PCI Express JPEG Frame Grabber Software Manual

Model 817 | Rev.C | November 2016

SENSORAY | embedded electronics

Designed and manufactured in the U.S.A

SENSORAY | p. 503.684.8005 | email: info@SENSORAY.com | www.SENSORAY.com

7313 SW Tech Center Drive | Portland, OR 97203

Table of Contents

Revision history

Rev.C, November 2016 – clarified deinterlacing capability.

Rev.B, April 2009 – reflects changes to the API implemented in the SDK versions 2.x. Please note: SDK versions 2.x are not compatible to versions 1.x. Data structures and functions have changed. Application software requires some modifications.

Rev. A, September 2008 – original release.

Software Installation and Demo Applications

Sensoray provides Software Development Kits (SDK's) for Windows and Linux. Each SDK includes some examples of programming the model 817 board (demo applications). Please refer to the readme file provided with the software for software installation instructions and demo applications' descriptions.

Overview

Model 817 's SDK's for Windows and Linux allow the development of custom application software. The Application Programming Interface (API) provided by the drivers offers maximum flexibility in the application development.

Both Windows and Linux SDK's include a number of demo applications illustrating the use of the API. Those provide a good starting point for the custom application development.

The API supports multiple 817 boards (currently up to 4). Each capture channel (an entity corresponding to a video source, for example, a camera) is configured and controlled individually. Wherever possible an effort was made to allow independent settings for video channels. Capture is performed either in single or continuous mode. In continuous mode the board captures frames with a selected frame rate and sends the data to the host automatically, provided the buffers on the host side are emptied promptly by the application. Capture frame rates for bitmaps and JPEGs can be set up independently. Image parameters (brightness, contrast, saturation), compression quality settings, even the image scale can be set individually for each capture channel.

A set of tools is provided for monitoring the board's status (see *Troubleshooting and Debugging Tools* section). Though not required to be used, those may provide some insight in cases when questions related to the performance arise.

The *Data Types* section describes special data types of the SDK and provides most of the information regarding the board's capabilities and operation restrictions. The *Functions* section lists the API's functions.

All API's functions return an error code. A non-zero value indicates an error. A brief description of error codes could be found in `s817.h`. It is strongly recommended to always check this code at the application level and trap error conditions. This usually simplifies troubleshooting and helps avoid severe application errors (crashes). When reporting an error condition to Sensoray, please provide the API function that caused an error and the value of the error code.

Data Types

All data types are defined in `s817.h`. Integer types are 32-bit.

VIDEO_MODE

```
typedef struct {
    MODE_TV_FORMAT    tvFormat;
    MODE_JPG_SIZE     jpgSize;
    MODE_BMP_TYPE      bmpType;
    int               brightness;
    int               contrast;
    int               saturation;
    int               hue;
} VIDEO_MODE;
```

`tvFormat`

Defines TV format:

`TV_NTSC` for NTSC,

`TV_PAL` for PAL.

`Note`: It is not recommended to switch the input signal from one TV standard to another while the board is capturing. This may result in a lock up requiring software restart.

`jpgSize`

Defines captured image size:

`SIZE_4CIF` - 640x480 (NTSC), 704x576 (PAL);

`SIZE_4CIFI` - 640x480 (NTSC), 704x576 (PAL), deinterlaced (JPEG only);

`SIZE_2CIF` - 640x240 (NTSC), 704x288 (PAL);

`SIZE_1CIF` - 320x240 (NTSC), 352x288 (PAL).

This parameter applies to *both* JPEG and bitmap captured images.

Images captured in `SIZE_2CIF` format represent one field of interlaced video. Such an image looks stretched in the horizontal direction, if displayed directly, but it does not have motion artifacts characteristic to interlaced images. Capture of fields is supported only at the frame rate (i.e. one field per frame, 30 fields per second for NTSC, 25 for PAL).

Images captured in `SIZE_1CIF` format represent one field of interlaced video scaled down horizontally by a factor of 2. The same capture rate limitation as for `SIZE_2CIF` applies.

Images captured in `SIZE_4CIFI` format have one field recreated by interpolating the lines of another field. This removes motion artifacts from captured JPEGs at the expense of some loss of vertical resolution. Bitmaps captured in this mode are still interlaced.

PAL images captured in `SIZE_4CIF` and `SIZE_1CIF` sizes have slightly distorted aspect ratio of 1.222 instead of 1.333.

