TW8832S, 33S
SPI OSD

Application Note

Techwell, Inc.
408 E. Plumier Drive
San Jose, CA 95134
http://www.techwellinc.com

Disclaimer

Document Revision History

Version	Date	Notes	By
0.1	5/20/2010	Initial	William Yu

Overview

This note describes the concept of SPI OSD feature supported by TW8832S, 33S and how to use it.

The integrated SPIOSD provides a flexible mapping between its display on the LCD and its bit mapped image stored in the SPI memory. In general a buffer in the SPI memory is specified for the image to be displayed. The buffer size is larger than the display size of the window. The bit mapped image stored is 8 bits per pixel. During display, the 8-bit pixel is fetched from the SPI memory and mapped to a 32-bit quantity by the window's LUT (Look Up Table). The 32-bit quantity consists of 24-bit RGB, 7-bit alpha blending attribute, and one bit blinking attribute. The pixel is then mixed with video before displaying on the LCD panel.

Two SPIOSD windows are provided. Each window has its own set of register but shares one LUT.

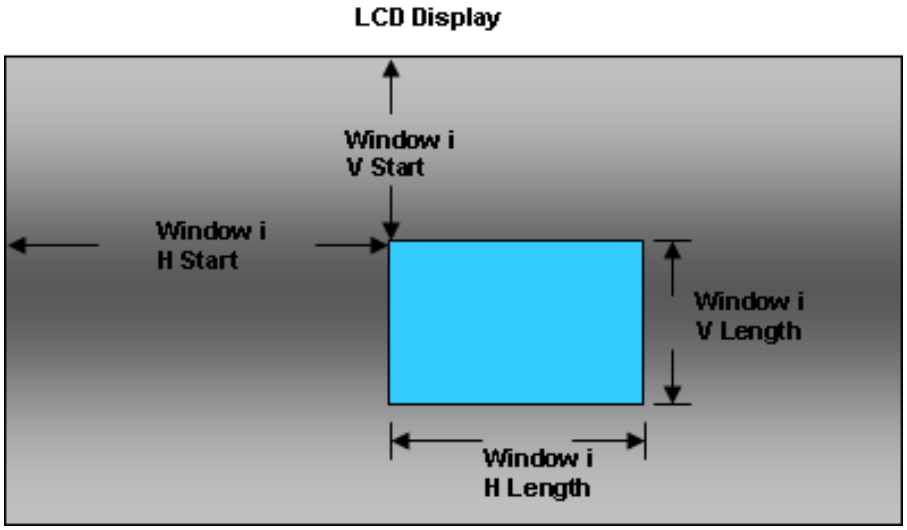
Note: both windows can be active at the same time but cannot be overlapped and have a minimum (16 pixels) horizontal spacing requirement.

Looping control for adjacent buffers is provided for each window. Animation can be achieved by properly allocating multiple buffers in the adjacent area and the looping control.

1. SPIOSD Window Display Starting Location and Sizes

There are four registers used to specify the staring location and size on the LCD:

- Window i Horizontal Start (Win0: Reg421[2:0],422[7:0], Win1: Reg441[2:0],442[7:0])
- Window i Vertical Start (Win0: Reg423[2:0],424[7:0], Win1: Reg443[2:0],444[7:0])
- Window i Horizontal Length (Win0: Reg425[2:0],426[7:0], Win1: Reg445[2:0],44[7:0])
- Window i Vertical Length (Win0: Reg427[2:0],428[7:0], Win1: Reg447[2:0],448[7:0])



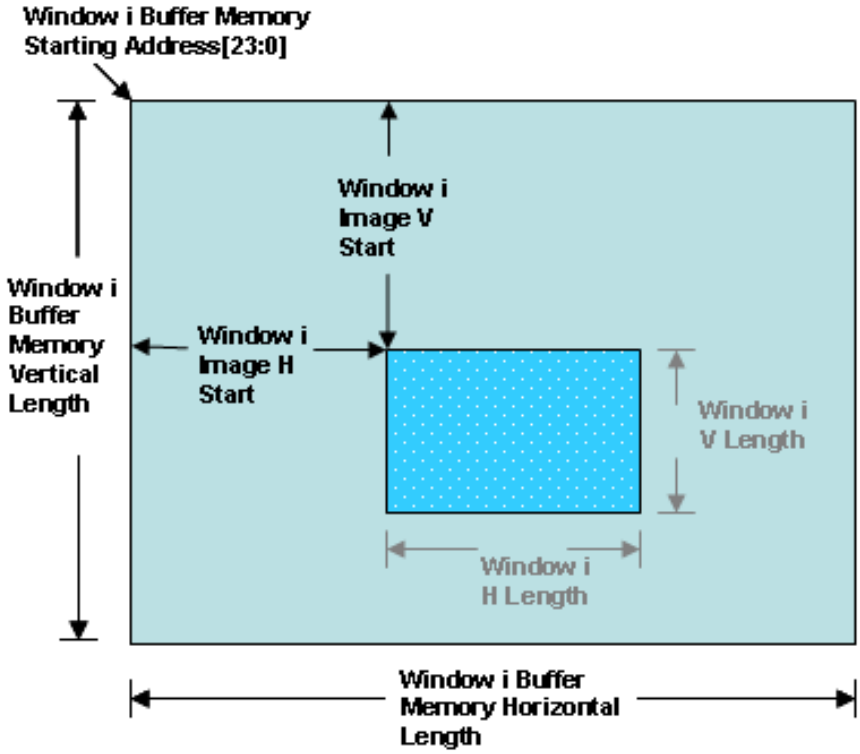
2. SPIOSD Window Buffer Memory

Three registers together define the buffer starting location and boundaries:

- Window i Buffer Memory Starting Address (Win0: Reg429,42A, 42B[7:0], Win1: Reg449,44A, 44B[7:0])
- Window i Buffer Memory Horizontal Length (Win0: Reg42C[3:0],42D[7:0], Win1: Reg44C[3:0],44D[7:0])
- Window i Buffer Memory Vertical Length (Win0: Reg42E[3:0],42F[7:0], Win1: Reg44E[2:0],44F[7:0])

Two registers point to the starting location of the image stored:

- Window i Image Horizontal Start (Win0: Reg430[2:0],421[7:0], Win1: Reg450[2:0],451[7:0])
- Window i Image Vertical Start (Win0: Reg432[2:0],423[7:0], Win1: Reg452[2:0],453[7:0])

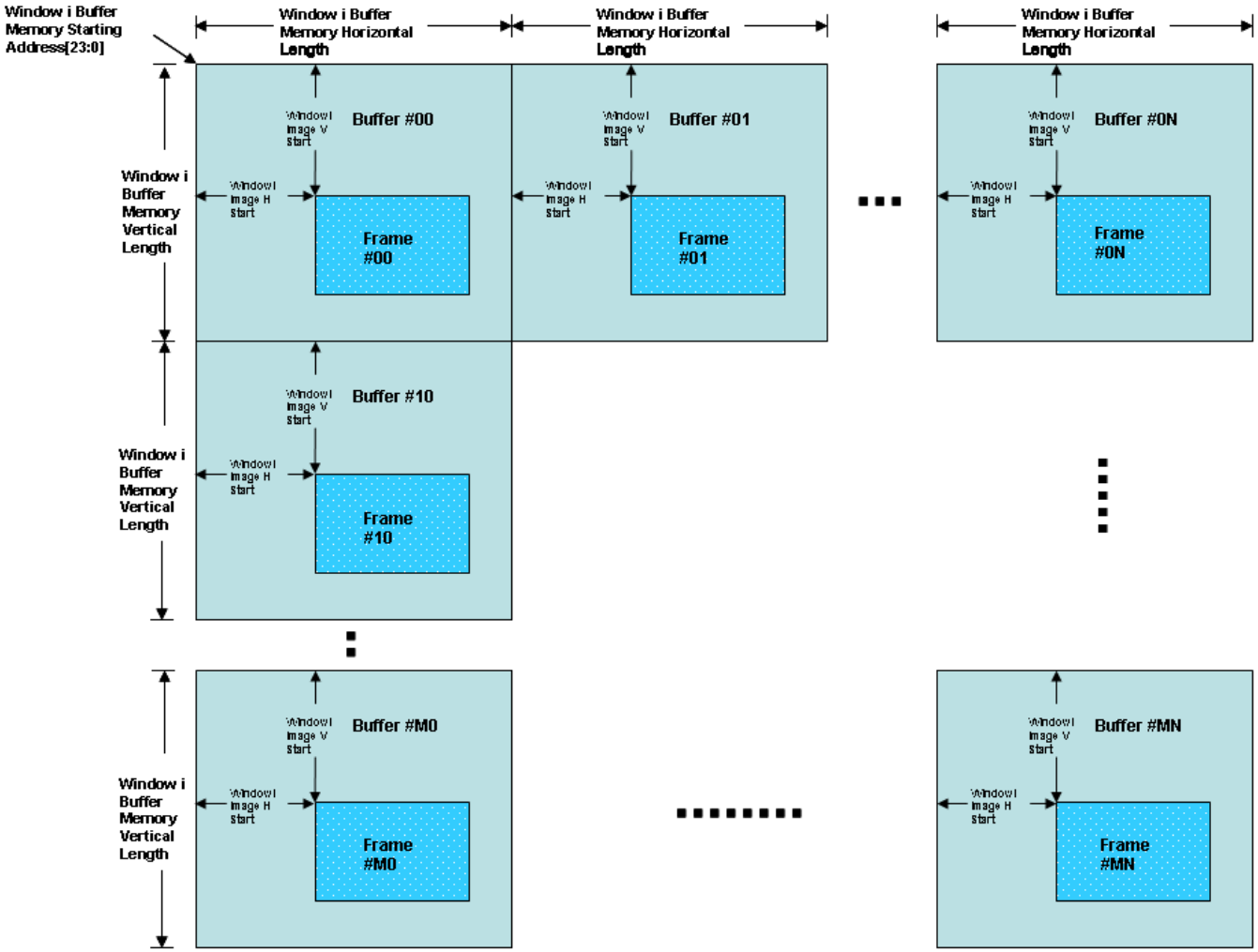


3. SPIOSD Window Loop Control

Three registers are used for loop control:

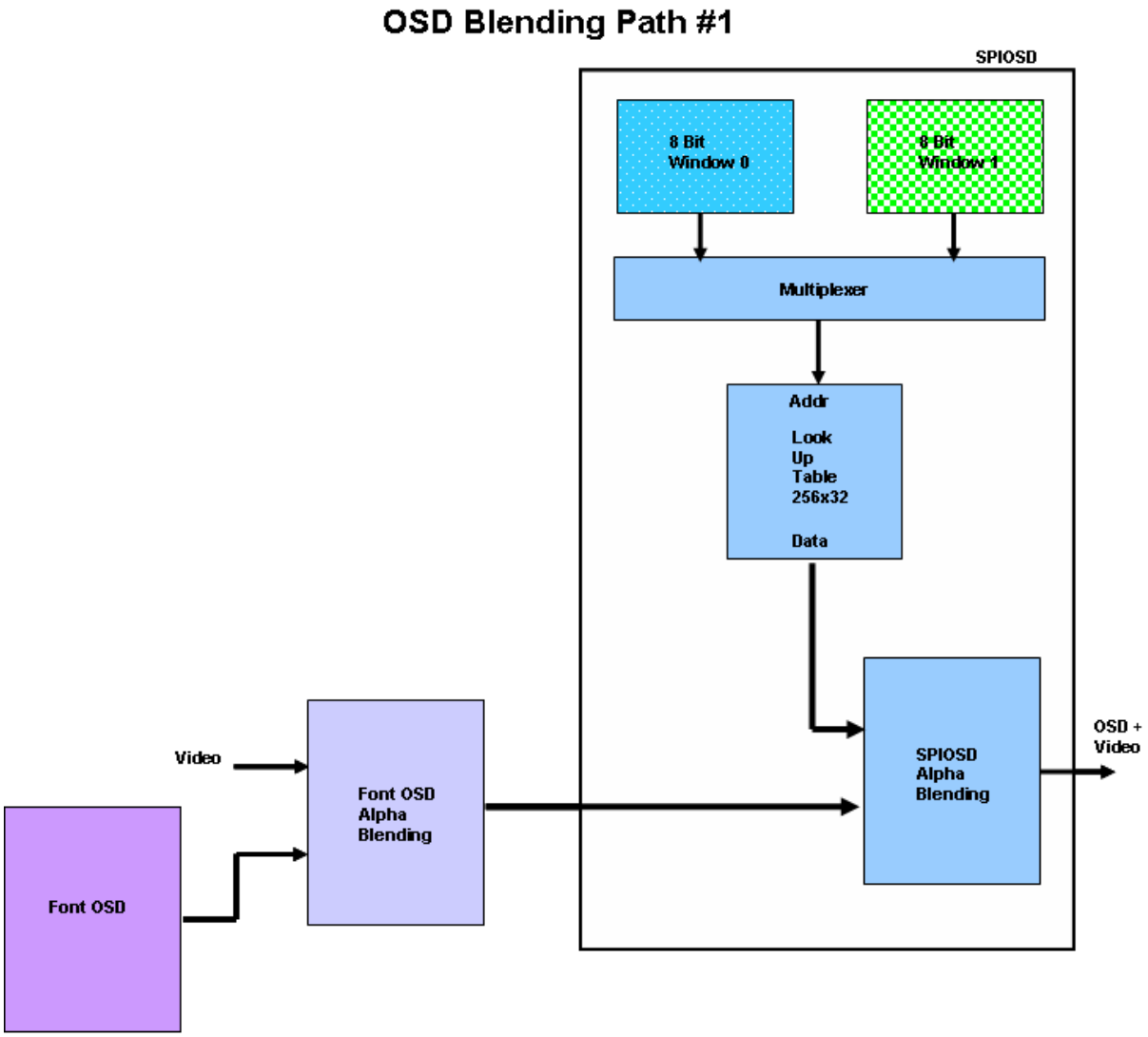
- Window i Looping Horizontal Frame Number (Win0: Reg435[7:0], Win1: Reg455[7:0])
- Window i Looping Vertical Frame Number (Win0: Reg436[7:0], Win1: Reg456[7:0])
- Window i Frame Duration (Win0: Reg437[7:0], Win1: Reg457[7:0])

In the diagram below, the **Looping Horizontal Frame Number** register contains a value N, and the **Looping Vertical Frame Number** register contains a value M. The display starts from Frame #00 and then moves horizontally to the right and then vertically down. The display order is #00, #01, #02, … 0N, #10, #11, #12, … #1N, …. #M0, #M1, …#MN. Each frame stays on for the time specified by **Frame Duration** register.

