AVI Library
Reference Guide

Publication Release Date: Jan. 2012

Support Chips:
W55FA series

Support Platforms:
Nuvoton

The information in this document is subject to change without notice.

The Nuvoton Technology Corp. shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

Table of Contents

1. AVI Library Overview

1.1. Video render

FA9x/VA9x can support JPEG decoder to output decoded packet data in DIRECT_RGB555, DIRECT_RGB565, DIRECT_RGB888 or DIRECT_YUV422 format. User application must initialize VPOST as corresponding format specified in AVI function call aviPlayFile(…). AVI player library will configure JPEG output format as specified format and use DMA to copy the decoded data to VPOST frame buffer in Vsync period to avoid the tearing issue.
In this way, three frame buffers are required. One is allocated in VPOST initialized function and two buffers are allocated in AVI library.

1.2. How to use AVI player library

The AVI player library has managed the file access, JPEG decode and audio decode. User only gives the AVI file name and render method to play the movie. The AVI player required user to prepare the following things before playing an AVI movie:

- Initialize system with cache on
- Initialize file system and storage interface (ex. SD card)
- Initialize timer 0
- Initialize VPOST

The VPOST frame buffer format should be consistent with the AVI playback render mode:

- Direct RGB555 – VPOST should select ***DRVVPOST_FRAME_RGB555***
- Direct RGB565 – VPOST should select ***DRVVPOST_FRAME_RGB565***
- Direct RGB888 – VPOST should select ***DRVVPOST_FRAME_RGBx888 or DRVVPOST_FRAME_RGB888x***
- Direct YUV422 – VPOST should select ***DRVVPOST_FRAME_CBYCRY or DRVVPOST_FRAME_YCBYCR or DRVVPOST_FRAME_CRYCBY or DRVVPOST_FRAME_YCRYCB***

Currently, if the decoded Video size is less then the panel size, it will be located at the center of panel. Moreover, decoded image scales by 1/2 in horizontal and vertical direction if the decoded video width is larger than the panel width.
The AVI playback function does not support (x, y) coordinate that are the second and third argument of ***aviPlayFile()*** used to specify the render location on LCD now.

1.3. AVI player user callback

While playing an AVI move, user application may want to draw information on screen or manage user inputs. AVI library provides a callback function to allow user application to grab pieces of CPU time. The callback function pointer was passed to AVI player as the last argument of ***aviPlayFile()***.
Depends on the loading of playing an AVI movie, the user callback will be called several times in each one second. User application should finish the execution of callback function as soon as possible. Otherwise, the AVI playback can be broken because of not enough CPU time.

1.4. AVI playback information

While playing an AVI move, user application can get AVI file information and playback progress information from AVI player. The AVI information will be passed to user application as a parameter of callback function. All information is packed in the AVI_INFO_T structure.

2. AVI Library APIs Specification

2.1. Enumeration

Name	Value	Description
JV_MODE_E		
DIRECT_RGB555	0x0	Direct RGB555 output format
DIRECT_RGB565	0x1	Direct RGB565 output format
DIRECT_RGB888	0x2	Direct RGB888 output format
DIRECT_YUV422	0x3	Direct YUV422 output format
AU_TYPE_E		
AU_CODEC_UNKNOWN	0x0	Unknown audio format
AU_CODEC_PCM	0x1	PCM audio format
AU_CODEC_IMA_ADPCM	0x2	ADPCM audio format
AU_CODEC_MP3	0x3	MP3 audio format

2.2. Structure

Table 1-1 : AVI_INFO_T structure

Field	Type	Description
uMovieLength	UINT32	The total length of input AVI movie (in 0.01 second unit)
uPlayCurTimePos	UINT32	The current playback position. (in 0.01 second unit)
eAuCodec	AU_TYPE_E	Audio format type
nAuPlayChnNum	INT	Audio channel number. (1: mono, 2: stereo, 0: video-only)
nAuPlaySRate	INT	audio sampling rate
uVideoFrameRate	UINT32	Video frame rate.
usImageWidth	UINT16	Video image width
usImageHeight	UINT16	Video image height
uVidTotalFrames	UINT32	total number of video frames

uVidFramesPlayed	UINT32	Indicate how many video frames have been played
uVidFramesSkipped	UINT32	The number of frames was skipped. Video frames may be skipped due to A/V sync

2.3. Functions

aviStopPlayFile

Synopsis

```
int
aviStopPlayFile(void);
```

Description

Stop current AVI file playback.

Parameter

None

Return Value

Successful: Success

ERRCODE: Error

Example

None.

aviPlayFile

Synopsis

```
int
aviPlayFile(
char *suFileName,
int x,
int y,
JV_MODE_E mode,
AVI_CB *cb
) ;
```

Description

Play an AVI file.

Parameter

suFileName [in]

The full path file name of input AVI file.

x [in]

The left-up corner x-coordinate of AVI video render area. Not used now.

y [in]

The left-up corner y-coordinate of AVI video render area. Not used now.

mode [in]

Video render mode.

cb [in]

User application callback function.

