Sequence files

The sequence file format, which describes files created by StreamPix while recording and saved to a file with the *.seq* extension, is described in the following lines:

A sequence file is made of a header section located in the first 1024 bytes. The header contains information pertaining to the whole sequence: image size and format, frame rate, number of images etc. Following the header, each images is stored and aligned to the disk sector size boundary. Please note that only the uncompressed sequence format is documented here, compressed sequences are handled in a different way.

Usually, pixels in the images are stored for top left to bottom right corner. Immediately following the image data comes 8 bytes, containing the absolute timestamp at which the image has been grabbed. The first 4 bytes are date and time and the last 2 bytes are the milliseconds. Assume, for instance, a sequence of 10 images of size 640 x 480 pixels in 8 bit monochrome in which the first image in the sequence file is at an offset of 1024 bytes.

Read 640 x 480 or 307200 bytes to get all the image pixels. Then read the next 32 bit (4 bytes) to get the timestamp in seconds, formatted according to the C standard *time_t data* structure. Read the next 16 bit (2 bytes) as an unsigned short to get the millisecond precision on the timestamp. Also, when using dedicated timing devices, the precision can be up to the microsecond. In those case, the microseconds are in the next 16 bits (2 bytes) immediatly after the milliseconds. Without timing devices, the microseconds will always be 0. Full timestamp information can be recombined as the *time_t data* converted to seconds plus the milliseconds read from the last 16 bit.

The next image will be at an offset of *HeaderSize* + *TrueImageSize*

The timestamp information for the second image will be at *HeaderSize* + *TrueImageSize* + *CImageInfo::ImageSizeBytes*

The Origin field is used in Pre/Post Recording and is the index of the frame received when the trigger occured.

Name	Content	File offset - size in bytes
long MagicNumber	Always 0xFEED	0 - 4
wchar_t Name[12]	Always "Norpix *seq*\n"	4 - 24

long Version	Sequence Header Version	28 - 4
long HeaderSize	Should always be 1024	32 - 4
BYTE Description[512]	User description	36 - 512
CImageInfo ImageInfo	See belows for a description of the CImageInfo struct	548 - 24
unsigned long AllocatedFrames	Number of frames allocated in the sequence	572 - 4
unsigned long Origin	Should be 0 if not Pre/Post recorded	576 - 4
unsigned long TrueImageSize	Number of bytes between the first pixel of each successive images	580 - 4
double FrameRate	Suggested Frame rate for playback (in fps)	584 - 8
long DescriptionFormat	The content of "Description" 0-UNICODE STRING 1-ASCII 2-DATA	592 - 4
BYTE Padding[428]	Unused bytes, reserved for future uses.	596 - 428

