Video Control Buffer

The video pixel control buffer register is used to format how the image to be displayed on the screen . Depending on how these registers are loaded depends how the video on the screen will look like

Device	Video Buffer	
Configuration	6 32-bit mapped registers	
Input/Output	Both	
Address Base	0xxFF203020	
Address	**R/W**	**Description**
base	R/W	**Front Buffer Address**: Holds video data to be displayed on monitor. If a 1 is written to this register then every 1/60 (vertical synchonization timer) the data from the back register is swapped into this register.
base+4	R/W	**Back Buffer Register**: While the front buffer is displaying data on the screen the back buffer updates the video display for the next cycle (1/60 second)
base+8	R/W	**Y** [bits 31-16] - y coordinate for pixel location **X** [bits 15-0] - x coordinate for pixel location
base+12	R/W	**M** [bits 31-24] - number of y address bits used **N** [bits 23-16] - number of x address bits used **B** [bits 7-4] - number of bytes used for each pixel (1 is minimum, 4 is maximum). **A** [bit 1] address format (1 expresses the X & Y FIELD) (0 expresses sequential [0 to pixel -1]) **S** [bit 0] status register - When swap mode is enabled (1 written to **Front Buffer**) then every 1/60 s the status bit goes low to indicate end of one video frame and beginning of another. It also indicates that the data in the **Back Buffer** has been swapped into the **Front Buffer**

Initialization See example below **Hardware Setup**Connect monitor to VGA interface on the DE1-SoC board

Notes:

1 - The Video control register works in conjuction with the **Pixel Buffer** 08000000 - 0803ffff. So for example if you want to put a pixel at location 250,110 and the pixel value is 16 bits

Base Address	08000000 (start location of video buffer)
Y	110 (y offset)
X	250 (x offset)
M	8 (number of address bits used for y coordinate)
N	9 (number of address bits used for x coordinate)
B	2 (number of video bytes used for **video pixels)**
A	0 (expressed as a **X** & **Y** field)
S	only active if swap mode is active

2 - The default values for N =9, M =8, B =2, A =0.
3 - If you are using **swap mode** (a value of 1 is written into **Front Buffer** FF203020) then each time a video frame is fiished 1/60 s, the **S** bit in the status will toggle from high to low. This indicates a swap of data from the back buffer to the front buffer. This is valuable if you are wanting to update your video while not changing the video displayed on the monitor. For more information go to the reference Double Buffer 4.2.2

Assembly Example: Enable Double Buffering and Video Streaming

```
.equ ADDR_Front_buffer, 0xFF203020
.equ ADDR_Video_in_controller, 0xFF20306C

  movia r2,ADDR_Video_in_controller
  movi  r3,0x4
  stwio r3,0(r2)        # Enable video in streaming
  movia r2,ADDR_Front_buffer
  movi  r3,0x1
  stwio r3,0(r2)        # Enable double buffer (swap pixel data from back buffer to front buffer)
```

Assembly Example: Enable Double Buffering using Slider Switch 0 to switch between two different Back Buffers locations

```
.equ ADDR_Front_Buffer, 0xFF203020
.equ ADDR_Slider_Switches, 0xFF200040

# SDRAM memory locations. Each will be the same memory size as the Front Buffer

 .equ VGA_Back_Buffer1, 0x01000000
 .equ VGA_End_Back1, 0x0103FFFF

 .equ VGA_Back_Buffer2, 0x02000000
 .equ VGA_End_Back2, 0x0203FFFF

# colour values for the Pixel buffer

 .equ red, 0xF100
 .equ blue, 0x001F
 .equ green, 0x07E0
 .equ white, 0xFFFF
 .equ yellow, 0xFFE0

# Fill back buffer1 memory locations with the colour red

 movia r2, red
 movia r3, VGA_Back_Buffer1
 movia r4, VGA_End_Back1

# counter to increment through each back buffer1 memory location until 256k locations have been filled with red Pixel value

 count1:
  sth r2, 0(r3)
  addi r3,r3,2
  ble r3,r4, count1

# Fill back buffer2 memory locations with the colour blue

 movia r2, blue
 movia r3, VGA_Back_Buffer2
 movia r4, VGA_End_Back2

# counter to increment through each back buffer2 memory location until 256k locations have been filled with blue Pixel value

 count2:
  sth r2, 0(r3)
  addi r3,r3,2
  ble r3,r4, count2

# load pointer for Back Buffer1

  movia r5, ADDR_Front_Buffer
  movia r3, VGA_Back_Buffer1
  stwio r3, 4(r5)       # set start location of back buffer1
  movi  r6, 1
  stwio r4, 0(r5)       # Enable double buffering

# wait for swap to be complete but waiting for status bit to go active low

  swapcheck:
  ldwio r3, 12(r5)  # load status bit
  andi r3, r3, 1
  beq r3,r6, swapcheck

# load pointer for Back Buffer2

  movia r3, VGA_Back_Buffer2
  stwio r3, 4(r5)       # set start location of Back Buffer2

# check slider switch
```

```
    movia r2,  ADDR_Slider_Switches
    mov r4, r0

   check:
    ldwio r3, 0(r2)
    andi r3,r3, 1
    beq  r3, r4, check
    mov r4, r3    # save switch setting

  # swap back buffers

   stwio r6, 0(r5)

  # wait for status bit to go low. This indicates that the pointer is properly set

   swapcheck2:
    ldwio r3, 12(r5)  # load status bit
    andi r3, r3, 1
    beq r3,r6, swapcheck2

   br check
```