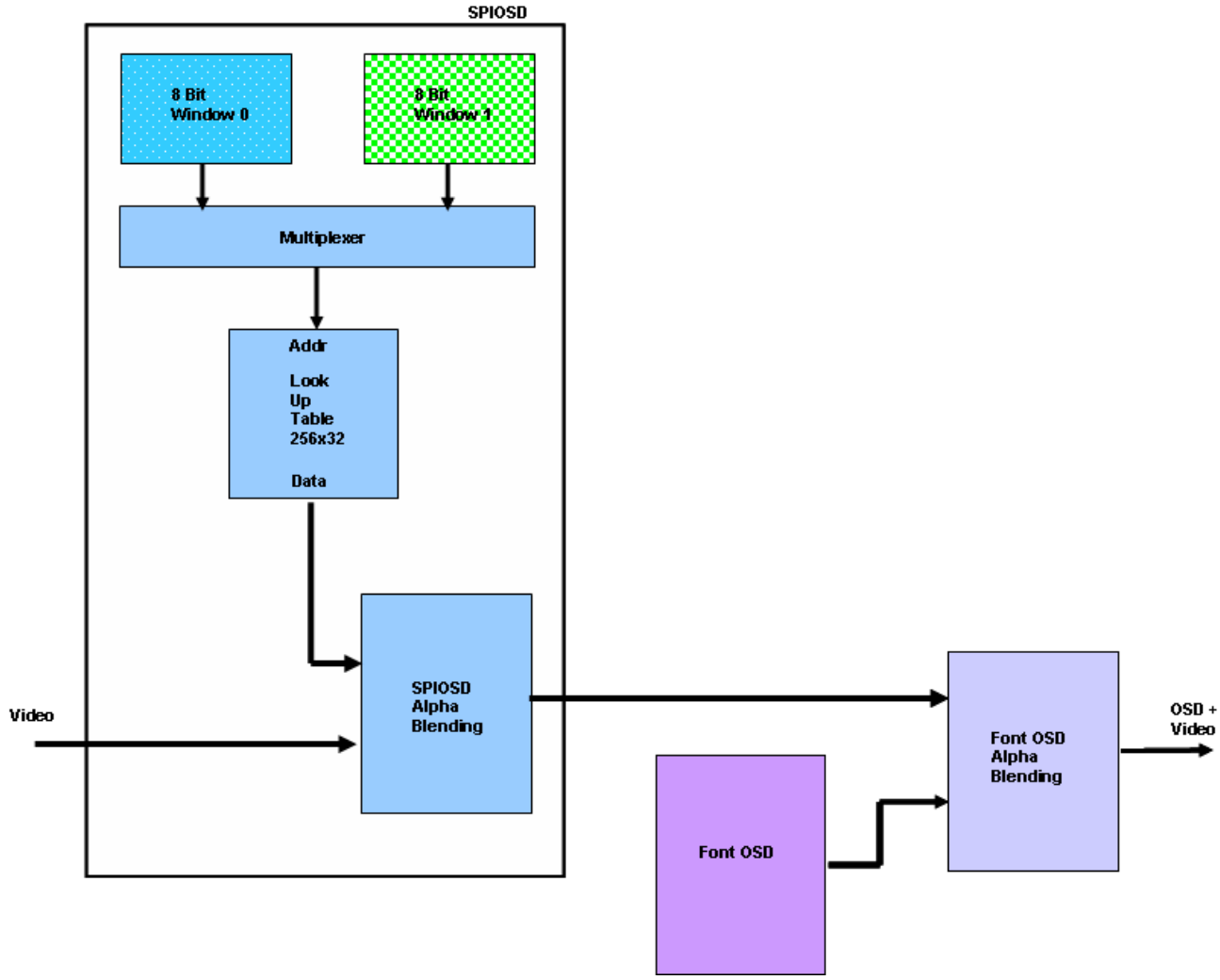


4. OSD Display Path

In normal mixing order, Video input is mixed with Font OSD first. The resultant output is then mixed with SPIOSD. Alternatively, Video input can be mixed with SPIOSD first and then Font OSD.



OSD Blending Path #2

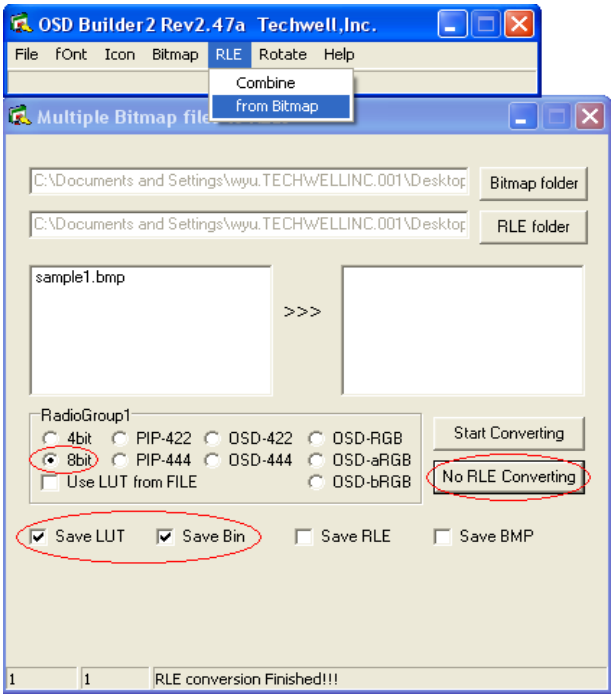


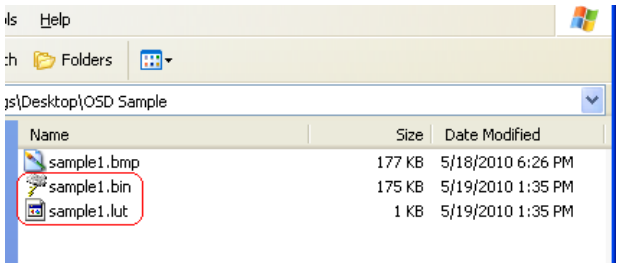
5. How to create SPI OSD/Animation

a. Make a **same or larger** resolution bitmap format picture which will be displayed on the screen.