CImageInfo is a simple structure that can be read in the following :

```
struct CImageInfo
{
   unsigned long ImageWidth;       //Image width in pixel
   unsigned long ImageHeight;      //Image height in pixel
   unsigned long ImageBitDepth;    //Image depth in bits (8,16,24,32)
   unsigned long ImageBitDepthReal;//Precise Image depth (x bits)
   unsigned long ImageSizeBytes;   //Size used to store one image.
   eHImageFormat ImageFormat;      //See formats below
};

enum eHImageFormat
{
      H_IMAGE_UNKNOWN = 0,        //Unknown format
      H_IMAGE_MONO = 100,         //Monochrome Image (LSB)
      H_IMAGE_MONO_BAYER = 101,   //Raw Bayer Image (treated as H_IMAGE_MONO)
      H_IMAGE_BGR = 200,          //BGR Color Image
      H_IMAGE_PLANAR = 300,       //Planar Color Image
      H_IMAGE_RGB = 400,          //RGB Color Image
      H_IMAGE_BGRx = 500,         //BGRx Color Image
      H_IMAGE_YUV422 = 600,       //YUV422
      H_IMAGE_UVY422 = 700,       //UVY422
      H_IMAGE_UVY411 = 800,       //UVY411
      H_IMAGE_UVY444 = 900,       //UVY444

      //The following formats are only use in sequence headers.
      H_IMAGE_MONO_JPEG = 102,         //JPEG Compressed sequence
      H_IMAGE_MONO_BAYER_JPEG = 103,
      H_IMAGE_BGR_JPEG = 201,
      H_IMAGE_PLANAR_JPEG = 301,
```

```
H_IMAGE_RGB_JPEG = 401,
H_IMAGE_BGRx_JPEG = 501,
H_IMAGE_YUV422_JPEG = 601,
H_IMAGE_UVY422_JPEG = 701,
H_IMAGE_UVY411_JPEG = 801,
H_IMAGE_UVY444_JPEG = 901,

H_IMAGE_BGR555_PACKED_JPEG = 217,
H_IMAGE_BGR565_PACKED_JPEG = 218,

H_IMAGE_MONO_RLE = 104,
H_IMAGE_MONO_BAYER_RLE = 105,
H_IMAGE_BGR_PACKED_RLE = 202,
H_IMAGE_BGR_PLANAR_RLE = 302,
H_IMAGE_RGB_PACKED_RLE = 402,
H_IMAGE_BGRx_PACKED_RLE = 502,
H_IMAGE_YUV422_PACKED_RLE = 602,
H_IMAGE_UVY422_PACKED_RLE = 702,
H_IMAGE_UVY411_PACKED_RLE = 802,
H_IMAGE_UVY444_PACKED_RLE = 902,

H_IMAGE_BGR555_PACKED_RLE = 1003,
H_IMAGE_BGR565_PACKED_RLE = 1004,

H_IMAGE_MONO_HUFFMAN = 106,
H_IMAGE_MONO_BAYER_HUFFMAN = 107,
H_IMAGE_BGR_PACKED_HUFFMAN = 203,
H_IMAGE_BGR_PLANAR_HUFFMAN = 303,
H_IMAGE_RGB_PACKED_HUFFMAN = 403,
H_IMAGE_BGRx_PACKED_HUFFMAN = 503,
H_IMAGE_YUV422_PACKED_HUFFMAN = 603,
H_IMAGE_UVY422_PACKED_HUFFMAN = 703,
H_IMAGE_UVY411_PACKED_HUFFMAN = 803,
H_IMAGE_UVY444_PACKED_HUFFMAN = 903,
H_IMAGE_BGR555_PACKED_HUFFMAN = 1103,
H_IMAGE_BGR565_PACKED_HUFFMAN = 1104,

H_IMAGE_MONO_LZ = 108,
H_IMAGE_MONO_BAYER_LZ = 109,
H_IMAGE_BGR_PACKED_LZ = 204,
H_IMAGE_BGR_PLANAR_LZ = 304,
H_IMAGE_RGB_PACKED_LZ = 404,
H_IMAGE_BGRx_PACKED_LZ = 504,
H_IMAGE_YUV422_PACKED_LZ = 604,
H_IMAGE_UVY422_PACKED_LZ = 704,
H_IMAGE_UVY411_PACKED_LZ = 804,
H_IMAGE_UVY444_PACKED_LZ = 904,
H_IMAGE_BGR555_PACKED_LZ = 1203,
H_IMAGE_BGR565_PACKED_LZ = 1204,

H_IMAGE_MONO_RLE_FAST = 2104,
H_IMAGE_MONO_BAYER_RLE_FAST = 2105,
