ADVANCED SYSTEM FEATURES

Examples and discussions of system features

for beginning machine-language programmers

1) Switching Screens
2) RAM Shadows
3) Display List Interrupts
4) Vertical Blank Interrupts
5) Display List Modification
6) Mixed Mode Screen
7) Using Central I/O
8) Direct Screen Write

ATARI, INC.
CONSUMER PRODUCT SERVICE
PRODUCT SUPPORT GROUP
1312 Crossman Ave.
Sunnyvale CA 94086

(800) 672-1404 inside CA
(800) 538-8543 outside CA

DEMOPAC #6
Rev.2 6-82/JB

DISCLAIMER OF WARRANTY ON PROGRAMS CONTAINED HEREIN

All computer programs contained herein are distributed on an "as is" basis by Atari, Inc. ("Atari") without warranty of any kind. Any statements concerning the capabilities or utility of the computer programs are not to be construed as express or implied warranties.

Atari shall have no liability or responsibility to the user or any other person or entity with respect to any claim, loss, liability, or damage caused or alleged to be caused directly or indirectly by the computer programs, contained herein. The entire risk as to the quality and performance of such programs is with the user.

Every effort has been made to ensure the accuracy of this document. However, because of ongoing improvements and updating of our computer software and hardware, Atari cannot guarantee the accuracy of printed material after the date of publication and disclaims any liability for changes, errors, or omissions.

Correspondence regarding this pack should be forwarded to Manager of Technical Support, Consumer Product Service, Atari, Incorporated, 1312 Crossman Avenue, Sunnyvale, CA 94086.

SWITCHING SCREENS
Display List Alternation
JB 2/82

It is often necessary or desirable to keep two separate screens of data in memory, and switch back and forth between them. You may wish, for example, to display one screen while updating the other, or you may simply wish to use the switching for an animation effect. The technique is sometimes called page-flipping or paging.

In order to switch screens on the Atari, you must create two separate display lists, each with its own data area. The switch is accomplished by simply changing the pointer to the display list. This two-byte pointer is located at decimal 560 and 561.

BASIC sets up a display list, complete with data area, every time you enter a new graphics mode. The location is based on the top of memory pointer, RAMTOP, located at decimal 106. In the example on the following page, an alternate display list is created by moving RAMTOP down 32 pages in memory, and then calling a new graphics mode. Since RAMTOP is a high-byte pointer (that is, it is always on a page boundary), the low byte of the new display list remains the same, while the high byte moves down 32 pages. In order to switch between the two, it is only necessary to change the high byte, at location 561.

If you wish to update one screen while displaying the other, you must keep track of the location of the screen data areas. The display list has its own pointer, called the LMS address, which is kept at the 5th and 6th bytes. (If the starting address of the display list, DL=PEEK(560)+PEEK(561)*256, then the LMS address is at DL+4 and DL+5.) BASIC, however, uses another pointer, called SAVMSC, located at decimal 88 and 89. When you enter a graphics mode and a new display list is set up, SAVMSC is updated from the LMS address. BASIC PLOT and DRAWTO statements use SAVMSC to find an address at which to store data.

In order to use BASIC to write data to one screen while displaying the other, you must change the high byte of SAVMSC (decimal 89) back to its alternate value. Your PLOT data then goes to the alternate screen, while the display list still uses the LMS address to get its data for the screen image. Remember that SAVMSC is only updated when you enter a graphics mode and set up a display list. When you switch between display lists, SAVMSC does not change.

When you are updating your alternate screen, you may get out-of-bounds errors if your two screens have different x/y limits. BASIC uses location 87 decimal to determine the mode for boundary checking. When you change SAVMSC to your alternate screen, you should also put the alternate screen mode number at 87. For example, if you are showing a mode 3 screen and updating a mode 7 screen, POKE 87,7 when you wish to write to the mode 7 screen.

```
1 REM *** DISPLAY LIST ALTERNATION ***
2 REM ********** CC/JB 2/82 **********
3 REM ******************************
10 GRAPHICS 7:REM set up first display list
20 DL1LO=PEEK(560):DL1HI=PEEK(561):REM keep address of first display list
30 COLOR 1:FOR X=150 TO 10 STEP -10
31 PLOT X,79:DRAWTO 159,0
32 NEXT X:REM draw something on screen 1
35 REM ****************************************
40 POKE 106,PEEK(106)-32:REM move RAMTOP down 32 pages
50 GRAPHICS 7:REM set up second display list
60 DL2LO=PEEK(560):DL2HI=PEEK(561):REM keep address of second display list
70 COLOR 2:FOR X=10 TO 150 STEP 10
71 PLOT X,0:DRAWTO 159,79
72 NEXT X:REM draw something on screen 2
75 REM ****************************************
80 REM now change the high byte pointer to the display list
81 REM with a delay, so each screen can be seen.
90 POKE 561,DL1HI
95 FOR WAIT=1 TO 200:NEXT WAIT
100 POKE 561,DL2HI
105 FOR WAIT=1 TO 200:NEXT WAIT
110 GOTO 90
```

