PCI Express 16-channel H.264 Encoder
Model 819
Linux Software Manual

Ver.1.0.1 | March 2015

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

Terminology

channel	video input. Each 819 has 16 input channels. Multiple boards are enumerated contiguously, so that channels are addressed by a single channel number (1-based): board #1 containing channels 1-16, board #2 – channels 17-32, etc.
stream	output of model 819 in a specific format corresponding to a channel. One channel may produce multiple streams, e.g. H.264 primary and secondary streams with different settings, JPEG stream, alarm stream, etc.
capture node	a set of streams produced by one or more channels combined to facilitate capture within a single application thread. Each node receives an individual signal from the driver once data is available for any of the attached streams. Depending on the desired application architecture specific streams may be combined into nodes to simplify data processing. For example, one node may combine archived streams and be handled in the archiving thread. Another node may combine streams that are being presented live and be handled by a streaming thread. Extreme cases of one thread per stream or one thread handling all streams are also possible, though may not be practical.

Limitations

The following data streams are currently ***not supported***:

- JPEG;
- audio;
- preview;
- caption.

Data Types

For complete reference to 819 data types please refer to s819.h.

Captured resolutions

```
typedef enum {
    RES_QCIF = 0,
    RES_CIF,
    RES_D1,
    RES_MAX,           //for array sizing only
} S819_RESOLUTION;
```

	NTSC	PAL
RES_QCIF	176x120	176x144
RES_CIF	352x240	352x288
RES_D1	720x480	720x576

TV standards

```
typedef enum {
    TVS_PAL,
    TVS_NTSC,
    TVS_MAX,        //for array sizing
} S819_TVSTANDARD;
```

Interlacing

```
typedef enum {
    MODE_INTERLACE = 0,
    MODE_PROGRESSIVE,
    MODE_MAX,           //for array sizing
} S819_INTERLACE;
```

Currently the only supported value is `MODE_INTERLACE`.

Bitrate control mode

```
typedef enum {
    BITRATE_H264_NO_RC,    //No rate control
    BITRATE_H264_CBR,      //constant bitrate
    BITRATE_H264_VBR,      //variable bitrate
    BITRATE_MAX,           //for array sizing only
} S819_BR_CTRL;
```

`BITRATE_H264_NO_RC` – no bitrate control. Actual bitrate depends on video contents.
`BITRATE_H264_CBR` – constant bitrate. Encoder makes best effort to keep bitrate constant regardless of video contents.
`BITRATE_H264_VBR` – variable bitrate. Output bitrate varies, but does not exceed the set value.

Capture mode

```
typedef struct {
    S819_TVSTANDARD     tvs;
    int                 bright;
    int                 contrast;
    int                 saturation;
    int                 hue;
    int                 sharpness;
} CHAN_MODE;
```

bright	brightness control. Valid values: -128 to 127. Default 0.
contrast	contrast control. Valid values: 0 to 255. Default 0x64. Adjustment rate – 1% per step.
saturation	saturation. The range of adjustment is 0 to 200%. Default value of 0x80 corresponds to 100%.
hue	color hue (for NTSC input only). Range: -128 to 127. Default 0. Two lower bits have no effect. Positive values increase greenish tones, negative – purplish.
sharpness	controls sharpness enhancement of video. Valid values are 0 (no enhancement) to 15 (most enhancement).

Stream mode

```
typedef struct {
    S819_RESOLUTION     resolution;
    int                 fps;
    S819_INTERLACE      interlace;
    int                 gopSize;
    S819_BR_CTRL        brCtrl;
    int                 bitrate;
} STREAM_MODE;
```

resolution	resolution of the captured stream.
fps	capture rate (frames per second). Range: 1-25 (PAL), 1-30 (NTSC) in 1 fps increments. This parameter does not affect JPEG capture rate, which is fixed at 2 fps.
interlace	interlacing mode.
gopSize	GOP size. 1 to 256, default 60. Irrelevant for JPEG stream.
brCtrl	Bitrate control mode. Irrelevant for JPEG stream.
bitrate	Requested bitrate (in bits per second). Please refer to s819.h for allowed ranges. Irrelevant for JPEG stream, JPEG quality is fixed.