b. Convert the Bitmap file to Binary format by using **OSD Builder** tool provided by TW:

Select “8bit” and “Save LUT, Save Bin”, click “No RLE Converting” button, 2 binary files **sample1.bin** (data) and **sample1.lut** (LUT, system default) will be generated in the selected folder (see pictures below).
(The OSD builder tool will be improved with more user friendly interface later)





c. The binary files can be loaded to SPI by using USB dongle with TW_Ternimal tool. When displaying, simply write the LUT then load the picture data to display window.

d. To display an animation, combine all the binary file of continuous pictures to one binary file in right order, using **vertical looping** and **frame duration** registers to control the frame number to be played and display speed.

Note:

1) SPIOSD doesn’t have zoom function, so the OSD picture stored in SPI needs to be the same as or larger size than what you want to display on the screen, you can display part of the picture by defining display window size and H/V offset.
2) The SPIOSD memory size (Horizontal/Vertical lengths) needs to be defined the exact the same as the original picture size (resolution), otherwise you will get broken image data.
3) The SPIOSD supports up to 8bit color, so all higher color depth will be converted to max 8bit format.
4) When two SPIOSD windows are opened at the same height, at least 16 pixels of space has to been kept horizontally between two windows otherwise the Window1 image will be broken.

6. Clock divider setting

When SPI-OSD is used, to synchronize SPI data and PCLKO, PCLKO has to be divided from PCLK, SPI CLK, PCLK and PCLKO dividers have to be configured very carefully.

For example, in case the panel is WVGA (40MHz) and SPI-OSD is used, set PLL as 80MHz and set SPI-CLK divider = 1, PCLK divider = 1, PCLKO divider = 2 and SPI read mode needs to be quad.

If the panel is WQVGA (10MHz) with TW8833, set PLL as 40MHz and set SPI-CLK divider = 1, PCLK divider = 1, PCLKO divider = 4 and SPI read mode as dual.

Please refer to "**AN-TW8832,33 Scaler & TCON.doc**" for details.

7. LUT program method

The LUT registers can be programmed by writing register externally or using SPI-DMA mode.

For both modes, the write order(pointer increment) has 2 options, which is controlled by Reg0x410[6:5],

"01h" means writing by color order, e.g.
B00→G00→R00→Attr00→B01→G01→R01→Attr01 … BFF→GFF→RFF→AttrFF

"10h" means writing by address order, e.g.
B00→B01→…BFF→G00→…GFF→R00→…RFF→Att00→…AttFF

B	G	R	Attribute
00	00	00	00
01	01	01	01
02	02	02	02
: : :	: : :	: : :	: : :
FE	FE	FE	FE
FF	FF	FF	FF

1) Example for Register method writing by address order:

```
w ff 04 ; page4
w 10 c0 ; Write En, write by address order, start from color B  (then c1, c2, c3)
w 11 00 ; LUT addr = 0

w 12 00 ; LUT data = 0
   :
   :
w 12 FF ; go to "w10 cx" for next color or attri
```

2) SPI-DMA method doesn't request address for each data, there is an option for SPIOSD LUT program which is set by Reg0x483[7:6] = 10h, just assign the SPI start address and LUT buffer start address, the LUT data will be "mapped" to LUT table. Below is an example:

```
w ff 04 ; page4
w 10 c0 ; LUT Write En, write by address order*
w 84 00 ; DMA stop

w 80 15 ; Host SPI mode Fast, OSD SPI Quad-io
w 83 85 ; SPIOSD LUT mode, 5 command bytes(refer to SPI chip spec)
w 8a 0b ; SPI Command = R (refer to SPI chip spec)

w 8b 2b ; SPI start Addr high byte
w 8c bb ; SPI start Addr mid  byte
w 8d 90 ; SPI start Addr low  byte

w 86 00 ; DMA buffer start address high byte
w 87 00 ; DMA buffer start address low byte

w 9a 00 ; DMA size high byte
w 88 04 ; DMA size mid  byte
w 89 00 ; DMA size low byte

w 84 01 ; DMA (write) Start
```

* "**OSD solution tool**" will include a mode to save LUT as color order later so user can use "color order" write mode. In this case, 0x410 will be "a0h".

