Stretch DVR Display SDK

API User's Guide

Version 1.0

Last modified: September 18, 2008

Part #: RU-0133-0001-000

Contents

Chapter 1

Stretch DVR Display SDK (sdvr_ui_sdk)

1.1 Include

```
#include "sdvr_ui_sdk.h"
```

1.2 Introduction

In conjunction with the Stretch DVR SDK, there is a UI SDK library that lets you display raw video frames on the display monitor.

This is an optional library that can be added to your Stretch DVR Application development environment. If you are developing an embedded DVR Application or choose to use a different display API, you may choose not to include this library.

This document describes the Application Programming Interface (API) implemented in the UI SDK.

The SDK provides the ability to:

- Display the specified raw video frame into one or more regions within a given window.
- Display a raw video frame in a rectangle within a display window.
- Clear the contents of a rectangle within a display window.

1.3 Linking with the UI SDK API in MS Window Platform

This library uses DirectX functions to draw directly into the video buffer. As result, `ddraw.lib` and `dxguid.lib` must be added to your development environment. (`ddraw.lib` and `ddraw.dll` are shipped as part of MSVC++ SDK.) Additionally, you need to add <*program files*>`/Microsoft DirectX`

`SDK (March 2008)\Lib\x86` to your library path and <*program files*>/`Microsoft DirectX SDK (March 2008)\Include` to your include path. Lastly, make sure WIN32 is added to your C/C++ preprocessor definitions.

You can choose to link with UI SDK API statically or dynamically.

1. Statically Linking - You must add `sdvr_sdk.lib` and `sdvr_ui_sdk.lib` to your library dependencies.
2. Dynamically Linking - To do so, you must add `sdvr_ui_sdk_dll.lib` and `sdvr_sdk_dll.lib` to your library dependencies. Additionally, you must include `sdvr_ui_sdk_dll.dll` and `sdvr_sdk_dll.dll` in the same directory as your DVR Application.

1.4 Linking with UI SDK API in Linux

This library requires an X Server that supports Xv extensions to display live video. As result, in addition to `sdvr_sdk.lib` and `sdvr_ui_sdk.lib`, you must include the following libraries to your development environment:

1. CentOS 5 - libXv-devel
2. Ubuntu 7.10 - libxv-dev

1.5 Using the UI SDK API

This section provides the steps required to display raw video frames on the display screen.

You must provide the SDK the largest video frame size that would ever be displayed. This size is passed as parameters to `sdvr_ui_init`(). This function must be called before any of the display functions can be called and should only be called once within your application.

The color key defines one specific color that should not be drawn to the render surface. After initializing the UI SDK, you can specify the color key by calling `sdvr_ui_set_key_color`(). This is an optional step. The SDK uses white as the default color key.

There are two different methods that you can choose in order to display raw video frames.

Method 1

This method is used if you don't need to access the YUV buffers to perform any video analytics, save into a file, or any other operations on raw video

frames. To start raw video display on a specific region within a window, call `sdvr_ui_start_video_preview`(). To stop raw video display, call `sdvr_ui_stop_video_preview`(). In cases where you are interested in displaying the raw video frames from one channel in different regions within a preview window, you should call `sdvr_ui_start_video_preview`() multiple times for each region. Once `sdvr_ui_start_video_preview`() is called for a specific encoder or decoder channel, the SDK starts displaying any raw video frame received from the DVR board into all the regions within the given window handle. You direct the SDK to stop displaying raw video for each region by calling `sdvr_ui_stop_video_preview`().

Method 2

This method is used if you need to have more control over displaying the video frames. To display YUV frames, get the YUV buffers by calling `sdvr_get_yuv_buffer`(). In general, you can either poll this function or wait to be called back in your video frame callback. After you acquire the YUV buffer, set the buffer by calling `sdvr_ui_set_yuv_buffer`() for displaying. Once a YUV buffer is set to be displayed, you can call `sdvr_ui_draw_yuv`() to draw the buffer into a specific region within a display window. There is no need to call `sdvr_ui_set_yuv_buffer`() multiple times to display the same video frame in different regions of the screen.

1.6 Important Restrictions

Note the following restrictions when using the UI SDK. These restrictions, apart from those explicitly noted, are not permanent and will be removed in future versions of the SDK.

- You must call `sdvr_ui_refresh`() in your application's paint message handler if your application can be covered by any other applications while there are still video frames displaying. This is a permanent restriction.
- The screen must not be locked while video is being displayed. Screen savers must not be activated while video is being displayed. Either of these actions can cause the display to stop refreshing. If this happens, the application must be shut down and restarted.
- There are only four display regions per channel when using `sdvr_ui_start_video_preview`() to display raw video frames.
- Dynamic linking of the display and board SDK libraries is only available in the MS Windows environment.
- DirectDraw acceleration must be enabled on your video card.