`bmpType`

Format of uncompressed image. Currently only monochrome 1 byte/pixel (Y8) format is supported.

`brightness`
Video brightness. Must vary between 0x00 (darkest) to 0xff (brightest). The default is 0x80.

`contrast`
Video contrast. Must vary between 0x00 (lowest) to 0xff (highest). The default is 0x80.

`saturation`
Video saturation. Must vary between 0x00 (no color) to 0xff (maximum). The default is 0x80.

`hue`
Video hue (irrelevant in PAL mode). The value is 8-bit signed: -128 (0x80) to 127 (0x7f). The default is 0.

CAPTURE_MODE

```
typedef struct {
     CAP_MODE      capMode;
     int           capType;
     int           compQuality;
     int           frameRateBmp;
     int           frameRateJpg;
} CAPTURE_MODE;
```

`capMode`
Defines capture mode :
`CAP_MODE_CONT` – continuous capture (the board sends captured frames at requested frame rate continuously;
`CAP_MODE_SINGLE` – a single image is captured. A call to `S817_SetCaptureMode` is required to capture one image.

`capType`
Types of images to capture (can be OR'ed):
`CAP_JPG` – enable JPEG capture;
`CAP_BMP` – enable bitmap capture;
`CAP_OFF` – disable capture.

`compQuality`
Defines the level of JPEG compression: must vary between 0 (highest compression, smallest file size, lowest quality) and 100 (lowest compression, largest file size, highest quality). The size of a valid JPEG file is limited to 80KB.

`frameRateBmp`
`frameRateJpg`
Define capture frame rates for JPEG and bitmap images, respectively. Apply to cases when `capMode` is set to `CAP_MODE_CONT.` The following values are allowed:

`frameRateBmp` `frameRateJpg`	Frame rate, frames per second	
	NTSC	PAL

FR_FULL	30	25
FR_4_5	24	20
FR_2_3	20	16.7
FR_1_2	15	12.5
FR_2_5	12	10
FR_1_3	10	8.3
FR_1_5	6	5
FR_1_10	3	2.5
FR_1_15	2	1.7

The frame rates in the table apply to JPEG and bitmap capture *independently*, that is, if `frameRateBmp = FR_1_2 and frameRateJpg = FR_1_2,` the application will capture 15 JPEGs *and* 15 bitmaps every second (NTSC). However, when JPEGs and bitmaps are captured simultaneously in 4CIF resolution on all 16 channels, the combined rate is limited to 400 fps (NTSC) and 330 fps (PAL).

Actual capture rate achieved in a particular case depends also on the application software and operating system performance. The application software must empty the capture buffers allocated in host's RAM promptly. Momentary "distractions" of the host computer can cause the lack of free capture buffers, and occasional frame loss. The resulting capture rate may be, strictly speaking, slightly lower than full. Usually this difference is not more than a fraction of a percent.

Real life application software will be performing additional tasks that could affect capture rate. It is recommended to set the capture rate such that it is limited by the setting, but not by the system's performance. That will result in more predictable and stable capture rates. For example, if the hard drive is capable of saving only 5 frames/second, one would achieve better results setting the capture rate to 5 frames/second, not 15.

MODE

```
typedef struct {
    VIDEO_MODE     vMode;
    CAPTURE_MODE   cMode;
} MODE;
```

Combines `VIDEO_MODE` and `CAPTURE_MODE` structures. Used by `S817_OpenChannel` function.

CAP_BUFFER

```
typedef struct {
    int       bufId;
    char      *pData;
    char      *pStat;
    int       width;
    int       height;
} CAP_BUFFER;
```

Defines the capture buffer structure. Capture buffers are allocated by the driver in the host's RAM. The driver allocates multiple (currently 4) buffers per capture channel. Once the requested captured images are transferred to a capture buffer, the buffer is marked as busy and an application may access the data using the pointers defined in `CAP_BUFFER.` The driver does not copy any image data, just provides the proper pointer values.
An application has to release the buffer promptly (see `S817_ReleaseBuffer` function), so that the board can transfer the data to the host.

`bufId`

Buffer index. The value of `bufId` is set by the driver and is used by an application when a buffer is being released back.

`pData`

A pointer to image data, which may be JPEG or bitmap data depending on the requested data type. The actual size of the data and its validity are indicated in the capture status structure pointed to by `pStat.`

`pStat`

A pointer to status data. See `CAP_STAT` structure.

`width, height`

Image dimensions based on the selected image size (see VIDEO_MODE structure). Those values are inserted by the driver.

