File Formats

Last update: 17 April 2018 Uview **15.4.0** and up

The following section describes the format of the proprietary file formats used in U-view.
A simple program can be provided to list all the header info and LEEM overlay data in a table.

Still Image File

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

Versions released:

Current FILEHEADER VERSION: ***8***

version 8: introduced with 3.3.0, 08/30/2010	CameraBitsPerPixel
version 7: introduced with 1.4.3, 03/21/06	recipe seperate: attachedRecipeSize
version 6: introduced with Uview 1.4.0, 11/29/05	image sequence
version 5: introduced with Uview 1.1.m, 2002	LEEMdata[256] added

Current IMAGEHEADER VERSION: ***7***

version 7: introduced 14.0.6, 2017	spareB2 changed to desired_rotation_fraction Inclusion of MACROs in 'markup section'
version 6: introduced with Uview 5.0, 6/22/2012	images are stored now without rotation but with infos desired_rendering; desired_rotation; applied_processing
version 5: introduced with Uview 1.5.3, 03/19/2008	block of markeup for image tools
version 4: introduced with Uview 1.1.m, 2002:	LEEMdata[256] added

Current LEEMDATAVERSION: ***2***

version 2: *introduced with Uview 1.4.3, 03/21/06:*
exposure time modified, average and seq. cycle added.
introduced with Uview 5.1, image header version 6:
when leemdataversion>2 an extra block of LEEM data is added

File Structure overview:

1. **FILE HEADER** (fixed size=104 Bytes)

2. **optional** for file version >=7:
   block of '**SEQUENCER RECIPE'** data (fixed size=128 Bytes)
   presence indicated by fileheader.attachedRecipeSize >0

3. **IMAGEHEADER**

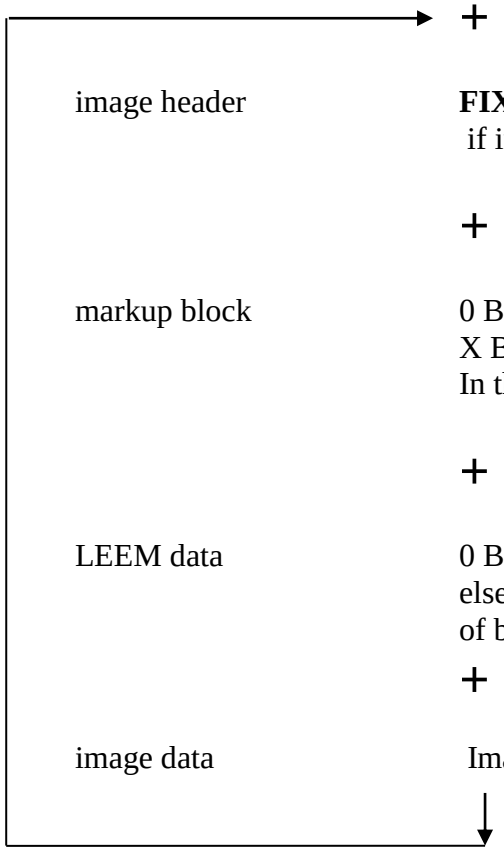
4. **optional** for image version >=5:
   block of **IMAGE MARKUP** data (multiples of 128 Bytes)
   presence indicated by imageheader.attachedMarkupSize > 0

5. **optional** LEEM data if LEEMDataVersion>2 then size is LEEMDataVersion

6. **IMAGEDATA**: Width x Height x Pixel in 2 Bytes (if BitsPerPixels=16)

File structure (.dat & .dav) for file header version 7 and up:

Description	Size
file header only 1x per file	**FIXED: 104** Bytes
	+
attached recipe	128 Bytes if *attachedRecipeSize* >0 else 0 Bytes (*for .dav always =0*)
	+
image header	**FIXED: 288 Bytes** if image header version >3
	+
markup block	0 Bytes if *attachedMarkupSiz*=0 X Bytes if *attachedMarkupSiz*>0 In that case X=128* ((*attachedMarkupSize*/128)+1) (integer division)
	+
LEEM data	0 Bytes if *LEEMdataSize* <=2 else *LEEMdataSize* contains number of bytes in LEEM data block
	+
image data	ImageWidth*ImageHeight*2 Bytes



