# EER file format documentation

***Release***

**M. Leichsenring**

Table of Contents

Chapter 1

# EER file format

A number of Thermo Fisher Scientific direct detection cameras have the ability to output EER encoded files which can be identified through the ".eer" file extension. Goal of this documentation is to describe the internal file format for these EER files. Note that interpretation of the data itself is outside the scope of this documentation.

As we introduce new cameras models, it is likely this will require changes the EER file format over time. This document provides a description of the TIFF tags used and how these should be applied to handle compatibility. Whenever possible python sample code will be provided.

## 1.1 BigTIFF

The EER files are encoded using the BigTIFF container format which is an extension of the original TIFF container format. BigTIFF format was defined by Aware Systems to support TIFF files larger than 4 GiB.

More information and a detailed description for the container file formats are available at:

- TIFF information and latest detailed specification
- BigTIFF extension

Several libraries are available which support reading (and writing) of BigTIFF files, such as the C++ based open source library LibTIFF. However, this documentation provides examples written in Python using the tifffile library. Note that the examples are written for readability instead of focusing on maximum processing performance.

## 1.2 Extensibility

The (Big)TIFF container format is defined to provide flexibility, readers should adhere to a set of requirements as specified in section 7 of the TIFF standard. The following requirements are the most important ones to ensure future extensibility of the file format:

- Entries with unknown tag numbers or field types must be ignored.
- Image file directories (IFDs) with an unexpected compression scheme must be skipped.

## 1.3 File structure

The following table provides an overview of the EER file structure. Each (Big)TIFF file can store one or more images stored using entries referred to as Image File Directories (IFDs).

Table 1.1. EER file structure

| IFD | | occurrence |
|---|---|---|
| Integrated (or final) image | | optional |
| | Integrated Image data | |
| | TIFF tag AcquisitionMetadata | once |
| | TIFF tag ImageMetadata | optional |
| EER frame | | repeated |
| | EER compressed frame data | |
| | TIFF tag AcquisitionMetadata | once |
| | TIFF tag FrameMetadata | optional |

As shown in the above table, a single EER file stores one or more EER frames and an optional integrated image. Additional TIFF tags within an IFD are used to describe the data including providing (optional) metadata. When applicable, entries in the file structure table provides links towards sections with detailed descriptions.

## 1.4 Example

The following python example uses the tifffile library to show the structure of an EER file. It shows the different files (referred to as IFDs or as "pages" by tifffile) and TIFF tags within each file.

Note that the example limits the number of lines output for readability. Furthermore, TIFF tags have been selectively filtered using the show_tags array to focus on the EER specific TIFF tags.

```
from tifffile import TiffFile

# iterate over tags in the specified TIFF file to show structure
def show_structure(filename, show_tags):
    with TiffFile(filename) as tif:
        for page in tif.pages:
            print("IFD: ", end="")
            opt_comma = ""
            for tag in page.tags:
                name = tag.name
                if tag.code == EER_METADATA_ACQUISITION:
                    name = "AcquisitionMetadata"
                elif tag.code == EER_METADATA_FRAME:
                    name = "FrameMetadata"
                elif tag.code == EER_METADATA_INTEGRATED:
                    name = "IntegratedImageMetadata"

                if tag.code in show_tags:
                    print(f"{opt_comma}{name}({tag.code})", end="")
                    opt_comma = ", "
            print()
```

Listing 1.1. EER file structure output

```
IFD: ImageWidth(256), ImageLength(257), Compression(259), Orientation(274),
AcquisitionMetadata(65001), FrameMetadata(65002)
IFD: ImageWidth(256), ImageLength(257), Compression(259), Orientation(274),
FrameMetadata(65002)
IFD: ImageWidth(256), ImageLength(257), Compression(259), Orientation(274),
FrameMetadata(65002)
...
IFD: ImageWidth(256), ImageLength(257), Compression(259), Orientation(274),
FrameMetadata(65002)
IFD: ImageWidth(256), ImageLength(257), Compression(259), Orientation(274),
FrameMetadata(65002)
IFD: ImageWidth(256), ImageLength(257), Compression(259), Orientation(274),
FrameMetadata(65002)
```

Chapter 2

# Integrated (or final) image

An EER file can contain an optional integrated (or final) image. This integrated image is only available when at the time of acquisition a flag was specified. When included, the integrated image is always the first IFD of the EER file and its presence can be detected using the compression tag. It will be set to "no compression" as opposed to EER frames which use one of the custom EER compression schemes, see EER compressed frame data.

The IFD for an integrated image contains one or more of the following TIFF tags:

Table 2.1. TIFF tags in integrated image IFD

| Name | Tag | Description |
|---|---|---|
| 256 | ImageWidth | Specifies the width of the integrated image in pixels. |
| 257 | ImageLength | Specifies the height of the integrated image in pixels. |
| 258 | BitsPerSample | The number of bits per channel. As the integrated image is considered a grayscale image, it will only have one channel and is typically set to 16 bits (or 2 bytes). |
| 259 | Compression | Specifies the compression scheme used, set to COMPRESSION_NONE (1) for the integrated image. |
| 274 | Orientation | Orientation of the image with respect to the rows and columns. Normally set to TOPLEFT (1) for the integrated image. |
| 278 | RowsPerStrip | The number of rows stored per data strip in the file. |
| 273 | StripOffsets | Specifies the offset within the TIFF file for the data within the individual strips. |
| 279 | StripByteCount | For each strip, the number of bytes stored as data. |
| 65001 | AcquisitionMetadata | Provides additional acquisition metadata in an XML formatted string, see TIFF tag AcquisitionMetadata for more details. |
| 65006 | ImageMetadata | Provides additional metadata in an XML formatted string applicable to the integrated image, see TIFF tag ImageMetadata for more details. |

## 2.1 Dose calculations

The integrated image is calculated by the image processing pipeline of the camera. It doesn't represent a single electron hit in the image as a value one, but instead represent it using a pixel "blob" to provide subpixel accuracy information. The total value for the "blob" is available n the "countsToElectrons" property of the metadata.