CAP_STAT

```
typedef struct {
    int     board;
    int     chanId;
    int     bufId;
    int     capType;
    int     videopresent;
    int     valid;
    int     jpgsize;
    int     tick;
    int     frameCnt;
    int     reserved[4];
} CAP_STAT;
```

Defines capture status structure. Capture status is returned with each captured buffer (see `CAP_BUFFER` structure).

`board`

Index of the board that the buffer was captured from (1 .. `SYS_MAXBOARDS`, see `s817.h`).

`chanId`

Index of an input channel that the buffer was captured from (1 .. 16). A channel is a combination of hardware and software allowing video capture from a specific video input of the 817 board.

`bufId`
: Buffer index. Same as `bufId` of `CAP_BUFFER` structure.

`capType`
: Reflects the type of captured data: JPEG or bitmap. May be either `CAP_JPG` or `CAP_BMP`.

`videopresent`
: Indicates if the board's front end was synchronized to incoming video at the moment the image was captured. A value of 0 means either no input video or loss of sync.

`valid`
: Indicates if the JPEG buffer contains valid data. Undefined for bitmap capture. A value of 0 indicates that JPEG data is invalid. The most common reason for that is `compQuality` set too high.

`jpgsize`
: Size of JPEG data, bytes. Undefined for bitmap capture.

`tick`
: A value of on-board 1 ms timer's tick corresponding to the start of data transfer to the host. The timer starts at 0 when the firmware is loaded onto the board (driver starts). There is a separate timer for each group of 4 input channels (1-4, 5-9, etc.). The timers may not start at exactly the same moment of time.

`frameCnt`
: A value of on-board captured frames counter individual for every input channel.

`reserved[4]`
: Reserved for internal use. Do not modify.

OSD_MODE

```
typedef struct {
  int         osdOn;
  int         osdBmp;
  int         transparent;
  int         positionTop;
  int         ddmm;
  int         year2;
  int         fraction;
  char        line[80];
} OSD_MODE;
```

Controls the on-screen display (OSD). OSD is implemented as 1 line of text overlaid on the captured image. OSD is displayed either in the top or bottom 13 lines of the image. The character width is 8 pixels, offset from the left edge is 4 pixels.

`osdOn`
: A non-zero value turns on OSD for JPEGs and allows OSD for bitmaps.

`osdBmp`
: A non-zero value turns on OSD for bitmaps if osdOn is not zero.

`transparent`
: If set to a non-zero value, OSD box is transparent (only the characters obstruct the underlying image). Otherwise OSD box is black.

`positionTop`
: If set to a non-zero value, OSD box is located on top of the image, otherwise – on the bottom.

`ddmm`
: If set to a non-zero value, the date format of the timestamp is dd-mm (date before month). Otherwise it is mm-dd (month before date).

`year2`
: If set to a non-zero value, the year value of the timestamp is 2 digits, otherwise 4 digits.

`fraction`
: Number of digits in a fraction of a second display of the timestamp: 0, 1 or 2. Other values are not allowed.

`line`
: An array containing the text of the OSD. The text is truncated if it does not fit within the image width.
The following special character combinations are used to control the display:
^d – inserts date;
^t - inserts time.
Date and time data is kept on the board separately for each group of 4 channels (1..4, 5..9, etc.). It is synchronized to the host's time by `S817_SetDateTime` function.

Functions

Overview

Most of the functions return `ECODE` type which is defined as `typedef int ECODE.`
A non-zero value is returned in case of an error. A brief description of error codes could be found in `s817.h`. It is highly recommended to check the return value of all functions that return error codes and report the error codes when contacting technical support.

S817_OpenChannel

`ECODE S817_OpenChannel (int board, int aChan, MODE *pMode)`

Opens a channel on a selected board, initializes operating mode. Required before any access to a channel. Capture buffers are allocated in the host's memory as a result of this call.

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`). See `s817.h` for constants definitions.

`aChan`

Index of the input video channel being addressed (1 .. 16).

`pMode`

Pointer to `MODE` structure defining required operating mode.

S817_CloseChannel

`ECODE S817_CloseChannel (int board, int aChan)`

Closes a previously opened channel on a selected board. Required to properly release the resources (capture buffers) allocated by `S817_OpenChannel`.

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`aChan`

Index of the input video channel being addressed (1 .. 16).

