Appendix

File Formats

last update: June 2, 2009 (latest changes highlighted)

The following section describes the format of the proprietary file formats used in U-view.

Still Image File

File Type: binary
File Extension: dat
Open with: U-view

File Contents:

1. **File header**
2. **if file version >=7: block of 'recipe' data (128 Bytes)**
   if fileheader.attachedRecipeSize >0
   note: the block is of fixed size (128 Bytes!)
3. **Imageheader**
4. **if image version >=5: block of markup data (128 Bytes)**
   if imageheader.attachedMarkupSize > 0
   note: the block is of fixed size (128 Bytes!)
5. **Width x Height x Pixel in 2 Bytes (if Bits PerPixels=16)**

Description of data structures used:

File header

current FILEVERSION = 7

File header: ***size in Bytes: 104***

```
struct UKFileHeader{   // sizeof(UKFileHeader):104
        char id[20];                    // 20 Bytes
        short size;                     // 2 Bytes
        short version;                  // 2 Bytes
        short BitsPerPixel;             // 2 Bytes
                                        (=16 for Sensicam, this refers to the storage,
                                              not the acquisition, which is 12 bits)
        // 6 Bytes inserted to get to next 8 byte boundary:
        // why?: because LONGLONG needs to start at 8 byte boundary
        LONGLONG spare;                 // 8 Bytes
        short ImageWidth,ImageHeight;   // 4 Bytes
        short NrImages;                 // 2 Bytes

//following 58 Bytes:
//file version 7:  >=1.4.3 03/21/06 recipe seperate: attachedRecipeSize
        short attachedRecipeSize;       // 2 Bytes
        BYTE  spare[56];                //  56 Bytes
//file version 6: >=1.4.0 11/29/05 image sequence 'recipe'
```

```
        short spareShort;               // 2 Bytes
        BYTE SeqRecipe[56];             // 56 Bytes
};
```

Sequencer recipe block

file version >=7:

file header may be followed by 128 Bytes of sequencer 'recipe' block

attachedRecipeSize=0: no attached block

>0 attached 128Byte block

attachedRecipeSize contains count of Bytes used for data within that block

sequencer commands	arg1	arg2	arg3	
<0 do nothing				(<0 in first node: list is empty)
0 acquire	image			
1 wait	msec			
2 subtract	image=	image	- image	
3 align	dest	reference	image	
4 setspin	1=spinup 0=spindown			
5 normalize difference/sum				
6 set LEEM supply absolute				
7 set LEEM supply relative				
8 LEEM preset				
9 add and accumulate	image=	image	+ image	
10 accumulate arg2 into internal buffer and display that buffer / nrcycles int arg 1				

the recipe contains *further data* – in case a more detailed explanation is needed please contact Elmitec.

Imageheader

current IMAGEVERSION = 5

4: >= 1.1.m: LEEMdata[256]

5: >= 1.5.3 03/19/08 block of markeup for image tools (cross section)

Imageheader *(fileheader.version >=5):* ***size in Bytes: 288***

```
struct UKImageHeader{
        short size;                             // 2 Bytes
        short version;                          // 2 Bytes
        short ColorScaleLow,ColorScaleHigh;     // from 1.3.10
                                                //  before 1.3.10 4 filler Bytes
        LONGLONG imagetime;                     // 8 Bytes
        short MaskXShift,MaskYShift;            // from 1.3.10
                                                //  before 1.3.10 4 filler Bytes
//following 4 Bytes:
//>=1.5.3 image version 5
        BYTE useMask                            // >= 1.6.5
        BYTE spare
        short attachedMarkupSize                //.dav never contain markups
//<1.6.5
        BYTE  spare[4];                         // 4 Bytes

        short spin;                             // 2 Bytes
        short LEEMdataVersion;                  // 2 Bytes currently version 2
        unsigned char LEEMdata[256];            // 256 Bytes Overlay data
                                                // 4 Bytes filler
};
```

notes:
The **'imagetime'** member of the image header structure holds the standard Windows *FILETIME* which is explained as the following:
"The *FILETIME* structure holds an unsigned 64-bit date and time value for a file. This value represents the number of 100-nanosecond units since the beginning of January 1, 1601." MS Visual C++ provides a number of functions to convert this time.