In order to perform calculation of total dose or dose rate related information, you need to access the acquisition metadata. It provides both the size of the pixel (in meters) based on the magnification settings at time of acquisition, and calculated total dose and dose rate values (without and without

coincidence compensation).

You can use the following formula to calculate the dose (without coincidence compensation):

```
calculatedDose = meanPixelValue * pixelValueToCameraCounts * countsToElectrons
```

## 2.2 Accessing integrated image properties

The following code example checks whether the first IFD in the file is an integrated image by checking the compression TIFF tag. If that is indeed the case, it retrieves both the acquisition and integrated image XML metadata and prints a number of fields from it.

Listing 2.1. example python code to access the integrated image with metadata

```
def retrieve_image_properties(self):
    # retrieve first IFD (page) and determine if this is an integrated image
    acqimage = self.file.pages[0]
    if acqimage.compression == TIFF_COMPRESSION_NONE:
        metadata_acq = check_tag_with_default(acqimage, EER_METADATA_ACQUISITION, None)
        vprint("  Acquisition Metadata:")
        parse_metadata(metadata_acq, ["acquisitionID", "cameraName",
                       "commercialName", "timestamp", "meanDoseRate", "totalDose"])

        metadata_img = check_tag_with_default(acqimage, EER_METADATA_INTEGRATED, None)
        vprint("  Integrated image Metadata:")
        parse_metadata(metadata_img, ["timestamp", "exposureTime", "binning",
                       "meanPixelValue", "pixelValueToCameraCounts", "countsToElectrons"])
    else:
        print("  File does not contain an integrated image")
```

The following examples provides the output of running the above example code on a EER file acquired using a Falcon C.

Listing 2.2. Output acquisition and integrated image metadata

```
Parsing EER file: examples/files/0.5e4kps_2e2kps_1s.eer
Found 238 EER frames
  Acquisition Metadata:
    acquisitionID = 20230523.171837.607
    cameraName = BM-Falcon
    commercialName = Falcon C
    meanDoseRate = 1.9219717956956817
    timestamp = 2023-05-23T16:18:38.589514+01:00
    totalDose = 1.9011491532611149
  Integrated image Metadata:
    binning = 1
    countsToElectrons = 0.013037
    exposureTime = 0.997208
    meanPixelValue = 144.216678
    pixelValueToCameraCounts = 1
    timestamp = 2023-05-23T16:18:38.589514+01:00
```

