SENSORAY CO., INC.

S615 Software Development Kit version 1.0

January 2002

7313 SW Tech Center Dr.
Tigard, OR 97223
Phone 503.684.8073 • Fax 503.684.8164
sales@sensoray.com
www.sensoray.com

Table of Contents

Introduction

Model 615 is a multifunctional video capture board. It allows simultaneous capture of JPEG compressed full resolution images and scaled down uncompressed images from 16 video inputs at a combined rate of close to 30 (25 for PAL/SECAM) frames per second. The board has a 16-to-4 video crosspoint switch connected to 4 image capture channels (decoders), and one JPEG compression engine. The crosspoint switch is used to connect any 4 of 16 video input signals to the decoders. The output of any decoder can be connected to the compression engine under software control. While the image is compressed and written to the output buffer, a scaled down uncompressed image is also being written to the buffer. Once the output of both compressed and uncompressed images is complete the board is ready to capture again.

A text caption of 64 characters may be overlayed on the compressed image. The position, size, and color of the caption are controlled through software.

The following general approach is implemented in the software. During the initialization the driver creates 2 queues: a command queue and a buffer queue. The application puts commands in the command queue. Each command controls the following features: whether JPEG capture is enabled or disabled, which channel (decoder) the image is captured from, which channel of the crosspoint video switch the selected decoder is switched to after the capture is complete, whether the caption (text overlay) is on or off, caption color selection, interlacing mode (currently only one mode is available). See the description of COMMAND structure below.

The driver captures the uncompressed and compressed images into a buffer (see the description of BUFFER structure below) and puts the buffer into a buffer queue, from which the application retrieves the completed buffers as necessary. Both queues are organized as FIFO's (first in first out).

The following algorithm can be implemented to poll all 16 channels in a cycle (assuming that initially input1 is connected to channel0, input2 to channel1, input3 to channel2, and input4 to channel3):
capture from channel0 (input1), switch channel0 to input5;
capture from channel1 (input2), switch channel1 to input6;
capture from channel2 (input3), switch channel2 to input7;
capture from channel3 (input4), switch channe3 to input8;
capture from channel0 (input5), switch channel0 to input9;
capture from channel1 (input6), switch channel1 to input10;
capture from channel2 (input7), switch channel2 to input11;
capture from channel3 (input8), switch channel3 to input12;
capture from channel0 (input9), switch channel0 to input13;
capture from channel1 (input10), switch channel1 to input14;
capture from channel2 (input11), switch channel2 to input15;
capture from channel3 (input12), switch channel3 to input16;
capture from channel0 (input13), switch channel0 to input1;
capture from channel1 (input14), switch channel1 to input2;
capture from channel2 (input15), switch channel2 to input3;
capture from channel3 (input16), switch channel3 to input4.

The application may choose a different algorithm if necessary, however one important condition has to be met: whenever a new video input is connected to the decoder channel, a period of at least 3 video frames has to be allowed for re-synchronization before capturing from this channel again. In the above example this condition is met by means of going through four channels in a cycle.

Software Installation

It is recommended to install the SDK before installing the board(s) into your PC. To install the SDK run setup.exe from the installation disk.

Windows98

The first time you boot your system after model 615 frame grabber is plugged in, Windows will launch "Add new hardware Wizard". Select "Display a list of all drivers", click "Next", check "Floppy drive" and uncheck all other check boxes, and insert a S615 SDK distribution disk into the floppy drive. Click "Next". Windows should automatically complete the rest of the installation procedure.

WindowsNT

To install the SDK run setup.exe from the installation disk.

Windows2000/XP

The first time you boot your system after model 615 frame grabber is plugged in, Windows will launch "Found new hardware Wizard". Select "Install from a list...", click "Next", check “Include this location in the search”, click on “Browse”, and select a floppy drive. Insert a S615 SDK distribution disk into the floppy drive and click “OK”. When a message appears saying that the driver does not have a Microsoft digital signature, click “Continue anyway”.

Note: under WindowsXP installing the driver before the SDK is installed may result in failure. In this case “Found new hardware…” message is not displayed any more. To install the driver properly go to Control Panel – System – Hardware – Device manager, locate the Sensoray 615 board under “Sound, video and game controllers” or “Other devices”, right click on it, select “Properties”, select “Driver” tab, and click on “Update driver”. Follow the steps described above.

Building an application with s615.dll

Files to be included in the project

The following files are distributed with the SDK:

- `s615.h` – contains data types and constants definitions;
- `s615f.h` – contains exported functions declarations;
- `s615app.c` – contains exported functions and helper functions definitions.

When building an application with `s615.dll`, it is necessary to include `s615app.c` in the project.

IMPORTANT:
If MFC is used, an option "Not using precompiled headers" must be set in the project settings for `s615app.c.`

All modules containing calls to the `s615.dll` functions must include `s615f.h`. Please refer to the sample source code for an example of building an application with S615 SDK.

Data types reference

System structure PCI

```
typedef struct {
    DWORD boards;
    DWORD mask;
    DWORD PCIslot[SYS_BOARDS];
} PCI;
```

The PCI structure contains information about the frame grabber boards identified by the system.

boards
Number of supported boards identified by the system.

mask
Board selection mask.