Return Value

Successful: Success

ERRCODE: Error

Example

```
/*-------------------------------------------------------------*/
/*    Direct RGB565 AVI playback !!                            */
/*-------------------------------------------------------------*/
lcdformatex.ucVASrcFormat = DRVVPOST_FRAME_RGB565;
vpostLCMInit(&lcdformatex, (UINT32 *)_VpostFrameBuffer);
fsAsciiToUnicode("c:\\Flip-20fps_640x480.avi", suFileName, TRUE);
aviPlayFile(suFileName, 0, 0, DIRECT_RGB565, avi_play_control);
```

aviGetFileInfo

Synopsis

```
int
aviGetFileInfo (
char *suFileName,
AVI_INFO_T *ptAviInfo
);
```

Description

Get the AVI file information.

Parameter

suFileName [in]

The full path file name of input AVI file.

ptAviInfo [in]

Return AVI parsing information.

Return Value

Successful: Success

ERRCODE: Error

Example

```
fsAsciiToUnicode("c:\\Flip-20fps.avi", suFileName, TRUE);
aviPlayFile(suFileName, &sAVIInfo);
```

aviSetPlayVolume

Synopsis

```
int
aviSetPlayVolume (
int vol
);
```

Description

Set the Left channel and Right channel playback audio volume.

Parameter

vol [in]

The audio volume

Return Value

Successful: Success

ERRCODE: Error

Example

```
aviSetPlayVolume(suFileName, 0x1F);
```

aviSetRightChannelVolume

Synopsis

```
int
aviSetRightChannelVolume (
int vol
);
```

Description

Set the Right channel audio playback volume only.

Parameter

vol [in]

The audio volume

Return Value

Successful: Success

ERRCODE: Error

Example

```
// Set Right Channel as Mute
aviSetPlayRightChannelVolume(suFileName, 0x0);
```

2.4. Error Code Table

Code Name	Value	Description
MFL_ERR_NO_MEMORY	0xFFFF8000	no memory
MFL_ERR_HARDWARE	0xFFFF8002	hardware general error
MFL_ERR_NO_CALLBACK	0xFFFF8004	must provide callback function
MFL_ERR_AU_UNSUPPORT	0xFFFF8006	not supported audio type
MFL_ERR_VID_UNSUPPORT	0xFFFF8008	not supported video type
MFL_ERR_OP_UNSUPPORT	0xFFFF800C	unsupported operation
MFL_ERR_PREV_UNSUPPORT	0xFFFF800E	preview of this media type was not supported or not enabled
MFL_ERR_FUN_USAGE	0xFFFF8010	incorrect function call parameter
MFL_ERR_RESOURCE_MEM	0xFFFF8012	memory is not enough to play/record a media file
MFL_ERR_FILE_OPEN	0xFFFF8020	cannot open file
MFL_ERR_FILE_TEMP	0xFFFF8022	temporary file access failure

MFL_ERR_STREAM_IO	0xFFFF8024	stream access error
MFL_ERR_STREAM_INIT	0xFFFF8026	stream was not opened
MFL_ERR_STREAM_EOF	0xFFFF8028	encounter EOF of file
MFL_ERR_STREAM_SEEK	0xFFFF802A	stream seek error
MFL_ERR_STREAM_TYPE	0xFFFF802C	incorrect stream type
MFL_ERR_STREAM_METHOD	0xFFFF8030	missing stream method
MFL_ERR_STREAM_MEMOUT	0xFFFF8032	recorded data has been over the application provided memory buffer
MFL_INVALID_BITSTREAM	0xFFFF8034	invalid audio/video bitstream forma
MFL_ERR_AVI_FILE	0xFFFF8080	Invalid AVI file format
MFL_ERR_AVI_VID_CODEC	0xFFFF8081	AVI unsupported video codec type
MFL_ERR_AVI_AU_CODEC	0xFFFF8082	AVI unsupported audio codec type
MFL_ERR_AVI_CANNOT_SEEK	0xFFFF8083	The AVI file is not fast-seekable
MFL_ERR_AVI_SIZE	0xFFFF8080	Exceed estimated size
MFL_ERR_MP3_FORMAT	0xFFFF80D0	incorrect MP3 frame format
MFL_ERR_MP3_DECODE	0xFFFF80D2	MP3 decode error
MFL_ERR_HW_NOT_READY	0xFFFF8100	the picture is the same as the last one
MFL_ERR_SHORT_BUFF	0xFFFF8104	buffer size is not enough
MFL_ERR_VID_DEC_ERR	0xFFFF8106	video decode error
MFL_ERR_VID_DEC_BUSY	0xFFFF8108	video decoder is busy
MFL_ERR_VID_ENC_ERR	0xFFFF810A	video encode error
MFL_ERR_UNKNOWN_MEDIA	0xFFFF81E2	unknow media type

3. Revision History

Version	Date	Description
V1.00	Jan. 30, 2012	▪ Created

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in equipment or systems intended for surgical implantation, atomic energy control instruments, aircraft or spacecraft instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for any other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications whereby failure could result or lead to personal injury, death or severe property or environmental damage.

Nuvoton customers using or selling these products for such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from their improper use or sales.