SSD192X
JPEG ENGINE
(Decoder)

Prepared by Tok Aik Hong

2008 – 04 - 10

Table of Content

1. INTRODUCTION 1

2. JPEG FILE 2

2.1 JPEG HEADER 2
2.1.1 SOI and JFIF APP0 marker 2
2.1.2 JPEG JFIF Marker 3

3. JPEG DECODER ENGINE 4

3.1 SETTING THE REGISTERS FOR DECODING 4
3.1.1 Initialization 4
3.1.2 Control Parameter 4
3.1.4 Input and Output 5
3.2 DECODING THE JPEG FILE 5
3.2.1 Decode process for header 5
3.2.2 Decode process for body 6

4. DISPLAYING JPEG FILE 8

4.1 JPEG PICTURE FIT PERFECTLY TO DISPLAY PANEL 8
4.2 JPEG PICTURE IS SMALLER THAN THE DISPLAY PANEL 8
4.2.1 Register to set before decoding JPEG body 9
4.2.2 Displaying picture on the display panel 9
4.2 DISPLAYING A PICTURE LARGER THAN DISPLAY SCREEN 11
4.2.1 Scaling down the picture 12
4.2.2 Stretching the picture 13
4.2.3 Actual setting of various register 13
4.3.4 Points to note 15
4.3.5 Memory management 16

1. Introduction

SSD192X had a JPEG engine which is able to encode a picture into JPEG format, and also able to decode a JPEG file and display on the display panel.

This document will be focusing on the decoder portion of the JPEG engine. It will be divided into 3 sections :

1) A brief introduction to JPEG picture format.

2) Description on usage of the decoder part of the JPEG engine, on how to set the various JPEG related registers for decoding a jpeg picture.

3) To discuss how to fit picture of various size on to different display panel size. (The document and code provided will be using QVGA (320 x240), 16 bit per pixel as a reference for explanation.)

This document aims to help the user to be able to familiarize with SSD192X JPEG decoder engine fast, and be able to use it for different project requirements.

The application note of SSD 192X will contain many important references on how to calculate and set the register values. It will not be repeated in this document. The document will indicate from time to time when to refer back to the application notes for further explanation.

2. JPEG file

JPEG (Joint Photographic Experts Group) refers to a standards organization, a method of file compression, and sometimes a file format.

(*This section is more for understanding of JPEG file and thus enable one to understand better how the JPEG decode work.*)

JPEG file can be “visually” divided into 2 portions.

a) The first part of the file is the information needed by the decoder to decode and display the JPEG file to its original size for display.

b) The second portion of the file contains the picture raw information.

2.1 JPEG header

2.1.1 SOI and JFIF APP0 marker

The beginning bitstream of any JPEG file will consist of the following :

```
BYTE SOI[2];          // 00h  Start of Image Marker
BYTE APP0[2];         // 02h  Application Use Marker
BYTE Length[2];       // 04h  Length of APP0 Field
BYTE Identifier[5];   // 06h  "JFIF" (0 terminated) Id
BYTE Version[2];      // 07h  JFIF Format Revision
BYTE Units;           // 09h  Units used for Resolution
BYTE Xdensity[2];     // 0Ah  Horizontal Resolution
BYTE Ydensity[2];     // 0Ch  Vertical Resolution
BYTE XThumbnail;      // 0Eh  Horizontal Pixel Count
BYTE YThumbnail;      // 0Fh  Vertical Pixel Count
```

SOI is the start of image marker and always contains the marker code values FFh D8h.

APP0 is the Application marker and always contains the marker code values FFh E0h.

Length is the size of the JFIF (APP0) marker segment, including the size of the Length field itself and any thumbnail data contained

in the APP0 segment. Because of this, the value of Length equals 16 + 3 * XThumbnail * YThumbnail.

Identifier contains the values 4Ah 46h 49h 46h 00h (JFIF) and is used to identify the code stream as conforming to the JFIF specification.