PCIslot
Array of slot numbers.

The PCI structure is initialized by S615_InitSystem function. The system constant SYS_BOARDS determines the maximum number of frame grabbers supported, and is defined in `s615.h`. The *boards* member is set by S615_InitSystem to the number of model 615 boards detected in the system. The *mask* member has to be set before the call to S615_InitSystem and determines which of the detected boards are required to be initialized by the calling process. A value of 1 in bits 0-3 of *mask* selects a corresponding board to be initialized. For example, to initialize boards 0 and 2, the value of *mask* has to be set to 5. The enabling bit position corresponds to the value of index by which the board is referenced further on. In the previous example, the index values would be 0 and 2. If another application has to use board 1, the mask value has to be set to 2, and the index will be 1. Note that no matter which boards are selected by *mask*, the *boards* and *PCIslot[]* members will reflect the actual number of boards in the system.

PCIslot member represents a PCI slot number for a given board. The PCI slot number is generated by Windows, and usually is not the same as the ordinal number of the slot. The PCI slot number is provided for reference only.

Image buffer structure BUFFER

```
typedef struct {
    void *pBmp;       //pointer to bitmap (uncompressed) buffer;
    void *pJpeg;      //pointer to compressed data buffer;
    DWORD jpegsize;   //actual size of jpeg data;
    LPBITMAPINFO lpbmi;     //pointer to BITMAPINFO structure;
    int xpchan;       //one of 16 channels (1-16) captured from;
    struct _timeb timestamp;      //timestamp structure;
    int videopresent; //video present flag;
    DWORD cqempty;    //command queue empty (cumulative);
    DWORD bqfull;     //buffer queue full (cumulative);
    DWORD incomplete; //compression incomplete (cumulative);
    DWORD zerror;     //compression error (cumulative);
} BUFFER;
```

The *lpbmi* member is provided to assist in displaying the bitmap, if necessary. Note: YCrCb bitmaps cannot be displayed using Windows API functions.

The *xpchan* member is provided to assist in identifying the channel that the image was captured from.

The *timestamp* structure provides the system time reflecting the start of the buffer's acquisition. Note that the millisecond value (timestamp.millitm) has the system timestamp granularity of about 55 ms (which means that some consecutively captured frames may have the same timestamp value).

The *videopresent* flag returns the status of selected decoder channel at the start of capture; it is set to 1, if the video is detected, and reset to 0 if no video is present. It is provided as means of distinguishing between a black field captured in case of no video present, and a dark video. This flag is not real time. The state of this flag is ahead of the captured video by one frame (due to the image buffering on the board). It is recommended to "integrate" the state of this flag over several frames before making a decision on whether video is present, or not.

Command structure COMMAND

```
typedef struct {
    int jpegon;       //JPEG capture ON/OFF
    int channel;      //one of 4 channels to capture from (0-3);
    int xpchan;       //input channel to switch to when done (1-16);
    int captionmode;  //caption mode: ON/OFF, caption color;
    int interlacemode;//interlaced, line doubling, line interpolation
    void *pCaption;   //pointer to caption data buffer (64 bytes);
    int offset;       //channel number offset;
} COMMAND;
```

Note: to assist in insertion of time stamp and the current channel number into the caption, the following formatting commands are available:

`^n`	New line
`^d`	Date stamp: Mon Oct 01 2001
`^tX`	Time stamp: 17:05:33.25, where X=0,1,2 controls the number of digits after the decimal point.
`^c`	Channel number (1-16 + *offset*). The *offset* member is provided to allow display of channel numbers > 16 in case of multiple boards (e.g. to display second board's channels as 17-32, rather than 1-16, set *offset* to 16).

Operation mode data structure MODE

```
typedef struct {
    DWORD format;           //NTSC, PAL
    DWORD bmpflip;          //bmp capture lines order
    DWORD bmpcolor;         //bitmap color format:RGB,monochrome,YCrCb
    DWORD bmpsize;          //bitmap size: 128x96 or 256x192
    DWORD jpgsize;          //jpeg size
    DWORD captionformat;    //format of the caption window
    DWORD captionpos;       //position of the caption window
    DWORD fontsize;         //caption font size
    DWORD compfactor;       //compression factor
    DWORD xpinit;           //initial crosspoint switch setting
} MODE;
```

The MODE structure contains information about the operation mode of the frame grabber. The settings that are controlled by MODE cannot be changed on a frame to frame basis.

format
Defines the format of the video signal:
FORMAT_NTSC – NTSC (60Hz) video;
FORMAT_PAL – PAL or SECAM (50Hz) video.

bmpflip
Uncompressed bitmap lines order:
FLIP_OFF – bitmap is stored in memory in “natural” lines order (top line first). If displayed by Windows API functions, the image will look upside down.
FLIP_ON – bitmap is stored in memory in Windows lines order (bottom line first).