1.7 Example of a Callback Function

```
 void av_frame_callback(sdvr_chan_handle_t handle,
                        sdvr_frame_type_e frame_type, sx_bool is_primary)
  {
      sdvr_av_buffer_t  *av_buffer;
      sdvr_yuv_buffer_t *yuv_buf;
      switch (frame_type)
      {
      case SDVR_FRAME_AUDIO_ENCODED:
          if (sdvr_get_av_buffer(handle, SDVR_FRAME_AUDIO_ENCODED,
                                 &av_buffer) == SDVR_ERR_NONE)
          {
                <add the buffer to an audio queue to be processed from
                a different thread>
                <The buffer should be released from that thread when
                you processed the audio frame.>
          }
          break;
      case SDVR_FRAME_RAW_VIDEO:
          if (sdvr_get_yuv_buffer(handle, &yuv_buf) == SDVR_ERR_NONE)
          {
              sdvr_ui_region_t ui_region;

              // set the YUV buffer so that we can display it within
              // any rectangle in any window handle.

              sdvr_ui_set_yuv_buffer(yuv_buf);

              // following code checks to see where this new
              // channel handle needs to be displayed.
              for (int screen = 0; screen < numScreens; ++screen)
              {
                  if (m_settings->displayChannel(screen) == handle)
                  {
sdvr_ui_draw_yuv(m_displayRect[screen].hWnd,&m_displayRect[screen].region);
                  }
              }

              // Once you are done displaying the frame,
              // you must release it.
              sdvr_release_yuv_buffer(yuv_buf);
          }
          break;
      case SDVR_FRAME_H264_IDR:
      case SDVR_FRAME_H264_I:
      case SDVR_FRAME_H264_P:
      case SDVR_FRAME_H264_B:
      case SDVR_FRAME_H264_SPS:
      case SDVR_FRAME_H264_PPS:
      case SDVR_FRAME_JPEG:
      case SDVR_FRAME_MPEG4_I:
      case SDVR_FRAME_MPEG4_P:
      case SDVR_FRAME_MPEG4_VOL:
          if (sdvr_get_av_buffer(handle,
```

```
            is_primary ? SDVR_FRAME_VIDEO_ENCODED_PRIMARY :
            SDVR_FRAME_VIDEO_ENCODED_SECONDARY, &av_buffer) == SDVR_ERR_NONE)
          {
               <add the buffer to a video encoded queue to be processed
               from a different thread.>
               <The buffer should be released from that thread when
               you processed the video frame.>
          }
          break;
      }
      NOTE: Any buffer that is not retrieved(i.e. no call to the
            "sdvr_get_xxx_buffer") will be freed by the SDK once
            the frame queue is full.
  }
```

1.8 Types

_sdvr_err_ui_e

```
typedef enum _sdvr_err_ui_e {
    SDVR_ERR_UI_INIT = 3000,
    SDVR_ERR_UI_INVALID_PARAM,
    SDVR_ERR_UI_SET_BUFFER,
    SDVR_ERR_UI_DRAW_YUV,
    SDVR_ERR_UI_INVALID_YUV_BUF,
    SDVR_ERR_UI_VIDEO_SIZE,
    SDVR_ERR_UI_NO_BUFFER,
    SDVR_ERR_UI_MAX_PREVIEW_REGIONS,
    SDVR_ERR_UI_DRAWINFO_FULL,
    SDVR_ERR_UI_CHANNEL_NOT_START,
    SDVR_ERR_UI_NO_INIT
} sdvr_err_ui_e;
```

Typedef for the errors returned by the UI SDK.

`SDVR_ERR_UI_INIT` - Error code for failure to initialize the UI.

`SDVR_ERR_UI_INVALID_PARAM` - Error code if the any of the parameters passed to any of the SDVR UI function is invalid.

`SDVR_ERR_UI_SET_BUFFER` - Error code for failure to lock the draw surface.

`SDVR_ERR_UI_DRAW_YUV` - Error code for failure to draw the surface.

`SDVR_ERR_UI_INVALID_YUV_BUF` - Error code if the given YUV buffer is not valid.

`SDVR_ERR_UI_VIDEO_SIZE` - Error code if the YUV buffer size is larger than the maximum supported video frame size.

`SDVR_ERR_UI_NO_BUFFER` - Error code if no buffer was set before calling `sdvr_ui_draw_yuv`().

`SDVR_ERR_UI_MAX_PREVIEW_REGIONS` - Error code if no more preview regions can be added.

SDVR_ERR_UI_DRAWINFO_FULL - Error code if the maximum number preview channel is reached.

SDVR_ERR_UI_CHANNEL_NOT_START - Error code if the preview was not started for the specified channel handle.

