SENSORAY CO., INC.

S617
Software Development Kit
version 2.0

July 2003

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 617 is a multifunctional video capture board. It allows simultaneous capture of JPEG compressed full or half resolution images (640x480 or 320x240) and scaled down uncompressed images (256x192 or 128x96) from 16 video inputs at a combined rate of 30 frames per second (NTSC/PAL/SECAM). The board has a 16-to-4 input video crosspoint switch connected to 4 image capture channels (decoders), and one JPEG compression engine. The crosspoint switch connects 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 (in the host computer's RAM), 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 board also features a second 16-to-4 video crosspoint switch which allows to select any 4 of 16 input video signals for viewing on external TV monitors. Each output video channel can be turned on/off under software control to allow connecting multiple boards to a single monitor.

Model 617 provides 32 bits of digital I/O with each individual bit configurable as an input or an output. The digital I/O is read from and/or written to at the boards capture rate (once per every captured frame).

The following general approach is implemented in the software. During the initialization, the driver creates 2 queues: a command queue and a buffer queue (each 16 elements deep). The application puts commands in the command queue. Each command controls the following features:

- whether JPEG and/or bitmap capture is enabled or disabled;
- which input channel (camera) the image is captured from;
- whether the caption (text overlay) is on or off, caption color selection;
- brightness, contrast, saturation and hue settings for current capture;
- JPEG compression factor.

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

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 617 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 S617 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 617 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 S617 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 “Model 617 JPEG capture 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.

Performance issues

In most cases, an application using the 617 is saving JPEG files to the hard drive. In case this is done continuously (similar to the sample application Sample1 provided with the SDK), the hard drive write speed affects the board’s performance dramatically. Slow hard drives result in the high number of acquisition errors and dropped frames. To achieve the best performance it is recommended to use the fastest available hard drive operating mode for saving the data (UDMA/33 or UDMA/66). Tests had shown that PIO mode does not provide adequate performance. Note that under Windows NT PIO is the default hard drive operating mode, and switching to DMA mode may not be straightforward. Please refer to Microsoft Knowledge Base Article Q158873 for the information on enabling the hard drive DMA.

A utility application DiskTest is provided with the SDK to estimate hard drive performance in the target system. To achieve good 617 performance (a few capture errors per 100,000 captured frames), the disk write speed reported by the utility has to be around 15-20 MB/sec. Lower write

speeds may result in higher error rate. Write speeds of around 3MB/sec (typical for PIO mode) are usually unacceptable.

Building an application with s617.dll

Files to be included in the project

The following files are distributed with the SDK:

- `s617.h` – contains data types and constants definitions;
- `s617f.h` – contains exported functions declarations;
- `s617app.c` – contains exported functions and helper functions definitions.

When building an application with `s617.dll`, it is necessary to include `s617app.c` in the project. Please, do not try the following: `#include "s617app.c"`, it is not going to work! Instead, use "Add files to project".

IMPORTANT:
If MFC is used, an option "Not using precompiled headers" must be set in the project settings for `s617app.c`.

All modules containing calls to the `s617.dll` functions must include `s617f.h`. Please refer to the sample source code for an example of building an application with S617 SDK.

Data types reference

System structure PCI

```
typedef struct {
    DWORD boards;
    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.

PCIslot
Array of slot numbers.

The PCI structure is initialized by S617_InitSystem function. The system constant SYS_BOARDS determines the maximum number of frame grabbers supported, and is defined in s617.h. The *boards* member is set by S617_InitSystem to the number of model 617 boards detected in the system.

PCIslot member contains PCI bus (upper 16 bits) and PCI slot (lower 16 bits) numbers for a given board. The PCI slot number is generated by BIOS and/or Windows, and usually is not the same as the ordinal number of the slot. On different systems, the order in which the boards are enumerated (from the CPU outwards, or reverse) may be different. The PCI bus/slot number is provided for reference only.

All SDK functions that need to access an individual board would address it with the help of *index* parameter, which is an index of a selected board in *PCIslot* array.

Example:

```
PCI   pci;        //PCI struct
ECODE ecode;      //error code
int   i;          //board index

if (!(ecode = S617_InitSystem(&pci)) {
  for (i = 0; i < pci.boards; i++) {
    ecode = S617_OpenBoard (i, Callback, NULL);
    . . . . .
```

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;
    int vxp;          //controls "video out" crosspoint switch
    struct timeb timestamp; //timestamp structure;
    int valid;        //0 if the buffer contains invalid data
    int videopresent; //"video present" flag;
    int dioin;        //data read from digital I/O
    DWORD cqempty;    //command queue empty (cumulative);
    DWORD bqfull;     //buffer queue full (cumulative);
    DWORD incomplete; //compression incomplete (cumulative);
    DWORD zerror;     //compression error (cumulative);
    int compfactor;   //compression factor used for current capture;
    int bright;       //brightness setting used for current capture;
    int contrast;     //contrast setting used for current capture;
    int sat;          //saturation setting used for current capture;
    int hue;          //hue setting used for current capture;
} BUFFER;
```

pBmp, pJpeg

Pointers to captured data (pBmp and pJpeg) are used to access the data. Note that those are only valid untill the buffer is released by the application (see the description of S617_ReleaseBuffer function).

jpegsize

Actual size of the JPEG file image in the buffer. May be set to 0 in case of capture errors.

lpbmi

This value is provided to assist in displaying the bitmap, if necessary. Note: bitmap will not be displayed correctly for YCrCb bitmaps.

xpchan

Identifies the input channel that the image was captured from.

vxp

Current state of output crosspoint switch. See COMMAND structure for details.

timestamp

Returns the timestamp of the captured image. 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 granularity of about 55 ms (which means that some consecutively captured frames may have the same timestamp value). The following statement is necessary in order to use the *timestamp* structure:

```
#include <sys/timeb.h>.
```

valid
Non-zero in case of successful capture. Set to zero if capture or compression errors occurred, in which case no valid data is present.

videopresent
Returns the status of video on the input captured from. Zero if video is not present.

dioin
Status of digital I/O bits. Bits configured as inputs reflect the signal status on the corresponding pins of the DIO connector. For bits configured as outputs the last written value is returned.

cqempty
command queue empty: application did not provide a command in time, the board was idle for 1 frame. This member returns the cumulative number of the errors of this kind since the application start.

bqfull
no buffers available: application did not read the data from the buffers fast enough, the board could not get a free buffer, and was idle for 1 frame. Cumulative value.

incomplete
the board was unable to complete compression within 1 frame's time, usually because of PCI bus holdup, the data was discarded. Cumulative value.

zerror
JPEG compression error: the compression engine generated an error, usually because of PCI bus holdup, the data was discarded. Cumulative value.

compfactor, bright, contrast, sat, hue
Those 5 values return the image settings used in the COMMAND structure which was issued to capture current image.

Command structure COMMAND

```
typedef struct {
    int jpegon;       //JPEG capture ON/OFF
    int bmpon;        //bitmap capture ON/OFF
    int xpchan;       //input channel to capture from (1-16);
    int compfactor;   //compression factor to use for current capture;
    int bright;       //brightness setting to use for current capture;
    int contrast;     //contrast setting to use for current capture;
    int sat;          //saturation setting to use for current capture;
    int hue;          //hue setting to use for current capture;
    int captionmode;  //caption mode: ON/OFF, caption color;
    void *pCaption;   //pointer to caption data buffer (64 bytes);
    int offset;       //channel number offset;
    int vxp;          //video out crosspoint switch control word;
    int dioout;       //digital i/o write (out) data;
} COMMAND;
```

jpegon

Controls whether JPEG image is captured or not (JPEG_ON , JPEG_OFF, see s617.h).

bmpon

Controls whether bitmap image is captured or not (BMP_ON , BMP_OFF, see s617.h).

xpchan

Defines the input channel (1-16) to capture from. All other values are invalid.

compfactor

Defines the level of JPEG compression. Greater values yield higher compression (lower quality). The minimum recommended value is 0x80. That results in the 640x480 JPEG file size of around 50KB (depending on the subject). A compfactor value of 0x100 results in an approximately 30KB JPEG file (for the same subject). Setting the compression factor too low may result in increased number of capture errors. The recommended JPEG file size for 640x480 capture should not exceed 55K, in 320x240 mode – 35K.

bright

Brightness setting for current capture (normal: 0x80, max: 0xff, min: 0x00).

contrast

Contrast setting for current capture (normal: 0x44, max: 0x7f, min: 0x00).

sat

Saturation setting for current capture (normal: 0x44, max: 0x7f, min: 0x00).

hue

Hue setting for current capture (0x00, max positive: 0x7f, max negative: 0x80).

captionmode

Turns caption on/off, determines caption color. The constants defining the color are OR'ed with on/off constants (CAP_ON, CAP_OFF). See s617.h for details.