Stream type

```
typedef enum {
    STREAM_H264_PRI = 0,
    STREAM_H264_SEC,
    STREAM_JPEG,
    STREAM_YUV,
    STREAM_AUDIO,
    STREAM_ALARM,
    STREAM_CAPTION,
    STREAM_MAX
} S819_STREAM_TYPE;
```

STREAM_H264_PRI	primary H.264 stream.
STREAM_H264_SEC	secondary H.264 stream. Resolution must be lower than that of the primary stream.
STREAM_JPEG	JPEG stream.
STREAM_YUV	preview stream.
STREAM_AUDIO	audio stream.
STREAM_ALARM	alarm stream.
STREAM_CAPTION	caption (on-screen display, OSD) stream (input).

Capture channel

```
typedef struct {
    int                ctrl;
} CONTROL;
```

Reserved.

```
typedef struct {
    CHAN_MODE          cMode;  //channel capture mode
    STREAM_MODE        psMode; //primary H.264 stream
    STREAM_MODE        ssMode; //secondary H.264 stream
    STREAM_MODE        jsMode; //JPEG stream
    STREAM_MODE        vsMode; //preview stream
    STREAM_MODE        asMode; //audio stream
    STREAM_MODE        lsMode; //alarm stream
    STREAM_MODE        csMode; //caption stream (in)
    CONTROL            ctrl;   //reserved
} S819_CHAN;
```

cMode	channel capture mode. Applies to all streams captured from a channel.
psMode	primary H.264 stream mode.
ssMode	secondary H.264 stream mode.
jsMode	JPEG stream mode.
vsMode	preview stream mode.
asMode	audio stream mode.
lsMode	alarm stream mode.
csMode	caption stream mode.
control	reserved.

Capture buffer

```
typedef enum {
    H264_PSLICE     = 1,
    H264_ISLICE     = 5,
    H264_IDRSLICE   = 6,
    H264_SPS        = 7,
    H264_PPS        = 8
} SLICE_TYPE;

typedef enum {
    ALARM_VIDEO_LOST = 0x01,
    ALARM_NIGHT = 0x02,
    ALARM_BLIND = 0x04,
    ALARM_STDCHG = 0x08,
    ALARM_MV = 0x10
} ALARM_TYPE;


typedef union {
    SLICE_TYPE           slice;
    ALARM_TYPE           atype;
} S819_SUBTYPE;

typedef struct {
    int                 chan;
    S819_STREAM_TYPE    stream;
    S819_SUBTYPE        subtype;
    char                *buf;
    int                 length;
    unsigned int        frm;
    unsigned int        pts;
    void                *rsv;
} BUFFER;
```

chan	channel number, 1 through maximum available channels based on the number of boards detected, but no more than 64.
stream	type of stream captured into buffer.
subtype	stream subtype
buf	pointer to buffer data.
length	length of valid data currently in buffer, bytes.
frm	frame number (H.264, JPEG streams).
pts	time stamp (H.264, JPEG streams).
rsv	Reserved.

`S819_SUBTYPE` provides additional information about the streams depending on stream's type. For H.264 streams it is current slice type, for alarm stream it is alarm type. Application should first determine the stream type using `.stream`, then use `.subtype` based on context.

Miscellaneous types

```
typedef int     ECODE;              //error code
typedef void    *HCNODE;            //capture node handle
```

SDK Functions

All functions returning an error code (type ECODE) return ECODE_OK, if success. Any other value indicates an error.

S819_Enumerate

```
ECODE S819_Enumerate (int *nChans)
```

Enumerates existing 819 boards and sets nChans to the number of available capture channels. This function needs to be called before any other SDK function.

S819_SetMode

```
ECODE S819_SetMode (int chan, S819_CHAN *s819chan)
```

If the argument *chan* is between 1 and the number of channels detected by S819_Enumerate, the function sets capture mode for a specified channel (*chan*) based on the values set in *s819chan*.
If *chan* = 0 the structure pointed to by *s819chan* is filled with default values.

S819_CreateCnode

```
HCNODE S819_CreateCnode (void)
```

Creates a new empty capture node. Streams are attached to a node by using *S819_AttachStreams* function.
Returns: a capture node handle or NULL in case of an error.

