I have an 800x600 pixels RGB565 image stored in the SDRAM of my STM32F7-discovery board starting at memory address 0xC0390000.

What I want to do is saving this image to an SD card.

The code for this is as follows:

```
FIL file; //file object
      char file_str[30] = "image01.bmp";//filename

      DMA2D_HandleTypeDef hdma2d_buf;

      int32_t i, j;
      uint32_t tmp = 0;

      uint16_t img_buf[800];//buffer for one line

      UINT bytes_written;

      FRESULT fs_error;

      file_str[6] = (char)((img_counter % 10) + 48);//increase the filename
      file_str[5] = (char)(((img_counter/10) % 10) + 48);

      //create a test pattern, black pixels
      for(j = 0; j < 640; j++)
      {
          if((j%2) == 0) img_buf[j] = 0x0;
          else img_buf[j] = 0x0;
      }

      fs_error = SD_Mount();
      if(fs_error != FR_OK) Filesystem_ErrorHandler(fs_error);

      fs_error = SD_OpenFile(&file, file_str, F_WR_CLEAR);
      if(fs_error != FR_OK) Filesystem_ErrorHandler(fs_error);

      fs_error = f_write(&file, bmp_fileheader800, 70, &bytes_written);//write the header

      if(fs_error != FR_OK)
      {
          Filesystem_ErrorHandler(fs_error);
      }

      for(i = 0; i < 600; i++)//600 lines
      {
          //copy one line in a RAM buffer using DMA2D

          /* Enable DMA2D clock */
          __HAL_RCC_DMA2D_CLK_ENABLE();

          /* Configure the DMA2D Mode, Color Mode and output offset */
          hdma2d_buf.Init.Mode         = DMA2D_M2M;
          hdma2d_buf.Init.ColorMode    = DMA2D_RGB565;
          hdma2d_buf.Init.OutputOffset = 0;

          /* Foreground Configuration */
          hdma2d_buf.LayerCfg[1].AlphaMode = DMA2D_NO_MODIF_ALPHA;
          hdma2d_buf.LayerCfg[1].InputAlpha = 0xFF;
          hdma2d_buf.LayerCfg[1].InputColorMode = CM_RGB565;
          hdma2d_buf.LayerCfg[1].InputOffset = 0;

          hdma2d_buf.Instance = DMA2D;
```

```
          /* DMA2D Initialization */
          if(HAL_DMA2D_Init(&hdma2d_buf) == HAL_OK)
          {
              if(HAL_DMA2D_ConfigLayer(&hdma2d_buf, 1) == HAL_OK)
              {
                  if (HAL_DMA2D_Start(&hdma2d_buf, (uint32_t)(CAPTURE_FRAME_BUFFER + tmp),
(uint32_t)&img_buf, 800, 1) == HAL_OK)
                  {
                      /* Polling For DMA transfer */
                      HAL_DMA2D_PollForTransfer(&hdma2d_buf, 10);
                  }
              }
          }

          fs_error = f_write(&file, img_buf, 1600, &bytes_written);
          if(fs_error != FR_OK)
          {
              Filesystem_ErrorHandler(fs_error);
          }
          tmp = tmp + 800*sizeof(uint16_t);
      }

      fs_error = SD_CloseFile(&file);
      if(fs_error != FR_OK) Filesystem_ErrorHandler(fs_error);

      fs_error = SD_UnMount();
      if(fs_error != FR_OK) Filesystem_ErrorHandler(fs_error);

      img_counter++;
```

As you can see, I always buffer one line of the image in the SRAM before wrting it to the file. This line is stored in `uint16_t img_buf[800]` which is first filled with zeros, so I get black pixels.