Notes for .dav files:

1. .dav file consists of concatenated .dat files with the exception of the file header (only one)

2. attachedRecipeSize always=0

3. To get from file start to the first byte of the first image skip the following number of bytes:

104+ 288+ attachedMarkupSize+ *LEEMdataSize*

Offset from file start:	attachedMarkupSize	126,127
	LEEMdataSize	130,131

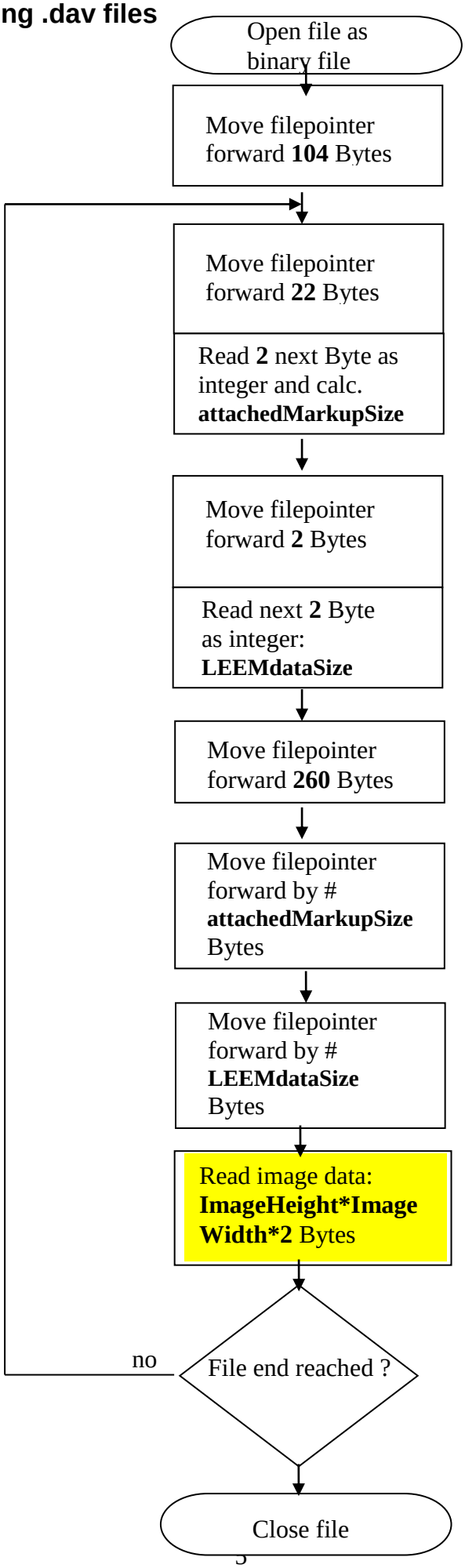
4. After reading the first image to get to the first byte if the following image skip the following number of bytes:

288+ attachedMarkupSize+ *LEEMdataSize*

Offset from file start:	attachedMarkupSize	22,23
	LEEMdataSize	26,27

byte offset from file start	**description**
0	file start (start of file header)
….	
104	start of image header #1
…	
104+22	integer to calculate attachedMarkupSize0
…	
104+26	integer containing LEEMdataSize0
…	
104+288	begin markup block of size calculated from attachedMarkupSize$_0$
…	
104+288+ **attachedMarkupSize**	begin LEEM data block of LEEMdataSize$_0$ length
…	
104+288+ attachedMarkupSize$_0$ **+LEEMdataSize**	begin image data of ImageHeight*ImageWith*2 length
…	

Flowchart for reading .dav files



C(++) program fragment to determine total size of (file) header for .dat with one image:

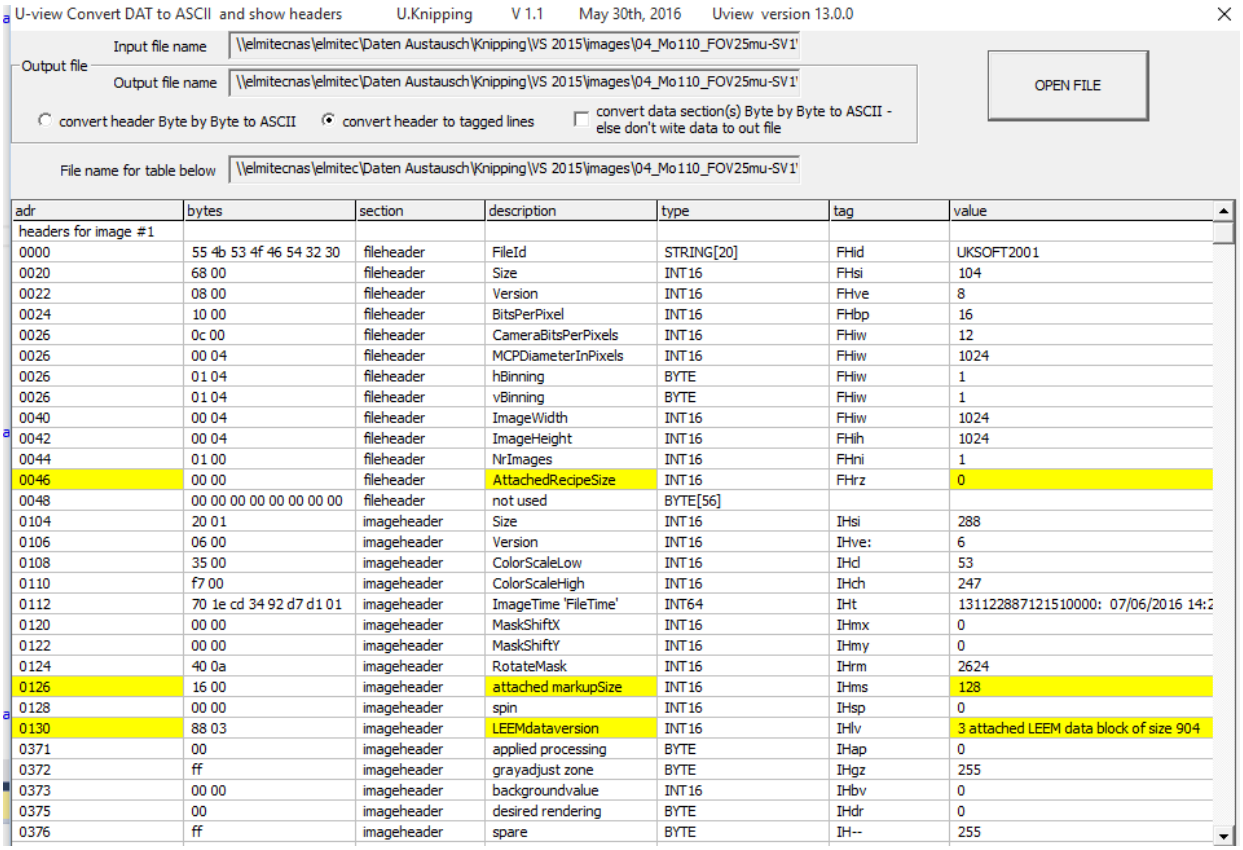
```
short s;
FILE *fd;
if((fd=fopen(currentfname,"r"))==NULL)return false;
fseek(fd,46,SEEK_SET);
fread(&s,2,1,fd);
int attachedRecipeSize  =(int)s;

fseek(fd,126 + attachedRecipeSize,SEEK_SET);
fread(&s,2,1,fd);
int attachedMarkupSize  =(int)s;
attachedMarkupSize  = 128*(( attachedMarkupSize  /128)+1);

fseek(fd,130 + attachedRecipeSize ,SEEK_SET);
fread(&s,2,1,fd);
int LEEMdataSize =(int)s;
int TotalSizeOfHeader=104+288+ attachedRecipeSize  + attachedMarkupSize;
if(leemdatasize>2)TotalSizeOfHeader=TotalSizeOfHeader+ LEEMdataSize;
fclose(fd);
```