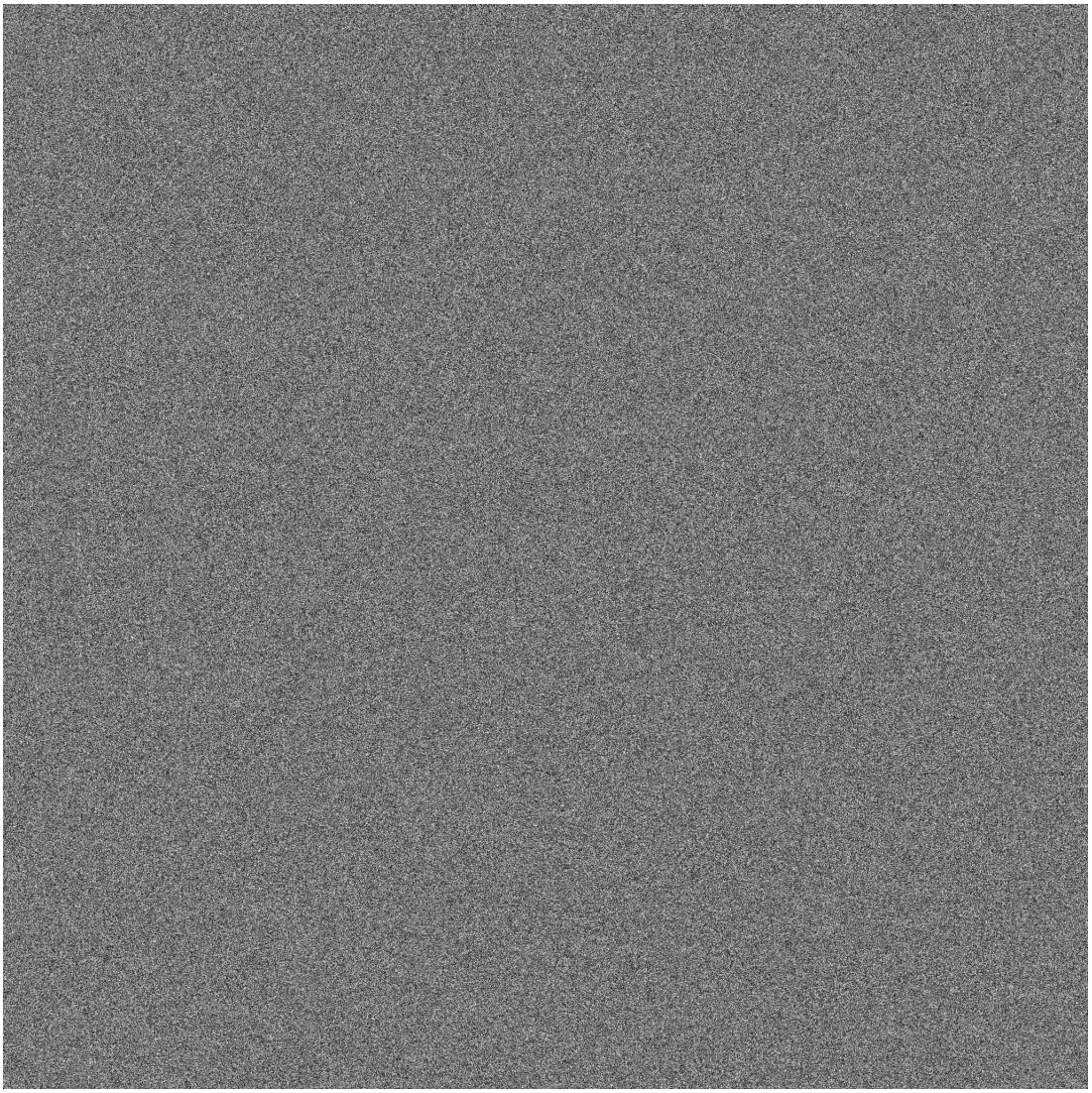
## 2.3 Retrieving the integrated image

The following example reads an EER file and checks whether it contains an integrated image. If that is indeed the case it will access the image array and provide some image statistics:

Listing 2.3. Retrieve the integrated image

```
Parsing EER file: examples/files/0.5e4kps_2e2kps_1s.eer
Found 238 EER frames
  Integrated image: (2048, 2048), dtype=uint16
  Image statistics: average 144.22, min 15, max 683
  Scaling image with factor 227 for visibility
```

As shown in the image statistics above, the average and maximum can be very low depending on the dose rate at time of acquisition. The following image has some additional image scaling applied for visibility.



# 3 EER frame

An EER file contains one or more image file directories (IFDs), with the IFD being either an EER frame or the (optional) integrated image. Typically an EER file will contain many EER frames, the number of EER frames determined during acquisition by the exposure time and camera frame rate.

An EER frame is stored as a bi-level image using a custom compression scheme. The EER frame itself might be stored in one or more strips. In case multiple strips are used, the entire frame can be reconstructed by concatenating the uncompressed data from each separate strip.

The IFD for an EER frame contains one or more of the following TIFF tags:

Table 3.1. TIFF tags in EER frame IFD

| Name | Tag | Description |
|---|---|---|
| 256 | ImageWidth | Specifies the width of the EER frame in pixels, without super resolution applied. |
| 257 | ImageLength | Specifies the height of the EER frame in pixels, without super resolution applied. |
| 262 | PhotometricInterpretation | Provides the color space for the image, set to BlackIsZero (=1) for EER. |
| 274 | Orientation | Orientation correction value for the EER frame to compensate for the camera mounting position and the EER encoder processing pipeline. |
| 259 | Compression | Specifies the compression scheme used, for an overview of possible values and decoding, see EER compressed frame data. |
| 65007 | PosSkipBits | The RLE code length to use in case compression is set to value 65002, see TIFF tag PosSkipBits for details. |
| 65008 | HorzSubBits | The number of horizontal sub-position bits in the EER stream. Only applicable when the compression value is set to 65002, see TIFF tag HorzSubBits for details. |
| 65009 | VertSubBits | The number of vertical sub-position bits in the EER stream. Only applicable when the compression value is set to 65002, see TIFF tag VertSubBits for details. |
| 278 | RowsPerStrip | The number of rows stored per data strip in the file. |
| 273 | StripOffsets | Specifies the offset within the TIFF file for the individual strips with the EER compressed data stream. |
| 279 | StripByteCount | For each strip, the number of bytes stored as (compressed) data. |
| 65001 | AcquisitionMetadata | Provides additional acquisition metadata in an XML formatted string, see TIFF tag AcquisitionMetadata for more details. Note that this information is only provided with the first EER frame in the file but applies to all others as well. |
| 65002 | FrameMetadata | Provides additional EER frame metadata in an XML formatted string, see TIFF tag FrameMetadata for more details. |

## 3.1 Accessing EER frame properties

The following python example reads a single eer frame from the BigTIFF file and accesses some items from both the acquisition and frame metadata. The acquisition metadata is only available on the first frame as shown in the example using the custom TIFF tag AcquisitionMetadata. The frame specific

metadata is provided with each frame using the TIFF tag FrameMetadata.

Listing 3.1. example python code to access an EER frame with metadata

```
def retrieve_frame_properties(self, frameno):
    # acquisition metadata is only available on the first frame in the file
    acqframe = self.file.pages[0]
    metadata_acq = check_tag_with_default(acqframe, EER_METADATA_ACQUISITION, None)
    if metadata_acq != None:
        vprint("  Acquisition Metadata:")
        parse_metadata(metadata_acq, ["acquisitionID", "cameraName",
                       "commercialName", "timestamp", "meanDoseRate", "totalDose"])

    #retrieve specific EER frame number and show properties (size, orientation)
    frame = self.get_frame(frameno)
    width, height = frame.imagewidth, frame.imagelength
    orientation = check_tag_with_default(frame, TIFF_ORIENTATION, 0)
    vprint(f"  EER frame: {width} x {height}, orientation: {orientation}")

    #retrieve frame metadata
    metadata_frame = check_tag_with_default(frame, EER_METADATA_FRAME, None)
    if metadata_frame != None:
        vprint("  EER frame Metadata:")
        parse_metadata(metadata_frame, ["frameID", "timestamp", "dose"])
```

