Apple II
Technical Notes

Developer Technical Support

Apple II Miscellaneous
#17: Buried Treasures of the Video Overlay Card

Written by: Dan Hitchens September 1990

This Technical Note describes some of the more esoteric features of the Video Overlay Card.

What All Does This Thing Do?

The Video Overlay Card contains a duplicate of the Apple IIGS video circuitry to help accomplish its task. This makes some pretty unusual (or esoteric, or downright weird) display behavior possible. Although this Note describes these techniques, the APDA document "Apple II Video Overlay Card Developers Notes" is considered the Bible of the VOC and should be consulted for further details, such as how to make the function calls described in this Note and what the constant values are.

RAM Page Select

Normally, the Mega II video circuitry on the Apple IIGS looks at bank $E1 (the auxiliary bank on Apple IIe) from $2000 to $9FFF for the Super High-Res graphics display buffer (pixel data, scan-line control bytes, and color palettes). This is the default for the VOC, and as such, a monitor connected to the VOC displays the same Super High-Res graphics as a monitor connected to the Apple IIGS main logic board.

A feature the VOC card has beyond the Apple IIGS video circuitry is its ability to select between bank $E1 (the auxiliary bank on the IIe) and bank $E0 (main bank on IIe) for its graphics display buffer. By specifying bank $E0 (or main bank on the IIe) , $E0/2000 to $E0/9FFF becomes the display buffer from which the pixel data, scan-line control bytes, and color palettes are found.

Another added feature of the VOC related to RAM Page Select is its ability to interlace between banks $E1 and $E0 (described later in this Note).

Note: When you select bank $E0 as the graphics display bank, you need to ensure that the display buffer memory is linear (remember, the Super High-Res graphics display buffer needs to be linear).

Technique

```
PASCAL:
        VDGGControl(VDRAMPageSel,VDAux); {selects Aux bank $E1}
        VDGGControl(VDRAMPageSel,VDMain); {selects Main bank $E0}
        VDGGControl(VDMainPageLin,VDEnable); {enable main page $E0 linearization}
        VDGGControl(VDMainPageLin,VDDisable); {disable main page linearization}
```

Uses

- **Dual monitors**. A monitor connected to the main logic board of the IIGS has its display buffer in the normal $E1 bank, while a monitor connected to the VOC has its display buffer in bank $E0.

- **Double-buffered graphics.** This is the ability to draw graphics in one bank while displaying from the other, then switching display banks when appropriate.

Interlacing

As mentioned earlier, another added feature of the VOC is its ability to perform 400 line interlacing. By enabling interlacing on the VOC, the card displays the even and odd lines from different memory areas depending on the setting of the RAM page select. If interlacing is enabled and the auxiliary bank is selected (the default at reset), then the VOC displays both the even and odd scan lines from bank $E1, giving the same display as if interlacing was not enabled at all (the 200 lines from bank $E1 are interlaced with the same 200 lines from bank $E1). If interlacing is enabled and the main bank is selected, then the VOC displays both the even and odd scan lines from bank $E0 (same sort of thing, the 200 lines from bank $E0 are interlaced with the same 200 lines from bank $E0.)

Now for the useful part,—if interlacing is enabled and RAM Page Select is set to interlace, then interlacing occurs between banks $E1 and $E0 (the odd 200 lines come from bank $E1 and the even 200 lines come from bank $E0).

Remember in Super High-Res interlace mode, not only the pixel data but also the scan line control bytes and color palettes can be different between the two banks and need to be loaded. Also be sure to turn main page linearization on before loading data into the $E0 bank (see discussion on RAM Page Select).

Technique

```
PASCAL:
        VDGGControl(VDRAMPageSel,VDInterlace); {selects interlacing between banks
                                                $E1 and $E0}
        VDGGControl(VDInterlaceMode,VDEnable); {enables 400 line interlace mode}
        VDGGControl(VDInterlaceMOde,VDDisable); {disables 400 lines interlace mode}
```

Uses

- **400-line display**. A monitor connected to the VOC can display 400 different lines when interlace is enabled and RAM Page Select is set to interlace.

Warning: The technique of interlacing can cause noticeable flicker when pixels on adjacent horizontal lines are different.