The **LEEMdata** array containing overlay data is structured as follows:
source 1, argument 1 ... source n, argument n

source tags:

0..99: 1. LEEM2000 module #
2. followed by name
3. followed by 1 ASCII digit identifying the unit
unit codes:
0=none,1=V,2=mA,3=A,4=C,5=K,6=mV,7=pA,8=nA,9=uA
4. 0 to terminate the string
5. data: 1 float (4 Bytes)

0xff: skip

100: Mitutoyo micrometer readout: 2 floats (x, y coordinate)
101: (before 1.3.10) FOV (string max 16 char's + 0)
102: (before 1.3.10) varian controller #1 gauge #1 value (float)
103: (before 1.3.10) varian controller #1 gauge #2 value (float)

104: camera exposure in ms (float)
105: title (string max 16 char's + 0)

new >= 1.3.10:
106: varian controller #1 gauge #1 label, units value (string max 16 char's+0,+string max 4 char's+0, float) 27
107: varian controller #1 gauge #2 label, units value (string max 16 char's+0,+string max 4 char's+0, float) 27
108: varian controller #2 gauge #1 label, units value (string max 16 char's+0,+string max 4 char's+0, float) 27
109: varian controller #2 gauge #2 label, units value (string max 16 char's+0,+string max 4 char's+0, float) 27
110: FOV , camera to FOV cal. factor (string max 16 char's+0+float)
111: phi, theta (float,float)
112: spin //>=1.4.1

obsolete:
Imageheade**r (fileheader.version <5):* ***size in Bytes: 48

```
struct UKImageHeader{
        short size;                 //  2 Bytes
        short version;              //  2 Bytes
                                    //  4 Bytes filler
        LONGLONG imagetime;         //  8 Bytes
        long  LEEMdata1_source;     //  4 Bytes
        float LEEMdata1_data;       //  4 Bytes
```

```
        short spin;                 //  2 Bytes
        short spareShort;           //  2 Bytes
        float LEEMdata2_data;       // 4 Bytes
        BYTE spare[16];             // 16 Bytes
};
```

Image markup block

image version >=7

image header may be followed by 128 Bytes of image markup block

attachedMarkupSize =0: no attached block

>0 attached 128Byte block

attachedMarkupSize contains count of Bytes used for data within that block

This block contains info about lines and markers (letters & numbers) which the used has placed on the image.

note: the word 'markup' does not imply a similarity to *html.*

Video File

File Type: binary
File Extension: dav
Open with: U-view
File Contents: Concatenated still images.

File contents:

Fileheader
if file version >=7: block of 'recipe' data (128Bytes)

Imageheader$_1$
ImageData$_{1:}$
Width x Height x Pixel in 2 Bytes (if Bits PerPixels=16)
…
Imageheader$_n$
ImageData$_{1n}$
Width x Height x Pixel in 2 Bytes (if Bits PerPixels=16)

Intensity data File

File Type:	text
File Extension:	ivs
Open with:	U-view or simple text editor : Microsoft Word-Pad Notepad does not display the text correctly, Word does not save correctly unless you follow the note below:
Note:	don't re-save the file with a formatting editor like MS Word. If this is done, the file can not be read back into U-view. You could use MS Word but must save as plain text file.

File Contents:

UK SOFT
software FileVersion
IRectangle left top right bottom
StartChannel StartChannel
DataSection #Channels n — *number of data pairs to follow*
$Time_1$ $intensity_1$ — *data pairs: time, intensity*
....
$Time_n$ $intensity_n$ — *Exponential format*
last_entry

example:

```
UK SOFT
software      1
IRectangle    254 174 274 194
StartChannel  0
DataSection   4
5.050000e+003    1.251472e+006
5.220000e+003    1.252496e+006
5.270000e+003    1.253216e+006
5.380000e+003    1.254112e+006
last_entry
```