bmpcolor
Color format of the uncompressed bitmap:
COLOR_MONO – monochrome, 1 byte per pixel;
COLOR_RGB – color, 3 bytes per pixel;
COLOR_YCRCB – color, 2 bytes per pixel, YCrCb (byte sequence: CbYCrY...);
COLOR_YCRCB_BS – color, 2 bytes per pixel, YCrCb (byte sequence: YCbYCr...).

bmpsize
Uncompressed bitmap size:
BMP_SIZE_LRG – large bitmap (256x192);
BMP_SIZE_SML – small bitmap (128x96).

jpgsize
Compressed (JPEG) image size:
JPG_SIZE_FULL - full size JPEG, 704x480 (NTSC) or 704x576 (PAL);
JPG_SIZE_FULL_C – clipped full size JPEG, 640x480 (NTSC), not valid for PAL;
JPG_SIZE_HALF - half size JPEG, 352x240 (NTSC) or 352x288 (PAL);
JPG_SIZE_HALF_C – clipped half size JPEG, 320x240 (NTSC), not valid for PAL;

captionformat
Caption window format:

CAP8X8 – 8 characters x 8 lines;
CAP16X4 – 16 characters x 4 lines;
CAP32X2 – 32 characters x 2 lines.

captionpos

Caption window position:
CP_UL – upper left corner;
CP_UR – upper right corner;
CP_LL – lower left corner;
CP_LR – lower right corner.

fontsize

Caption font size:
CF_SMALL – small font (8x16 pixels);
CF_LARGE – large font (16x32 pixels).

compfactor

JPEG compression factor. Determines the level of JPEG compression. Greater values yield higher compression (lower quality). The minimum allowed value is 0xA0. That results in the 704x480 JPEG file size of around 40-50KB (depending on the subject). A *compfactor* value of 0x100 results in a 30KB JPEG file (for the same subject).

xpinit

Determines the initial crosspoint switch setting at the start of the acquisition. The format of the *xpinit* member is:
bits 0-7 – input crosspoint channel (1-16) for channel 3,
bits 8-15 – input crosspoint channel (1-16) for channel 2,
bits 16-23 – input crosspoint channel (1-16) for channel 1,
bits 24-31 – input crosspoint channel (1-16) for channel 0.

Functions reference

S615_InitSystem

```
ECODE S615_InitSystem (
    PCI *pPci         //pointer to a PCI structure
);
```

Parameters

pPci
Pointer to the structure of PCI type.

Return values

Returns 0 in case of success, or an error code (a list of error codes is included in S615.h).

Notes

The function initializes the driver, searches for, and initializes all boards supported by the SDK. This function has to be called **only once** per application. The system resources allocated by S615_InitSystem are released by a call to S615_CloseSystem. A member of PCI structure *mask* has to be set to allow board(s) initialization before a call to S615_InitSystem. See the description of the PCI structure for details.

S615_CloseSystem

```
void S615_CloseSystem (
    void
)
```

Parameters

None.

Return values

None.

Notes

S615_CloseSystem releases all resources allocated by the calls to S615 functions. A call to this function does not affect the boards not selected before a call to S615_InitSystem.

S615_SetMode

```
ECODE S615_SetMode (
    int index,         //board index
    MODE *pMode        //pointer to MODE variable
)
```

Parameters

index
A value by which the boards are addressed. The board selected by *index* has the PCI slot value of PCI.PCIslot[index]. The value of index has to be between 0 and PCI.boards.
pMode
Address of the variable (of MODE type) defining the frame grabber operation mode.

Return values

Returns 0 in case of success, or an error code.

S615_CmdQueueFull

```
BOOL S615_CmdQueueFull (
    int index   //board index
)
```

Parameters

index
Board index.

Return values

Returns TRUE if command queue is full, FALSE otherwise.

Notes

The function has to be called before placing a command into the queue.

S615_QueueCmd

```
void S615_QueueCmd (
    int index          //board index
    COMMAND *pCmd      //pointer to the COMMAND structure
)
```

Parameters

index
Board index.
pCmd
Address of the COMMAND structure.

Return values

None.

S615_BufferReady

```
BOOL S615_BufferReady (
    int index         //board index
)
```

Parameters

index
Board index.

Return values

TRUE if buffer queue is not empty, FALSE otherwise.

S615_WaitBuffer

```
ECODE S615_WaitBuffer (
    int index,        //board index
    BUFFER *pBuf,     //pointer to BUFFER structure
    DWORD timeout     //timeout value (milliseconds)
)
```

Parameters

index
Board index.
pBuf
Address of the BUFFER structure to receive the data.
timeout
The length of time after which the function returns if no buffers were available.

Return values

Returns 0 in case of success, or an error code in case of a time out.

Notes

S615_WaitBuffer does not consume CPU time while waiting.

S615_ReleaseBuffer

```
void S615_ReleaseBuffer (
    int index,        //board index
    BUFFER *pBuf,     //pointer to BUFFER structure
)
```

Parameters

index
Board index.
pBuf
Address of the BUFFER structure to be released.

Return values

None.

Notes

S615_ReleaseBuffer is used to signal the driver that the application is done with a buffer, and it may be used for capture.