I agree this has to be accounted for somewhere so the user does not think he actually has temperature data. In the examples you have provided, I know for a fact that these DSLRs do not have a digital thermometer anywhere. Values of -128, to me, are flag values covering for the absence of such data.
Forget the color temperature, this has nothing to do with the issue. Hence my description telling this is the temperature of the "sensor electronics".

So, we could use the flag value of -128, that are simply telling the data values are bogus, and maybe we act upon on this in the print function (that would call the printValueMinus128 among other tests).
Finally, I intended to get several raw CR2 images from several "recent" canon DSLRs and check we still have realistic data. Best case scenario is that either we have the flag value of -128, or we have a true temperature measurement. What do you say?

#44 - 11 Sep 2016 14:38 - Robin Mills

I'm confident that we can detect the -128 and remove this "old" metadata from ExifData, So we only report "bona fide" Exif.CanonSi.CameraTemperature.

Sridhar is working on something similar #1203.

When Andreas comes back to you, please try to engage him about this because I'm very overloaded.

#45 - 11 Sep 2016 14:41 - Raphael Attie

Robin Mills wrote:

> *I'm confident that we can detect the -128 and remove this "old" metadata from ExifData, So we only report "bona fide" Exif.CanonSi.CameraTemperature.*
>
> *Sridhar is working on something similar #1203.*
>
> *When Andreas comes back to you, please try to engage him about this because I'm very overloaded.*

Agreed. Ive already started setting up a more exhaustive series of DSLR raw files released across the last years. Shouldn't take me too long to check this out reliably and exhaustively.

#46 - 11 Sep 2016 15:21 - Robin Mills

Now Raphael, we agreed when your brought up this issue that it couldn't be got ready for v0.26. That was 3 weeks ago and you and Ben and Sridhar are putting very heavy demands on my time. And I'm trying to get to v0.26. I've got to ask you to relax about this and let me get to "feature complete v0.26". This might make it into v0.26, however it may not. There will be no v0.26 if I have no time to work on it.

#47 - 11 Sep 2016 15:55 - Robin Mills

- Target version changed from 0.26 to 0.28
- % Done changed from 60 to 30
- Estimated time changed from 10.00 to 30.00

I was too optimistic when I set the target to 0.26. I've changed it back to v0.27.

At the moment, we're only printing the value. We haven't investigated the business of writing that value. We could make Exif.CanonSi.CameraTemperature a read only value. However, in your telescope work, you may have lab notes that enable you set the temperature correctly in an image. There's more to this issue than we thought and we should defer it to v0.27.

#48 - 15 Sep 2016 11:47 - Sridhar Boovaraghavan

Hi Dr. Attie,

I have been in hibernation for a while and am jumping in at the very end without possibly understanding your issue completely.

There seems to be two separate issues:

1. The signedShort value is interpreted as unsignedShort by exiv2. This is the same as what I reported in #1203.
2. The value -128 is a default that Canon populates and is meant to be ignored. This is the same as what I have tried fixing in #1206.

It would seems that Robin has deferred your patch to 0.27 because of unexplained effects that it has on the test suite - the same that happened with

my patch for #1206.

I suggest that you look at the code that I submitted for #1206 (possibly r4438, not positive) and see if that can be adapted to fix the temperature issue. Now, the code that I had written was mostly concentrating on the reading side and it could have/create issues when data is written back. If that is indeed the case, then augmenting the code to write back the default values that we ignored while "reading" should not be that difficult with the framework that has been established to store default values.

I am not back in exiv2 "development" yet because of other personal issues that have taken my time, but I would like to look at how you tried to solve this problem, even though the fix seems to be specific to this case (again, not having delved into the code). My fix could be more generic.

Thanks,
Sridhar

#49 - 15 Sep 2016 12:36 - Robin Mills

Sridhar

Thanks for stepping into this. Raphael is getting married tomorrow! So, I think he has other things on his mind this week!

Raphael and I have been discussing a visit to my home in England sometime in October. When he comes, we'll investigate this issue if he hasn't made progress. Perhaps the three of us can discuss it on Skype (or one of those messenger things). I'm planning to be in Chennai late November.

Good News about v0.26 feature complete. Very close to done - possibly today. Over the weekend for sure.

Robin

Files

CR2_Scrutinized.pdf	1.81 MB	01 Sep 2016	Raphael Attie
CR2_Scrutinized.pdf	1.81 MB	01 Sep 2016	Raphael Attie
CR2_Scrutinized_updated.pdf	1.81 MB	01 Sep 2016	Raphael Attie
Exiv2_Explained_V1.pdf	42.4 KB	06 Sep 2016	Raphael Attie
Exiv2_Explained_V2.pdf	55.5 KB	08 Sep 2016	Raphael Attie
Exiv2_Explained_V3.pdf	57 KB	09 Sep 2016	Raphael Attie
canonmn_int.hpp	6.49 KB	09 Sep 2016	Raphael Attie
canonmn.cpp	112 KB	09 Sep 2016	Raphael Attie