H_IMAGE_BGR_PACKED_RLE_FAST = 2202,
H_IMAGE_BGR_PLANAR_RLE_FAST = 2302,
```

```
      H_IMAGE_RGB_PACKED_RLE_FAST = 2402,
      H_IMAGE_BGRx_PACKED_RLE_FAST = 2502,
      H_IMAGE_YUV422_PACKED_RLE_FAST  = 2602,
      H_IMAGE_UVY422_PACKED_RLE_FAST  = 2702,
      H_IMAGE_UVY411_PACKED_RLE_FAST  = 2802,
      H_IMAGE_UVY444_PACKED_RLE_FAST  = 2902,

      H_IMAGE_BGR555_PACKED_RLE_FAST     = 2003,
      H_IMAGE_BGR565_PACKED_RLE_FAST  = 2004,

      H_IMAGE_MONO_HUFFMAN_FAST = 3106,
      H_IMAGE_MONO_BAYER_HUFFMAN_FAST = 3107,
      H_IMAGE_BGR_PACKED_HUFFMAN_FAST = 3203,
      H_IMAGE_BGR_PLANAR_HUFFMAN_FAST = 3303,
      H_IMAGE_RGB_PACKED_HUFFMAN_FAST = 3403,
      H_IMAGE_BGRx_PACKED_HUFFMAN_FAST = 3503,
      H_IMAGE_YUV422_PACKED_HUFFMAN_FAST  = 3603,
      H_IMAGE_UVY422_PACKED_HUFFMAN_FAST  = 3703,
      H_IMAGE_UVY411_PACKED_HUFFMAN_FAST  = 3803,
      H_IMAGE_UVY444_PACKED_HUFFMAN_FAST  = 3903,
      H_IMAGE_BGR555_PACKED_HUFFMAN_FAST = 3003,
      H_IMAGE_BGR565_PACKED_HUFFMAN_FAST = 3004,

      H_IMAGE_MONO_LZ_FAST = 4108,
      H_IMAGE_MONO_BAYER_LZ_FAST = 4109,
      H_IMAGE_BGR_PACKED_LZ_FAST = 4204,
      H_IMAGE_BGR_PLANAR_LZ_FAST = 4304,
      H_IMAGE_RGB_PACKED_LZ_FAST = 4404,
      H_IMAGE_BGRx_PACKED_LZ_FAST = 4504,
      H_IMAGE_YUV422_PACKED_LZ_FAST = 4604,
      H_IMAGE_UVY422_PACKED_LZ_FAST = 4704,
      H_IMAGE_UVY411_PACKED_LZ_FAST = 4804,
      H_IMAGE_UVY444_PACKED_LZ_FAST = 4904,
      H_IMAGE_BGR555_PACKED_LZ_FAST = 4003,
      H_IMAGE_BGR565_PACKED_LZ_FAST = 4004,

      H_IMAGE_BGR555_PACKED = 905,  // PhynxRGB
      H_IMAGE_BGR565_PACKED = 906,

// Only for > 8 bit per pixel, MSB align litle endian 10 bit: JIHGFEDC BA000000
      H_IMAGE_MONO_MSB = 112,

// Only for > 8 bit per pixel, MSB align
      H_IMAGE_MONO_BAYER_MSB = 113,

// Only for > 8 bit per pixel, MSB align big endian 10 bit: BA000000 JIHGFEDC
      H_IMAGE_MONO_MSB_SWAP = 114,

// Only for > 8 bit per pixel, MSB align
      H_IMAGE_MONO_BAYER_MSB_SWAP = 115,

// Only for > 8 bit per pixel, LSB align
      H_IMAGE_MONO_PPACKED = 121,
```

```
// Only for > 8 bit per pixel, LSB align
      H_IMAGE_MONO_BAYER_PPACKED = 122,

// Only for  10 bit per pixel, LSB align
      H_IMAGE_BGR10_PPACKED = 123,

// Only for  10 bit per pixel, LSB align, RRRRRRRR RR00GGGG GGGGGGBB BBBBBBBB
      H_IMAGE_BGR10_PPACKED_PHOENIX = 124,

// Only for  10 bit per pixel, LSB align, BBBBBBBB BB00GGGG GGGGGGRR RRRRRRRR
      H_IMAGE_RGB10_PPACKED_PHOENIX = 125,

// Only for > 8 bit per pixel, MSB align
      H_IMAGE_MONO_MSB_PPACKED = 131,

// Only for > 8 bit per pixel, MSB align
      H_IMAGE_MONO_BAYER_MSB_PPACKED = 132,

      H_IMAGE_BASLER_VENDOR_SPECIFIC = 1000,
      H_IMAGE_EURESYS_JPEG = 1001,
      H_IMAGE_ISG_JPEG = 1002
};
```