Version identifies the version of the JFIF specification, with the first byte containing the major revision number and the second byte containing the minor revision number. For version 1.02, the values of the Version field are 01h 02h; older files contain 01h 00h or 01h 01h.

Units, Xdensity, and Ydensity identify the unit of measurement used to describe the image resolution. Units may be 01h for dots per inch, 02h for dots per centimeter, or 00h for none (use measurement as pixel aspect ratio). Xdensity and Ydensity are the horizontal and vertical resolution of the image data, respectively

XThumbnail and YThumbnail give the dimensions of the thumbnail image included in the JFIF APP0 marker.

To understand more about these marker, one may go to this webpage for further reading :
http://netghost.narod.ru/gff/graphics/summary/jfif.htm

2.1.2 JPEG JFIF Marker

The bitstream that follow after the SOI and APPO marker are the information of the JPEG picture on how its encoded, and its actual horizontal and vertical size. Each information header is marked by "**FFXX**", thus the jpeg decoder will scan for this header and abstract the data accordingly.

The header which indicate that the value follow after it is the picture data is "FFDA", thus once JPEG decoder read this value, it will set a flag to indicate that the decoding of header is done.

For more understanding and reading of the marker, the following webpage is a good reading material :

http://www.impulseadventure.com/photo/jpeg-decoder.html

3. JPEG decoder engine

The figure below is used to describle the JPEG decoder engine and how to various setting of the register will control the JPEG engine.

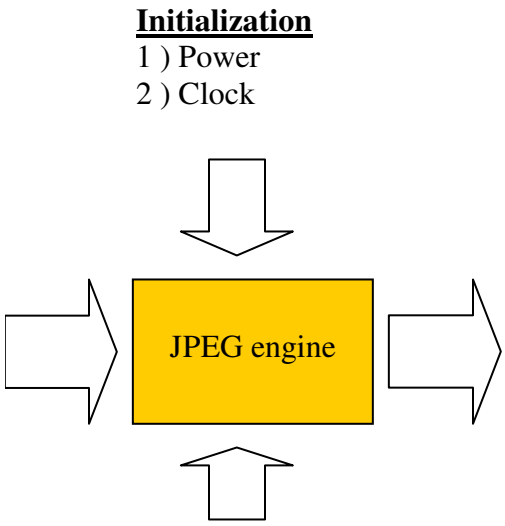

Figure 1 : JPEG engine system

3.1 Setting the registers for decoding

The steps below describle on the setting of various registers to decode the header and body of JPEG with reference to the above diagram

3.1.1 Initialization

1) Enable the JPEG Codec (REG[380h] bit 0 = 1). This will turn on the codec and the clock to the JPEG engine

3.1.2 Control Parameter

1) Software reset the JPEG Codec (REG[402h] bit 7 = 1).

2) Set operation mode to decode (REG[400h] bit 2 = 1).

3) Clear JPEG MJPEG Mode (REG[400h]) bit 5, as still JPEG is being decoded and not motion JPEG.

4) Set the RST Marker Operation Setting (REG[41Ch]) to 0x02. (Refer to application note on the various setting of RST, 0x02 was chosen so the JPEG continue to function even there is a corrupt input file)

5) Register 386 and 387 is the interrupt status register. Set it accordingly to different user requirement. (Refer to application notes for more detail)

3.1.4 Input and Output

1) Set the JPEG Source Start Address (REG[414h] – REG[416h]). This address refer to the JPEG address where the source data is going to be written in (input).

2) Set the JPEG Destination Address (REG[410h] – REG[412h]). This address refer to the JPEG address where the decoded data for display is stored (output).

(Note as JPEG engine is 32 bits addressing, hence the maximum address to be written shall not be more than 0xFFFF)

3) Set the JPEG FIFO Size (REG[3A4h]).
(Set the size in multiply of 8, Example, to set a 64K FIFO size, the value to write is 0x0F. For actual calculation, please refer to the application note.)

4) Set the JPEG File Size (REG[3B8h] – REG[3BAh]).

