User Guide

SiRFprima Video Codec SDK API

July, 2008
Document Number: CSM-00416
Revision B

Table of Contents

INTRODUCTION

This document describes the Application Programming Interface (API) of the SiRFprima^{TM} Video Codec Software Development Kit (SDK) as well as how to use this API. The API defined in this document will apply to all the video codecs developed for SiRFprima series platforms. The API is cross-platform and can be used on all supported operating systems.

DESCRIPTION

The following is the detailed system diagram of the SiRFprima Video Codec System.

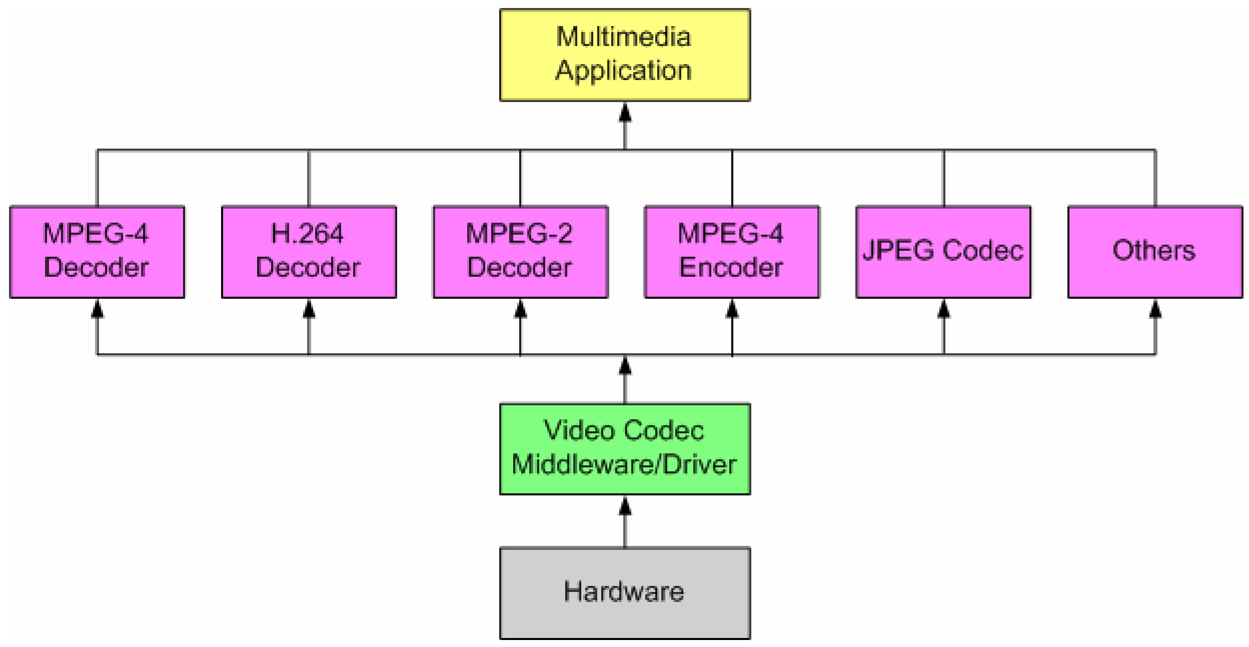

Figure 1: SiRFprima Video Codecs System Diagram

All codecs depend on the video codec middleware/driver, which covers hardware and platform-dependent operations and provides accelerated service to the video codecs. The video codec interfaces are designed to be portable and unified to applications.

The video profiles and levels supported by the SiRFprima Video Codecs are described below.

Video Encoding

- MPEG-4 SP@L3
- H.264 BP@L3

Video Decoding

- H.264 BP@L3, FMO currently not supported.
- MPEG-4 ASP@L5, GMC not supported.

- MPEG-2 MP@ML
- WMV V9 SP@ML

JPEG Codec

JPEG Baseline, 3 Mega Pixel still picture encoding and up to 10 Mega Pixel still picture decoding.

DATA TYPES

<codec>_dec_handle, <codec>_enc_handle

The codec can be MPEG4, H264, MPEG2 and WMV, which indicates the handle for each video codec.

VIDEO_DEC_CONTEXT

The following code shows the decoder interface structure for video decoders:

```
typedef struct
{
    UINT32  size;                   // size of this struct
    UINT32  width;                  // [out], width from decoder
    UINT32  height;                 // [out], height from decoder
    UINT8   *buf_ptr;               // [in], buffer of bitstream
    UINT32  buf_len;                // [in, out], buffer size
    void    *pSurfHdl;              // [out], handler of output surface
                                    // (when shared surface is enabled)
} VIDEO_DEC_CONTEXT;
```

VIDEO_ENC_CONTEXT

The following code shows the encoder interface structure for video encoders.

```
typedef struct
{
    UINT32  size;                 // size of this struct
    UINT8   *buf_ptr;             // [in], output bitstream buffer
    UINT32  buf_len;              // [in/out], output buffer size
    UINT32  type;                 // [in/out], frame type
} VIDEO_ENC_CONTEXT;
```

NOTE – The video resolution should be 4-pixel aligned.

