SOFTWARE APPLICATION

READING VIDEO DATA FROM A PROGRAM

In the October, 1982 issue of *SOFTALK* magazine, Bob Bishop reported an exciting discovery. Any time a 6502 program reads an address from which there is no data response, the video data from the previous scanner access is read by the MPU. Mr. Bishop made a big mistake publishing this information. He could have made a living making bets with Apple experts that you can sync an unmodified Apple to the video scan under program control. He'd have made about a thousand dollars per sucker. Who would have thought that the data on the floating data bus would remain valid for over half a microsecond. The article "Have an Apple Split" contains programming examples and is highly recommended reading. It is partly because of "Have an Apple Split" that this book has endeavored to present extensive memory scanning maps for programmer reference.

One reason for syncing a program to the video scan is to create displays that are a mixture of the normal Apple screen modes. For example, if you switch to HIRES before line 5 of every vertical scan and switch to LORES before line 10 of every vertical scan, you will have a stable combination of HIRES and LORES graphics displayed on the screen.

The method of reading video sync from a program is to set up flags in scanned memory at the point of the television scan where the program needs to take an action, such as switching from LORES to HIRES. Then the program polls a nonresponding address such as the cassette output port until it detects the flag. The choice of flags and their location in memory will be dictated by the application.

Programming a combination display requires the normal programmer's imagination to conceive of the display. Additionally, a thorough grasp of memory scanning details is a necessity. The purpose of this application note is to provide some discussion of techniques that might be used.

For starters, here is a list of addresses from which video data can be read:

$C01X	KEYBOARD STROBE FF Reset
$C02X	Cassette Output
$C03X	Speaker Output
$C04X	C040 STROBE'
$C05X	Screen Switches and Annunciator Outputs
$C06X	Serial Inputs (D0-D6 only)
$C07X	Timer Trigger
$C080-$C7FF	I/O Control (use if slot is empty)
$CFFF	Expansion ROM Disable

Of all these choices, the screen switches stand out as having no chance of interfering with the operation of some device. The screen switches will normally be used, because the task of polling and switching can then be combined. Also, the polling loop is self documenting to investigators of the program. In the rare instance that one of the screen switches will not do for video polling, the annunciators, the keyboard strobe reset, the $C040 strobe, and the cassette output addresses are likely choices.

Figure 5.17 is a diagram designed to aid the programmer in selecting locations for syncing flags. It should be used in conjunction with the memory maps from earlier sections of this chapter. From this diagram, one can quickly see the prospects for successful syncing at a given point in a given mode. The most obvious feature discerned from Figure 5.17 is that HBL scanned memory in HIRES overlaps HBL' scanned memory, but the HBL and HBL' scanned memory areas of TEXT/LORES are distinct. This means that different problems will arise when choosing flags for these two different memory scanning modes.

A problem with HIRES is that most memory gets scanned more than once every vertical scan. Only the first 16 bytes of the SECOND 40 and the first 16 bytes of the THIRD 40 are scanned just once per vertical frame. This makes it more difficult to uniquely flag a scan position. In TEXT/LORES, memory is more uniquely scanned with only a partial overlap between the area scanned during VBL and during the top of the screen. However, one must deal with the fact that critical non-displayed memory areas are scanned. HBL scanned memory is in an area normally taken up by Applesoft programs or Integer BASIC variables, and the undisplayed eight bytes at the end of each 128 byte segment are used by the DOS for critical disk information (like active slot number).

The following is a typical video polling loop:

```
POLLIT  CMP $C050   ;FLAG VALUE
                    ;IN ACCUMU-
                    ;LATOR.
        BNE POLLIT
```