ConvertDATto ASCII.exe

This simple program opens .dat as well as .dav files and displays all header information.
It also writes those information and optionally the data section as text into an output file.



adr	bytes	section	description	type	tag	value
headers for image #1						
0000	55 4b 53 4f 46 54 32 30	fileheader	FileId	STRING[20]	FHid	UKSOFT2001
0020	68 00	fileheader	Size	INT16	FHsi	104
0022	08 00	fileheader	Version	INT16	FHve	8
0024	10 00	fileheader	BitsPerPixel	INT16	FHbp	16
0026	0c 00	fileheader	CameraBitsPerPixels	INT16	FHiw	12
0026	00 04	fileheader	MCPDiameterInPixels	INT16	FHiw	1024
0026	01 04	fileheader	hBinning	BYTE	FHiw	1
0026	01 04	fileheader	vBinning	BYTE	FHiw	1
0040	00 04	fileheader	ImageWidth	INT16	FHiw	1024
0042	00 04	fileheader	ImageHeight	INT16	FHih	1024
0044	01 00	fileheader	NrImages	INT16	FHni	1
0046	00 00	fileheader	AttachedRecipeSize	INT16	FHrz	0
0048	00 00 00 00 00 00 00 00	fileheader	not used	BYTE[56]		
0104	20 01	imageheader	Size	INT16	IHsi	288
0106	06 00	imageheader	Version	INT16	IHve:	6
0108	35 00	imageheader	ColorScaleLow	INT16	IHcl	53
0110	f7 00	imageheader	ColorScaleHigh	INT16	IHch	247
0112	70 1e cd 34 92 d7 d1 01	imageheader	ImageTime 'FileTime'	INT64	IHt	131122887121510000: 07/06/2016 14:2
0120	00 00	imageheader	MaskShiftX	INT16	IHmx	0
0122	00 00	imageheader	MaskShiftY	INT16	IHmy	0
0124	40 0a	imageheader	RotateMask	INT16	IHrm	2624
0126	16 00	imageheader	attached markupSize	INT16	IHms	128
0128	00 00	imageheader	spin	INT16	IHsp	0
0130	88 03	imageheader	LEEMdataversion	INT16	IHlv	3 attached LEEM data block of size 904
0371	00	imageheader	applied processing	BYTE	IHap	0
0372	ff	imageheader	grayadjust zone	BYTE	IHgz	255
0373	00 00	imageheader	backgroundvalue	INT16	IHbv	0
0375	00	imageheader	desired rendering	BYTE	IHdr	0
0376	ff	imageheader	spare	BYTE	IH--	255

Detailed description of data structure used in the still image file:

FILE HEADER

size in Bytes: 104

	latest version bold	**addressoffset**
struct UKFileHeader{		
char **id[20];**	20 Bytes: contains "UKSOFT2001" + 0's	**0**
short **size**;	2 Bytes: sizeof(UKFileHeader) = 104	**20**
short **version**;	2 Bytes : FILEHEADER VERSION	**22**
short **BitsPerPixel**;	2 Bytes, bits per pixel in file =16	**24**
FILEHEADERVERSION>=8:		
short **CameraBitsPerPixel**;	2 Bytes, bits per pixel of camera	**26**
FILEHEADERVERSION<8:		
short spare;		
FILEHEADERVERSION>=8:		
short **MCPDiameterInPixels**	2 Bytes only !=0 for files taken by Uview 8.5 and up	**28**
BYTE **hBinning,vBinning**	1+1 Bytes only !=0 for files taken by Uview 8.5 and up	**30**
FILEHEADERVERSION<8:		
long **spare**;	4 Bytes (because the following LONGLONG is aligned by the compiler at an 8-Byte boundary, this variable was not shown in some earlier versions of the file header description)	
LONGLONG **spare**;	8 Bytes (was redundant 'starttime' in early versions)	**32**
FILEHEADER VERSION>=2:		
short **ImageWidth**,**ImageHeight**;	2+2 Bytes	**40,42**
short **NrImages**;	2 Bytes	**44**
FILEHEADER VERSION<2:		
short spare1,spare2,spare3;	6 Bytes	
following 58 Bytes:		
FILEHEADER VERSION >=7: >=Uview 1.4., 03/21/06 recipe separate block		
short **attachedRecipeSize**;	2 Bytes	**46**
attachedRecipeSize values:		
=0:	no attached block	
>0:	attached 128Byte block	
attachedRecipeSize contains count of Bytes used for data within that block		
BYTE **spare[56];**	56 Bytes	**48**
FILEHEADER VERSION =6: >=Uview 1.4.0,11/29/05 image sequence 'recipe'		
(only if more than 1 image in file)		
short spareShort;	2 Bytes	
BYTE SeqRecipe[56];	56 Bytes	
FILEHEADER VERSION <6:		
short spare;	2 Bytes	
BYTE spare[56];	56 Bytes	
		104