Listing 3.2. Output acquisition and frame metadata

```
Parsing EER file: examples/files/3e4kps_12e2kps_1s.eer
Found 238 EER frames
  Acquisition Metadata:
    acquisitionID = 20230523.172157.127
    cameraName = BM-Falcon
    commercialName = Falcon C
    meanDoseRate = 16.364915398260688
    timestamp = 2023-05-23T16:21:57.602170+01:00
    totalDose = 16.187617904835932
Processing EER frame: 1
  EER frame: 2048 x 2048, orientation: 5
  EER frame Metadata:
    dose = 0.039978
    frameID = 1
    timestamp = 2023-05-23T16:21:57.606191+01:00
```

## 3.2 Accessing EER data stream

The EER data itself is stored within the IFD in one or more (data) strips. The number of strips can be calculated using the following formula (as specified in the TIFF standard):

```
StripsPerImage = floor ((ImageLength + RowsPerStrip-1) / RowsPerStrip)
```

The following example calculates the number of strips and reads the indivual data strips and passes them to the EER decoder library. As shown, each individual strip must be decoded as a separate EER stream representing the number of rows in the resulting (counted) image.

Listing 3.3. example python code to read and trigger decoding of EER data

```
def decode_eer_frame(self, frameno, image):
    frame = self.get_frame(frameno)

    #retrieve EER decoder settings and resulting image dimensions
    code_len, horz_subpix, vert_subpix = self.get_decoder_settings(frame)
    decoder = EerDecoder(code_len, horz_subpix, vert_subpix)
    width, height = frame.imagewidth, frame.imagelength

    rows_strip = frame.rowsperstrip
    strips_image = math.floor((height + rows_strip-1) / rows_strip)

    #retrieve strip strip data from file and perform EER decode
    fh = self.file.filehandle
```

```
    strip_offset = 0
    events = 0
    vprint(f"  EER data stored as {strips_image} strip(s):")

    for i in range(0, strips_image):
        offset = frame.dataoffsets[i]
        len_bytes = frame.databytecounts[i]
        vprint(f"    strip {i+1}: file offset = {offset}, length = {len_bytes} bytes")

        _ = fh.seek(offset)
        eer_data = fh.read(len_bytes)

        events += decoder.decode_eer_data(eer_data, width, rows_strip, image, strip_offset)
        strip_offset += rows_strip

    vprint(f"  found {events} events in frame, {events/(width*height):.4f} events/pixel")
    return events
```

When run on an EER file it will show the following output mentioning the EER decoder settings, number of strips found and for each the offset and length. As part of the EER stream decoding, it also shows the number of electron events recorded in the frame with the calculated average number of events per pixel.

Listing 3.4. EER frame decode output

```
Parsing EER file: examples/files/3e4kps_12e2kps_1s.eer
Found 238 EER frames
Processing EER frame: 1
  decoder: len 7 bits, subpixel 1+1 bits
  EER data stored as 1 strip(s):
    strip 1: file offset = 53118028, length = 189392 bytes
  found 167680 events in frame, 0.0400 events/pixel
```