5) Set JPEG YUV Output Data Range Select (REG[380h]) bit 4.

3.2 Decoding the JPEG file

3.2.1 Decode process for header

1) Clear all status of JPEG Line Buffer and FIFO (REG[382h] – REG[383h] = 0000h).

2) Start JPEG operation (REG[402h] bit 0 = 1).

3) Start decoding (REG[38Ah] bit 0 = 0.

4) Write data to JPEG FIFO (REG[414h – REG[416h]]
(The actual decoding only start when data is written to this address.)

5) Keep writing data to JPEG FIFO till, "Raw JPEG Decode Marker Read Flag" (REG[385h], bit 4) is set.

6) Once set, read the Vertical Pixel Size (REG[3DCh] – REG[3DDh]) and Horizontal Pixel Size (REG[3D8h] – REG[3D9h]). This will be use to set the registers related to display panel.

7) The usual size for the JPEG header information shall not be more than 2kbytes.

3.2.2 Decode process for body

1) Clear all status of JPEG Line Buffer and FIFO (REG[382h] – REG[383h] = 0000h).

2) Enable required interrupts (REG[386h] – REG[387h]).

3) Software reset the JPEG Codec (REG[402h]) by setting bit 7.

4) Start decoding (REG[38Ah] bit 0 = 0).

5) Write data to JPEG FIFO (REG[414h – REG[416h]]

6) Poll for "Raw JPEG Decode Complete Flag" and " Raw JPEG Codec Interrupt Flag" (REG[385h], bit 5 & 1).

7) If both are set at the same time, the decoding is complete. But if "Raw JPEG Codec Interrupt Flag" is set before "Raw JPEG Decode Complete Flag", there is some problem in the decoding.

8) Another situation in which the JPEG decoding will stop is when the size of the input data is equal to the file size stated in REG[3B8h].

9) Turn off the JPEG codec as this will free up the memory space for other usage. Set REG[380h] bit 0 = 0. This will turn off the codec and the clock to the JPEG engine.

Verify the decode process is complete with the JPEG Operation Status (REG[404h] bit 0 = 0). Check JPEG RST Marker Operation Status Register (REG[41E]) to ensure there is no error in the decoding of the JPEG picture.

4. Displaying JPEG file

After decoding the JPEG file, the next step will be showing the picture on the display panel. In most case, the height and width of the JPEG file may be too small or big for the display panel. Seldom would one encounter a perfect fit for the display panel.

In view of this, we could use one of the JPEG engine features, 2D engine and software code to best fit the picture into the display panel nicely.

This section will cover 3 cases.

1) The jpeg picture fit perfectly to the display panel

2) The jpeg picture is smaller than the display panel

3) The jpeg picture is bigger than the display panel

As 192X is usually use for QVGA display, the section below will use 320 x 240 as a reference for the above 3 cases. For other display panel size, using the same principal below shall also achieve the same effect.

4.1 JPEG picture fit perfectly to display panel

If the picture fit the size of the display panel nicely or a bit smaller (the tolerance will depend on individual. Generally if the decoded picture is 10 or lesser pixels smaller than the display side, no resizing is really needed), no other registers setting are needed.

Hence, to test the JPEG engine and the code written, its advisable to resize the JPEG picture to the display panel size before pumping it to 192X, as minimal registers are need to be set in this situation.

4.2 JPEG picture is smaller than the display panel

If the JPEG picture is much smaller than the display panel, it would be nice to display the picture in the middle of the screen, rather than on the top left corner of the screen.

To achieve this, we will be using the floating window feature in 192X chip.

4.2.1 Register to set before decoding JPEG body

This registers are set before the JPEG engine start to decode the body, preferably after getting the horizontal and vertical value of the picture.

1) Set REG[360h] to 0x01. This is to enable the resize viewing mode

2) Set REG[360h] to 0. This is the start position of the X co-ordinate for resize viewing mode.

