FSTS and ADV File Formats

Version 1.3 Specification

Hristo Pavlov

Scope

The Flexible Stream Transport Format (FSTF) defines a file format for storing streams of data such as measurements from scientific equipment. The format is built as a generic and extensible binary file format which is fitted for storing of large amounts data such as 16 bit video frames.

The Astronomical Digital Video (ADV) file format is built on the top of the FSTF and is designed to particularly store astronomical time-stamped video observations. The ADV file format is used by the Astronomical Digital Video System (ADVS).

Purpose

This document formally defines the FSTS and ADV formats for data structuring and exchange. It specifies the organisation and content of FSTS and ADV data sets. A list of specific metadata tags used by the ADVS system is given.

Definitions and notations

Conformance

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Numeric values

Decimal and hexadecimal are used within this specification. Decimal numbers are most frequently used to represent quantities or counts. Addresses and sample binary data is represented by hexadecimal numbers.

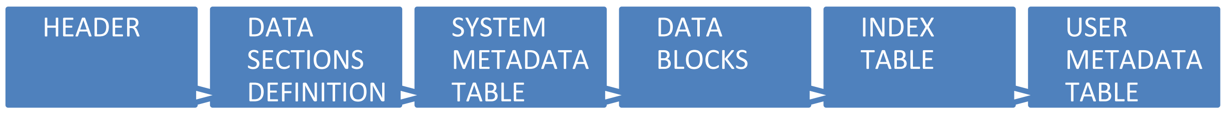
Byte ordering

The ordering convention of bytes within 32-bit and 64-bit words in the FSTS binary data is little-endian.

FSTF File Format

File Structure

The FSTF file is organized into the following 6 blocks:



The Header identifies the file format and defines the offsets of the blocks inside the file. The Data Sections Definition block defines the number, type and settings of the data sections saved for each data frame. The Data Blocks contain the actual recorded data frames.

The Index Table contains the offsets of each data frame in the file for a faster access and the Metadata tables contain generic name/value pairs of metadata related to the current record. There are two Metadata tables – System Metadata Table which was created and populated at the time of the recording and a User Metadata Table, which is created blank and can used by data reduction software to add extra information to the file as necessary.

Data Types

Throughout this document a number of data type abbreviations are used to indicate the type of data of the fields. The following data types are used:

Type	Definition
UInt8	An 8 bit unsigned integer number
UInt16	A 16 bit unsigned integer number
UInt32	A 32 bit unsigned integer number
UInt64	A 64 bit unsigned integer number
Int64	A 64 bit signed integer number
PascalString	An ASCII string with a maximum length of 255 characters where the first byte shows the length of the string. The maximum length of a PascalString data block is 256 where the first byte is 0xFF followed by 255 characters.
Bytes16	A block of 16 bytes
Bytes	A block of bytes with variable length

Glossary

Type	Definition
Reader Software	Any software that reads the file format for example to do data reduction

The format of all file blocks is specified next.

Header

The first section of the FSTF file is the Header. It is positioned at the beginning of the file and its binary content is as follows:

Offset	Type	Description
0x00	UInt32	Magic value - this is the number 0x46545346 (the characters 'FSTF') which identifies the file format.
0x04	UInt8	The revision of the file format. This document describes revision 01.
0x05	UInt32	The number of data frames saved in the file.
0x09	UInt64	The offset of the index table from the beginning of the file.
0x11	UInt64	The offset of the system user metadata table from the beginning of the file.
0x19	UInt64	The offset of the user metadata table from the beginning of the file.

The offsets to the Index Table and Metadata Tables are 64-bit and a Reader Software MUST be able to handle 64-bit addressing.

Data Section Definition

The Data Section Definition is placed immediately after the Header. Its binary format is presented in the table below. The offsets in the first column are from the beginning of the file as the Data Section Definition can be viewed as a continuation of the Header.

Offset	Type	Description
0x21	UInt8	Number of data sections
0x22	PascalString	Name of the first section
	UInt64	Offset from the beginning of the file of the configuration header for the first section
	PascalString	Name of the second section
	UInt64	Offset from the beginning of the file of the configuration header for the second section
...	...	...
	PascalString	Name of the N-th section
	UInt64	Offset from the beginning of the file of the configuration header for the N-th section

The section name SHOULD contain only alpha numerical characters.

