**OSD Engine Spec**

|  |  |  |  |
| --- | --- | --- | --- |
| Ver. | Author | Date | Revision |
| 0.1 | Chen Changhong | 2018/4/23 | Initial version |
| 0.2 | Chen Changhong | 2018/5/9 | Add bpp of Bitmap/Glyph  Add palette color format selection |

1. Full Diagram



1. User cases

## Monitor

For monitor mode without OSD frame buffer, OSD engine is enabled.



## TV with OSD Engine

ATV with OSD engine use same diagram with monitor mode



## ATV without OSD Engine

For some cases, if OSD cannot meet the requirement of OSD drawing, CPU can draw OSD frame to frame buffer directly. In this mode, OSD Engine is disabled. CPU can utilize DMA to improve OSD performance.



1. Features

## Entity relationship



Resource RAM: font glyph + bitmap

Index RAM: Font or bitmap index in the place, each column take 3 bytes (index + color + width\_delta)

(Each rectangle take 4 Bytes)

|  |  |  |
| --- | --- | --- |
|  | Monitor | ATV with OSD Engine |
| Resource RAM | 16K  For 16x16 1bpp font  16\*1024/(16\*16/8) = 512 glyphs  For 16x16 2bpp font  16\*1024/(16\*16/4) = 256 glyphs | 32K  For 16x16 1bpp font  32\*1024/(16\*16/8) = 1024 glyphs  For 16x16 2bpp font  32\*1024/(16\*16/4) = 512 glyphs |
| Index RAM | 16K  For 16x16 font, 1920x1080 OSD  16\*1024/(1920/16\*3) =45 rows | 32K  For 16x16 font, 960x540 OSD  32\*1024/(960/16\*3) =182 rows |
| Rectangle RAM | 1K  1024/4 = 256 rectangle | 2K  2048/4 = 512 rectangles |
| ATV without OSD Engine | only DDR used, 960x540x4=2M | |

## Width\_delta field of glyph/bitmap

Width\_delta field is to adjust width of current cell, it’s can be positive or negative.



## OSD Engine Supported

|  |  |  |
| --- | --- | --- |
| Object | Max Count | Description |
| Window | 8 | Support z-order  Support alpha blending  Support any overlap between windows  All RAM (Index/Resource/Rectangle) are configurable between windows. (1K unit?)  One window can support 32 rows |
| Row | Limit by Index RAM | Each window has several rows  Height of row can be different  Width of cell in one row is same  Row’s x and y can be any position in window  Height unit: 4, total row should not larger than index RAM count  One row can be a glyph row or a bitmap row  Glyph row is forbidden to overlay glyph row  Bitmap row is forbidden to overlay bitmap row  Bitmap row allow overlay with glyph row in any order  Row has a field called resource address, which pointer to address of resource RAM  Different size of glyph use different coding sequence.  For example:  For 16x16 glyph: index 1 means ‘A’  For 20x20 glyph: index 1 may means ‘B’  Glyph address = resource\_address + index \* cell\_width \*cell\_height / 8  Bitmap address = resource\_address + index \* cell\_width \*cell\_height |
| Cell | Limit by Index RAM | Width unit: 4  Index: font or bitmap index of resource RAM, 10bit  Color: color of character, unused for bitmap, 8 bit  width\_delta: 6 bit, highest bit can be negative or positive, so its value is between -32 and 32 |
| Palette | 2 | Each window has 2 palettes  The one for font and rectangle, and another for bitmap  Palette support 256 color  Palette value can be RGB888 or RGB565 |
| Bitmap | Limit by Resource RAM | Bitmap support 1/2/4/8 bpp  The type of bpp is defined in the Row data |
| Glyph | Limit by Resource RAM | Pixel information of font  Glyph should support 1/2/4 bpp font  For 1 bpp, 1 means draw the pixel, 0 means nothing  For 2/4 bpp, the value means intensity of font color to blending to background.  For example:  font\_color(r1,g1,b1)  background\_color: (r2, g2,b2)  if font pixel value: 2  r = 2/3 \* r1 + 1/3 \* r2  g = 2/3 \* g1 + 1/3 \* g2  b = 2/3 \* b1 + 1/3 \* b2  For 4 bpp  if font pixel value: 5  r = 5/15 \* r1 + 10/15 \* r2  g = 5/15 \* g1 + 10/15 \* g2  b = 5/15 \* b1 + 10/15 \* b2  The type of bpp is defined in the Row data |
| Rectangle | Limit by Rectangle RAM | One window support 4 group rectangle  Rectangle in one group should not overlay  Rectangle in difference group can overlay  Support 4 colors for 4 border, border weight, border style  Gradient color fill  left to right  top to bottom  left/top to right/bottom  left/bottom to right/top  corner to center |

## Others

Output format: ARGB8888

Configuration Double Buffer: no extra hardware needed, we just change window’ RAM range in runtime