3) Set REG[366h] to 0. This is the start position of the Y co-ordinate for resize viewing mode.

4) Set REG[368h] to 1 pixel lesser than the display panel width (in QVGA case, 319). This is the end position of the X co-ordinate for resize viewing mode.

5) Set REG[36Ah] to 1 pixel lesser than the display panel height (in QVGA case, 239). This is the end position of the Y co-ordinate for resize viewing mode.

6) Set REG[36Ch] to 1. This register indicates how many times the JPEG picture is scale down from its original size. (Refer to application note for different value of scaling).

7) Set REG[366h] to 1. This register indicates the operation of the resize engine (Refer to application note for further explanation).

Once the above registers are set, continue to proceed on with the decoding of the main body of the JPEG picture.

4.2.2 Displaying picture on the display panel

The figure below show the various register that need to be set for floating window. (Please refer to application notes on the details on how to set each register)

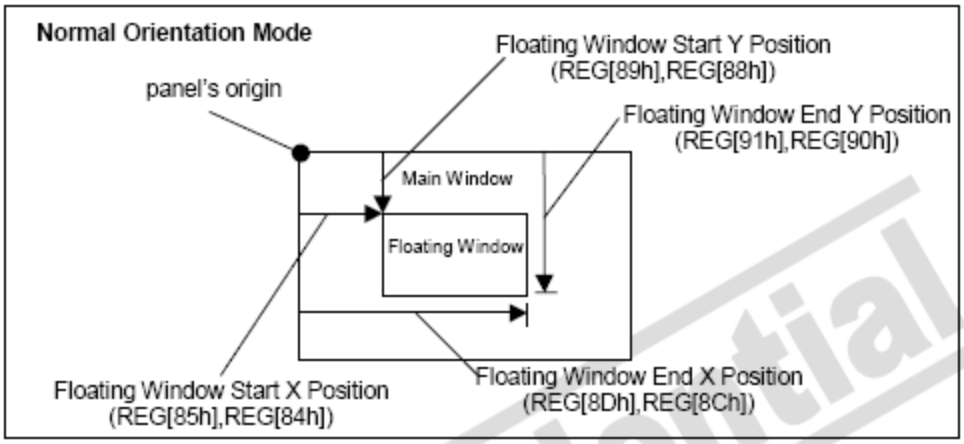

Figure 2 : Floating window

Do take note the width & height of the floating window is equal to the decoded picture horizontal and vertical pixel.

An example, to display 200 x 180 on a QVGA (320 x 240) panel.

Register 84 = 60 (0x3C)
Register 89 = 30 (0x1E)
Register 8C = 159 (0x9F)
Register 91 = 209 (0xD1)

In addition, some other register that need to be set for floating window :

1) Register 0x7C (the start address of the floating window). For ease of programming, The value can be the same as JPEG decoded address (REG [410h – 412h]).

2) Register 0x80 (floating window line offset). This is the width offset to the next line.

3) Register 0x71, to enable the floating window.

4) Register 0x1A4. This is to set the floating window to YUV or RGB output.

Lastly, to display the picture in the center of the screen with a black background, please set the following register to the value as follow :

1) Register 0x1A4. Bit 6 = 1, this will force the main window to be RGB.

2) Register 0x74. Set the main window address to 0x10000. Our frame buffer is only up to 0xFFFF addressing. By placing a 0x10000 value in it, the display panel will show a black background.

When the user enables the display, it shall be nicely in the center of the screen with a black background.

4.2 Displaying a picture larger than display screen

This section will describe one of the methods that can be used to fit the picture to the display screen. There can be many other ways to do it, and its up to the different engineer creativity and preference.

Generally this method will use 3 features of SSD192X.

1) Use the JPEG decoder scaling feature to downsize the picture to fit into the display panel. The picture can be scale down to 2 times, 4 times and up to a maximum of 8 times.

2) Use the 2D engine, stretch feature. This will stretch the picture to the desired width and height.

3) Use the floating window to position the picture in the center of the display panel.