It is RECOMMENDED the configuration headers of the individual sections to be placed immediately after the Data Section Definition and before the System Metadata Table. The data layout of the configuration headers of the individual sections is content specific and is not defined in the FSTF format specification. The data sections configuration header formats for the ADV file format is defined later in this document.

Here is an example of a Header and Data Section Definition data blocks that define 163 data frames with 3 sections called IMAGE1, IMAGE2 and STATE. The offset of the Index Table is 0x000000000028C463, the offset of the System Metadata Table is 0x000000000028C507 and the offset of the User Metadata Table is 0x0000000000320A35. All values in the first table are hexadecimal and the second table displays the corresponding ASCII codes for clarity where non-displayable characters are shown with a dot.

46	53	54	46	01	A3	00	00	00	63	C4	28	00	00	00	00
00	07	C5	28	00	00	00	00	00	35	0A	32	00	00	00	00
00	03	06	49	4D	41	47	45	31	47	00	00	00	00	00	00
00	06	49	4D	41	47	45	32	AB	00	00	00	00	00	00	00
06	53	54	41	54	55	35	01	00	00	00	00	00	00		

F	S	T	F	.	.	.	.	.	C	(	.	.	.	.	.
.	.	.	(	.	.	.	.	.	5	2	.	.	.	.	.
.	.	.	I	M.	A	G	E	1	G	.	.	.	.	.	.
.	.	I	M	A	G	E	2	.	.	.	.	.	.	.	.
.	S	T	A	T	U	S	.	.	.	.	.	.	.		

Data Block

The obtained data is saved in the Data Blocks as sequence of frames. There is one frame for each interval where data was obtained. Each frame consists of one or more sections of data. A frame MAY contain more than one section of data and each frame MUST have exactly the same number and order of sections of data. A section could contain for example the image bytes and/or other information such as a collection of measurements including timestamps, settings of the equipment and any other information that can be useful and goes with the frame. All data sections must be present in the file for each and every frame. If for a given frame there is no data in a specific section then the section is still present but blank.

Data Blocks section SHOULD be placed immediately after the System Metadata Table. The FSTF file doesn't provide the offset of the Data Blocks sections in the Header however the offset is given in the Index Table (see below) for each data frame.

Data frames SHOULD be placed sequentially in the Data Blocks and usage of padding between the data frames is OPTIONAL.

Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
...
Frame N

The entries in the Index Table are used to find the position of each data frame. Further to this each data frame starts with the same magic number which makes it possible to find the data frames even when the Index Table hasn't been saved in the file for example due to a power outage during the recording.

The format of the individual data frames is presented below:

Offset	Type	Description
0x00	UInt32	Magic value - this is the word 0xEE0122FF which identifies the beginning of a data frame.
0x04	Int64	Timestamp of the beginning of the frame represented as the time in milliseconds elapsed since 1 Jan 2010, 00:00:00.000 Universal Time (JD 2455197.5). The time can be negative for timestamps before this moment.
0x0C	UInt32	Exposure (duration) of the frame in 0.1 milliseconds.

0x10	Bytes	Contents of the first section of the data frame
	Bytes	Contents of the second section of the data frame
	...	...
	Bytes	Contents of the N-th section of the data frame

The offsets given in the table above are from the beginning of the individual data frames. The offset of the data frame and the total length of the data frame bytes are given in the Index Table. The four bytes used for the magic value identifying the start of the frame are not included in the bytes counts saved in the Index Table.

Index Table

The Index Table contains the positions and sizes of each data frame saved in the file. The offset of the Index Table from the beginning of the file is given in the Header. The Index Table has the following format:

Offset	Type	Description
0x00	UInt32	Number of index entries
0x04	Bytes16	First frame index entry
0x14	Bytes16	Second frame index entry
0x24	Bytes16	Third frame index entry
...	...	...
	Bytes16	N-th frame index entry

Where each frame index entry contains 16 bytes in the following format:

Offset	Type	Description
0x00	UInt32	Elapsed time in milliseconds since the first frame
0x04	UInt64	The offset of the data frame from the beginning of the file
0x0C	UInt32	The length of the data frame in bytes

The elapsed time SHOULD NOT be used to calculate the timestamp of a frame based on the first frame timestamp. The timestamp value given at the beginning of each data frame should be used instead. The elapsed time in milliseconds is provided so Reader Software can display visually the relative position of any frame in the file and also to do a search for a frame based on the time elapsed from the beginning of the recording.