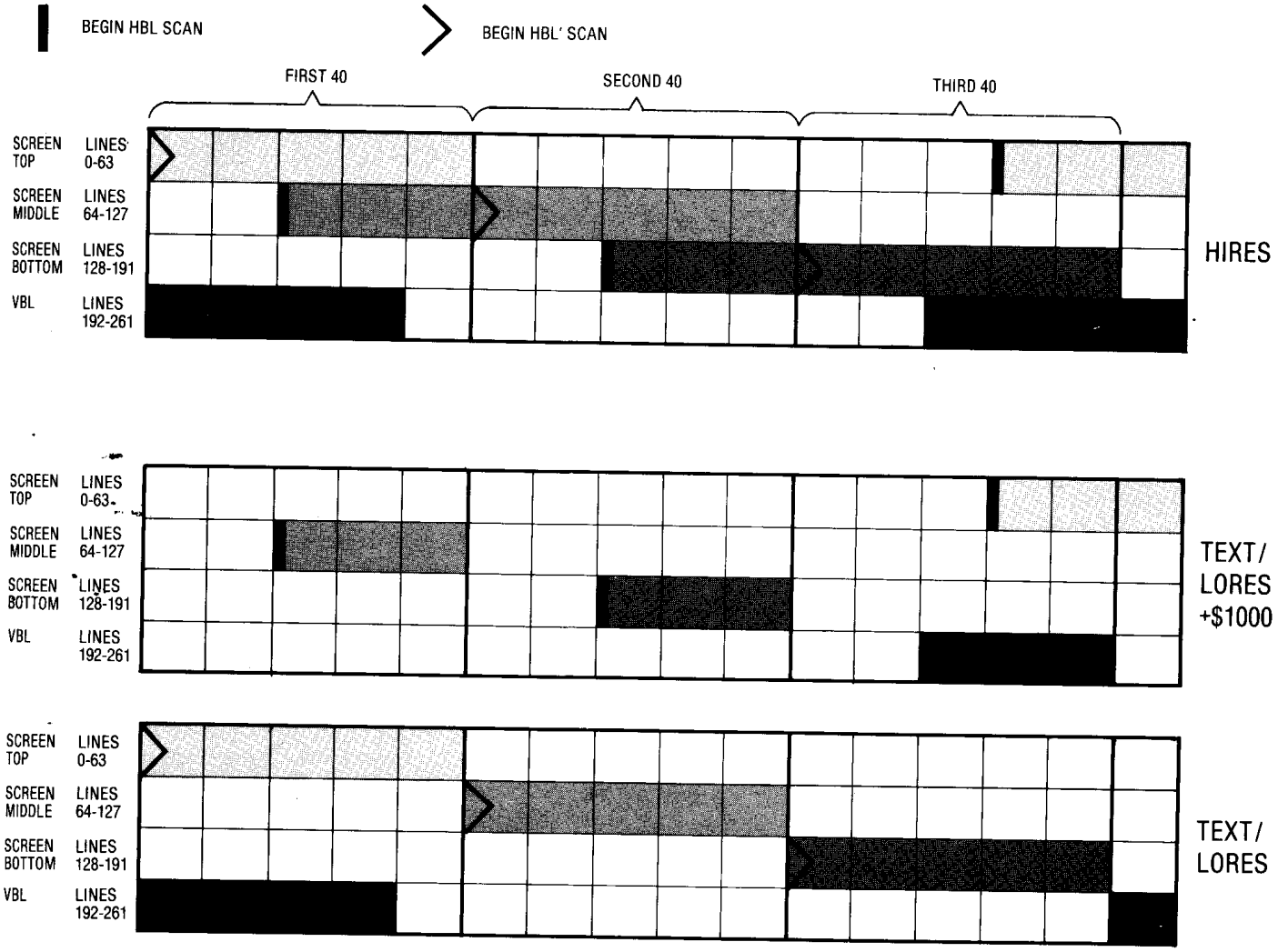

Figure 5.17 Screen Memory Scanning.

The loop takes seven clockpulses to execute and thus establishes one criteria for a screen flag. It must occupy a minimum of seven adjacent scanned bytes of memory or six bytes if one of the bytes is the first byte of a HBL scan. The first byte of a HBL scan is scanned twice so six bytes here is the same as seven elsewhere.

