Apple II Computer Information

Apple Soft BASIC Info:
Notes on Hi-Res Graphics Routines in AppleSoft
Mesztenyi - Apple Orchard - Spring 1981

Document # 47

Ex Libris David T. Craig
NOTES ON HI-RES GRAPHICS Routines in Applesoft

by C. K. Mesztenyi
Washington Apple Pi

Checking out the entry points given by J. Crossley in the article "APPLESOFT INTERNAL ENTR YPOINTS" in the March/April 1980 Apple Orchard, I found the given entry points were 4 bytes off from the given ones in our AP PLE II Plus. Furthermore, after checking out the routines in more detail, I thought to share my experiments with other APPLE II Plus owners interested in machine language programming. In the first section, I describe the essential data storage area, in the second I give the entry points of the subroutines somewhat more detailed than in the above article, and in the last section I give some listings of instructions following the entry points so that one could identify it for different versions of Applesoft.

I have called the above set of data as external cursor data since the actual point plot is performed by the following five instructions:

```
LDA $1C
EOR ($26),Y
AND $30
EOR ($26),Y
STA ($26),Y
```

which uses data located at $1C, $26, $27, register Y and $30. The contents of register Y are always picked up from location $E5 prior to the above instructions, thus we may call the data in the following five locations as internal cursor data:

```
$E5: The integer part of the horizontal screen coordinate divided by 7.
$E6: Page indicator ($20 for Page 1, $40 for Page 2)
```

These two cursor data (external and internal) are equivalent in the sense that given one, the other can be derived from it. There would be no need to make any distinction if they would correspond to each other all the time, but unfortunately, this is not always the case, e.g. the following sequence of BASIC instructions:

```
HCOLOR = 1
HPL T 0,0 TO 10,10
HCOLOR = 2
HPL T 0 TO 10,50
```

plots two lines, (0,0) to (10,10) and (10,10) to (10,50), both with color 1, i.e. HCOLOR = 2 has no effect. Actually it resets the color code in $E4 but it does not change $1C, and the statement HPL T TO picks up whatever was left in $1C.

A machine language programmer can write his/her own graphics routines which takes time and uses sometimes much-needed memory space. Thus using the available programs in Applesoft ROM can be advantageous. If execution time is also important, as in the case of animation, then one should concentrate only on the internal cursor data, and modify the external cursor only when it is necessary. The entry points INTX and INTY, provide the basic routines for incremental plotting which are not available directly in BASIC. Also, modifying the external cursor coordinates allows the use of HLINE with off-set.
2. ENTRY POINTS IN APPOLESOFT

Page and Color:

HGR2 ($F3D8): Displays page 2
with all graphics mode,
sets $6E to $40,
clears
page 2 (black) and
sets
$1C to zero (black I).

HGR ($F3E2): Displays page 1
in mixed mode,
sets $6E to
$20,
clears
page 1 (black) and
sets $1C to zero
(black I).

BKGND ($F3F4): Clears the page
defined by $E6 to the
color defined by the con-
tents of register A, which
should be one from the
Color Masking Table. Also
stores register A in $1C.

HCOLOR ($F6F0): Assumes regis-
ter X contains the color
index (0 to 7). The rou-
tine picks up the ap-
propriate color code from
the Color Masking Table
and stores it in $E4.

Positioning Entries:

HPOSN ($F411): Assumes the in-
put upon entry in the reg-
isters as:
register X = low order bits of
the horizontal screen
cordinate,
register Y = high order bit of
the horizontal screen
cordinate,
register A = vertical screen
cordinate.
The routine stores the reg-
isters in $E0, $E1 and
$E2. Then, using $E6, sets
$26, $27, $30 and $E5
with register Y, and
sets $1C to the con-
tents of $E4. Thus this
routine makes the internal
cursor equivalent to the
external one.

INTX ($F465): Modifies the internal
cursor data in $1C, $E5,
register Y and $30 so
that it corresponds to
incrementing/decre-
menting the horizontal

APPLE ORCHARD

screen coordinate X by
one. Upon entry, if the
N-flag is zero (positive)
then it increments; if N
is set (negative) then it
decrements. The modifi-
cation has a wrap
around feature, i.e., in-
crementing/decrement-
ing at the extreme sides
of the screen defined
by the internal cursor
causes it to come back
on the other side. The
routine assumes that reg-
ister Y corresponds to
$E3 upon entry, and
leaves the routine cor-
rectly modified if
necessary.