Dual Monitors

Because the VOC has a duplicate of the video circuitry of the Apple IIGS, it has the capability to have two monitors displaying the same or totally different information at the same time. The VOC doesn't have to be in the same display mode as the main logic board. It can be in the same video mode or in a totally different one (i.e., the main logic board is in text mode while the VOC is in Super High-Res mode.)

The technique required for putting the VOC into a video mode different from that of the main logic board's circuitry is to enable and disable, at the appropriate time, the bus to the VOC. To go into the video mode in which you want the VOC to remain, you must first make sure the bus is enabled (power on default). Once you have the VOC in the mode you want, disable the bus with `VDGGControl(VDGGBus, VDEnable)`, then go into the video mode in which you want the main logic board. After you have the main logic board into the mode you want, reenable the bus with `VDGGControl(VDGGBus, VDDisable)`. Remember to reenable the bus or the VOC screen remains frozen—memory writes to the video display buffers are not written through to the VOC's internal memory display buffers.

Technique

```
PASCAL:
        VDGGControl(VDGGBus,VDEnable);  {Enables the GGBus_Disable circuitry which
                                         inhibits writes to the VOC display memory.}
        VDGGControl(VDGGBus,VDDisable); {Disables the GGBus_Disable circuitry which
                                         enables writes to the VOC display memory
                                         (power on default)}
```

Uses

- **Two monitors with different display modes**. A monitor connected to the VOC can display one mode while a monitor connected to the main logic board is displaying another. There are many different applications that could benefit from this type of monitor arrangement. For example, you might need one monitor to display control information, while another is displaying graphics (a debugger could trace code in text mode on one monitor, while the other monitor is displaying the actual graphics in real time.)

Apple IIe

One of the outstanding features of the VOC is its ability to provide all the video modes found on the IIGS to the Apple IIe community, including Super High-Res graphics (320x200 or 640x200 bit resolution). Not only can Apple IIe owners experience the world of Super High-Res graphics

with the VOC, they also can perform all the previously mentioned functions (such as RAM page select, interlacing, and dual monitors).

Super High-Res graphics mode on the IIe is enabled the same way as on the IIGS. The New Video register, which is located at $C029, is the register which controls the mode. The following is a short description of the bits for this register as found in the *Apple IIGS Hardware Reference*:

Bit 7:	0	Enables old Apple IIe video modes
	1	Enables Super High-Res video mode
Bit 6:	0	Memory linearization is disabled (old Apple IIe memory map configuration)
	1	Memory linearization is enabled (memory from $2000 to $9D00 becomes one linear address space.)
Bit 5:	0	Double Hi-Res graphics are displayed in color
	1	Double Hi-Res graphics are displayed in black and white
Bits 4-0:		Undefined

From this description, all that is needed to switch into Super High-Res display mode, is to set bits six and seven at memory location $C029. When you want to go back to the old Apple IIe video modes, just clear bits six and seven.

Note: The Super High-Res graphics display buffer on the Apple IIe is located in Auxiliary memory from $2000 to $9FFF (unless its been changed to Main page with RAM page select, as described earlier in this Note). Within this display buffer, you must set up the following three types of data:

$2000-$9CFF	Pixel Data
$9D00-$9DFF	Scan-line control bytes
$9E00-$9FFF	Color palettes

For more details, see the *Apple IIGS Hardware Reference*.

Implementation Technique

The thing to remember is to make sure memory linearization is on ($C029 bit 6=1) before you fill the Super High-Res buffer ($2000-$9FFF). If you write a program that moves data into the display buffer with memory linearization off ($C029 bit 6=0), then enable Super High-Res by turning bits six and seven on, you do not get the results you anticipate. What you see is a scrambled mess on the screen, because you filled the display buffer with memory linearization off (the VOC filled its display buffer in a nonlinear manner), and when you turned Super High-Res mode on, the VOC expected to see memory in a linear manner.