Here is one way to flag PAGE 1 of TEXT memory at the beginning of line 12 (the thirteenth line from the top if the first line is line 0). Store the value $5B at locations $1610-$1627. Figure 5.6 shows this is HBL scanned memory before line 12, and Figure 5.17 shows that this memory will only be scanned once per vertical scan. $5B is the ASCII value of a flashing left bracket which is very unlikely to be found in text memory. It may be found in the UNUSED 8 or in other HBL scanned memory, which creates a problem. One way to solve it is to blank the other HBL scanned memory. Then poll for the flag as follows:

```
POLLIT    CMP  $C051   ;ACCUMULATOR
                       ;IS $5B
          BNE  POLLIT
          NOP
          CMP  $C051
          BNE  POLLIT
```

This loop will not be exited without finding two $5Bs separated by eight bytes. This excludes accidentally syncing on anything in the UNUSED 8 and solves the problem without messing up the memory locations used in disk I/O. Any Applesoft program beginning at $800 and extending beyond $140F is clobbered.

It is not necessary to uniquely flag a scan position. It is only necessary that there is no interference in detecting a scan flag between the present scan position and the flagged scan position. For example, assume we switched from HIRES to LORES at the middle of the display screen and we wish to detect the beginning of VBL. A flag at $1460 through $1477 will serve this purpose nicely. Even though parts of this memory are also scanned at the top of the screen, the next time they are scanned is during HBL, before VBL. There is no interference between the present location and the detection point.

Mr. Bishop showed an interesting flagging technique in his *SOFTALK* article. He flags the middle of the TEXT/LORES display area with a string of $E0s. He uses this to switch from TEXT to LORES. These $E0s are printed as spaces on the text screen and as a row of black blocks and a row of aqua blocks in LORES. By detecting this row of $E0s from TEXT mode then switching to LORES mode for the lower part of the screen, the program effortlessly transits from an empty text line to a LORES display with an upper aqua border. The program is short and simple and illustrates that programming mixed screen displays is a bit of an art form.

The further possible flagging techniques have not all been dreamed up yet. Here are some related ideas and facts:

1. Switching modes during HBL or VBL eliminates the unsightly display of switch points. This can be accomplished by flagging VBL or HBL or by flagging a display area and waiting for HBL or VBL with timed execution loops.
2. The UNUSED 8 is the only undisplayed area of HIRES available for flagging. It is scanned during HBL before the top third of the display and just after HBL during VBL. The undisplayed eight is of minimal use in flagging PAGE 1 of TEXT/LORES because of interference with the DOS.
3. Bit 7 of HIRES may be used as a flag and checked with the BIT instruction. This is one way of flagging the displayed HIRES area if bit 7 isn't critical for color or positioning.
4. Bit 7 is a good TEXT flag if there is no inverse video on the screen. Other likely TEXT flags are flashing ASCII, control ASCII, lowercase ASCII, and ASCII of characters not supported by the Apple keyboard.
5. When considering LORES flags in displayed areas, bits 0-3 control the upper block, and bits 4-7 control the lower block.
6. The video polling method works very well in conjunction with timed execution loops. There are 65 cycles in a horizontal scan—25 cycles of HBL and 40 cycles of HBL'. There are 262 horizontal scans in a vertical scan—64 in each third of the display screen and 70 during VBL. When a group of flagged bytes is located, it is then possible to find a precise byte in the group by slewing backwards in 17029 cycle loops until the first flagged byte is found. 17031 cycle loops can be used to slew forwards.
7. The first byte scanned during HBL is scanned twice in a row. In TEXT/LORES, every memory line is scanned eight times in a row, except the last line of VBL is scanned 14 times. In HIRES, every memory line is scanned once, but lines 250-255 are rescanned after line 255 (250 is same as 256, 251 is same as 257, etc.).
8. Switching rapidly between GRAPHICS and TEXT mode will cause many televisions to lose color sync. This is a factor of alignment and response of the 3.58 MHz oscillator inside the television. Because of this unpredictability from TV to TV, it is not possible to say what percentage of the time a program can leave the Apple in TEXT mode and still hope to maintain color sync. Commercial programmers could be well advised to keep at least a 50% GRAPHICS mode to TEXT mode ratio in their products if color stability is important.
9. A very heavily loaded data bus may not reliably store video data long enough to be read by the MPU. Apparently, some commercially available peripheral cards cause this condition when plugged in. Beware!

Figures 5.18 and 5.19 are example programs that will create a mixed display in which all the text on the screen except the bottom line is underlined. NORMAL, INVERSE, and FLASHING text are all underlined. The program works by setting up a LORES map of PAGE 1 text in PAGE 2. Then the screen mode is switched to LORES PAGE 2 for the top of every text line except line 0, then back to TEXT PAGE 1 for the next seven lines.