RAM Shadows of Hardware Registers

JB 6/82

A number of hardware registers are associated with RAM locations, known as shadows. Shadow registers are used to update the actual hardware registers during the vertical blank routine. Each sixtieth of a second, after the screen is updated, the OS VBLANK routine reads the value from each shadow register in RAM and writes the value into the corresponding hardware register.

The shadow registers can be used along with display list interrupts to produce different effects. The color registers, for example, are all shadowed, so a display list interrupt that changes a color can update either the hardware register or its shadow. When the hardware register is changed directly, the new color appears on the screen immediately, wherever the interrupt occurs. After the screen is drawn, the VBLANK routine reads the original value from the shadow register, and restores it to the hardware register. The original color then appears at the top of the screen, and remains there until it encounters the interrupt again.

If you wish to make a permanent change, which affects the entire screen, you would change the shadow register. The change is not apparent until the following VBLANK. No change occurs at the line of the interrupt , but as soon as a new screen is drawn, the shadow value goes into the hardware register, and the new color appears. This change affects the whole screen, and lasts beyond the frame in which the interrupt occurred.

The register and its RAM shadow can also be used together. On the following page is an example of a display list interrupt routine. The BASIC program POKEs in the values of the object code from the machine language service routine listed below it. In the routine, the hardware register is changed to produce an immediate color change on the screen. The original value is still in the shadow, so the routine reads the shadow to restore the original color. The rest of the screen then contains the original color.

Some of the familiar locations such as CH (last key pressed, 764), CHBASE (character set pointer, 756) and even the display list pointer (560,561) are actually the RAM shadows of the hardware registers. The controller locations (paddle and joystick) are also shadowed.

```
1 REM DISPLAY LIST INTERRUPT
2 REM JB 6/82
3 REM use a display list interrupt to change the color of the first line
4 REM of a graphics mode 2 screen.
5 REM ************************************************************
10 GRAPHICS 2
20 FOR ADDRESS=1536 TO 1536+28:REM set up service routine on page 6
30 READ BYTE:POKE ADDRESS,BYTE:REM get opcode value, put in address
40 NEXT ADDRESS
48 REM these data statements contain the opcode values of the machine
49 REM language service routine.
50 DATA 72,138,72,141,10,212,169
51 DATA 194
52 REM 194 is the color, in this case green.
53 DATA 141,26,208,162,15,141,10,212,202,208,250,173,200
54 DATA 2,141,26,208,104,170,104,64
55 REM ************************************************************
60 POKE 512,0:POKE 513,6:REM point DLI vector to page 6, where code is.
70 DL=PEEK(560)+PEEK(561)*256:REM start address of display list
80 POKE DL+2,112+128:REM set interrupt flag before first mode line
90 POKE 54286,192:REM enable display list interrupts
100 PRINT #6;" THIS IS COLOR 194"
```

```
                10 ; DISPLAY LIST INTERRUPT SERVICE ROUTINE
                20 ; This routine saves the registers, waits for synchronization
                30 ; with the screen, and changes the background color.
                40 ; It waits for 16 scan lines (one mode line), then changes
                50 ; the color back, and restores the registers
                60 ;
D40A            70 WSYNC    =    $D40A
D01A            80 COLBAK   =    $D01A
02C8            90 SHADOW   =    $2C8
00C2            0100 COLOR  =    194
                0110 ;
0000            0120        *=   $600
0600 48         0130        PHA
0601 8A         0140        TXA
0602 48         0150        PHA
0603 8D0AD4     0160        STA  WSYNC
0606 A9C2       0170        LDA  #COLOR
0608 8D1AD0     0180        STA  COLBAK
060B A20F       0190        LDX  #$F
060D 8D0AD4     0200 LOOP   STA  WSYNC
0610 CA         0210        DEX
0611 D0FA       0220        BNE  LOOP
0613 ADC802     0230        LDA  SHADOW
0616 8D1AD0     0240        STA  COLBAK
0619 68         0250        PLA
061A AA         0260        TAX
061B 68         0270        PLA
061C 40         0280        RTI
```