VIDEO DECODER INTERFACE

NOTE — <codec> can be replaced by mpeg2, mpeg4, h264 and wmv.

<codec>_dec_open

This function is called to create the decoder handler.

```
<codec>_dec_handle <codec>_dec_open(UINT32 width, UINT32 height,
                                    UINT32 flag);
```

- Parameters:
  - width

    [in] Width of the source video
  - height

    [in] Height of the source video
  - flag

    [in] Render flag: SURFACE_MGR_DRIVER for video buffer in driver-SURFACE_MGR_RENDER for video buffer in render
- Return Values

  Decoder handle. Returns NULL if failed.
- Remark

  Using flag SURFACE_MGR_RENDER can directly use video renderer to render video to the screen to improve performance greatly.

<codec>_dec_get_caps

This function is called to retrieve capability of the decoder.

```
INT32  <codec>_dec_get_caps(<codec>_dec_handle h,
                            VIDEO_CAPS *caps_ptr);
```

- Parameters:
  - h

    [in] Handle of the decoder
  - caps_ptr

    [out] VIDEO_CAPS structure pointer
- Return Values

  VC_STATE_OK. Returns VC_FATEL_INVALID_PARAMS if failed.

<codec>_dec_header

This function is called to decode video header information from input bit stream.

```
INT32 <codec>_dec_header(<codec>_dec_handle h,
                         VIDEO_DEC_CONTEXT *context_ptr,
                         void *info_ptr);
```

- Parameters:
  - h

    [in] Handle of the decoder
  - context_ptr

    [in/out] Decoder context structure pointer
  - info_ptr

    [in/out] Decoder information (reserved)
- Return Values

  VC_STATE_OK. If failed, the returned codes depend on the syntax.

<codec>_dec_process

This function is called to decode one frame.

```
INT32 <codec>_dec_process (<codec>_dec_handle h,
                           VIDEO_DEC_CONTEXT *context_ptr);
```

- Parameters:
  - h

    [in] Handle of the decoder
  - context_ptr

    [in/out] Decoder context structure pointer
- Return Values

  VC_STATE_OK. If failed, the returned codes depend on the syntax.

NOTE – When <codec>_dec_process() is called, the input bit stream should contain whole frame data.

<codec>_dec_close

This function is called to destroy decoder handler.

```
void <codec>_dec_close (<codec>_dec_handle h);
```

- Parameters:
  - h

    [in] Handle of the decoder

- Return Values

  None

<codec>_dec_surface_lock

This function is called to lock the surface to get the frame buffer.

```
INT32 <codec>_dec_surface_lock (<codec>_dec_handle h,
                                 VIDEO_DEC_CONTEXT *context_ptr,
                                UINT8 *ptr[3], UINT32 stride[3]);
```

- Parameters:
  - h

    [in] Handle of the decoder

  - context_ptr

    [in/out] Context structure.pointer

  - ptr

    [out] YUV pointer of the video frame buffer

  - stride

    [out] YUV pointer and stride of the video frame buffer

- Return Values

  None

- Remark

  When SURFACE_MGR_RENDER is set, it is unnecessary to call lock and unlock to get frame buffer.

<codec>_dec_surface_unlock

This function is called to unlock the surface to release the frame buffer.

```
INT32 <codec>_dec_surface_unlock (<codec>_dec_handle h);
```

- Parameters
  - h

    [in] Handle of the decoder

- Return Values

  None

- Remark

  This function should be called together with lock operation.

VIDEO ENCODER INTERFACE

NOTE — <codec> can be replaced by mpeg4 and h264.

<codec>_enc_open

This function is called to create the encoder handler.

```
<codec>_enc_handle  <codec>_enc_open (UINT32 width, UINT32 height,
                                      UINT32 flag);
```

- Parameters
  - width

    [in] Width of the source video

  - height

    [in] Height of the source video

  - flag

    [in] Reserved, it should be 0

- Return Values

  Handle of the encoder. Returns NULL if failed.

<codec>_enc_get_caps

This function is called to retrieve capability of the encoder.

```
INT32 <codec>_enc_get_caps (VIDEO_CAPS *caps_ptr);
```

- Parameters:
  - h

    [in] Handle of the encoder

  - caps_ptr

    [out] VIDEO_CAPS structure pointer

- Return Values

  VC_STATE_OK. Returns VC_FATEL_INVALID_PARAMS if failed.