The most challenging part of this case is the limited frame buffer area. One had to be very mindful that the JPEG engine had an input and output area, the 2D engine also had a source and destination area, hence all this address, and area allocated shall not overlap else the data will be corrupted. A section on the memory management will be discussed in the last part.

The process can be summarize into the figure show below. A simple example will be used to demonstrate on what value to put into the various registers.

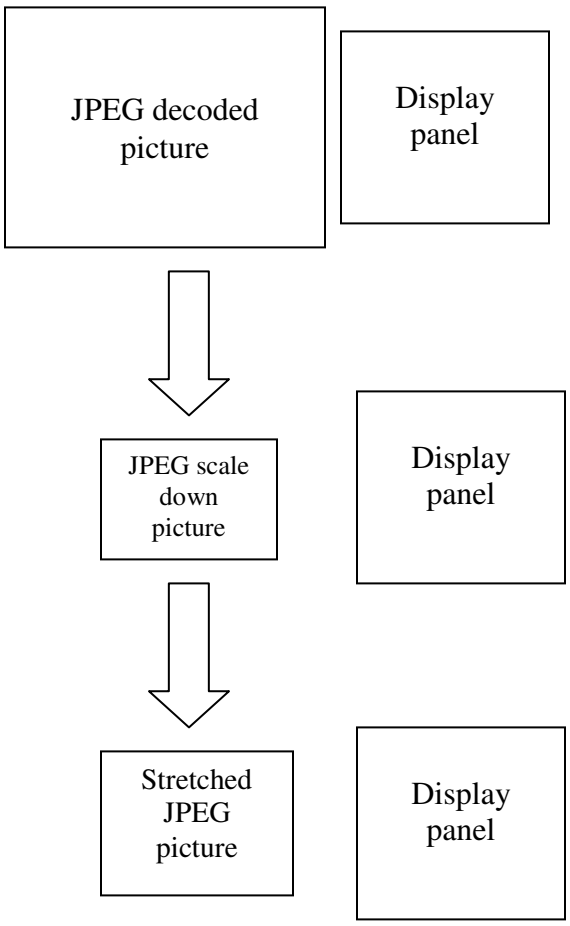

Figure 3 : Displaying a larger picture on the display panel

4.2.1 Scaling down the picture

To determine the scaling factor, one would need to ensure that the final value of the horizontal and vertical is smaller than the display panel width and height.

After the decoding the JPEG header, the code will know the picture size. It will then determine the scaling factor, write to the related registers, so that after the JPEG engine decode the body of the JPEG, it will output the wanted scale down size.

4.2.2 Stretching the picture

This step involves the use of 2D engine of our SSD192X. Do note that the addressing of the 2D engine is different from the JPEG engine.

For example, to access 0x12000, one had to write (0x12000)/2 to the source or destination address of the 2D engine. If using JPEG engine, the value would be (0x12000)/4

Another point to take note is that the line buffer of the stretching engine is only 128 pixels. Thus to stretch a 200 pixels wide picture to fit to a 320 pixel display panel, one had to do 4 times, each time stretching 50 pixels of the picture to 100 pixels of display panel.

The height of the picture is of no impact, since the stretching is done line by line. If user specifies the height to be 50, it will automatically stretch 50 lines.

This step may be give a miss if after the JPEG picture is down sized, the picture can almost fit the display panel well. Doing additional stretching may have very little improvement on the display in this case.

4.2.3 Actual setting of various register

To show a ***600 x 360*** jpeg file on a ***320 x 240*** display, below are the steps to be taken (assume 16bpp display).

After decoding the header, we would know the size of the picture. Using software, one would derive that the resize factor shall be 2. Hence the register to set after decoding the header would be :

1) Set REG[360h] to 0x01. This is to enable the resize viewing mode

2) Set REG[360h] to 0. This is the start position of the X co-ordinate for resize viewing mode.

3) Set REG[366h] to 0. This is the start position of the Y co-ordinate for resize viewing mode.