Metadata Tables

The System and User Metadata tables contain series of name/value pairs referred in this specification as Tags. The type of the information saved as System Metadata Tags is controlled by the recorder that created the file. Examples of such information include the name and version of the recoding software, version of the hardware, geographical coordinates of the observer and others. The System Metadata Table SHOULD be placed after the Data Section Definition block and before the Data Blocks.

The User Metadata Table MUST be at the very end of the file so any Reader Software can easily add new Tags and update existing Tags. Reader Software SHOULD NOT update any of the entries in the System Metadata Table.

The two Metadata Tables have the following format:

Offset	Type	Description
0x00	UInt32	Number of Metadata tags
0x04	PascalString	Name of the first tag
	PascalString	Value of the first tag
	PascalString	Name of the second tag
	PascalString	Value of the second tag
	…	…
	PascalString	Name of the N-th tag
	PascalString	Value of the N-th tag

Reconstruction of a Corrupted File

In rare cases the Index Table and User Metadata Table MAY be missing. This could happen if the recorder didn't complete the recording due to an interruption for example caused by a power failure. In such a case is could be possible for a Reader Software to restore the Index Table by searching the FSTF file for the data frame magic number (0xEE0122FF). Data frames contain the timestamp and duration of the exposure and this information is sufficient to build the Index Table entries.

In a case of an interrupted recording the User Metadata Table MAY be missing as well however it SHOULD be empty.

Any software that reads the data frames SHOULD handle gracefully the case where a data frame hasn't been flushed completely as a result of an interrupted recording.

To determine whether the recording has been interrupted a Reader Software could use the following criteria:

- The offset of the Index Table or User Metadata Table specified in the Header are either not specified (are zero) or appear to be after the end of the file
- The Index Table could not be read correctly

If the software reconstructs the Index Table of a file it MAY also add an entry into the User Metadata Table about this reconstruction adding information such as:

- Date when the reconstruction was done
- Reason for reconstructing the file
- Identification of the software and/or user that has reconstructed the file

ADV File Format

The FSTF file format defines a flexible and extensible data layout for storing scientific data. However to be complete the file format needs to be extended and the data format of the Data Section Definition and Data Blocks needs to be defined for a specific usage.

The Astronomical Digital Video System (ADVS) is a device that can record 12bit digital video time-stamped by a GPS with a precision of a millisecond. To record the data frames ADVS uses a file format based on the Flexible Stream Transport Format called Astronomical Digital Video (ADV). The ADV file format is defined next.

The ADV file uses two sections called *IMAGE* and *STATUS*. The *IMAGE* section stores the pixels from each video frame image and the *STATUS* sections stores any additional measurements/information associated with the video frame. The ADV file format was created with the intension for it to allow very fast recording at high frame rates and to record compressed data that would minimize the total file size for long records at low frame rates. Because of the scientific nature of the data being stored a lossless compression is in use.

The ADV format is also extensible allowing the format to define new data structures in future for example when better cameras or better data compression becomes available.

IMAGE Section

The name of the first section is IMAGE in capital letters. The PascalString hexadecimal bytes of the name of the section are:

05	49	4D	41	47	45

.	I	M	A	G	E

The configuration of the IMAGE section is saved immediately after the Data Section Definition block. The offset of the configuration from the beginning of the file is given in the Data Section Definition for the *IMAGE* section. The configuration has the following format:

Offset	Type	Description
0x00	UInt8	Version of the *IMAGE* section. This specification defined version 01
0x01	UInt32	The width of the video images in pixels
0x05	UInt32	The height of the video images in pixels
0x09	UInt8	The native bits per pixels (bpp) of the detector. This may be different from the bpp used to save individual data frames. For example a video camera may have a native resolution of 12 bpp but the images may be saved in 16 bpp format.
0x0A	UInt8	Number of Image Layouts that could be found in the file
0x0B	UInt8	ID of the first defined Image Layout
	Bytes	Configuration of the first Image Layout
	UInt8	ID of the second defined Image Layout
	Bytes	Configuration of the second Image Layout
	…	…
	UInt8	ID of the N-th defined Image Layout