SDVR_ERR_UI_NO_INIT - Error code if sdvr_ui_init() was not called prior calling this API.

No error

```
#define SDVR_UI_ERR_NONE
```

SDVR_UI_ERR_NONE - Error coder if no error.

_sdvr_ui_region_t

```
typedef struct _sdvr_ui_region_t {
    sx_uint32 top_left_x;
    sx_uint32 top_left_y;
    sx_uint32 width;
    sx_uint32 height;
} sdvr_ui_region_t;
```

This data structure is used to define a display region. A display region is the area within a window in which the video frame is displayed. The top left corner of a window is specified by (0,0). Each window can include many of such regions.

top_left_x - X-coordinate of the upper left corner.

top_left_y - Y-coordinate of the upper left corner.

width - The width of display region within the given window.

height - The height of display region within the given window.

```
typedef HWND sdvr_ui_hwnd_t;
```

The window handle for where to display the raw video frames. In MS Windows environment this is HWND.

```
typedef DWORD sdvr_ui_color_key_t;
```

The color key defines one specific color, for example white, which should not be drawn to the render surface. The color key usually is defined by the RGB macro.

1.9 Defines

Maximum region size

```
#define SDVR_MAX_DRAW_REGIONS 4
```

It is possible to display raw video frames for each encoder or decoder channel in multiple regions within a given window handle.
SDVR_MAX_DRAW_REGIONS defines the maximum number of display regions that can be added to the display preview regions.

Line type

```
#define SDVR_UI_LS_SOLID 0
#define SDVR_UI_LS_DASH 1
#define SDVR_UI_LS_DOT 2
```

The line style to use to draw a frame around a region.

`SDVR_UI_LS_SOLID` - Use solid lines. ____

`SDVR_UI_LS_DASH` - Use dashed lines. -----

`SDVR_UI_LS_DOT` - Use dotted lines. .....

Chapter 2 API Syntax Definitions

2.1 sdvr_rgb

```
#define sdvr_rgb (r,g,b) { ... }
```

A macro to construct an RGB color scheme.

2.2 sdvr_ui_init

```
sdvr_err_ui_e sdvr_ui_init (sx_uint32 max_width, sx_uint32 max_lines);
```

This function initializes video surface and sets up the maximum supported video frame size. (i.e., `sdvr_ui_init(720, 480)` sets the largest video frame size in your system as NTSC D1.)

This function must be called before any of the display functions and should be called only once per session.

Parameters

Name	Description
max_width	The maximum width video frame that will ever be displayed.
max_lines	The maximum number of lines in a video frame that will ever be displayed.

Returns

`SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

2.3 sdvr_ui_close

```
sdvr_err_ui_e sdvr_ui_close ();
```

This function closes the UI SDK and frees up the system resources used by it. You must call this function prior to exiting your DVR Application for a clean shutdown of your system. No other UI API function calls, except `sdvr_ui_init`(), are allowed after this function is called.

Parameters None.

Returns `SDVR_ERR_NONE` - On success.

2.4 sdvr_ui_set_key_color

```
void sdvr_ui_set_key_color (sdvr_ui_color_key_t color_key);
```

The color key defines one specific color that should not be drawn to the render surface. The color key is used to set the background window color when clearing a region. The default color key is white.

NOTE: Color key is not supported by all the video cards.

Parameters

Name	Description
color_key	The new color key. It is usually defined by the `sdvr_rgb` macro.

Returns

None.

2.5 sdvr_ui_set_yuv_buffer

```
sdvr_err_ui_e sdvr_ui_set_yuv_buffer (sdvr_yuv_buffer_t *yuv_buffer);
```

Before a video frame can be displayed, you must specify a raw video frame in the format of 4-2-0. The same buffer is used every time `sdvr_ui_draw_yuv`() is called. This means that the same video frame can be displayed in multiple window regions. To display the next video frame, you must call this function again with a new `yuv_buffer`. In general you get YUV buffers by calling `sdvr_get_yuv_buffer`() in the video frame callback, as you are notified when YUV buffers arrive.

Parameters

Name	Description
yuv_buffer	A pointer to a YUV buffer structure.

Returns

`SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

2.6 sdvr_ui_draw_yuv

```
sdvr_err_ui_e sdvr_ui_draw_yuv (sdvr_ui_hwnd_t hwnd, sdvr_ui_region_t * region);
```

This function displays a YUV video frame that was previously set by `sdvr_ui_set_yuv_buffer`() in the given region within the given window handle.

The specified region must be within the maximum video frame width and lines that were set by the call to `sdvr_ui_init`().

Name	Description
hwnd	The window handle to display the video frame.
region	The rectangle within the given window handle in which to display the video frame.

