ANALOG DEVICES

ITU656 VIDEO UTILITIES

DATE: FEBRUARY 13, 2006

Table of Contents

Document Revision History

Date	Description of Changes
07/02/2006	Initial Release

1. ITU656 video utilities Overview

The ITU656 video utilities can be used to format an area of memory into a selected (ITU656) video frame and fills the whole frame or a particular column/row of the frame with the specified colour value. The utilities has of four different functions to perform the above operations and accepts YCbCr (4:2:2) colour values.

2. ITU656 video utilities Files

The files listed below comprise the Video utilities API and source files.

2.1. Include Files

adi_itu656_utilities.h

2.2. Source Files

adi_itu656_utilities.c

3. Theory of Operation

The ITU656 video utilities have four different functions.

- adi_video_FrameFormat( ) – Formats an area of memory into a selected ITU656 video frame
- adi_video_FrameFill ( ) – Fills active video portions of formatted frame with specified colour
- adi_video_RowFill ( ) – Fills a row of pixels in active video portion of formatted frame with specified colour
- adi_video_ColumnFill ( ) – Fills a column of pixels in active video portion of formatted frame with specified colour

3.1. Function – adi_video_FrameFormat ( )

This function formats an area of memory into a selected ITU656 video frame

Prototype

void adi_video_FrameFormat (u8 *frame_ptr, FRAME_TYPE frametype);

Arguments

*frame_ptr – Pointer to the area of memory to be formatted
frametype – the selected memory area will be formatted to this frame type

FRAME_TYPE is an enumerated data type which can take one of the following values

Data type	Description
NTSC_IL	Format the memory area to NTSC Interlaced frame
PAL_IL	Format the memory area to PAL Interlaced frame
NTSC_PR	Format the memory area to NTSC progressive frame
PAL_PR	Format the memory area to PAL progressive frame

3.2. Function – adi_video_FrameFill ( )

This function fills the active video portions of formatted frame with specified colour

Prototype

void adi_video_FrameFill (u8 *frame_ptr, FRAME_TYPE frametype, u8 *ycbcr_data);

Arguments

*frame_ptr – Pointer to the formatted video frame in memory area
frametype – formatted video frame type
ycbcr_data – Pointer to 4 byte array of 32 bit colour value of YCbCr data

3.3. Function – adi_video_RowFill ( )

This function fills a row of pixels in the active video portion of formatted frame with specified colour. The remaining area of the active video portion will be left blank.

Prototype

void adi_video_RowFill (u8 *frame_ptr, FRAME_TYPE frametype, u32 row_value, u8 *ycbcr_data);

Arguments

*frame_ptr – Pointer to the formatted video frame in memory area
frametype – formatted video frame type
row_value – Row number of the active field which is to be filled with the specified colour
ycbcr_data – Pointer to 4 byte array of 32 bit colour value of YCbCr data

3.4. Function – adi_video_ColumnFill ( )

This function fills a column of pixels in the active video portion of formatted frame with specified colour. The remaining area of the active video portion will be left blank.

Prototype

void adi_video_ColumnFill (u8 *frame_ptr, FRAME_TYPE frametype, u32 column_value, u8 *ycbcr_data);

Arguments

*frame_ptr – Pointer to the formatted video frame in memory area
frametype – formatted video frame type
column_value – Column number of the active field which is to be filled with the specified colour
ycbcr_data – Pointer to 4 byte array of 32 bit colour value of YCbCr data

3.5. ITU656 video utilities specific Return Codes

There are no specific return codes for ITU656 video utilities.

4. Annexure

Few colour values in YCbCr (4:2:2) format

```
char black[ ]   = { 0x80, 0x10, 0x80, 0x10 }   // Black pixel ycbcr format
char blue[ ]    = { 0xF0, 0x29, 0x6E, 0x29 }   // Blue pixel ycbcr format
char red[ ]     = { 0x5A, 0x51, 0xF0, 0x51 }   // Red pixel ycbcr format
char magenta[ ] = { 0xCA, 0x6A, 0xDE, 0x6A }   // Magenta pixel ycbcr format
char green[ ]   = { 0x36, 0x91, 0x22, 0x91 }   // Green pixel ycbcr format
char cyan[ ]    = { 0xA6, 0xAA, 0x10, 0xAA }   // Cyan pixel ycbcr format
char yellow[ ]  = { 0x10, 0xD2, 0x92, 0xD2 }   // Yellow pixel ycbcr format
char white[ ]   = { 0x80, 0xEB, 0x80, 0xEB }   // White pixel ycbcr format
```