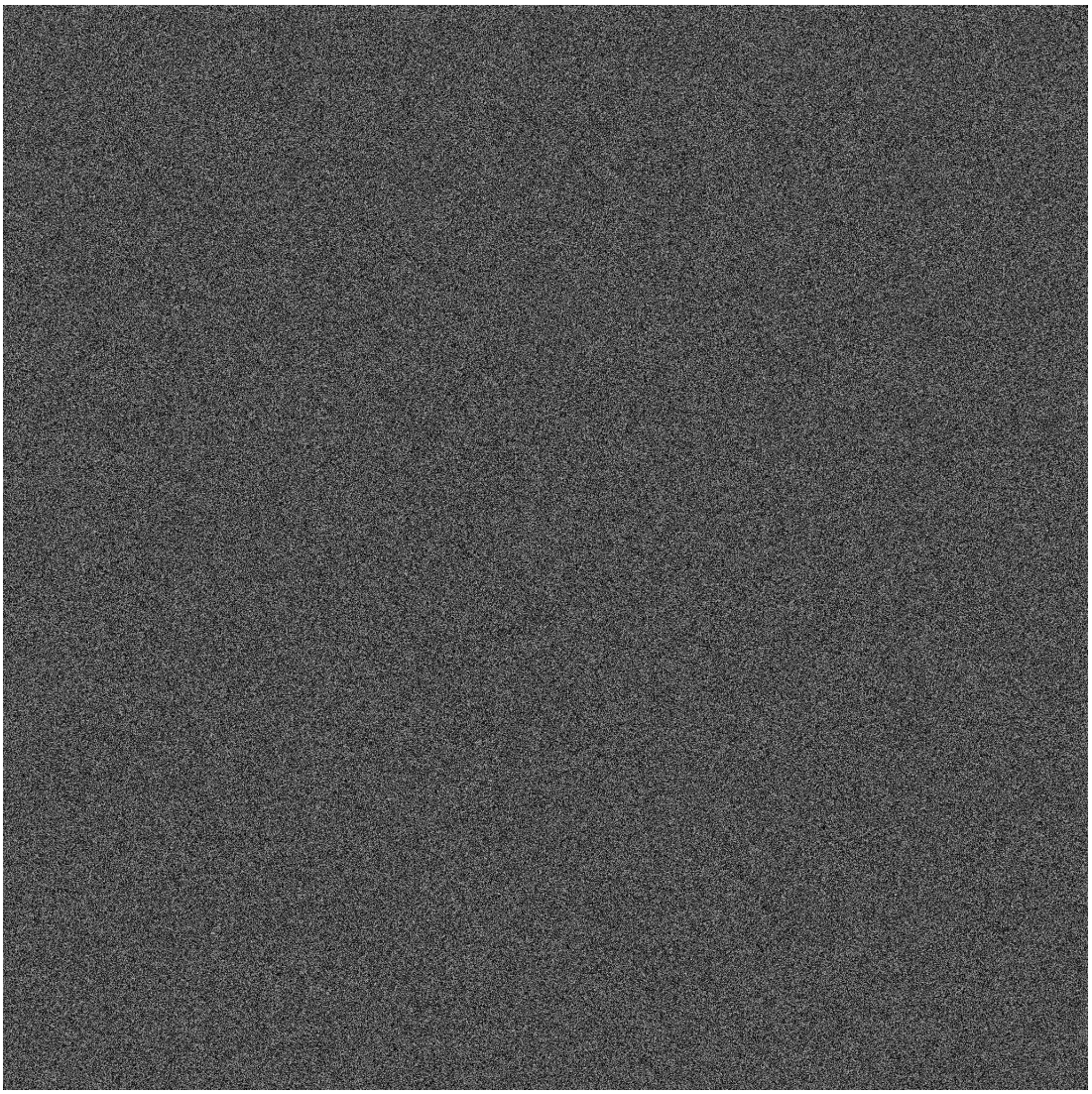
## 3.3 Combining multiple EER frames

The following example reads a subset of the available EER frames from a file and creates an integrated image at native resolution of the sensor. As the EER data was acquired with a Falcon C, the resulting output has a 2048 x 2048 resolution.

Listing 3.5. EER frame decode output

```
Parsing EER file: examples/files/3e4kps_12e2kps_1s.eer
Found 238 EER frames
  Output image: 2048 x 2048, orientation: 5
  Decoding EER frames, from 0 to 20
  Generating resulting image
  Total time: 6527.828ms
  Image statistics: average 0.80, min 0, max 8
  Scaling image with factor 160 for visibility
```

Due to the low number of electrons the average and maximum are typically low, as shown by the statistics in the decoder output above. The following image is the result of the above processing but with some additional scaling applied for visibility.



# 4 EER compressed frame data

The EER file format is used to provide an electron-event representation for cryoEM file storage while fully preserving the spatial and temporal resolution. It is supported by a number of direct detection cameras which support electron counting.

Each individual frame, as acquired by the sensor, is electron counted and the EER format preserve the location of each detected electron event. The encoding supports super-resolution by providing subpixel accuracy information when applicable.

## 4.1 Compression scheme

Each individual EER frame is stored in the EER TIFF file as a separate file/IFD. As this is a custom compression scheme, it uses custom values for the TIFF compression field (tag 259). The following table provides an overview of parameters for the EER decoder:

| Value | #RLE bits | #subpixel bits | Applicable camera(s) | Status |
|---|---|---|---|---|
| 65000 | 8 | 2 | Falcon 4 | obsolete |
| 65001 | 7 | 2 | Falcon 4, Falcon 4i | |
| 65002 | [PosSkipBits] | [HorzSubBits] and [VertSubBits] | Falcon C | |

The compression scheme 65002 has been introduced to increase flexibility for both EER encoding and decoding to support future cameras. Instead of using predefined values for the number of RLE and subpixel bits, these are provided by additional custom TIFF tags with the EER frame IFD. For more information see: TIFF tag PosSkipBits, TIFF tag HorzSubBits and TIFF tag VertSubBits.

## 4.2 Determine EER decoder settings

The following python example determines the EER decoder settings based on the custom compression value used. As shown in lines 21 to 23, the new custom EER decoder settings code requires reading additional custom TIFF tags for the individual EER decoder settings.

Listing 4.1. Example python code to determine EER decoder settings

```
#determine the EER decoder settings based on compression value
def get_decoder_settings(self, frame):
    if frame.compression == EER_COMPRESSION_FIXED82:
        code_len = 8
        horz_subbits = 2
        vert_subbits = 2
    elif frame.compression == EER_COMPRESSION_FIXED72:
        code_len = 7
        horz_subbits = 2
        vert_subbits = 2
    elif frame.compression == EER_COMPRESSION_VARIABLE:
        code_len = check_tag_with_default(frame, EER_DECODER_SKIPBITS, 7)
        horz_subbits  = check_tag_with_default(frame, EER_DECODER_HORZBITS, 2)
        vert_subbits = check_tag_with_default(frame, EER_DECODER_VERTBITS, 2)
    else:
        raise Exception(f"Unexpected EER compression value found: {frame.compression}")

    vprint(f"  decoder: len {code_len} bits, subpixel {horz_subbits}+{vert_subbits} bits")
    return code_len, horz_subbits, vert_subbits
```

As shown in the output below, typically all EER frames within a single file use the same EER decoder settings. Note however that this might change for future cameras or use cases and therefore you should check the decoder settings per EER frame stored in the file.

Listing 4.2. EER frame decode output

```
Parsing EER file: examples/files/falconc.eer
Found 238 EER frames
Processing EER frame: 1
  decoder: len 7 bits, subpixel 1+1 bits
```

## 4.3 EER stream decoding

The EER data stream should be considered a variable length bitstream which encodes detected electron events with sub pixel information. The EER encoded stream consists of bytes from which bits are grabbed from the LSB towards the MSB position.