Returns `SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

2.7 sdvr_ui_clear_yuv

```
sdvr_err_ui_e sdvr_ui_clear_yuv (sdvr_ui_hwnd_t hwnd, sdvr_ui_region_t * region);
```

This function uses the previously defined color key that was set by calling `sdvr_ui_set_key_color`() to clear the contents of the given rectangle within the given window handle.

The specified region must be within the maximum video frame width and lines that were set by the call to `sdvr_ui_init`()

Parameters

Name	Description
hwnd	The window handle to clear the video frame.
region	The rectangle within the given window handle to clear.

Returns

`SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

2.8 sdvr_ui_draw_frame

```
sdvr_err_ui_e sdvr_ui_draw_frame (sdvr_ui_hwnd_t hwnd,
                                  sdvr_ui_region_t region,
                                  sdvr_ui_color_key_t rgb_color,
                                  int line_style);
```

Use this function to draw a frame around the given region within the given window.

You can specify the frame color or the line style.

Parameters

Name	Description
hwnd	The current window handle.
region	The rectangle within the given window handle around which to draw a frame.
rgb_color	The line frame color.
line_style	The line frame style. (i.e., `SDVR_UI_LS_SOLID`)

Returns

`SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

2.9 sdvr_ui_version

```
void sdvr_ui_version (sx_uint8 * major, sx_uint8 * minor,
                      sx_uint8 * revision, sx_uint8 * build);
```

This function returns the display SDK version.

Stretch follows the convention of using four numbers for version control. A change in the major number indicates major changes to functionality, a change in the minor number indicates minor changes to functionality, and a change in the revision number indicates significant bug fixes that were introduced in the minor change functionality. A change to the build number indicates only bug fixes that do not change functionality.

This function can be called before or after `sdvr_ui_init`();

Parameters

Name	Description
major	Pointer to a variable that will hold the major version of the display SDK when this function returns.
minor	Pointer to a variable that will hold the minor version of the display SDK when this function returns.
revision	Pointer to a variable that will hold the revision version of the display SDK when this function returns.
build	Pointer to a variable that will hold the build or bug fix version of the display SDK when this function returns.

Returns

Nothing.

2.10 sdvr_ui_start_video_preview

```
sdvr_err_ui_e sdvr_ui_start_video_preview (sdvr_chan_handle_t handle,
                                           sdvr_ui_hwnd_t wnd_handle,
                                           sdvr_ui_region_t * region,
                                           int * preview_id);
```

You can request the SDK to display video frames directly into one or more regions within a given window handle. Each consecutive call to this function causes the given display region within the given window handle to be added to the preview lists. There is a unique preview ID returned for each region that can be used to stop the preview on it by calling `sdvr_ui_stop_video_preview`().

This function does not enable video streaming. You must call `sdvr_stream_raw_video`() to enable streaming of the raw video frames for the specific encoder or decoder channel. Then, if this function is called, the corresponding video frame will be displayed in the given regions.

Parameters

Name	Description
handle	An encoder or decoder channel to display its raw video frames.
hwnd	The window handle to display the video frames.
region	The rectangle within the given window handle to display the video frames. You can add up to `SDVR_MAX_DRAW_REGIONS` regions by calling this function multiple times. Each region displays the same raw video frame for the given encoder or decoder handle.
preview_id	A pointer to the variable holding the ID for the region that is added successfully.

Returns

`SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

Remarks

You still can call `sdvr_get_yuv_buffer`() to retrieve raw video frames for further processing if it is needed.

NOTE: For decoder channels, you must send each encoded frame to the board to decode them to receive raw video frames.

2.11 sdvr_ui_stop_video_preview

```
sdvr_err_ui_e sdvr_ui_stop_video_preview (sdvr_chan_handle_t h_channel_handle,
                                          int preview_id);
```

This function stops previewing of the raw video frames of the given encoder or decoder channel for the given region ID.

Parameters

Name	Description
handle	An encoder or decoder channel to display its raw video frames.
preview_id	The ID of the region to stop raw video frame previewing. This ID was returned from the call to `sdvr_ui_start_video_preview`().

Returns

`SDVR_ERR_NONE` - On success. Otherwise, see the error code list.

Remarks

Calling this function will not stop the actual raw video streaming. Call `sdvr_stream_raw_video` to stop raw video streaming.

2.12 sdvr_ui_refresh

```
void sdvr_ui_refresh ();
```

Occasionally, the window region displaying the video frames may need to be repainted as the result of another window or application covering it. In this case, you will receive a paint message to repaint the region.

Call this function to refresh the content of all the regions that were being previewed.

Parameters None.

Returns None.

Remarks It is recommended, although not required, to call `sdvr_ui_clear_yuv()` for the desired region before calling this function.

Index

Stretch Inc.
1322 Orleans Drive
Sunnyvale, CA 94089
tel 408.543.2700 • fax 408. 747.5736
www.stretchinc.com