LUMITRON
Thermal Imaging Products

Video and Overlay Palette File Formats for the ECS-320A Embeddable Camera Electronics System

(Document Number 700-00000042-R2)

10503 Timberwood Circle
Suite 120
Louisville, KY 40223
(502) 423-7225

Video and Overlay Palette File Formats

Document Number 700-00000042
April 2003

Lumitron
10503 Timberwood Circle
Suite 120
Louisville, KY 40223
(502) 423-7225
FAX (502) 423-7064

Table of Contents

1 Introduction

This guide has been written to provide an end user of the ECS-320A electronics based cameras to create custom palettes.

With this guide users can convert existing RGB based palettes to a format compatible with the embedded application.

2 Video Palette Components

The video palette that ultimately is loaded into the video encoder is of the YCrCb or Y'CrCb format. The Y' component is the user specified gamma corrected value of the standard Y component.

The palette is made up of 256 colors and an 8 character mnemonic that can be retrieved from a host connected to the camera. Each of the components (Y, Y', Cr, Cb) that make up a single color is 8-bits wide.

The data for the palette is spread across four serial data flash pages on the camera. Each colors Y/Y' values and Cr/Cb values are stored in the same 16-bit word. Also the complete table of Y/Y' values are stored separately from the table of Cr/Cb values. This is due to the way the data eventually is loaded into FPGA memory space. The following data structures are used to define the separate components and allow for ease of transferring the data to the camera.

```
struct _VID_Y_COLORS {

      BYTE  yStd;       // Standard luminance component
      BYTE  yGamma;     // Gamma corrected luminance component

};
typedef struct _VID_Y_COLORS VID_Y_COLORS;


struct _VID_CX_COLORS {

      BYTE  cB;         // Chrominance component (blue)
      BYTE  cR;         // Chrominance component (red)

};
typedef struct _VID_CX_COLORS VID_CX_COLORS;
```

The same format is used by the embedded application to extract the color components from flash and load them to FPGA memory space without rearranging or modification.

3 Computing Video Color Components

The color components used can be computed from a standard RGB palette color with the following equations:

$$Y = (BYTE)(0.297 \times R + 0.582 \times G + 0.113 \times B + 1)$$

$$Y' = (BYTE)(254 \times (\frac{Y}{254})^{\frac{1}{\gamma}})$$

$$Cr = (BYTE)(0.496 \times R - 0.416 \times G - 0.080 \times B + 128)$$

$$Cb = (BYTE)(-0.167 \times R - 0.329 \times G + 0.496 \times B + 128)$$

Where:

R = 8-bit Red Value (0 – 255);

G = 8-bit Green Value (0 – 255);

B = 8-bit Blue Value (0 – 255);

γ Is the desired gamma correction value from 1.0 – 2.5;

Y, Y’, Cr, Cb: must be between 1 and 254 (inclusive);

Set Cr & Cb to 128 for gray scale palettes.

4 Building Video Palette File

Since the palette will be stored in serial data flash and each page has 264 bytes, padding will be used to create a file that is exactly four serial page sizes. To accomplish this it is necessary to pad the array of colors in the following manner.

```
VID_Y_COLORS        yColors[264];
VID_CX_COLORS       cXColors[264];
```

Where:

- yColors[0 – 255] are valid color components
- yColors[256 – 259] is used to store the palette mnemonic (8 bytes mapped into this location)
- yColors[260 – 263] is padding
- cXColors [0 – 255] are valid color components
- cXColors [256 – 263] is padding

In this format the file is 264 * 4 bytes in size (4 serial flash pages). This format makes for an efficient transfer of data to the camera using the PC Master serial interface. No other header information is needed.

5 Overlay Palette Components

The overlay palette that ultimately is loaded into the video encoder is of the YCrCb format. This format is similar to the video palettes with the exception that the Cr and Cb values are only 4-bits each instead of 8. Therefore computed Cx values are mapped using a lookup table.

The palette is made up of 16 colors (0 – 15) with the first being reserved for transparent (all components being 0). Colors at indexes 14 and 15 are also specific in nature. The camera reticles use colors at these indexes when enabled. To create a properly sized reticle that is interpolated correctly by the video encode 2 colors are used to paint the vertical lines. A full intensity pixel is sandwiched between two 50% intensity pixels. Therefore the color at index 15 can be user defined, but the color at index 14 must be at 50% of that color’s intensity.

The data for each the palettes is contained in a single serial data flash page on the camera. There is room for up to 8 overlay palettes on the camera with each palette being 32 bytes in size. The following data structure is used to define the separate components and allow for ease of transferring the data to the camera.

```
struct _OVL_COLORS {

      BYTE  yVal;       // Y Component
      BYTE  cxVal;      // MSB: Cb Component, LSB: Cr Component

};
typedef struct _OVL_COLORS OVL_COLORS;
```

The same format is used by the embedded application to extract the color components from flash and load them to FPGA memory space without rearranging or modification.

When creating an overlay palette for cameras that will be using the range reticle, further restrictions should be noted. This palette reserves not only color index pair 14/15 but also pair 12/13. As described above, two colors are needed to create the vertical lines for reticles. The second pair of reserved colors allows for the normal/inverted video polarity modes. This will allow the user to create a palette color that will remain visible if the white hot/black hot setting is changed. Colors 12/13 are used to draw the reticle in standard mode and colors 14/15 are used when the video polarity is toggled.

6 Computing Overlay Color Components

The color components used can be computed from a standard RGB palette color with the following equations:

$$Y = (BYTE)(0.297 \times R + 0.582 \times G + 0.113 \times B + 1)$$

$$Cr = (BYTE)(0.496 \times R - 0.416 \times G - 0.080 \times B + 128)$$

$$Cb = (BYTE)(-0.167 \times R - 0.329 \times G + 0.496 \times B + 128)$$

Where:

R = 8-bit Red Value (0 – 255);

G = 8-bit Green Value (0 – 255);

B = 8-bit Blue Value (0 – 255);

Y: must be between 1 and 254 (inclusive);

The Cr/Cb components must be indexed to a 4-bit value using the following transformation.

Computed Cr/Cb Range	Cr/Cb Index	Actual Cr/Cb
0 <= Cx < 24	1	16
24 <= Cx < 40	2	32
40 <= Cx < 56	3	48
56 <= Cx < 72	4	64
72 <= Cx < 88	5	80
88 <= Cx < 104	6	96
104 <= Cx < 120	7	112
120 <= Cx <= 136	8	128
136 < Cx <= 152	9	144
152 < Cx <= 168	10	160
168 < Cx <= 184	11	176
184 < Cx <= 200	12	192
200 < Cx <= 216	13	208
216 < Cx <= 232	14	224
232 < Cx <= 255	15	240

7 Building Overlay Palette File

All of the overlay palettes will be stored in a single page of serial data flash. A page has 264 bytes so there is room for 8 overlay palette files sized at 32 bytes each. The file will be the binary output of the following array.

OVL_COLORS ovlyColors[16];

Where:

- ovlyColors [0] set to zero

- ovlyColors [1 - 13] user defined colors
- ovlyColors [14].yVal = ovlyColors [15].yVal >> 1 (Cx components the same)
- ovlyColors [15] user defined color

In this format the file is 32 bytes in size.