S817_SetVideoMode

`ECODE S817_SetVideoMode(int board, int aChan, VIDEO_MODE *pVMode)`

Sets video mode for a selected channel on a selected board. Changes in video mode affecting `tvFormat` and/or `jpgSize` may cause capture delays of 1-2 frames. Other changes do not cause any capture delays.

`board`
    Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`aChan`
    Index of the input video channel being addressed (1 .. 16).

`pVMode`
    Pointer to `VIDEO_MODE` structure defining required video mode.

S817_SetCaptureMode

`ECODE S817_SetCaptureMode (int board, int aChan, CAPTURE_MODE *pCMode)`
    Sets capture mode for a selected channel on a selected board.

`board`
    Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`aChan`
    Index of the input video channel being addressed (1 .. 16).

`pCMode`
    Pointer to `CAPTURE_MODE` structure defining required capture mode.

S817_GetBuffer

`ECODE S817_GetBuffer (int board, int aChan, CAP_BUFFER *cBuf, int capType)`
    Copies the pointers to captured data into the `CAP_BUFFER` structure . New capture to this buffer is not allowed and the data remains unchanged until the buffer is released back to the driver (see `S817_ReleaseBuffer` below).

`board`
    Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`aChan`
    Index of the input video channel being addressed (1 .. 16).

`cBuf`
    Pointer to `CAP_BUFFER` structure.

`capType`
    Type of captured data requested: bitmap or JPEG. Must be either `CAP_JPG` or `CAP_BMP.`

S817_ReleaseBuffer

`ECODE S817_ReleaseBuffer (int board, int aChan, int bufId, int capType)`

Releases a buffer back to the driver. Capture to this buffer is allowed.

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`aChan`

Index of the input video channel being addressed (1 .. 16).

`bufId`

Buffer index. See `CAP_BUFFER` structure.

`capType`

Type of buffer to be released: bitmap or JPEG. Must be either `CAP_JPG` or `CAP_BMP.`

S817_BlockDone

`ECODE S817_BlockDone(int board, int to, int *aChan, int *capType)`

Waits for the board to capture requested data. It is a blocking function, i.e. it does not consume CPU time when waiting.

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`to`

Timeout value in milliseconds. The function returns after this period of time even if the selected channel is not ready. The return value of the function in this case is non-zero. If `timeout` is set to 0, the function checks the capture status and returns immediately.

`aChan`

A pointer to a variable receiving the index of the channel having data. The driver implements a round-robin algorithm, so that all the channels are fairly represented.

`capType`

A pointer to a variable receiving the type of data available for a given channel. Can be `CAP_JPG` or `CAP_BMP`, or their logical OR.

S817_SetVideoSwitch

`ECODE S817_SetVideoSwitch(int board, int *switchOut)`

Controls the analog video crosspoint switch.

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`switchOut`

An array of 4 integers. Each member of `switchOut` defines an input video channel connected to the corresponding output video channel. For example, setting `switchOut[2]` to 5 connects video input 5 to video output `2.`

S817_SetVideoOut

`ECODE S817_SetVideoOut(int board, unsigned int outEn)`

Controls the output switch of the analog video crosspoint switch. This feature allows connecting outputs from multiple boards to the same video monitor (in parallel).

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`outEn`

Four lower bits of this parameter control the output switches of the video crosspoint switch. Bit 0 controls output 0, bit 1 controls output 1, etc. Setting a bit to 1 closes the switch (connects the signal to the output connector of the 817 board). Setting a bit to 0 opens (disconnects) the switch.

S817_SetDateTime

`ECODE S817_SetDateTime(int board)`

Copies system date and time settings from the host to the board. The date and/or time values may be used for on screen display (OSD).

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

S817_SetOsdMode

`ECODE S817_SetOsdMode(int board, int aChan, OSD_MODE *osd)`

Controls OSD mode for a selected input channel.

`board`

Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`aChan`

Index of the input video channel being addressed (1 .. 16).

`osd`

Pointer to `OSD_MODE` structure.

Troubleshooting and Debugging Tools

This section describes data types and functions used for obtaining various status information from the 817 board. Though not required to be used during normal operation, those could provide assistance in resolving various performance issues.

CHAN_STATUS

```
typedef struct {
    int             lastCmd;
    unsigned int    chanTick;
    int             putToSleep;
    int             lockPend;
    int             fvidStat;
    unsigned int    noFreeBufJpg;
    unsigned int    noFreeBufBmp;
} CHAN_STATUS;
```

Defines a status structure for a capture channel. Returned as part of `SYS_STATUS` structure by `S817_GetStatusInfo` function.