4) Set REG[368h] to 1 pixel lesser than the display panel width (in QVGA case, (320*resize factor) -1 = 639). This is the end position of the X co-ordinate for resize viewing mode.

5) Set REG[36Ah] to 1 pixel lesser than the display panel height (in QVGA case, , (240*resize factor) -1 = 479). This is the end position of the Y co-ordinate for resize viewing mode.

6) Set REG[36Ch] to **2.** This register indicates how many times the JPEG picture is scale down from its original size.

7) Set REG[366h] to 1. This register indicates the operation of the resize engine

Once the above registers are set, continue to proceed on with the decoding of the main body of the JPEG picture. The output picture will be in the size of 300 x 180.

The next step will determine the amount of stretching to be done. The stretching must be done to keep the aspect ratio of the picture intact. In addition, we must also consider the display panel aspect ratio.

Hence there are 2 scenarios on how the picture can be stretched.

1) Picture aspect ratio < Display panel aspect ratio

2) Picture aspect ratio > Display panel aspect ratio

Case 1 :

1) Get the stretching ratio for the horizontal side. Divide the display vertical pixel divide by the picture vertical pixel. (Assume we will stretch to the maximum vertical display panel size)

2) Multiply the stretching ratio by the horizontal picture pixel to determine how much the picture is to be stretched horizontally.

3) Determine how many times that is need to be stretched horizontally (remember each stretched horizontal value cannot be more than 128 pixels)

4) Do the stretching. The routine “stretch” in the attached code will show what values are to be put into the 2D register. Generally, it’s the original width, stretched width, line offset, and address of original and stretched picture.

Case 2 :

1) Get the stretching ratio for the vertical side. Divide the display horizontal pixel divide by the picture horizontal pixel.. (Assume we will stretch to the maximum horizontal display panel size)

2) Multiply this value by the vertical picture pixel to determine how much the picture is to be stretched vertically.

3) The number of time to stretch horizontally is 4 for a QVGA display. For a QVGA display, the total number of bytes in 1 line is 640 bytes, and the max line buffer is 256 bytes (128 pixels). In order to have the same number of horizontal bytes to be stretched to each round, 4 is the nearest number.

Base on the above example of a 300 x 180 decoded picture, the picture vertical side will be stretched to :

(display panel pixel width / 300) * 180
= (320 / 300) * 180
= 192

4) Due to the frame buffer limitation, we will divide the picture into 4 parts vertically, and stretched each portion so as to ensure there is no overlap.

4.3.4 Points to note

1) In using this method, the the picture is first downsized to a small picture before stretching back. This will degrade the quality of the picture view. However, as the display is QVGA, it is still acceptable in most case.

2) When a picture is downsize and have odd number, (for example, 193 x 137) , the code had to cater for this situation as the 2D stretching engine for YUV picture must be even number. The principal is to forgo the last horizontal / vertical line as this will not affect the quality of the picture.

Thus for a 193 x 137 picture, we will only stretch 136 x 192 to 320 x 226. The code attached will cater for this situation.

4.3.5 Memory management

This section discuss particularly on how my code manage the memory, it may not be applicable to other user who had other way to do scaling.

In my code, when the JPEG engine downsize any picture and output it, the maximum memory it would needed to hold the decoded picture would be the display panel size (320 x240 x2 bytes, assuming 16bpp).

After downsizing, as there may be a need to do stretching, hence the code also needs to allocate an area for the output of the 2D engine.

All this could be summarize in the memory picture below, step by step on the reason how I allocate the address for JPEG decoder, and the 2D engine input and output address.