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).

vxp

Controls the output crosspoint switch. Bits [7:0] control output 0, [15:8] – output 1, [23:16] – output 2, [31:24] – output 3. A value between 1 and 16 connects the corresponding input to the output, a value of 0 turns the output off. Values above 16 are invalid.

dioout

Value to be written to digital I/O. Bits configured as inputs are not affected.

Operation mode data structure MODE

```
typedef struct {
    DWORD format;           //NTSC, PAL
    DWORD bmpflip;          //Windows or natural 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 diomask;          //digital i/o mask
    DWORD boardver;         //board version
} 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.
Note: The output (captured) image resolution is the same for both NTSC and PAL/SECAM sources. The capture frame rate is also the same (30 Hz).

bmpflip

Uncompressed bitmap lines order:
FLIP_OFF – bitmap is captured in “natural” lines order (top line first). If displayed by Windows API functions, the image will look upside down.
FLIP_ON – bitmap is captured 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, 640x480, both fields captured;
JPG_SIZE_HALF - half size JPEG, 320x240, one field captured.

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).
Note: Large font is only available in full (640x480) JPEG size mode (jpgsize = JPG_SIZE_FULL).

diomask

Defines the function of individual I/O bits: a value of 0 sets up the corresponding bit as output (bit 0 corresponds to DIO0 pin, etc.). On power up all I/O bits are configured as inputs.

boardver

Board version. The default value for a 16-input version is BOARD_S. For a 4-input version of the board BOARD_S4 has to be used. Other values are not supported.

Functions reference

S617_InitSystem

```
ECODE S617_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 s617.h).

Notes

The function initializes the driver, searches for all boards supported by the SDK. This function has to be called **only once** per application. The system resources allocated by S617_InitSystem are released by a call to S617_CloseSystem. See the description of the PCI structure for details.

S617_CloseSystem

```
void S617_CloseSystem (
    void
)
```

Parameters

None.

Return values

None.

Notes

S617_CloseSystem releases all resources allocated by the calls to S617 functions.

S617_OpenBoard

```
ECODE S617_OpenBoard (
    int index,                          //board index
    void(*Callback)(DWORD, void *),     //callback
    void *pObj                          //optional pointer
)
```

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 - 1.

Callback

A pointer to a callback function that is called when the data buffer is ready (capture complete). This function is being passed 2 arguments by the calling thread: a DWORD indicating an error code (0 in case of success), and a pointer to ***void***, which is the same pointer as the third argument of S617_OpenBoard function. This allows passing a parameter to the callback, such as the board's index (see sample application), or, in case of C++ programming, a ***this*** pointer.

The only non-zero value passed as a first argument to the callback may be ERR_ACQ_TIMEOUT, which means that the board stopped generating interrupts. This is a severe error, which requires reinitializing the board, or possibly a system reboot.

pObj

A pointer passed as a second parameter of a callback function.

Return values

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

S617_CloseBoard

```
void S617_CloseBoard (
    int index                           //board index
)
```

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 - 1.

Return values

None.

Note

A call to S617_CloseBoard disconnects the application from the selected board. Other applications are able to access the board after that. A call to S617_CloseSystem closes all open boards automatically.

S617_SetMode

```
ECODE S617_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.

Notes

This function has to be called at least once before the capture starts. Calling this function while capture is in progress stops the capture and resets command and buffer queues.

S617_CmdQueueFull

```
BOOL S617_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.

S617_QueueCmd

```
void S617_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.

S617_GetBuffer

```
ECODE S617_GetBuffer (
    int index,        //board index
    BUFFER *pBuf      //pointer to BUFFER structure
)
```

Parameters

index
Board index.
pBuf
Address of the BUFFER structure to receive the data.

Return values

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

S617_ReleaseBuffer

```
void S617_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

S617_ReleaseBuffer is used to signal the driver that the application is done with a buffer, and it may be used for capture. After this function is called the pointers in BUFFER structure become invalid.