The following example shows 7 bits for the code length (shown as "s"), followed by a single bit for both vertical and horizontal subpixel positions (shown as "v" and "h"). The byte values are shown at the top with the binary representation below it, with (partial) decoding results:

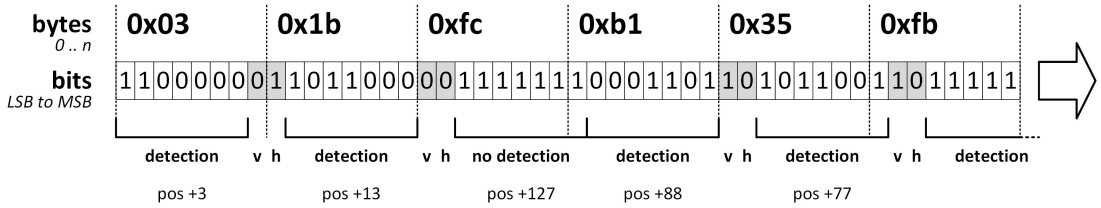



Figure 4.1. Example on how to interpret the EER bitstream

## 4.4 EER subpixel position

The output stream provides subpixel accuracy information for each detected electron event. With the number of subpixel information bits provided as part of the decoder settings, see above.

The subpixel position information is provided as both vertical and horizontal values. Note that the position itself is stored as an integer using two's complement registration. The 0,0 position is defined as the first pixel bottom/right from the center location. This is shown in the following image as the green dots for both 2x2 and 1x1 subpixel accuracy:

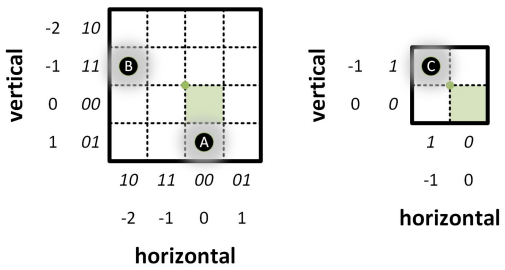



Figure 4.2. Subpixel information encoding, green square marks the 0,0 position

The above figure shows for each axis, both the integer and its two's complement value as stored in the EER bitstream. The left image uses 2 bits per direction, meaning a super-resolution scale of up to 4 times in each direction. So point A is represented by v=01, h=00 so "0100" in the bitstream and point B as "1110".

The right image uses 1 bit per direction, meaning a super-resolution scale of up to two. Point C would be represented by "11" in the bitstream.

Note: due to the nature of the two's complement encoding, you can also xor the sign bit to determine the absolute position using top,left as 0,0. This is also done it the example code to prevent integer position calculations.

## 4.5 Example EER decoding

The EER decoding process in shown in the Python code snippet below. You repeatedly grab pixel skip values from the bitstream. The value indicates the number of pixel positions to skip. If the value is below the maximum based on the number of bits, an electron event has been detected and the next bits in the stream provide the subpixel position. If the value represents the maximum, you skip the number of pixels and continue.

Listing 4.3. example Python code for EER decoding

```
def decode_eer_data(self, eer_data, width, height, image, offset):
    pos = 0
    last_pos = width * height
    max_skip = (2**self.code_len)-1
    events = 0
    vsignmask = self.vert_subpix
    hsignmask = self.horz_subpix

    eer_stream = BitStream(eer_data)
    while pos < last_pos:
        skip = eer_stream.get_bits(self.code_len)
        pos += skip
        if pos == last_pos:
            return events
        elif pos > last_pos:
            raise Exception(f"Incomplete EER frame, pos: {pos}, last: {last_pos}, bits left:
{eer_stream.no_bits_left()}")

        if skip == max_skip:
            continue
        else:
            ysub = eer_stream.get_bits(self.vert_subpix) ^ vsignmask
            xsub = eer_stream.get_bits(self.horz_subpix) ^ hsignmask

            #xpos = pos % width
            #ypos = (pos // width) + offset
            if image is not None:
                image[pos] += 1

        pos += 1
        events += 1
    return events
```

The loop should continue until the end position is equal to or above the total number of pixels based on the strip width and height. Note that something is wrong with the encoded EER file if the end position is exceeded as shown in the code example above. Some example output from the start of an EER frame and its interpretation is shown below.

Listing 4.4. EER frame decode output

```
Parsing EER file: examples/files/falconc.eer
Found 238 EER frames
Processing EER frame: 1
  decoder: len 7 bits, subpixel 1+1 bits
  get_bits(7): skip=3  , get_bits(1): xsub=0, get_bits(1): ysub=1 -- EVENT at 3,0 [0,1]
  get_bits(7): skip=13 , get_bits(1): xsub=1, get_bits(1): ysub=1 -- EVENT at 17,0 [1,1]
  get_bits(7): skip=127, MAX VALUE => no event
  get_bits(7): skip=88 , get_bits(1): xsub=1, get_bits(1): ysub=0 -- EVENT at 233,0 [1,0]
  get_bits(7): skip=77 , get_bits(1): xsub=1, get_bits(1): ysub=0 -- EVENT at 311,0 [1,0]
  get_bits(7): skip=127, MAX VALUE => no event
  get_bits(7): skip=7  , get_bits(1): xsub=1, get_bits(1): ysub=0 -- EVENT at 446,0 [1,0]
  get_bits(7): skip=81 , get_bits(1): xsub=1, get_bits(1): ysub=0 -- EVENT at 528,0 [1,0]
  get_bits(7): skip=127, MAX VALUE => no event
```

```
  get_bits(7): skip=127, MAX VALUE => no event
  get_bits(7): skip=127, MAX VALUE => no event
...
```

# 5 Custom TIFF tags

An EER encoded file contains several TIFF standard tags for example to define the image width and height, however includes several custom tags for metadata. The section provides an overview of the custom tags.