8. Display position control

Due to the internal clock delay caused in each processing block, the final OSD display start position may not match the video start position perfectly, and current design can't adjust it automatically either, it needs to be re-calculated and fine tuned manually. Here is the formula for SPIOSD data read timing (0x40F) setting, all the involved registers are (the value in { } are mapped to the register value):

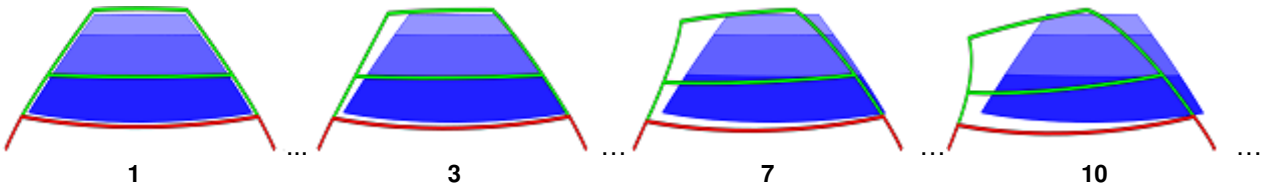
```
HDE    = REG(0x210[7:0])
Mixing = REG(0x400[1:1]) {0,3}
PCLKO = REG(0x20D[1:0])
SPI_MODE  = REG(0x480[2:0]) {30,21,26,38,22,28,37}
CLK_DELAY = REG(0x481[4:4])
```

The final result for 0x40F = HDE + Mixing + PCLKO - (SPI_MODE + CLK_DELAY)

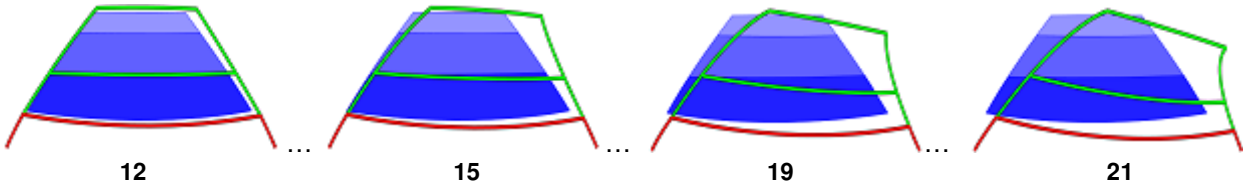
9. Example of Park Guider with the Demo board

A demo of park guider for overlaying on the image from backup camera is implemented in the EVB, customer can create their own based on the sample. The following procedure how it is created.

1) Total 21 continuous pictures are created for an animation effect of moving grid from center to left(1~11) then center to right(12~21), the actual size of each picture is 730 x 245, below are 8 of them.

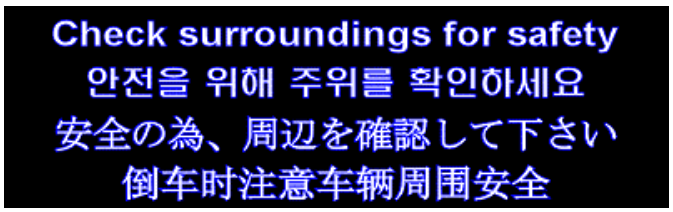


Grid moving from center to left



Grid moving from center to right

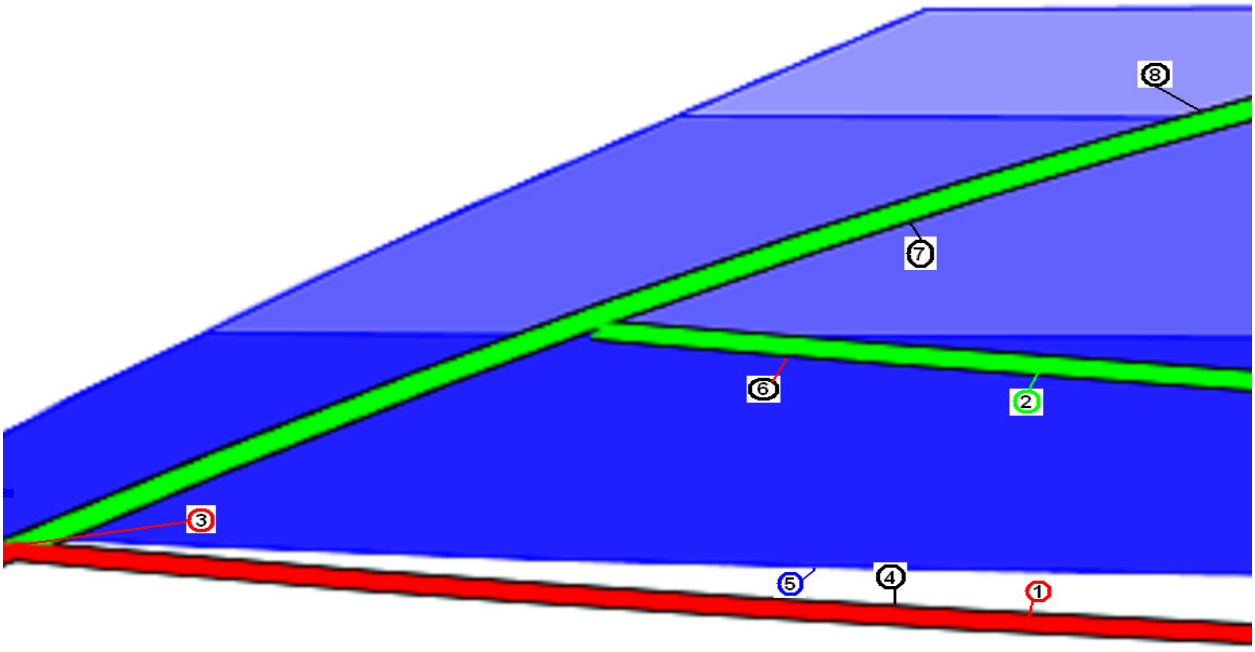
2) At the bottom of the grid, there is a warning message displayed in 4 different languages alternately, the whole image is showed right side below.