Figure 5.18 is an Integer BASIC program which sets up the PAGE 2 map and flags HBL scanned memory for both PAGE 1 and PAGE 2. The process is pretty slow in BASIC, but this program is included for illustration and BASIC is easy to follow. Just give the program about thirty seconds to work. Integer BASIC is used because Integer is very easy to work around when modifying low memory just above $800. The actual screen splitting is done by the assembly language program in Figure 5.19.

```
  1 REM
  2 REM
  3 REM  CALL UNDERLINE
  4 REM
  5 REM
  6 REM  FIRST MOVE INTEGER LOW POINTERS UP TO $1C40.
  7 REM
  8 REM
 10 POKE 74,64: POKE 75,28: POKE 204,64: POKE 205,28
 20 PRINT "BLOAD UNDERLINE.OBJ0"
 22 POKE -16304,0: POKE -16299,0: POKE -16298,0: REM LOOK AT LORES PAGE 2.
 25 REM
 26 REM
 27 REM  NOW FILL PAGE 2 WITH THE UNDERLINES FOR THE CURRENT TEXT DISPLAY.
 28 REM
 29 REM
 30 FOR P2=2048 TO 2176: POKE P2,0: NEXT P2: REM BLANK TOP LINE OF PAGE 2
 40 FOR P1=1024 TO 1919:P2=P1+1152: GOSUB 160
 45 NEXT P1
 50 FOR P1=1920 TO 1999:P2=P1+168: GOSUB 160
 55 NEXT P1
 60 REM
 61 REM
 62 REM  NOW FLAG HBL SCANNED MEMORY WITH "$5B"S.
 63 REM
 65 REM
 70 FOR A=0 TO 1920 STEP 128: FOR B=0 TO 23
 75 FLAG=0: IF B<6 THEN FLAG=91: REM  91 IS FLASHING LEFT BRACKEET
 80 POKE 5136+A+B,FLAG: REM HBL BEFORE MIDDLE SCREEN = FLAG
 90 POKE 5176+A+B,FLAG: REM HBL BEFORE BOTTOM SCREEN = FLAG
100 NEXT B
110 FOR B=0 TO 31
115 FLAG=0: IF B<14 THEN FLAG=91: REM   91 IS FLASHING LEFT BRACKET
120 POKE 5216+A+B,FLAG: REM HBL BEFORE VBL AND TOP SCREEN = FLAG
130 NEXT B: NEXT A
140 CALL 7168: REM  GO SPLIT THE SCREEN
150 END
155 REM
156 REM COLOR = 0 OR 1 OR 15
157 REM
160 T= PEEK (P1):COLR=1: REM  UNDERLINE COLOR = 1
165 IF T=32 THEN COLR=15: REM  INVERSE SPACE?
170 IF T=96 OR T=160 OR T=224 THEN COLR=0: REM  SPACE?
180 POKE P2,COLR: RETURN
```

Figure 5.18 Integer BASIC Listing: Underline Program.

The difficult part about a program of this nature is designing a flagging method. UNDERLINE uses strings of $5Bs as flags stored in the HBL areas. $5B is the code for a flashing left bracket, which can safely be said to rarely be printed on the TEXT screen. A string of 14 $5Bs is stored beginning at HBL before VBL. Note from Figure 5.17 that this also stores six $5Bs in HBL before screen top. Additionally, six $5Bs are stored in HBL before screen middle and HBL before screen bottom. All other HBL scanned bytes are cleared. Since the first byte of HBL scanned memory is driven out twice, this flagging method results in seven $5Bs being driven out every HBL before display and fifteen $5Bs being driven out at HBL before VBL.