## 5.1 XML encoded metadata

Several custom tags encode metadata in an XML string. The XML string has the following formatting:

```
<metadata>
  <item name="[name]" unit="[unit]">[value]</item>
  <item name="[name]">[value]</item>
  …
</metadata>
```

Note that the unit attribute may be omitted for values that do not have a unit, as shown in the 3rd line of the example above.

## 5.2 TIFF tags Reference

The following list provides detailed information on the custom tags and its contents:

### 5.2.1 TIFF tag AcquisitionMetadata

**IFD** image

**code** 65001

**Name** AcquisitionMetadata

**Type** String (XML)

**count** 1

**default** none

Written as part of the first IFD specifying metadata items which apply to all images stored within the file. The metadata is provided as an XML string using the following XML tags:

Table 5.1. Acquisition Metadata

| Name | Unit | Description |
|---|---|---|
| acquisitionID | | Unique acquisition identifier (or label) used to identify the data set. |
| cameraName | | Camera name used for the acquisition e.g. BM-Falcon |
| commercialName | | Commercial name of the camera e.g. Falcon 4i |
| serialNumber | | Unique serial number of the sensor within the camera. Can be used to identify the camera or relate set of EER gain references. |
| sensorImageHeight | #pixels | The height of the sensor. |
| sensorImageWidth | #pixels | The width of the sensor. |
| sensorPixelSize.height | m | The height of a pixel compensated for magnification. |
| sensorPixelSize.width | m | The width of a pixel compensated for magnification. |

| | | |
|---|---|---|
| timestamp | ISO 8601 | Tme stamp of the acquisition with millisecond accuracy, including time zone information. |
| exposureTime | s | Duration of the acquisition. |
| numberOfFrames | | Total number of frames in the acquisition, corresponds to number of EER frames in the file. |
| meanDoseRate | e/pixel/s | Average dose rate for the acquisition, in number of electrons per pixel per second. |
| totalDose | e/pixel | Average total dose for the acquisition, in number of electrons per pixel. |
| coincidenceCompensatedDose | e/pixel | Average total dose for the acquisition, in number of electrons per pixel. The value is compensated for the known coincidence loss of the camera. |
| coincidenceCompensatedDoserate | e/pixel/s | Average dose rate for the acquisition, in number of electrons per pixel per second. The value is compensated for the known coincidence loss of the camera. |
| eerGainReference | | Name of the gain reference file which can be used to correct for variations in the sensitivity of the sensor. |

Listing 5.1. Example Acquisition metadata

```
Parsing EER file: examples/files/example.eer
Found 770 EER frames
Processing EER frame: 0
<metadata>
    <item name="acquisitionID">TEMApps_20220718_164035</item>
    <item name="cameraName">EF-Falcon</item>
    <item name="commercialName">Falcon 4</item>
    <item
name="eerGainReference">ImagesForProcessing/EF-Falcon/300kV/20220711_104630_EER_GainReference.gain</item>
    <item name="exposureTime" unit="s">3.2000000000000002</item>
    <item name="meanDoseRate" unit="e/pixel/s">7.5698134278181328</item>
    <item name="numberOfFrames">770</item>
    <item name="sensorImageHeight" unit="pixel">4096</item>
    <item name="sensorImageWidth" unit="pixel">4096</item>
    <item name="sensorPixelSize.height" unit="m">6.42429665e-10</item>
    <item name="sensorPixelSize.width" unit="m">6.42429665e-10</item>
    <item name="serialNumber">20-44-A11-G4H</item>
    <item name="timestamp">2022-07-18T15:40:36.141-08:00</item>
    <item name="totalDose" unit="e/pixel">24.198384735639088</item>
</metadata>
```

### 5.2.2 TIFF tag FrameMetadata

**IFD** EER frame

**code** 65002

**Name** FrameMetadata

**Type** String (XML)

**count** 1

**default** none

Table 5.2. EER frame metadata

| Name | Unit | Description |
|---|---|---|
| frameID | | Sequence number of the EER frame in the acquisition. |

| | | |
|---|---|---|
| orientation | | The TIFF orientation value (see TIFFTAG_ORIENTATION=274) to correct for the camera mounting and image processing pipeline. |
| timestamp | ISO 8601 | The time stamp at which the frame was acquired with millisecond accuracy, expressed using ISO8601 representation with timezone offset indication. |
| dose | e/pixel | The average dose per pixel in the frame as detected by the camera. |
| decompressionAlgorithmVersion | | Version of the EER encoding/decoding algorithm required to correctly process the EER data. |
| rleCodeLength | | The run-length-encoding code length |
| nrOfSubPixelPerDirection | | The number of sub pixels per pixel in width and height |
| pixelFormat | | The pixel format (?) |

Listing 5.2. Example EER frame metadata

```
Parsing EER file: examples/files/example.eer
Found 770 EER frames
Processing EER frame: 10
<metadata>
    <item name="decompressionAlgorithmVersion">V1</item>
    <item name="dose" unit="e/pixel">0.023866</item>
    <item name="frameID">10</item>
    <item name="nrOfSubPixelPerDirection">2</item>
    <item name="orientation">2</item>
    <item name="pixelFormat">EerEncoding</item>
    <item name="rleCodeLength">7</item>
    <item name="timestamp">2022-07-18T15:40:36.182-08:00</item>
</metadata>
```