SEQUENCER RECIPE block

Optional for file header version >=7

size in Bytes: 128

overview:

sequencer commands	arg1	arg2	arg3	
<0 do nothing (<0 in first node: list is empty)				
0 acquire	image			
1 wait	msec			
2 subtract	image=	image2 -	image3	
3 calculate shift between image[arg1] & image[arg2]				>=Uview 6.1
4 set spin	1=spin up 0=spin down			
5 normalize difference/sum				
6 set LEEM supply absolute				
7 set LEEM supply relative				
8 LEEM preset				
9 add and accumulate	image=	image2 +	image3	
10 accumulate image[arg2] into internal buffer and display that buffer / nr cycles in arg 1				
12 shift and accumulate image[arg2] into internal buffer and display that buffer in image[arg 1] Shift needs to be calculated prior to '12'				>=Uview 6.1
13 apply median filter	image	filter-window 3x3...		>=Uview 6.1
14 divide	image=	factor*(image2 /	image3)	>=Uview 6.1

IMAGEHEADER

Image header version 5 & 6 & 7

size in Bytes: 288

struct UKImageHeader{		addressoffset
short **size**;	2 Bytes: sizeof(UKImageHeader): 288	**0**
short **version**;	2 Bytes: IMAGEHEADER VERSION	**2**
short **ColorScaleLow**,**ColorScaleHigh**;	4 Bytes	**4,6**
LONGLONG **imagetime**;	8 Bytes (see notes)	**8**
short **MaskXShift**,**MaskYShift**;	4 Bytes: shift of overlayed mask in x and y (pixels)	**16**
WORD **RotateMask**;	2 Bytes: bit 7-15: 0-359 degrees image rotation, bits 0,1 UseMask before Uview Version 1.6.5 not used (=0)	**20**
short **attachedMarkupSize**;	2 Bytes	**22**

attachedMarkupSize values:

=0: no attached block

>0: IMAGE MARKUP block is attached, value is number of bytes used for data within that block.
8.6.0: size of this block on disk is a multiple of 128 bytes instead of just 128 bytes as in previous versions.
To determine the actual length of the block calculate:
Block size in bytes: 128* ((attachedMarkupSize/128)+1)
(note: integer arithmetic is used here i.e. 90/128=0)

short **spin**;	2 Bytes	**24**
short **LEEMdataVersion**;	2 Bytes: = 1 or 2 or >2 if >2 then extra block of LEEM data of the size equals LEEMDataVersion is attached after the IMAGE MARKUP block. This is implemented for IMAGEHEADER VERSION>5.	**26**

IMAGEHEADER VERSION>5

unsigned char **LEEMdata**[239];	240 Bytes Overlay data	**28**
BYTE **applied_processing**;	1 Byte	**267**

bit	5	4	3	2	1	0
bad pixel replacement	background subtracted		reverseZ	mirrorY	mirrorX	90 degree rotation