	Bytes	Configuration of the N-th Image Layout
	UInt8	Number of configuration name/value Tags
	PascalString	The name of the first *IMAGE* section tag
	PascalString	The value of the first *IMAGE* section tag
	PascalString	The name of the second *IMAGE* section tag
	PascalString	The value of the second *IMAGE* section tag
	...	...
	PascalString	The name of the N-th *IMAGE* section tag
	PascalString	The value of the N-th *IMAGE* section tag

The *IMAGE* section configuration defines some generic parameters such as the size and bit depth of the images and then defines specific Image Layouts that may be used to store the video frames. Image Layouts allow different compression and data layout to be used for the video frames and with this to either achieve fast recording time or smaller file size. The *IMAGE* section also defines its own Tags that give extra information about the recording. The Tags used by the Astronomical Digital Video System are given in the table below:

Tag Name	Tag Value
IMAGE-BYTE-ORDER	Indicates the byte order of the pixel bytes. Possible values are the **LITTLE-ENDIAN** and **BIG-ENDIAN**
SECTION-DATA-REDUNDANCY-CHECK	When a data redundancy check is used the value indicates the algorithm being used. When this Tag is present its value can be only **CRC32**

Each of the Image Layout configurations has the following format:

Offset	Type	Description
0x00	UInt8	Version of the Image Layout configuration. This specification defines version 01
0x01	UInt8	Bits per pixel for each pixel in the Image Layout. This may differ from the bpp of the detector
0x02	UInt8	Number of name/value Tags of the Image Layout
	PascalString	Name of the first tag
	PascalString	Value of the first tag
	PascalString	Name of the second tag
	PascalString	Value of the second tag
	...	...
	PascalString	Name of the N-th tag
	PascalString	Value of the N-th tag

All specifics of an Image Layout are given in Tag values and the following Tags are used by ADVS:

Tag Name	Tag Value
DATA-LAYOUT	The type of the data layout used. ADVS uses two data layouts and they are **FULL-IMAGE-DIFFERENTIAL-CODING** and **FULL-IMAGE-RAW**. Those layouts will be

	described below.
DIFFCODE-BASE-FRAME	This Tag indicates the type of differential coding (when used) and could have a value of either **KEY-FRAME** or **PREV-FRAME** (when specified)
DIFFCODE-KEY-FRAME-FREQUENCY	When differential coding is used this Tag indicates the frequency of the key frame
SECTION-DATA-COMPRESSION	Specifies the used compression algorithm. Possible values are **UNCOMPRESSED** and **QUICKLZ**

When each data frame is saved in the ADV file it begins with the 4 byte magic number followed by the 8 byte timestamp and the 4 byte exposure duration. After that follow the data from the IMAGE and STATUS sections as defined below:

Offset	Type	Description
0x00	UInt32	The size of the *IMAGE* data block starting from the next byte (saved next)
0x04	UInt8	The zero based Image Layout index used to save the pixels in the current image. The indexing of the Image Layouts is done based on the order in which they appear in the *IMAGE* section configuration
0x05	UInt8	The type of the recorded frame (this is dependent on the DATA-LAYOUT used). Possible values are: 0 = This is a frame that does not use differential coding; 1 = This is a key frame; 2 = This is a differentially coded frame
0x06	Bytes	Pixel values according to the Data Layout used (see below)
	UInt32	The size of the *STATUS* data block
	Bytes	Data associated with the STATUS data block (saved next)

If the specified Image Layout uses compression (the value of the SECTION-DATA-COMPRESSION tag is QUICKLZ) then the bytes need to be decompressed before reading them further. The QuickLZ algorithm is used by ADVS. This is an open source fast compression library developed by Lasse Mikkel Reinhold and source code for compression and decompression in a number of different languages is available on http://www.quicklz.com/.

After a decompression (where required) the data format of the pixels for the two layout types is given below. If SECTION-DATA-REDUNDANCY-CHECK is specified in the Image Section tags at the end of the pixel data section a 4 byte CRC value is present.