Apple IIe Super High-Res Demo Program

```
*************************************************************
*
*   A simple Apple IIe super high res. screen demo load program for
```

```
*   use with the "Apple II Video Overlay Card"
*
*    Copyright Apple Computer, Inc. 1989
*    All rights reserved.
*
*    Programmed by: Dan Hitchens May 1989
*
*****************************************************************
*
*    This program loads super high res. screen dumped files which
*   were named "SCREEN.x" (where x starts at zero and increments
*   up sequentially.)  When this program doesn't find the next
*   sequential file, it beeps and starts looking back at
*   "SCREEN.0" again (this program loops endlessly.)
*
        longi off
        longa off
        MACHINE M65c02
        ENTRY begin:CODE

*****************************************************************

begin       PROC
BEll1       EQU $FBDD             ;beep subroutine
MLI         EQU $BF00             ;ProDos-8 machine language interface
AuxMove     EQU $C311             ;Aux. move firmware routine
OpenCMD     EQU $C8               ;Open command
ReadCMD     EQU $CA               ;Read command
CloseCMD    EQU $CC               ;Close command

            lda #$40
            sta $C029             ;turn on memory linearization

NextFile    nop
;First init. a few counters

            lda #$20
            sta AuxAddr           ;init aux. address counter

;Now Open the file
            jsr MLI               ;perform an open command
            dc.b OpenCmd          ;passed command (open)
            dc.w OpenBlk          ;passed block address
            beq OKOpen2           ;branch if able to open file

;We arrive here if unable to open the file (probably wasn't there)
InitZero    jsr Bell1             ;sound the bell
            lda #'0'              ;re-init to "SCREEN.0"
            sta FileNum-1         ;init. SCREEN.x to .0
            jmp NextFile          ;go try opening "SCREEN.0"

OKOpen2     lda ORefNum           ;get ref. num from open command
            sta RRefNum           ;store for read command
            sta CRefNum           ;store for close command

OKOpen      nop
;Now read $4000 bytes into $3000
            jsr MLI               ;call machine language interface
            dc.b ReadCMD          ;passed command (read)
            dc.w ReadBlk          ;passed block address
            sta MyError           ;save returned error (if any)
            bcs AllDone           ;branch if error occurred

;Set starting address to $3000
            lda #0
            sta $3c
```

```
             lda #$30
             sta $3d
;Set ending address to $6fff
             lda #$ff
             sta $3e
             lda #$6f
             sta $3f
;Set destination starting addr to AuxAddr (its incremented)
             lda #0
             sta $42
             lda AuxAddr
             sta $43

             sec                    ;to indicate move from main to aux mem.
             jsr AuxMove            ;now move the data to aux. memory

;Now increment AuxAddr:=AuxAddr+$4000
             lda AuxAddr
             clc
             adc #$40
             sta AuxAddr
             lda MyError            ;get error (if any)
             beq OKOpen             ;continue reading until error (eof)

;Now we can turn on the super high res screen
AllDone      nop
             lda #$c0
             sta $c029
```

```
;Now delay for awhile (just a big delay loop)
             lda #0
             tax
             tay
             lda #12
xloop        dex
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop

             bne xloop
yloop        dey
             bne xloop
             dec a
             bne xloop

;Now Close the file
             jsr MLI                ;do a Close command
             dc.b CloseCmd          ;passed command (close)
             dc.w CloseBlk          ;passed block address

;Now increment to next file
             lda FileNum-1          ;get current file number
             cmp #'9'               ;is it nine
             beq InitZero           ;branch if already at nine
             clc
             adc #1                 ;else add one to next "SCREEN.x"
             sta FileNum-1
             jmp NextFile           ;go try and read next file

;---------------------------------------------------------------
        STRING PASCAL               ;we want pascal strings
OpenBlk      dc.b 3
             dc.W Pathname
             dc.W $2700
ORefNum      ds.b 1                 ;returned ref_num
Pathname     dc.b 'SCREEN.0'
FileNum      ds.b 1
ReadBlk      dc.b 4                 ;no. of parameters
RRefNum      ds.b 1                 ;ref. number (stuffed from Open command)
             dc.w $3000             ;data buffer pointer
             dc.w $4000             ;request count
             ds.w 1                 ;actual data transferred count

CloseBlk     dc.b 1                 ;no. of parameters
CRefNum      ds.b 1                 ;ref. number (Stuffed from Open command)

AuxAddr      ds.b 1                 ;aux. address pointer

MyError      ds.b 1
;---------------------------------------------------------------
            ENDP
        END
```

Further Reference

- *Apple IIGS Hardware Reference*
- *Apple II Video Overlay Card Developers Notes* (APDA)