Using the Vertical Blank

JB 1/82

Machine language code which alters the screen display should be synchronized with the screen in order to avoid unsightly glitches. If a change is made while the screen is being drawn, it occurs in plain sight and in unpredictable places. In order to make sure that your changes occur between screens, the code should be placed in the vertical blank, which occurs every sixtieth of a second. On the following page is an example of a simple Vertical Blank Interrupt (VBI) or VBLANK routine.

There are two places to put a vector to your own code, one at the beginning and one at the end of the OS VBLANK routines. If you put your code before the OS routines, it is in the "immediate mode". If you put it after, it is "deferred mode". An immediate VBLANK routine has 840 machine cycles available, and a deferred routine has 1470 cycles available. The last part of a deferred routine might be visible on the screen, so display changes should be made in immediate mode, or at the beginning of deferred mode. The example uses deferred mode.

There is a built-in routine for setting the vector to the VBLANK code, called SETVBV. The address is passed with the low byte in the Y register and high byte in the X register. The accumulator should contain a 7 to select deferred mode, as shown in the example, or a 6 to select immediate mode. After calling SETVBV, continue with the main-line program. Since there is no mainline program in the example, the machine simply hangs in an infinite loop. Exit the program with SYSTEM RESET.

The example routine itself is located at an arbitrary location on page six. It checks the trigger of joystick 0 by masking out all but the least significant bit, and checking for the 0 which indicates that the trigger has been pressed. If it hasn't, we exit to the main-line program, and wait for VBLANK to come around again. If it has, we get the background color for mode 0 from the shadow register, add one to the number, and put it back. We then exit normally. To exit from a deferred VBLANK routine, use the vector given ($E462). To exit back to the OS VBLANK routines from immediate mode, jump to location $E45F (SYSVBV).

There is an excellent discussion of VBLANK processing in De Re ATARI, chapter 8. This manual is available from the ATARI Program Exchange (APX), and can be ordered by calling the toll-free line, (800) 538-1862 (outside CA) or (800) 672-1850 (inside CA).

```
                10 ;VERTICAL BLANK ROUTINE
                20 ; change color of screen on
                30 ; joystick trigger  JB 1/82
                40 ;*******************************
0000            50          *=   $600
                55 ;set up vector to VBLANK routine
0600 A050       60          LDY  #$50       ;address of routine, lo
0602 A206       70          LDX  #$06       ;address, hi byte
0604 A907       80          LDA  #$07       ;specify deferred mode
0606 205CE4     90          JSR  $E45C      ;call SETVBV
0609 4C0906     0100 LOOP   JMP  LOOP       ;continue with main-line program
                0110 ;
060C            0120        *=   $650
                0125 ;VBLANK routine
0650 AD8402     0130        LDA  $0284      ;check trigger 0
0653 2901       0140        AND  #$01       ;least significant bit
0655 C900       0150        CMP  #$0        ;is it pressed?
0657 D007       0160        BNE  EXIT       ;no, forget it
0659 AEC602     0170        LDX  $2C6       ;yes, get color 2 from shadow register
065C E8         0180        INX             ;change the color
065D 8EC602     0190        STX  $2C6       ;put it back
0660 4C62E4     0200 EXIT   JMP  $E462      ;jump to XITVBV
```

DISPLAY LIST MODIFICATION

JB 8/82

The GRAPHICS command in BASIC sets up a default display list, with three blank-8-line instructions, an LMS (Load Memory Scan) address, and an appropriate number of ANTIC mode lines. All BASIC modes have a text window, 4 lines of mode 0 (ANTIC mode 2) at the bottom. The text window can be suppressed by calling <mode number> +16. The GRAPHICS command opens the S: device, which clears out the data area, erasing the screen image. Erasure can be suppressed by calling <mode number> +32, although the data left over from another mode will appear differently in the new mode.