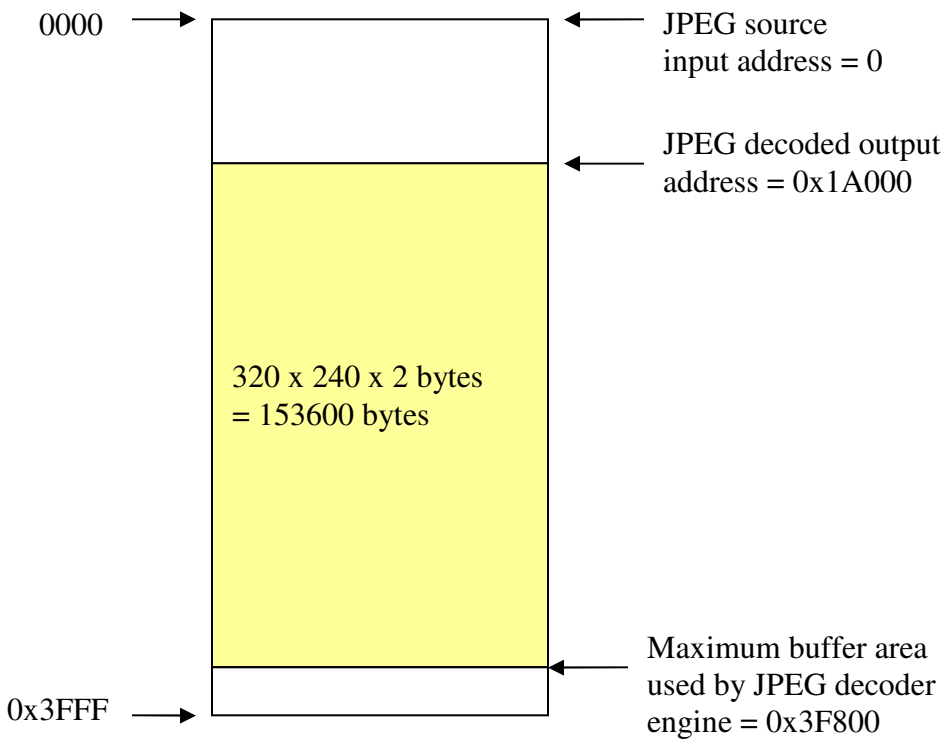

Figure 4 : JPEG decoder memory management

After the JPEG engine had done it job, it will be turn off. Hence the picture will now be at location 0x1A000 to 0x3F800. If no resizing is needed, we set the main window display address to be 0x1A000.

If resizing is needed, we will use address 0 as the 2D engine destination address, and 0x1A000 as the source address as shown below.

