Apple II Technical Notes

Developer Technical Support

Apple IIGS
#39: Mega II Video Counters

Revised by: Dave Lyons July 1989
Written by: J. Rickard May 1988

This Technical Note describes the Mega II video output registers, which your applications can use to get information about where the beam is located on the Apple IIGS display.

Changes since November 1988: Corrected description of when VBL begins and simplified example code to read the scan line number.

The Mega II controls video timing for the Apple IIGS with a 16-bit counter split into a 7-bit horizontal and a 9-bit vertical part (Figure 1). The counter outputs are made available to programs running on the machine through two addresses in the I/O space, $C02E for the vertical count and $C02F for the horizontal count. These outputs can be used by a program for finer control over display update timing.

Vertical Counter									Horizontal Counter						
V5	V4	V3	V2	V1	V0	VC	VB	VA	HPE	H5	H4	H3	H2	H1	H0
$E0C02E									$E0C02F						

Figure 1 – Mega II Video Counter

You can see that one bit of the nine-bit vertical counter is in location $E0C02F with the seven bits of the horizontal counter. Keep this location in mind when reading the counters.

The seven-bit horizontal counter starts at $00 and counts from $40 to $7F (the sequence is $00, $40, $41,…,$7E, $7F, $00, $40,…). The active video time consists of 40 one µsec clock cycles starting with $58 and ending with $7F. Since this count changes at 980 nanosecond intervals, it will probably be of little use to most programs.

The nine-bit vertical counter ranges from $FA through $1FF (250 through 511) in NTSC mode (vertical line count of 262) and from $C8 through $1FF (200 through 511) in PAL video timing mode (vertical line count of 312). Vertical counter value $100 corresponds to scan line zero in NTSC mode. The vertical count changes at 63.7 µsec intervals, giving a program time to respond to a specific count before it changes. The vertical counter **byte**, at $E0C02E, only changes half as often (at 127 µsec intervals) since the lowest bit of the nine-bit counter is actually stored in the next byte (at $E0C02F).

The nine-bit counter consists of bits VA, VB, VC, V0, V1, V2, V3, V4 and V5. Bits V0 through V5 can be read as a six-bit value. If this value is between 0 and 23, it is the line on the text screen currently being updated. Other values indicate the vertical blanking cycle is occurring. Bits VA through VC can be read as a three-bit value (0-7) indicating which scan line of a text character (characters are composed of eight lines) is currently being drawn.

The vertical counter can also be used to determine which scan line (0-191 for most video modes, including high-resolution and double high-resolution, and 0–199 for super high-resolution) is being updated at any given moment.

Example

Suppose you want to repaint a portion of the super high-resolution screen that will require more time than the vertical blanking period allows. You will have a tear in your animation when the screen's refresh cycle catches up with your drawing.

One solution to this problem would be locating the approximate place the tear occurs and starting your drawing when the system is scanning that line of graphics. Let's say you are painting an area that is about (for example) 100 pixels wide and 200 pixels tall in 320 mode, and that the tear will occur somewhere around scan line 80. To avoid the tear, you would wait until the system is scanning line 80, then you would start redrawing at the top of the screen. This way, you should be finished drawing when the system is back to scanning line 80 again and you will have flicker-free screen updating.

The tricky part is trying to determine just when the system is scanning any given scan line. One way to determine this is to examine the Mega II video counter registers at $E0C02E (vertical) and $E0C02F (horizontal),

described above. By using some simple arithmetic you can come up with the exact scan line being updated. The following piece of code computes the current scan line number (assuming eight-bit native mode):

```
lda    >$E0C02F
asl    A                    ;VA is now in the Carry flag
lda    >$E0C02E
rol    A                    ;roll Carry into bit 0
```

The result (in A) is the low byte of the vertical counter. This value is 0 for the first scan line, 1 for the second scan line, etc. Values $FA to $FF are used twice, since you ignore the high byte of the vertical counter. (The six scan lines immediately above scan line 0 are numbered $0FA to $0FF, and the six above those are $1FA to $1FF.) The example code leaves the highest bit of the vertical counter in the Carry flag, if you really want it.

Note that the VBL interrupts always trigger at scan line 192, even in Super Hi-Res display mode, and that the $C019 soft switch indicates vertical blanking is in effect starting at scan line 192. Be careful polling for a specific scan line number—if interrupts are enabled, it is conceivable that the system will be busy processing an interrupt every time that scan line is being scanned, so your program will hang forever waiting for it.

Setting a scan line interrupt is another way to determine when a particular super high-resolution scan line is being drawn. However, you must be careful in turning scan line interrupts on and off so that you do not interfere with the cursor in QuickDraw II (which uses scan line interrupts).

Further Reference

- *Apple IIGS Toolbox Reference*, Volume 2
- Apple IIGS Technical Note #40, VBL Signal