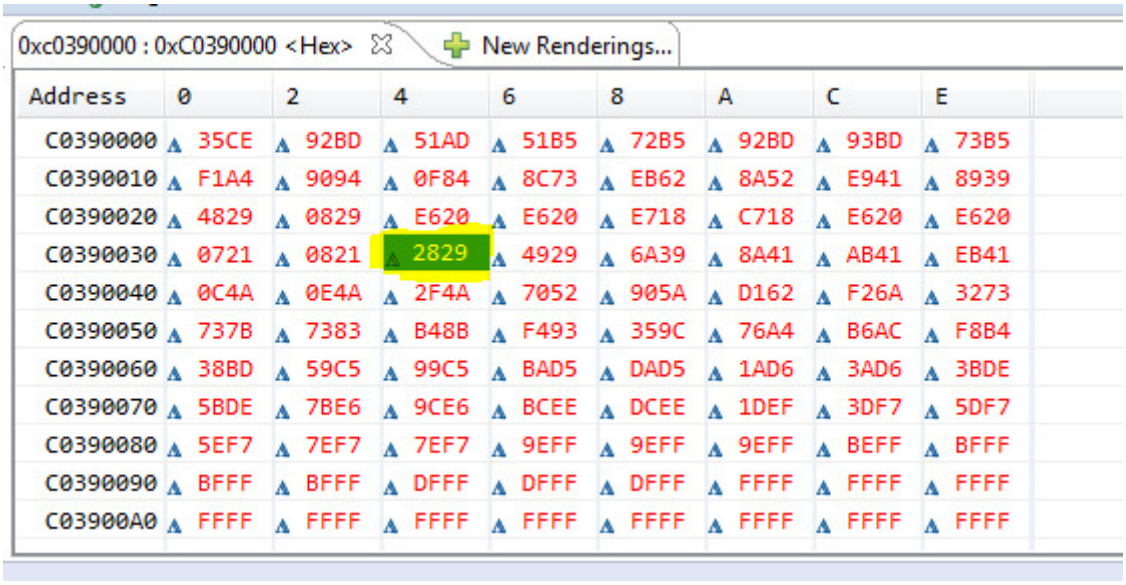
After that I want to copy one line to my RAM buffer using DMA2D. This is done 600 times, once for each line.

After each loop iteration I increment my SDRAM base address for the DMA2D by 1600, since each line has 1600 bytes (2 bytes per pixel).

This is where the problems start:

I get incorrect data in my SRAM buffer. I ran a debug session to verify this behaviour.

Here are the contents of the SDRAM and and the line array `img_buf`:



img_buf	uint16_t [800]	537196556 (Decimal)
[0...99]	uint16_t [100]	0x2004f80c (Hex)
img_buf[0]	uint16_t	0x0 (Hex)
img_buf[1]	uint16_t	0x0 (Hex)
img_buf[2]	uint16_t	0x0 (Hex)
img_buf[3]	uint16_t	0x0 (Hex)
img_buf[4]	uint16_t	0x0 (Hex)
img_buf[5]	uint16_t	0x0 (Hex)
img_buf[6]	uint16_t	0x0 (Hex)
img_buf[7]	uint16_t	0x0 (Hex)
img_buf[8]	uint16_t	0x0 (Hex)
img_buf[9]	uint16_t	0x0 (Hex)
img_buf[10]	uint16_t	0x0 (Hex)
img_buf[11]	uint16_t	0x0 (Hex)
img_buf[12]	uint16_t	0x0 (Hex)
img_buf[13]	uint16_t	0x0 (Hex)
img_buf[14]	uint16_t	0x0 (Hex)
img_buf[15]	uint16_t	0x0 (Hex)
img_buf[16]	uint16_t	0x0 (Hex)
img_buf[17]	uint16_t	0x0 (Hex)
img_buf[18]	uint16_t	0x0 (Hex)
img_buf[19]	uint16_t	0x0 (Hex)
img_buf[20]	uint16_t	0x0 (Hex)
img_buf[21]	uint16_t	0x0 (Hex)
img_buf[22]	uint16_t	0x0 (Hex)
img_buf[23]	uint16_t	0x0 (Hex)
img_buf[24]	uint16_t	0x0 (Hex)
img_buf[25]	uint16_t	0x0 (Hex)
img_buf[26]	uint16_t	0x2928 (Hex)
img_buf[27]	uint16_t	0x2949 (Hex)
img_buf[28]	uint16_t	0x396a (Hex)
img_buf[29]	uint16_t	0x418a (Hex)
img_buf[30]	uint16_t	0x41ab (Hex)
img_buf[31]	uint16_t	0x41eb (Hex)

As you can see the first few array elements stay zeros while the SDRAM content is clearly not zero. Then I get some correct values starting at the yellow marked elements, but after some time zeros agian and so on.

This behaviour is the same for all lines.

The image is displayed correctly on the onboard LCD.

So, the question is: Why am I getting wrong data via DMA2D?

I am pretty desperated right now because the DMA2D works well in several similar applications...

Thank you for any helpful hints.