<codec>_enc_set_params

This function is called to set parameters of the encoder.

```
INT32 <codec>_enc_set_params (<codec>_enc_handle h,
                              const UINT8 *params, UINT32 size);
```

- Parameters: Parameter string pointer and string size
  - h

    [in] Handle of the encoder
  - params

    [in] Parameter buffer pointer
  - size

    [in] Parameter buffer size (in byte)
- Return Values

  VC_STATE_OK. Returns VC_FATEL_INVALID_PARAMS if failed.

<codec>_enc_process

This function is called to encode one frame.

```
INT32 <codec>_enc_process (<codec>_enc_handle h,
                           VIDEO_ENC_CONTEXT *context_ptr);
```

- Parameters
  - h

    [in] Handle of the encoder
  - context_ptr

    [in/out] Encoder context structure pointer
  - Return Values

    VC_STATE_OK. The returned codes depend on the syntax if failed.

<codec>_enc_close

This function is called to destroy encoder handler.

```
void <codec>_enc_close (<codec>_enc_handle h)
```

- Parameters
  - h

    [in] Handle of the encoder

- Return Values

  None

<codec>_enc_surface_lock

This function is called to lock the surface to retrieve the video frame buffer.

```
INT32 <codec>_enc_surface_lock (<codec>_enc_handle h,
                                 VIDEO_ENC_CONTEXT *context_ptr,
                                UINT8 *ptr[3], UINT32 stride[3])
```

- Parameters
  - h

    [in] Handle of the encoder
  - context_ptr

    [in/out] Encoder context structure pointer
  - ptr

    [out] YUV pointer of the video frame buffer
  - stride

    [out] YUV pointer and stride of the video frame buffer
- Return Values

  None

<codec>_enc_surface_unlock

This function is called to unlock the surface to release the frame buffer.

```
INT32 <codec>_enc_surface_unlock (<codec>_enc_handle h)
```

- Parameters
  - h

    [in] Handle of the encoder
- Return Values

  None
- Remark

  This function should be called together with lock operation.

Sample Codes

These examples describe how to use functions listed above. Please refer to the corresponding interface file for detailed comments and description. You can get the sample code for each codec in the corresponding SDK directory.

Video Decoders

The following pseudo-codes describe how to use video decoders.

```
// create a H.264 decoder
handle = h264_dec_open(width, height, SURFACE_MGR_DRIVER);
…
// decode one frame
state = h264_dec_process(handle, &dec_context);
…
// choice 1: you can get video buffer directly
// lock surface get decoded buffer
state = h264_dec_surface_lock(handle, &dec_context, ptrs, steps);
//…
// get the YUV pointer in ptrs and do other operation
//…
// free surface
state = h264_dec_surface_unlock(handle);
…

// choice 2: render it directly to screen using SiRFprima video render
// SDK
// directly render it to screen
state = BSM_OutputSurf(dec_context.pSurfHdl);
//…

// destroy decoder
H264_dec_close(handle);
//…
```

Video Encoders

The following pseudo-codes describe how to use video encoders.

```
// create a H.264 encoder
handle = h264_enc_open(width, height, 0);
…
// lock out video buffer to fill in YUV data
// lock surface get YUV buffer for encoding
state = h264_enc_surface_lock(handle, &dec_context, ptrs, steps);
```

```
//…
// get the Y,U,V pointer in ptrs and copy YUV data to it
// commit surface
state = h264_enc_surface_unlock(handle);
//…

// encode one frame
state = h264_enc_process_frames(handle, &dec_context);
//…


// destroy encoder
h264_enc_close(handle);
//…
```

REFERENCES

- Industry Standards
  - ISO/IEC 14496-2/10: Information technology – Coding of audio-visual objects
  - VC-1 compressed video bitstream format and Decoding Process
- SiRF Documents
  - *CSM-00399: SiRFprima Video Renderer SDK V1 API User Guide*
  - *CSM-00400: SiRFprima Video Renderer SDK (for WinCE) User Guide*
  - *CSM-00401: SiRFprima Video Renderer SDK V1 (for WinCE) Release Notes*

WORLDWIDE SALES OFFICES

North America	Europe	Asia Pacific	
North America	**Europe**	**Asia Pacific**	
Corporate HQ (1) (408) 467-0410 ✉ Sales@sirf.com	United Kingdom (44) (1344) 668390 ✉ SalesUK@sirf.com	China (86) (21) 5854-7153 ✉ SalesChina@sirf.com	
	Germany (49) (81) 529932-90 ✉ SalesGermany@sirf.com	Taiwan (886) (2) 8174-8966 ✉ SalesTaiwan@sirf.com	India (91) (80) 41966000 ✉ SalesIndia@sirf.com
	Belgium (32) (496) 152969 ✉ SalesBelgium@sirf.com	Japan (81) (44) 829-2186 ✉ SalesJapan@sirf.com	South Korea (82) (2) 3424-3150 ✉ SalesKorea@sirf.com