BYTE **gray adjust zone**;	1 Byte 8.6.4 0: invalid, 1=center half, 2= custom areas <0: entire image	**268**
unsigned short **backgroundvalue**;	2 Bytes	**273**
BYTE **desired_rendering**;	1 Byte Values: LINEAR_RENDERING 0 HISTogramEQUALisation_RENDERING 1 GAMMA_RENDERING 2 LOG_RENDERING 3 SQRT_RENDERING 4 ASinH_RENDERING 5 GAUSS_RENDERING 6 CLAHE_RENDERING 7	
BYTE **desired_rotation_fraction**;	fractional part of desired_rotation *100 version 7	
short **rendering_argShort**;	2 Bytes: additional argument for the selected rendering mode, not all modes use this argument	
float **rendering_argFloat**;	4 Bytes: additional argument for the selected rendering mode, not all modes use this argument	

short **desired_rotation**; 2 Bytes: angle the image was last displayed at
short **rotation_offset**; 2 Bytes: for images<6 to avoid rotating them again
short **spare,spare**; 4 Bytes

IMAGEHEADER VERSION=5
unsigned char LEEMdata[256]; 256 Bytes Overlay data
DWORD **spare**; 4 Bytes (to get to a total size divisible by 8
}

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:
Highest bit of this byte is set when this item is not shown on the image: example: if 0x26 is tag for Start Voltage then 0xA6 is used as tag when Start voltage is recorded but not show on image.

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 (float) in seconds - mislabeled previously as ms
LEEMdataVersion >1 :
2 bytes B1,B2 follow
If B1>0 average is on , B2= number of averaged images 2 to 127
if B1=0 average is off
if B1<0 sliding average (<0 in this case -1: hex: 0xff, decimal :255)
105: title (string max 16 char's + 0)
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	
113:	FOV rotation (from LEEM presets)	[15.2.0]
114:	Mirror state	[11.0.1]
115:	MCP screen voltage in kV	[11.0.1]
116:	MCP channelplate voltage in kV	[11.0.1]

120-130 additional gauges (#5,#6....

IMAGEHEADER (obsolete)

Image header version 4

Used in file header version 5 and 6 into file header version 7

size in Bytes: 288

```
struct UKImageHeader{
        short size;                 2 Bytes
        short version;              2 Bytes
                                    4 Bytes filler
        LONGLONG imagetime;         8 Bytes
        BYTE  spare[8];             8 Bytes
        short spin;                 2 Bytes
        short spareShort;           2 Bytes
        BYTE LEEMdata[256];         256 Bytes
                                    4 Bytes filler
};
```

IMAGEHEADER (obsolete)

Image header version 3 and below

Used in file header 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

Optional for file header version >=7

size in Bytes: 128, 256 etc. (multiples of 128)

size depends on actual space needed by the markers

The image header may be followed by 128 bytes or multiples of 128 bytes of an image markup block. This block contains info about lines and markers (letters & numbers) which the user has placed on the image.

The markup block consists of an array of 2-byte words:

[] description

0 block size
1 reserved
2 type of marker:
- 1 horizontal cross section
- 2 vertical cross section
- 3 arbitrary cross section
- 4 reserved
- 6 markers
- 7 inclusion or exclusion areas on image for histogram calculation (8.6.4)
- 8 marker label (11.0.0) follows #6 immediately
- 9 macro (from 14.0.0)

3… followed by:

in case of the cross sections:
- First index x and y into image data array
- Last index x and y into image data array
- Cursor Data index x and y into cross section data

in case of the markers:
- First index x and y into image data array
- Last index x and y into image data array, y index ORed with type of markers
- Color
- Size and line weight
- Followed in case of a marker containing text by the characters making up the text

in case of inclusion or exclusion areas:
- index of rectangle (0,1,2,3)
- type: 1: inclusion, 2 exclusion
- rectangle coordinates (left, top, right, bottom)

in case of marco:
- followed by macro operation tag and arguments

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

Data File containing multiple images

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

Fileheader
Optional sequencer recipe block

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

Video File (.dav)

File Type: binary
File Extension: dav
Open with: U-view
File Contents: Concatenated still images (except that the file header appears only once)

Fileheader
Optional sequencer recipe block

Imageheader$_1$
Optional markup block$_1$
Optional LEEM data block$_1$
ImageData$_1$:
Width x Height x Pixel in 2 Bytes (if Bits PerPixels=16)
…
Imageheader$_n$
Optional markup data block$_n$
Optional LEEM data block$_n$
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*
.... *Exponential format*
$Time_n$ $intensity_n$
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