`lastCmd`
: Last command received by a channel.

`chanTick`
: Value of system 1 ms timer corresponding to the last frame captured by a channel (frame capture rate always equals the video source frame rate regardless of the frame decimation setting which affects the output frame rate).

`putToSleep`
`lockPend`
: A non-zero value of any of those indicates a critical hardware error on a channel.

`fvidStat`
: A value of 1 indicates normal operation. Other values indicate a critical hardware error.

`noFreeBufJpg, noFreeBufBmp`
: This value is a running count of the conditions when there was no free buffer available on the host to accept data from the board. It does not indicate a critical error, but could be used to gauge the overall performance of the system. Usually a small number of those cases is registered when the application starts. Steady increase of this value indicates an application's performance bottleneck and results in some decrease of capture rate.

SYS_STATUS

```
typedef struct {
  CHAN_STATUS      chanData[4];
  unsigned int     extErrAddr;
  unsigned int     dspAddr;
  unsigned int     fwVer;
  unsigned int     inPciRead;
  unsigned int     inPciWrite;
  unsigned int     hostLate;
  unsigned int     reserved[8];
} SYS_STATUS;
```

Defines system status structure. System status is obtained with the help of `S817_GetStatusInfo` function for a group of channels (1 .. 4, 5 .. 9, etc.).

`chanData[4]`
: An array of 4 structures of `CHAN_STATUS` type (see above), one for each of 4 channels belonging to a group.

`extErrAddr`
: Internal use, do not modify.

`dspAddr`
: Internal use, do not modify.

`fwVer`
: Firmware version (major in bits [31:16], minor in bits [15:0]).

`inPciRead`
: The board is attempting to read data from host. Normally is 0.

`inPciWrite`
: The board is attempting to write data to host. Normally is 0.

`reserved[8]`
: Internal use, do not modify.

EXTERR

```
typedef struct {
  unsigned int       tick;
  unsigned int       ecode;
  unsigned int       reserved[2];
} EXTERR;
```

Defines the extended error structure. The extended error information for each capture channel is kept in an array of `EXTERR` elements of the size `SYS_EXTERR_BUF_SIZE` (defined in `s817.h`, currently 64). This array is copied from the board to the buffer in the host's memory by a call to `S817_GetExtErrInfo` (see below). The first element of the array is used to keep track of the total number of errors registered for a given capture channel (in `.tick` member), the rest are used as a ring buffer. Each valid entry of the ring buffer keeps the information on the type of the error (`.ecode`) and the time of occurrence in units of the 1 ms timer (`.tick`). Let's assume, for example, that the buffer is `err[64]`, where `err` is of `EXTERR` type. If `err[0].tick` is 17, then the elements `err[1]` to `err[18]` contain valid error information. If `err[0].tick is 100`, then members `err[1]` to `err[63]` contain error information for the last 63 out of total 100 errors that were registered for a selected channel.

`tick`
: Timer tick corresponding to the occurrence of the error.

`ecode`
: Error code. The error codes returned in `EXTERR` structure are different from those returned by the API functions. Please record the diagnostics output prior to contacting the technical support.

`reserved[2]`
: Internal use, do not modify.

S817_GetStatusInfo

`ECODE S817_GetStatusInfo (int board, int dsp, SYS_STATUS *sysStat)`

Gets status information for a group of 4 channels.

`board`
: Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`dsp`
: Index of the group of channels being addressed (0 .. 3). Use 0 to obtain status for channels 1-4, 1 for channels 5-9, etc.

`sysStat`
: Pointer to `SYS_STATUS` structure accepting the data.

S817_GetExtErrInfo

`ECODE S817_GetExtErrInfo (int board, int dsp, int errChan, EXTERR *err)`

Gets extended error information from the board.

`board`
: Index of the board being addressed (1 .. `SYS_MAXBOARDS`).

`dsp`
: Index of the group of channels being addressed (0 .. 3). Use 0 to obtain status for channels 1-4, 1 for channels 5-9, etc.

`errChan`
: Index of the requested channel within a group (0..4). For example, to obtain extended error information for channel 7, set `dsp` to 1 and `errChan` to 2. The value of 4 within each group selects extended error information of the communication channel common for four capture channels within one group.

`err`
: An array of `SYS_EXTERR_BUF_SIZE` elements of `EXTERR` type accepting the data. Must be allocated by the application software before a call to `S817_GetExtErrInfo`.