S819_DeleteCnode

```
void S819_DeleteCnode (HCNODE hCnode)
```

Deletes a capture node.

S819_AttachStreams

```
ECODE S819_AttachStreams (HCNODE hCnode, int chan, int streams)
```

Attaches stream(s) produced by a selected channel to a capture node.

hCnode	a valid capture node handle.
chan	selected channel number (1-based).
streams	streams bitmask. To select a stream, set a bit of the parameter *streams* to 1 by using an SMASK macro. For example,

`streams = SMASK(STREAM_H264_PRI);` or
`streams = SMASK(STREAM_H264_PRI) | SMASK(STREAM_H264_SEC);`

S819_StartStreams

```
ECODE S819_StartStreams (HCNODE hCnode, int chan, int streams)
```

Starts capture of a combination of streams attached to a capture node. Other streams that may be attached to the same capture node are not affected.

hCnode	a valid capture node handle.
chan	selected channel number (1-based).
streams	streams bitmask corresponding to desired stream combination. To select a stream, set a bit of the parameter *streams* to 1 by using an SMASK macro. For example, `streams = SMASK(STREAM_H264_PRI);` or `streams = SMASK(STREAM_H264_PRI) | SMASK(STREAM_H264_SEC);`

S819_StopStreams

```
ECODE S819_StopStreams (HCNODE hCnode, int chan, int streams)
```

Stops capture of a combination of streams attached to a capture node. Other streams that may be attached to the same capture node are not affected.

hCnode	a valid capture node handle.
chan	selected channel number (1-based).
streams	streams bitmask corresponding to desired stream combination. To select a stream, set a bit of the parameter *streams* to 1 by using an SMASK macro.

S819_StartAll

```
ECODE S819_StartAll (HCNODE hCnode)
```

Starts capture of all streams attached to a selected capture node.

S819_WaitBuffer

```
ECODE S819_WaitBuffer (HCNODE hCnode, BUFFER *buffer)
```

A blocking function that waits for data to become available for a selected capture node. The function times out if the data is not ready in 5 seconds and returns an error code. If the data is available the function fills in the structure pointed to by *buffer* with data parameters. A buffer needs to be returned back to the driver by calling `S819_ReleaseBuffer`.

S819_ReleaseBuffer

`ECODE S819_ReleaseBuffer (BUFFER *buf)`

Returns a buffer to the driver. It is essential to call this function after the data has been handled by the application. Once the buffer is released members of the `*buf` become invalid.

S819_Close

`void S819_Close (void)`

SDK cleanup. Must be called before an application terminates. Must be the last call to the SDK.

S819_SetXPSwitch

`ECODE S819_SetXPSwitch (int brd, int inp, int out)`

Controls output video crosspoint switch. Connects selected input to selected output.

brd	board index (0-based).
inp	selected input channel number (1-16).
out	selected output (0-3).

S819_SetXPOut

`ECODE S819_SetXPOut (int brd, int mask)`

Enables/disables outputs of video crosspoint switch. This allows connecting outputs of multiple boards together and enabling one of them at a time.

brd	board index (0-based).
mask	*bits 0-3 control outputs 0-3 (1 – enabled, 0 – disabled).*

Demo application

A simple demo application (common819) is provided to illustrate the use of the SDK. The source file (common819.c) is included. The application allows capture of multiple H.264 streams and JPEG frames from any number of available channels. The H.264 data from each stream/channel is written into a separate file. The files are of a fixed size, so once the end of the file is reached the recording continues from the start of the file. This allows execution of the demo for an indefinite period of time without running out of the disk space. The JPEG files are captured at a low rate, overwriting the old data for each channel.

SDK Installation

1. Decompress the archive.
2. Change directory to the top level of the SDK.
3. Run “make”. That makes the driver and the demo application. The 819 SDK is distributed in the form of a library, libs819.a.
4. Run “sudo make load” to load the driver.
5. To run the demo application change directory to /s819/common819, run “sudo ./common819”.

Release History and Notes

SDK Release	Notes
Ver.1.0.1, March 2015	“Limitations” section added.
Ver.1.0.0, July 2013	Initial release. The following SDK features are not supported: 1. Audio capture. 2. Preview stream. 3. Caption (OSD) stream.