FULL-IMAGE-RAW

The raw image layout is only supported for 16bpp pixel data and the recorded frame types will be 0 (frames that don't use differential coding). The data block contains 2 * Width * Height bytes i.e. 2 bytes per pixels. Pixels in a row are given in order from left to right and rows are added to the data block from top to bottom. This data layout is used with fast frame rates and does not support CRC checksums. The order of the bytes in the 16 bit pixel value is determined by the IMAGE-BYTE-ORDER Image Layout tag value. The default byte order, when IMAGE-BYTE-ORDER is not specified, is little-endian.

FULL-IMAGE-DIFFERENTIAL-CODING

The differential coding data layout could have frames of two different recorded types – a key frame (second byte of the Image Layout data block has a value of 1) and differentially coded frame (second byte of the Image Layout data block has a value of 2).

For key frames the pixels of the current frame are given while for differentially coded frames the saved value for each pixel is the difference (as a signed integer number) between the current frame pixels and either the last key frame or the previous frame pixels. Whether the key frame or the previous frame is used for the differential coding is determined from the value of the DIFFCODE-BASE-FRAME tag of the used Image Layout.

The pixels are recorded in lines starting from the top to the bottom of the image and from left to right in the individual line. Depending on the bits per pixel of the Image Layout and the Byte Order of the Image Section there is a number of byte configurations:

Mode	Values to Save	Saved Bytes
12bpp	0x123 0x456 0x789 0xABC	12 34 56 78 9A BC
little-endian, 16bpp	0x1234 0x5678 0x9ABC	34 12 78 56 BC 9A
big-endian, 16bpp	0x1234 0x5678 0x9ABC	12 34 56 78 9A BC

Key frames are always used in a case of Differential Coding and their frequency is given by the DIFFCODE-KEY-FRAME-FREQUENCY tag of the current Image Layout. Because it is possible in the same ADV file any two Image Layouts to be used in any two consecutive frames, a Reader Software MUST NOT rely on the DIFFCODE-KEY-FRAME-FREQUENCY to find the closest previous key frame. To decode the current frame a Reader Software SHOULD use the following algorithm:

1) Read the first two bytes of the frame data block for the current frame. The first byte is the Image Layout index and the second byte is the Frame Type
2) Read the size of the *IMAGE* data block and read all the content bytes
3) If the used Image Layout uses compression then decompress the content bytes
4) If the Image Layout is FULL-IMAGE-RAW then the content is the raw pixels
5) If the Image Layout is FULL-IMAGE-DIFFERENTIAL-CODING then check the value of the Frame Type to determine what to do next
    a. If Frame Type is 1 (key frame) then the content is the raw pixels
    b. If Frame Type is 2 (differentially coded frame) then the previous key frame needs to be found in order to reconstruct the image. For this keep going one frame backwards until a frame data block is found that has a Frame Type of 1 (key frame). Once the key frame is found read its pixels.
        i. If the DIFFCODE-BASE-FRAME is KEY-FRAME then add the raw pixels to the key frame pixels to get the current frame pixels
        ii. If the DIFFCODE-BASE-FRAME is PREV-FRAME then continue reconstructing frames forward until the current frame is reached

When Differential Coding is used the operation of finding the previous key frame should not normally be done very often. If the file is played forward then this will not be necessary at all unless possibly for the very first frame. If a random frame needs to be displayed or the file is played backwards then a key frame may need to be searched every time.

STATUS Section

The name of the second section is STATUS in capital letters. The PascalString hexadecimal bytes of the name of the section are:

06	53	54	41	54	55	53

.	S	T	A	T	U	S

The configuration of the *STATUS* section is saved immediately after the Data Section Definition block. The offset of the configuration is given in the Data Section Definition for the *STATUS* section. The configuration has the following format:

Offset	Type	Description
0x00	UInt8	Version of the *STATUS* section configuration. This specification defines version 01
0x01	UInt8	Number of status entries in the section
	PascalString	Name of the first status entry
	UInt8	Data type of the first status entry
	PascalString	Name of the second status entry
	UInt8	Data type of the second status entry
	…	…
	PascalString	Name of the N-th status entry
	UInt8	Data type of the N-th status entry

The names of the status entries SHOULD be short but descriptive and indicate the type of the value being saved in this entry. The values of the entries are always saved as a PascalString. The data type represented in the string value is given with the Data type configuration entry. The following data types are defined:

Data Type	Description
0	UInt8 – unsigned 16 bit number
1	UInt16 – unsigned 16 bit number
2	UInt32 – unsigned 32 bit number
3	UInt64 – unsigned 64 bit number
4	Real – 32 bit floating point single precision number (IEEE 745)
5	PascalString – a string with maximum length of 255 characters and a leading byte indicating the string length
6	List16PascalString – a list of maximum 16 PascalString values each with max length of 255 and a leading byte indicating the string length. The first byte is the number of strings followed by the PascalString content

The data block representing the *STATUS* section has the following format:

Offset	Type	Description
0x00	UInt32	The size of the STATUS section data block for the current frame
0x04	UInt8	Number of status entries given in the current data block. This value can be zero if no status entries are given
0x05	UInt8	The 0-based index of the first recorded status entry
	PascalString	The value of the first recorded status entry
	UInt8	The 0-based index of the second recorded status entry
	PascalString	The value of the second recorded status entry
	…	…
	PascalString	The 0-based index of the N-th recorded status entry
	UInt8	The value of the N-th recorded status entry

The values of some or all of the defined status entries MAY be omitted. The blank status section data block has the length of 5 bytes and looks like this:

01	00	00	00	00

The following status entries may appear in the ADV file:

Gain
The gain value returned by the video camera for each frame represented as decibels.

Gamma
A value of gamma is only given for the first frame in which gamma has been changed by ADVS.

Shutter
The shutter value returned by the video camera for each frame represented as milliseconds.

Offset
The brightness (offset) value returned by the video camera for each frame, represented in percentages.

SystemTime
The system time as number of milliseconds elapsed since 1 Jan 2010, 00:00:00.000 Universal Time. The system time is a low resolution time as returned by the system clock at the time when the frame was saved to the ADV file. The system time is only given as a reference and should not be used for precise timing of events. It could however give a low precision timescale relative to unknown time offset from UTC which may be useful in a case of no GPS fix.

GPSTrackedSatellites
The number of GPS satellites tracked by the timing module.

GPSAlmanacStatus
The status of the GPS almanac reported by the timing module. The value could be 0 for ‘uncertain’ or 1 for ‘good’.

GPSFixStatus
The status of the GPS fix reported by the timing module. The value could be: 0 for 'No Fix', 1 for 'G Fix' or 2 for 'P Fix'

GPSFix
List of messages generated when the GPS obtains a fix for example after it has lost it or if the recording has started before the GPS had a fix. If the GPS fix is lost this is recorded as a SystemError (see below)

UserCommand
List of descriptive text messages generated when the user has requested any ADVS parameter to be changed. This also includes commands to change the frame rate/exposure.

SystemError
List of descriptive text messages generated when a system error had occurred or when a system error has been resolved.

Metadata Tags
The Astronomical Digital Video System records the following System Metadata Tags in the ADV file:

Tag Name	Tag Value
RECORDER	The name of the recording system. The value is **ASTRO DIGITAL VIDEO SYSTEM**
ADVR-SOFTWARE-VERSION	The version of the recording software running on the ADVS system
HTCC-FIRMWARE-VERSION	The version of the firmware running on the HTCC module of the ADVS system
HC-FIRMWARE-VERSION	The version of the firmware running on the hand controller module
CAMERA-MODEL	The model of the used video camera
CAMERA-SERIAL-NO	The serial number of the used video camera
CAMERA-VENDOR-NAME	The vendor name of the video camera
CAMERA-SENSOR-INFO	Information about the CCD sensor of the video camera
CAMERA-SENSOR-RESOLUTION	Resolution of the CCD sensor
CAMERA-FIRMWARE-VERSION	The version of the firmware running on the video camera
CAMERA-FIRMWARE-BUILD-TIME	Firmware build time for the video camera firmware
CAMERA-DRIVER-VERSION	The version of the software driver used to control the video camera
LONGITUDE	The Geographical Longitude where the observation was made shown in degrees as a floating point number
LATITUDE	The Geographical Latitude where the observation was made shown in degrees as a floating point number
ALTITUDE	The altitude above the sea level in meters as a floating point number
WIDTH	The width of the images in pixels
HEIGHT	The height of the images in pixels
BITPIX	The bit rate (bit per pixels) of the video camera