3) To make the grid moving look nature and smooth, a black border is put surrounding the lines and characters, this makes procedure much more complicated because all the colors used in the picture have been considered how alpha blending will affect each other when they cross or overlay.

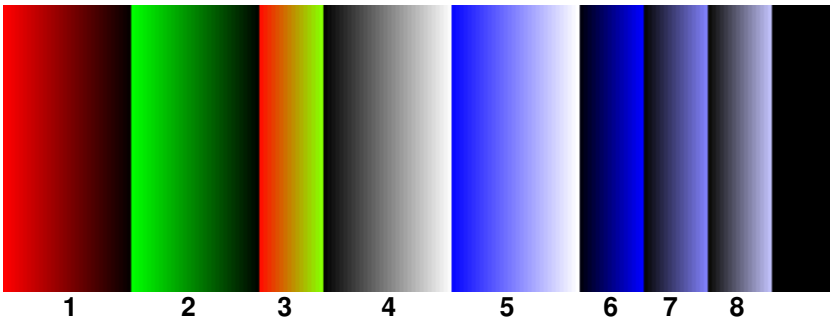
A zoomed picture of the grid below shows there are 8 color crossing cases need to be taken care of:

1. Red to Black
2. Green to black
3. Red to Green
4. Black to White
5. Blue to White
6. Black to Blue
7. Black to SkyBlue128
8. Black to SkyBlue192



Zoomed grid picture shows the color crossing cases need to be taken care of

In the case of 1~3 the changes are between solid colors, so no transparency is required, for case 4~8, several levels of alpha blending is added to make it look smooth when the grid moving over different background colors. The picture below is created for each pair of crossed colors, and the detail is shown in the chart below the picture.

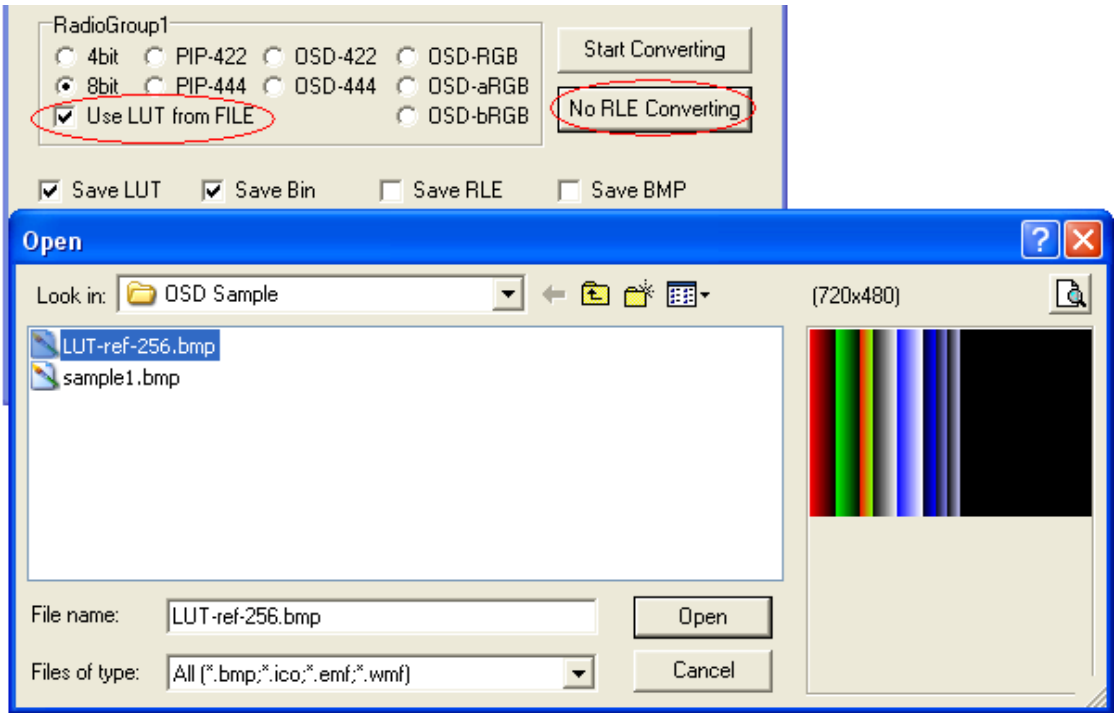


The look up colors for the picture

						Bitmap			LUT			
	From	To	Index	Step	Inc	B	G	R	B	G	R	Alpha
1	Red	Black	00~1F	32	8	00->00	00->00	FF->07	00->00	00->00	FF->07	00->00
2	Green	Black	20~3F	32	8	00->00	FF->07	00->00	00->00	FF->07	00->00	00->00
3	Red	Green	40~4F	16	16	00->00	0F->FF	FF->07	00->00	0F->FF	FF->07	00->00
4	Black	White	50~6F	32	8	07->FF	07->FF	07->FF	07->FF	07->FF	07->FF	03->7F
5	Blue	White	70~8F	32	8	FF->FF	07->FF	07->FF	FF->FF	07->FF	07->FF	60->7F
6	Black	Blue	90~9F	16	16	0F->FF	00->00	00->00	0F->FF	00->00	00->00	05->6F
7	Black	SkyBlue128	A0~AF	16	16	08->F8	00->00	00->00	08->F8	00->00	00->00	06->6F
8	Black	SkyBlue192	B0~BF	16	16	08->F8	00->00	00->00	08->F8	00->00	00->00	07->7F

The detail of the color crossing and transparency level involved in the picture

4) Generate Binary file and LUT by using tool with option of "**Use LUT from FILE**", select "**LUT-ref_256.bmp**" then convert.