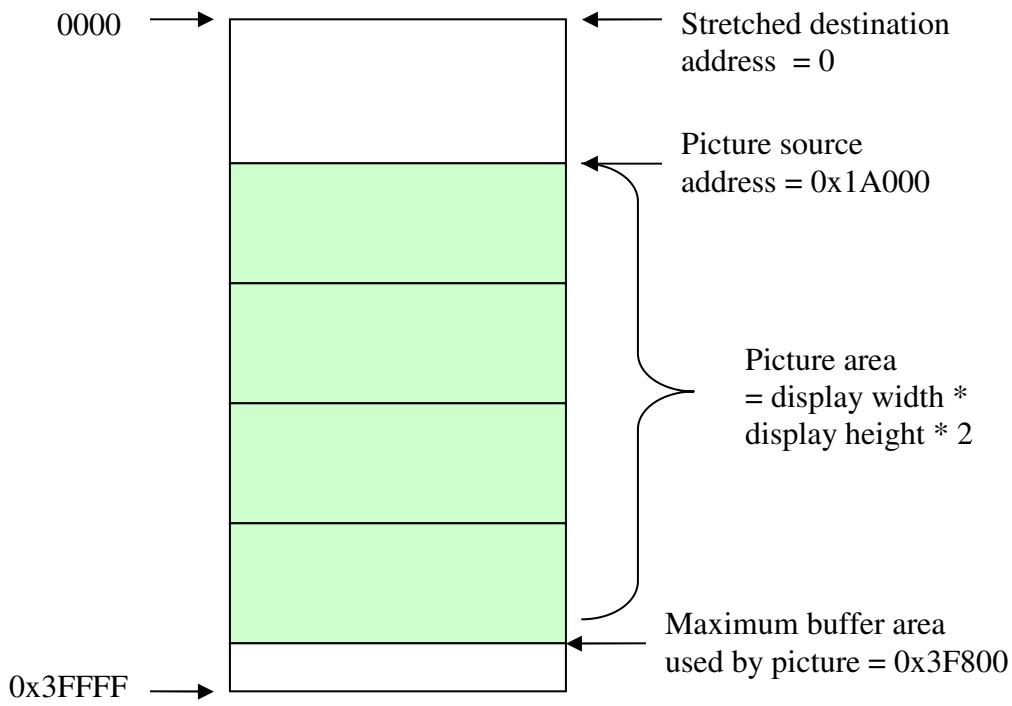

Figure 5 : 2D engine memory management

We cannot resize the whole picture at 1 time because there is only 0x1A000 bytes available, and there is a total of 0x0x25800 bytes in the picture.

To overcome this issue, we will divide the source picture into 4 portions, and stretch them to the destination address accordingly. This will ensured that when doing the stretching, there will always be enough memory space to hold the stretched picture.

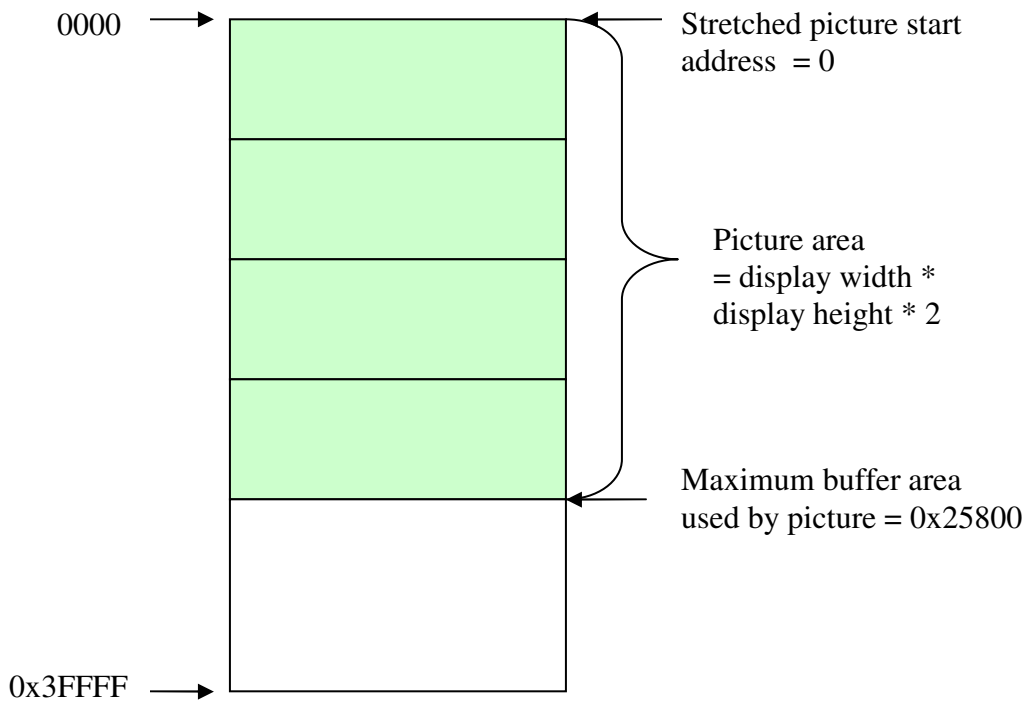

Figure 5 : Location of picture after stretching

After the stretching is done, the start address of the picture will be at 0. Thus by setting the main window address to be 0, the picture will be shown.

Revision	Date	Changes	Author
1.0	10-Apr-2008	Initial Revision	Aik Hong
1.1	24-Apr-2008	1) Stretching of picture had to take in consideration of the display aspect of the display panel 2) Stretching YUV picture must be in even number	Aik Hong