Upon testing the N-flag
the routine jumps to
DECRX or INCRX.

DECRY ($F4D5): The routine modi-
ifies the internal cursor
data by decrementing the
vertical screen coordinate
by 1 (see INTY).

INCRX ($F504): The routine modi-
ifies the internal cursor
data by incrementing the
vertical coordinate by 1
(see INTY).

IPOSN ($F5CB): Sets the external
cursor data in $E0, $E1,
$E2 equivalent to the
internal cursor coordinate.

Plotting Entries:

HPLOT ($F457): Assumes input data
in the registers as
HPOSN:

register X: low order bits
of horizontal screen
cordinate,
register Y: high order bit
of horizontal screen
cordinate,
register A: vertical screen
cordinate.
The routine calls HPOSN
with the above data, then
goes to PLOT.

PLOT ($F45A): The routine exe-
cutes the five instructions
listed in the beginning of
the article which plots a
point using the internal
cursor data. If this entry
is used directly, then the
user should make sure
that register Y contains
the data from $E5.

HLINE ($F53A): The routine as-
sumes input in the
registers:

register A: low order bits
of horizontal screen
cordinate,
register X: high order bit
of horizontal screen
cordinate,
register Y: vertical screen
cordinate.

(Note that it is in different
order than (HPOSN).)
The routine draws a line from the internal cursor position to the point defined by the input. Upon exit, it leaves the external cursor data corresponding to the input, the internal cursor data corresponding to the last plot point of the line. If the internal and external cursor data were not equivalent, then an off-set occurs. This can be visualized as follows:

Draw a line segment from the external cursor coordinates to the input coordinates. Now move this line segment parallel to itself so that the endpoint at the external cursor position gets into the internal cursor position. This is the actual line segment which will be drawn. If it gets outside of the screen, then a wrap-around occurs, i.e. it comes back on the opposite side of the screen.

**APPENDIX**

The first few instructions are listed for each entry point so that one could identify them using the Monitor list feature.

### Bit Position Table:

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F5B2</td>
<td>$81 = 10000001</td>
<td>$81</td>
</tr>
<tr>
<td>$F5B3</td>
<td>$82 = 10000010</td>
<td>$82</td>
</tr>
<tr>
<td>$F5B4</td>
<td>$84 = 1000100</td>
<td>$84</td>
</tr>
<tr>
<td>$F5B5</td>
<td>$88 = 10001000</td>
<td>$88</td>
</tr>
<tr>
<td>$F5B6</td>
<td>$90 = 10010000</td>
<td>$90</td>
</tr>
<tr>
<td>$F5B7</td>
<td>$A0 = 10100000</td>
<td>$A0</td>
</tr>
<tr>
<td>$F5B8</td>
<td>$C0 = 11000000</td>
<td>$C0</td>
</tr>
</tbody>
</table>

### Color Masking Table:

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F6F6</td>
<td>$00 = 00000000 (black I)</td>
<td>$00</td>
</tr>
<tr>
<td>$F6F7</td>
<td>$2A = 00101010</td>
<td>$2A</td>
</tr>
<tr>
<td>$F6F8</td>
<td>$55 = 01010101</td>
<td>$55</td>
</tr>
<tr>
<td>$F6F9</td>
<td>$7F = 01111111 (white I)</td>
<td>$7F</td>
</tr>
<tr>
<td>$F6FA</td>
<td>$80 = 10000000 (black II)</td>
<td>$80</td>
</tr>
<tr>
<td>$F6FB</td>
<td>$AA = 10101001</td>
<td>$AA</td>
</tr>
<tr>
<td>$F6FC</td>
<td>$D5 = 11010101</td>
<td>$D5</td>
</tr>
<tr>
<td>$F6FD</td>
<td>$FF = 11111111 (white II)</td>
<td>$FF</td>
</tr>
</tbody>
</table>

### Source: David T Craig