### 5.2.3 TIFF tag ImageMetadata

**IFD** Integrated Image

**code** 65006

**Name** ImageMetadata

**Type** String (XML)

**count** 1

**default** none

Table 5.3. Integrated Image metadata

| Name | Unit | Description |
|---|---|---|
| timestamp | ISO 8601 | The time stamp at which the integrated image was acquired with millisecond accuracy, expressed using ISO8601 representation with timezone offset indication. |
| exposureTime | s | The exposure time of the integrated image. |
| numberOfFrames | | Total number of frames used to calculate the integrated image. |
| roi.top | | Top coordinate of the region of interest for the image in sensor coordinates, where 0 is the topmost pixel. |
| roi.left | | Left coordinate of the region of interest for the image in sensor coordinates, where 0 is the leftmost pixel. |

| | | |
|---|---|---|
| roi.bottom | | Bottom coordinate of the region of interest for the image in sensor coordinates. The value is exclusive, so for a 4096 x 4096 image the top coordinate will be 0 and the bottom coordinate will be 4096. |
| roi.right | | Right coordinate of the region of interest for the image in sensor coordinates. The value is exclusive, so for a 4096 x 4096 image the left coordinate will be 0 and the right coordinate will be 4096. |
| binning | | The amount of pixels that have been binned together. |
| darkCorrection | | A yes/no indication on whether the image is dark corrected |
| gainCorrection | | A yes/no indication on whether the image is gain corrected. |
| pixelValueToCameraCounts | | The conversion from a pixel value to a camera count |
| meanPixelValue | | The average pixel value of the final image |
| countsToElectrons | | The conversion from camera counts to electrons. The following formula can be used to calculate the dose for a pixel: pixelValue x pixelValueTo-CameraCounts x countsToElectrons |
| driftCorrectionInformation.driftCorrected | | A yes/no indication on whether drift correction was enabled. |
| driftCorrectionInformation.confidence | | A confidence indication for the drift vectors ranging from 0 to 1. |
| driftCorrectionInformation.clipping | | A yes/no indication on whether either vector was clipped. |
| driftCorrectionInformation.vectorXCoordinate | | The drift in X direction |
| driftCorrectionInformation.vectorYCoordinate | | The drift in Y direction |
| checksum | | An indication on whether the checksum was valid. |

Listing 5.3. Example Integrated image metadata

```
Parsing EER file: examples/files/0.5e4kps_2e2kps_1s.eer
Found 238 EER frames
<metadata>
    <item name="binning">1</item>
    <item name="checksum">Valid</item>
    <item name="countsToElectrons">0.013037</item>
    <item name="darkCorrection">Yes</item>
    <item name="driftCorrectionInformation.clipping">No</item>
    <item name="driftCorrectionInformation.confidence">0.000000</item>
    <item name="driftCorrectionInformation.driftCorrected">No</item>
    <item name="driftCorrectionInformation.vectorXCoordinate">0.000000</item>
    <item name="driftCorrectionInformation.vectorYCoordinate">0.000000</item>
    <item name="exposureTime" unit="s">0.997208</item>
    <item name="gainCorrection">Yes</item>
    <item name="meanPixelValue">144.216678</item>
    <item name="numberOfFrames">238</item>
    <item name="pixelValueToCameraCounts">1</item>
    <item name="roi.bottom">2048</item>
    <item name="roi.left">0</item>
    <item name="roi.right">2048</item>
    <item name="roi.top">0</item>
    <item name="timestamp">2023-05-23T16:18:38.589514+01:00</item>
</metadata>
```

### 5.2.4 TIFF tag PosSkipBits

**IFD** EER image

**code** 65007

**Name** PosSkipBits

**Type** short

**count** 1

**default** 7

**Description** Specifies the number of RLE bits to use by the EER decoder as the position skip count. Only applies when Compression value is set to 65002.

### 5.2.5 TIFF tag HorzSubBits

**IFD** EER image

**code** 65008

**Name** HorzSubBits

**Type** short

**count** 1

**default** 2

**Description** Specifies the number of bits in the EER stream which provide the horizontal subpixel position for an detected electron event. Only applies when Compression value is set to 65002.

### 5.2.6 TIFF tag VertSubBits

**IFD** EER image

**code** 65009

**Name** VertSubBits

**Type** short

**count** 1

**default** 2

**Description** Specifies the number of bits in the EER stream which provide the vertical subpixel position for an detected electron event. Only applies when Compression value is set to 65002.

Chapter 6

# Document version history

Table 6.1. Version History

| | Date | Changes | Author(s) |
|---|---|---|---|
| 1.0 | 30 Aug 2019 | Initial version | P. Bootsma |
| 2.0 | 02 Dec 2019 | Added 7-bit compression | A. Dumitrescu |
| 2.1 | 10 Dec 2019 | Added Final image directory | M. Balasubramanian |
| 2.2 | 17 Jan 2020 | Added acquisition metadata | A. Dumitrescu, P. Bootsma |
| 2.3 | 25 Feb 2021 | Added orientation | D. van der Steen |
| 2.4 | 1 April 2021 | Update acquisition metadata | L. Oosterhof |
| 2.5 | 4 April 2021 | Update Fields | D. van der Steen |
| 2.6 | 11 May 2021 | Sensor pixel size units changed to meters | G. Singla |
| 2.7 | 18 May 2021 | Update acquisition metadata | G. Singla |
| 2.8 | 20 May 2021 | Added camera name | D. van der Steen |
| 2.9 | 15 Sep 2021 | Removing camera type from metadata | G. Singla |
| 3.0 | 03 Mar 2023 | Added compression details | D. van der Steen |