Here's the scheme for switching. First, wait for VBL. HBL before VBL is the only area which will respond to two consecutive polls with $5B. Once the

```
SOURCE FILE: UNDERLINE
0000:              1 ************************************************
0000:              2 *                                              *
0000:              3 *                                              *
0000:              4 *                  UNDERLINE TEXT              *
0000:              5 *                                              *
0000:              6 *                  BY JIM SATHER               *
0000:              7 *                                              *
0000:              8 *                      1/4/83                  *
0000:              9 *                                              *
0000:             10 *                                              *
0000:             11 ************************************************
0000:             12 *
0000:             13 *
C050:             14 GRAPHIX EQU  $C050
C051:             15 TEXT    EQU  $C051
C052:             16 NOMIX   EQU  $C052
C054:             17 PAGE1   EQU  $C054
C055:             18 PAGE2   EQU  $C055
FCA8:             19 WAIT    EQU  $FCA8
0000:             20 *
0000:             21 *
0000:             22 *
----- NEXT OBJECT FILE NAME IS UNDERLINE.OBJ0
1C00:             23          ORG  $1C00
1C00:8D 52 C0     24 DOSCRN   STA  NOMIX
1C03:8D 54 C0     25          STA  PAGE1
1C06:A9 5B        26          LDA  #$5B
1C08:CD 51 C0     27 SCRNLP   CMP  TEXT      LOOK FOR STRING OF 15 "$5B"S
1C0B:D0 FB        28          BNE  SCRNLP
1C0D:EA           29          NOP
1C0E:CD 51 C0     30          CMP  TEXT
1C11:D0 F5        31          BNE  SCRNLP
1C13:A9 28        32          LDA  #$28      GOT VBL
1C15:20 A8 FC     33          JSR  WAIT      WAIT 4553 CYCLES
1C18:A9 5B        34          LDA  #$5B
1C1A:A0 17        35          LDY  #23       DO 23 LINES
1C1C:A2 08        36          LDX  #8
1C1E:D0 02        37          BNE  TEXTLP
1C20:             38 *
1C20:             39 *
1C20:A2 07        40 DOLINE   LDX  #7        7 TEXT LINES FOR 1 LINE OF GRAPHICS
1C22:CD 51 C0     41 TEXTLP   CMP  TEXT
1C25:D0 FB        42          BNE  TEXTLP
1C27:CA           43          DEX            GOT HBL
1C28:D0 F8        44          BNE  TEXTLP
1C2A:8D 55 C0     45          STA  PAGE2     SWITCH TO LORES
1C2D:CD 50 C0     46 LORESLP  CMP  GRAPHIX
1C30:D0 FB        47          BNE  LORESLP
1C32:8D 54 C0     48          STA  PAGE1     GOT HBL
1C35:8D 51 C0     49          STA  TEXT
1C38:88           50          DEY
1C39:D0 E5        51          BNE  DOLINE
1C3B:F0 CB        52          BEQ  SCRNLP

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Figure 5.19 Assembler Listing: Underline Program.

first line of VBL is located, wait 4553 cycles. This is a ball park figure which waits until after HBL of the first displayed line. Now that the scan is past VBL, Figure 5.17 shows that there is no further possibility of reading $5B in polling loops before the next VBL, except that $5B will be read once within the first seven bytes of every HBL. This creates a flagged situation in which HBL can be detected before any displayed line.

The given task is to underline the current PAGE 1 TEXT display. The ground work for doing this is laid in the BASIC program. First understand that the top line of every text character pattern is blank. This creates the space between the text lines. To underline the character at position 1 of line 0, you map a LORES block into the top of position 1 of line 1 in PAGE 2. Then for the first scan of text, line 1 you switch to GRAPHICS PAGE 2. During HBL of the next scan, you switch back to TEXT PAGE 1. This creates an underline appearance for line 0. The BASIC program simply checks the PAGE 1 display memory for space or no space, then maps corresponding LORES blocks into the adjacent line of PAGE 2 memory. The bottom line of text is not underlined because there is no adjacent lower line to switch to LORES.

With screen memory fully mapped, the task of the screen splitting program is reduced to finding the top of the screen precisely, then switching to LORES PAGE 2 for one line every eighth line. It's easy once the flags are properly set.

Since the Apple is in TEXT mode most of the time during UNDERLINE, probably any television will lose color sync and the underline will be white rather than colored. Readers may wish to experiment with colors by changing lines 160 through 180 of the BASIC program. Just don't POKE 91 into LORES displayed memory. 91 ($5B) is our flag, and it is reserved only for undisplayed memory.