When BASIC opens the screen in a mode, the default display list is located under RAMTOP, the top of memory pointer. If the value of RAMTOP is changed before the GRAPHICS call, the new display list location is based on the new value. The pointer to the display list is in two hardware registers, called DLISTL/DLISTH, at 54274/54275. These are shadowed in RAM, at locations 560/561. The shadow pointers are used to change or find the location of the display list.

The easiest way to set up your own mixed-mode screen, using BASIC, is to start with a default display list set up by the GRAPHICS call. To modify, find the location with the pointers:

```
DL=PEEK(560)+PEEK(561)*256
```

The lines which you wish to modify will be found at an offset from the starting location. The first mode line on the screen starts at DL+3, and has the READ LMS bit set. This is followed by the LMS address, which points to the data area. The second mode line is at DL+6. To change a mode line, POKE in the new ANTIC mode number at the desired offset.

BASIC mode numbers are not the same as ANTIC mode numbers. A chart of ANTIC modes, and all other display list instructions, may be found in Chapter II of the Hardware Manual (TECH USER NOTES, Part no. C016555), and on the programming card in DE RE ATARI (available from APX).

Different modes take up different numbers of bytes per line, and different numbers of horizontal scan lines. When mixing modes, you must take care that the number of bytes per line comes out even, so that data will not be offset. For example, if a mode 2 (ANTIC mode 7) line, taking 20 bytes of memory, is poked into a mode 0 (ANTIC mode 2) display list, with 40-byte lines, the remaining data will be half a line (20 bytes) off. To get the start of each line back to the left side of the screen, you must use two mode 2 lines, for a total of 40 bytes.

You must also take care that the total number of scan lines does not exceed 192. If more scan lines are used, the bottom of the image will be off the screen. If more than about 200 lines are used, the image will roll.

```
1 REM DISPLAY LIST MODIFICATION
2 REM JB 6/82
3 REM An example of a modified display list, combining text and map modes.
4 REM **********************************************************************
10 GRAPHICS 0:REM set up default mode 0 display list
20 DL=PEEK(560)+PEEK(561)*256:REM starting location of display list
30 POKE DL+7,6:POKE DL+8,6:REM 2 lines of mode 1
40 POKE DL+9,7:POKE DL+10,7:REM 2 lines of mode 2
50 FOR I=11 TO 14:POKE DL+I,8:NEXT I:REM 4 lines of mode 3
60 REM **********************************************************************
70 REM The screen now contains modes 0,1,2,and 3.
80 REM The data is interpreted differently in each mode.
85 POKE 82,0:REM move left margin out to column 0
90 PRINT :PRINT "THIS IS IN MODE 0"
100 PRINT "THIS IS IN MODE 1"
110 PRINT "THIS IS IN MODE 2"
120 PRINT "THIS IS IN MODE 3"
130 PRINT "   MODE 3 INTERPRETS DATA AS COLORS,"
140 PRINT "   RATHER THAN CHARACTERS."
```

```
1 REM MIXED MODE SCREEN
2 REM LW/JB 6/82
3 REM Draws curves on a screen which is half mode 8,
4 REM and half mode 7 1/2 (ANTIC mode E)
5 REM ************************************************************
10 GRAPHICS 8:REM set up original mode 8 display list
20 POKE 708,86:POKE 709,122:POKE 710,0:POKE 711,0:POKE 712,4:REM set colors
30 POKE 87,7:REM make BASIC think screen is mode 7
40 OUT=0:REM initialize out-of-bounds flag
50 DL=PEEK(560)+PEEK(561)*256:REM start address of display list
60 POKE DL+3,78:REM first mode E line with LMS bit set
70 FOR I=0 TO 92:POKE DL+6+I,14:NEXT I:REM another 93 mode E lines
80 REM ************************************************************
100 FOR Z=1 TO 3:COLOR Z:REM 3 curves in 3 colors
110 OP=0:X=0:GOSUB 1000:GOSUB 2000
120 OP=1:FOR X=1 TO 159
130 GOSUB 1000:GOSUB 2000
140 NEXT X:U=U+2:NEXT Z
200 END
998 REM ************************************************************
999 REM -- routines called from plotter --
1000 Y=60*SIN(0.04*Z*X)*(0.5+U/10)+85:RETURN :REM calculate point
1100 POKE 89,PEEK(89)+12*T:POKE 88,PEEK(88)+128*T:RETURN :REM adjust SAVMSC
1200 YLON=ABS(YNEW-YOLD):YSHO=ABS(YNEW-79):XLON=XN-XOLD:XA=INT(XN-YSHO/YLON*XLON
):XD=SGN(INT(XN-XOLD)):RETURN
1998 REM ************************************************************
1999 REM -- subroutine to plot curves --
2000 XN=X:YNEW=Y:IF OUT=1 THEN Y=Y-80:GOTO 2040
2010 IF Y<80 THEN GOTO 2070
2020 IF INT(Y)>79 THEN OUT=1:T=1:GOSUB 1200:IF OP=0 THEN GOSUB 1100:PLOT XN,Y-80
:GOTO 2100
2030 DRAWTO XA,79:GOSUB 1100:PLOT XA+XD,0:DRAWTO XN,Y-80:GOTO 2100
2040 IF Y>0 THEN GOTO 2070
2050 OUT=0:T=-1:GOSUB 1200:IF OP=0 THEN GOSUB 1100:PLOT XN,Y+80:GOTO 2100
2060 IF OP=1 THEN DRAWTO XA+XD,0:GOSUB 1100:PLOT XA+XD,79:DRAWTO XN,Y+80:GOTO 21
00
2070 IF OP=0 THEN PLOT XN,Y:GOTO 2100
2080 DRAWTO XN,Y
2100 XOLD=XN:YOLD=YNEW:RETURN
```