5) Edit the LUT by adding Alpha Blending and Blinking attribute value from the table below to it. The 32-bit LUT consists of 24-bit RGB, 7-bit alpha blending attribute and one bit blinking attribute.

From	To	Index	B	G	R	Alpha
Red	Black	00~1F	00->00	00->00	FF->07	00->00
Green	Black	20~3F	00->00	FF->07	00->00	00->00
Red	Green	40~4F	00->00	0F->FF	FF->07	00->00
Black	White	50~6F	07->FF	07->FF	07->FF	03->7F
Blue	White	70~8F	FF->FF	07->FF	07->FF	60->7F
Black	Blue	90~9F	0F->FF	00->00	00->00	05->6F
Black	SkyBlue128	A0~AF	08->F8	00->00	00->00	06->6F
Black	SkyBlue192	B0~BF	08->F8	00->00	00->00	07->7F
N/A	N/A	C0~CF	00->00	00->00	00->00	00->00
Red	Black	D0~EF	00->00	00->00	FF->07	00->00
Green	Black	D0~EF	00->00	FF->07	FF->07	00->00
Black	Black	F0~FF	00->00	00->00	00->00	07->5F

The data is in order of B(0000~00FF)→G(0100~01FF)→R(0200~02FF)→Attr(0300~03FF) in the table. D0~EF is for warning message color, there are 2 type, Red and Yellow with different LUT, F0~FF is for Black line surrounding characters and grid, the final LUT of Blue background plus Yellow warning message looks like below.

Also see User Manual of EVB for detail operation instruction for the SPI OSD demo.

```
00000000  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
00000070  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000080  FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF FF
00000090  0F 1F 2F 3F 4F 5F 6F 7F-8F 9F AF BF CF DF EF FF
000000A0  08 18 28 38 48 58 68 78-88 98 A8 B8 C8 D8 E8 F8
000000B0  08 18 28 38 48 58 68 78-88 98 A8 B8 C8 D8 E8 F8
000000C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000000D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000000E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
000000F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

LUT value of Blue Color (0000~00FF)

00000100	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000110	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000120	FF	F7	EF	E7	DF	D7	CF	C7-BF	B7	AF	A7	9F	97	8F	87
00000130	7F	77	6F	67	5F	57	4F	47-3F	37	2F	27	1F	17	0F	07
00000140	0F	1F	2F	3F	4F	5F	6F	7F-8F	9F	AF	BF	CF	DF	EF	FF
00000150	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000160	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000170	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000180	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000190	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000001A0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000001B0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000001C0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000001D0	FF	F7	EF	E7	DF	D7	CF	C7-BF	B7	AF	A7	9F	97	8F	87
000001E0	7F	77	6F	67	5F	57	4F	47-3F	37	2F	27	1F	17	0F	07
000001F0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00

LUT value of Green Color (0100~01FF)

00000200	FF	F7	EF	E7	DF	D7	CF	C7-BF	B7	AF	A7	9F	97	8F	87
00000210	7F	77	6F	67	5F	57	4F	47-3F	37	2F	27	1F	17	0F	07
00000220	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000230	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000240	FF	F7	EF	E7	DF	D7	CF	C7-BF	B7	AF	A7	9F	97	8F	87
00000250	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000260	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000270	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000280	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000290	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000002A0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000002B0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000002C0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000002D0	FF	F7	EF	E7	DF	D7	CF	C7-BF	B7	AF	A7	9F	97	8F	87
000002E0	7F	77	6F	67	5F	57	4F	47-3F	37	2F	27	1F	17	0F	07
000002F0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00

LUT value of Red Color (0200~02FF)

00000300	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000310	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000320	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000330	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000340	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
00000350	03	07	0B	0F	13	17	1B	1F-23	27	2B	2F	33	37	3B	3F
00000360	43	47	4B	4F	53	57	5B	5F-63	67	6B	6F	73	77	7B	7F
00000370	60	61	62	63	64	65	66	67-68	69	6A	6B	6C	6D	6E	6F
00000380	70	71	72	73	74	75	76	77-78	79	7A	7B	7C	7D	7E	7F
00000390	05	0B	11	17	1D	23	29	2F-35	3B	41	47	4D	53	59	5F
000003A0	06	0D	14	1B	22	29	30	37-3E	45	4C	53	5A	61	68	6F
000003B0	06	0D	14	1B	22	29	30	37-3E	45	4C	53	5A	61	68	6F
000003C0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000003D0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000003E0	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
000003F0	07	0F	17	1F	27	2F	37	3F-47	4F	57	5F	5F	5F	5F	5F

LUT value of Alpha Blending (0300~03FF)