USING CENTRAL I/O
OS Routines for Input/Output
JB 6/82

The Operating System has built-in routines for talking to any device. This centralized input/output procedure is used to put and get data to and from the printers, tape drive, disk drive, screen and keyboard. The procedure for using this set of routines (known as CIO) is the same, regardless of which particular device you are addressing. Two example programs are given on the following pages, showing how to use CIO to write on the screen.

To use CIO, you must set up an I/O Control Block (IOCB) with a command, and the appropriate parameters. Some commands which can be sent through the IOCB are the same for all devices: OPEN,CLOSE,GET, and PUT, for example. Some are device-specific, such as DRAWLINE for the screen, or POINT for the disk drive. The details of which commands are available, and which parameters go with which commands, are given in the OS Manual (TECH USER NOTES,C016555) under the heading "Device Specific Information".

There are eight IOCBs, 0-7. Each one is 16 bytes long, so each begins at an offset of $10 from the previous one. IOCB#0 begins at $340, IOCB#1 begins at $350, #2 at $360, and so on. #0 is used by the OS for screen editing, #6 and #7 are used by the OS at times, so we generally use #1-5.

Each of the 16 bytes in the IOCB has a special meaning. Some are set up by the system, and some by the user. The bytes we are interested in here are:

mnemonic	location	contains
ICCOM	IOCB+2	command
ICBAL	IOCB+4	buffer address, lo-byte
ICBAH	IOCB+5	buffer address, hi-byte
ICBLL	IOCB+8	buffer length, lo-byte
ICBLH	IOCB+9	buffer length, hi-byte
ICAX1	IOCB+A	aux 1 byte (read/write info)
ICAX2	IOCB+B	aux 2 byte (graphics mode)

When all of the parameters are set in the IOCB, the command is executed. To execute a command, put the offset to the IOCB you are using into the X register, the jump through the Central I/O Vector (CIOV) at $E456. If you are using IOCB#3, for example, store $30 in X, then JSR CIOV.

Two examples programs are given. The first, MEMOPAD, opens two devices, the screen and the keyboard. Characters are input from the keyboard, then output to the screen, as in the default mode of the OS, with no cartridge present. The second program, DRAWLINE, opens the screen in graphics mode 7, then plots a point and draws a line on the screen. A display list can be set up automatically, as in BASIC by specifying the graphics mode in the AUX 2 byte.

For all the codes, locations, and device-specific details, refer to Tech User Notes, C016555 Locations are also listed in the OS Source Listing, C017893, and in De Re ATARI. Tech User Notes and the OS Listing can be ordered through retailers. De Re ATARI is available from the ATARI Program Exchange (APX), and can be ordered by calling 800 538-1862 (outside CA)or 800 672-1850 (inside CA).

```
                10 ;MEMOPAD
                20 ;JB 6/82
                30 ; Use Central Input/Output (CIO) to accept input
                40 ; from keyboard, and output characters to screen.
                50 ;
0000            60           .OPT NOEJECT
0342            70 ICCOM     =    $0342      ;Command code
0344            80 ICBAL     =    $0344      ;Buffer address LO
0345            90 ICBAH     =    $0345      ;Buffer address HI
0348           0100 ICBLL    =    $0348      ;Buffer length LO
0349           0110 ICBLH    =    $0349      ;Buffer length HI
034A           0120 ICAX1    =    $034A      ;Aux 1 byte
E456           0130 CIOV     =    $E456      ;Central I/O Vector
               0140 ;
0003           0150 OPEN     =    $03        ;Code for OPEN
000C           0160 CLOSE    =    $0C        ;Code for CLOSE
0007           0170 GETCHR   =    $07        ;Code for GET CHARACTER
000B           0180 PUTCHR   =    $0B        ;Code for PUT CHARACTER
               0190 ;
0000           0200         *=    $0600
               0210 ;initialize IOCBs for OPEN
0600 A003      0220         LDY   #OPEN      ;open both devices
0602 8C5203    0230         STY   ICCOM+$10
0605 8C6203    0240         STY   ICCOM+$20
0608 A054      0250         LDY   #KCOLON&255 ;lo-byte of device pointer
060A 8C5403    0260         STY   ICBAL+$10 ;in buffer address bytes
060D A006      0270         LDY   #KCOLON/256 ;hi-byte
060F 8C5503    0280         STY   ICBAH+$10
0612 A004      0290         LDY   #$4        ;Set READ bit in aux1
0614 8C5A03    0300         STY   ICAX1+$10
0617 A056      0310         LDY   #SCOLON&255 ;lo-byte of device pointer
0619 8C6403    0320         STY   ICBAL+$20 ;
061C A006      0330         LDY   #SCOLON/256 ;hi-byte
061E 8C6503    0340         STY   ICBAH+$20
0621 A008      0350         LDY   #$8        ;Set WRITE bit in aux1
0623 8C6A03    0360         STY   ICAX1+$20
               0370 ;execute OPEN commands
0626 A210      0380         LDX   #$10       ;execute OPEN for IOCB#1
0628 2056E4    0390         JSR   CIOV
062B A220      0400         LDX   #$20       ;execute OPEN for IOCB#2
062D 2056E4    0410         JSR   CIOV
               0420 ;initialize IOCBs for GET/PUT
0630 A000      0430         LDY   #0         ;set buffer lengths to 0
0632 8C5803    0440         STY   ICBLL+$10 ;(length of 0 means data
0635 8C5903    0450         STY   ICBLH+$10 ; is passed in accumulator)
0638 8C6803    0460         STY   ICBLL+$20
063B 8C6903    0470         STY   ICBLH+$20
063E A007      0480         LDY   #GETCHR    ;GET command for keyboard
0640 8C5203    0490         STY   ICCOM+$10
0643 A00B      0500         LDY   #PUTCHR    ;PUT command for screen
0645 8C6203    0510         STY   ICCOM+$20
               0520 ; execute GET from keyboard, PUT to screen
0648 A210      0530 LOOP    LDX   #$10
064A 2056E4    0540         JSR   CIOV
064D A220      0550         LDX   #$20
064F 2056E4    0560         JSR   CIOV
0652 D0F4      0570         BNE   LOOP
               0580 ;
0654 4B        0590 KCOLON .BYTE "K:"      ;Reserve ASCII K for device pointer
0655 3A
0656 53        0600 SCOLON .BYTE "S:"      ;Reserve ASCII S for device pointer
0657 3A
```

```
                10 ;DRAWLINE
                20 ;JB 6/82
                30 ; Use CIO to open mode 7 screen and draw a line.
                40 ;
0000            50          .OPT NOEJECT
0352            60 ICCOM    =    $352       ;Command byte of IOCB#1
0354            70 ICBAL    =    $354       ;Buffer address lo byte
0355            80 ICBAH    =    $355       ;Buffer address hi
0358            90 ICBLL    =    $358       ;Buffer length lo byte
0359            0100 ICBLH  =    $359       ;Buffer length hi
035A            0110 ICAX1  =    $35A       ;Aux 1 byte
035B            0120 ICAX2  =    $35B       ;Aux 2 byte
E456            0130 CIOV   =    $E456      ;Central I/O vector
                0140 ;
0003            0150 OPEN   =    $3         ;Command codes
000B            0160 PUTCHR =    $B
0011            0170 DRAW   =    $11
                0180 ;
0054            0190 ROWCRS =    $54        ;Cursor location - row
0055            0200 COLCRS =    $55        ; - column
02FB            0210 ATACHR =    $2FB       ; Color information
                0220 ;
0000            0230        *=   $600
                0240 ;
0600 A903       0250        LDA  #OPEN      ;open screen using IOCB#1
0602 8D5203     0260        STA  ICCOM      ;already defined with offset
0605 A908       0270        LDA  #$08       ;set write bit in AUX1
0607 8D5A03     0280        STA  ICAX1
060A A907       0290        LDA  #$07       ;set mode in AUX2
060C 8D5B03     0300        STA  ICAX2
060F A952       0310        LDA  #SCOLON&255 ;point to screen
0611 8D5403     0320        STA  ICBAL
0614 A906       0330        LDA  #SCOLON/256 ;hi byte of pointer
0616 8D5503     0340        STA  ICBAH
0619 A900       0350        LDA  #$0        ;buffer length of 0 causes data to be
061B 8D5803     0360        STA  ICBLL      ;passed in the accumulator
061E 8D5903     0370        STA  ICBLH
0621 A210       0380        LDX  #$10       ;keep offset to IOCB#1 in the X register
0623 2056E4     0390        JSR  CIOV       ;call CIO
                0400 ;
0626 A900       0410        LDA  #$0        ;store cursor position
0628 8554       0420        STA  ROWCRS     ;pos(0,0)
062A 8555       0430        STA  COLCRS
062C A903       0440        LDA  #$3        ;use color register 3
062E 8DFB02     0450        STA  ATACHR
0631 A90B       0460        LDA  #PUTCHR    ;plot a point
0633 8D5203     0470        STA  ICCOM
0636 A210       0480        LDX  #$10       ;keep offset
0638 2056E4     0490        JSR  CIOV       ;call CIO
                0500 ;
063B A949       0510        LDA  #$49       ;set drawto location
063D 8554       0520        STA  ROWCRS
063F A999       0530        LDA  #$99
0641 8555       0540        STA  COLCRS
0643 A903       0550        LDA  #$3        ;still using register 3
0645 8DFB02     0560        STA  ATACHR
0648 A911       0570        LDA  #DRAW      ;drawline command
064A 8D5203     0580        STA  ICCOM
064D A210       0590        LDX  #$10       ;keep offset
064F 2056E4     0600        JSR  CIOV       ;call CIO
                0610 ;
0652 53         0620 SCOLON .BYTE "S",$9B   ;store ascii data for handler pointer
0653 9B
0654 4C5406     0630 LOOP   JMP  LOOP       ;keep image on screen
```

```
                10 ;DIRECT SCREEN WRITE
                20 ;JB 6/82
                30 ;this program fills the screen with a character by storing it
                40 ;directly into the screen data area
                50 ;*************************************************************
0000            60          *=    $6000
00CC            70 SCREEN   =     $CC         ;start address of screen
                80 ;$CC is one of the free bytes on zero page
                90 ;
6000 A558       0100        LDA   $58         ;starting address of screen from SAVMSC
6002 85CC       0110        STA   SCREEN      ;lo-byte
6004 A559       0120        LDA   $59
6006 85CD       0130        STA   SCREEN+1    ;hi-byte
6008 A921       0140        LDA   #$21        ;internal code for character
600A A204       0150        LDX   #04         ;hi-byte of screen length
600C A000       0160        LDY   #00         ;lo-byte of screen length
600E 91CC       0170 STUFF  STA   (SCREEN),Y ;store character at screen location
6010 88         0180        DEY               ;next location
6011 D0FB       0190        BNE   STUFF       ;256 locations
6013 E6CD       0200        INC   SCREEN+1    ;increment hi-byte
6015 CA         0210        DEX               ;count down # of pages in screen length
6016 D0F6       0220        BNE   STUFF       ;put up next 256 characters
6018 4C1860     0230 END    JMP   END         ;loop to keep image on screen
```