ADVANCED GRAPHICS

Examples and discussions of Player-Missile Graphics

and features of Graphics Mode 8

1) Moving a Player
2) Setting Priority
3) Collision Detection
4) Using Missiles
5) String-Player
6) Color Artifacts
7) Text in Mode 8
8) VBLANK Player Move

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

(800) 672-1404 inside CA
(800) 538-8543 outside CA
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.
PLAYER-MISSILE GRAPHICS
Moving a Player
JB 8/82

The programs on the following pages, Moving a Player, versions 1 and 2, illustrate the use of the joystick in moving a player around the screen. Both programs are entirely in BASIC.

In these, and in all of the following P/M Graphics programs, a subroutine is used to initialize the player. The subroutine performs the major housekeeping tasks for setting up a player:

1) P/M Graphics is enabled at DMACTL and GRACTL, selecting single-line resolution;
2) An area of free RAM is allocated for player data by setting PMBASE;
3) The starting location of the player is computed from PMBASE;
4) The color and horizontal position are initialized;
5) The player data area is cleared.

In these programs, the RAM area selected for the players is computed from RAMTOP, the top of memory pointer. Since RAMTOP is also used in setting up the display list and screen data area, the program steps back a number of pages from RAMTOP in order to place the player data area directly below the screen data area. This is not the only way to do it; you can simply select an area of free RAM. For example, to start the player data area at page 14 of memory ($1400, or decimal 5120), simply POKE PMBASE,14*1024/256.

The body of the program reads the joystick and moves the player image. The image is not drawn on the screen except as part of the vertical movement routine, so it does not appear until the stick indicates a move down. An ON...GOSUB statement is used to read the stick, eliminating the need for time-consuming IF statements.

In version 1 of the program, a single byte of data (231) is used for all 8 lines of the player. This simplifies the vertical movement, and speeds it up considerably. Because all 8 lines have the same bit pattern, it is only necessary to erase the top line and draw the bottom line to move down, or vice-versa to move up. Version 2 illustrates the more realistic situation, in which different lines have different bit patterns, creating an interesting shape. The data for the player shape is contained in line 120. It takes noticeably longer to redraw the entire player for each vertical movement.

Using BASIC to move players vertically is quite slow. If more than one player is used, movement is even slower. Horizontal movements are much faster, since it is a simple matter of poking a register. If fast vertical movement is required, it is best to use a machine-language subroutine. An example of a VBLANK routine to read the joystick and move a player is provided on a later page.
1 REM MOVING A PLAYER
2 REM JB 5/82
3 REM -- select one of the four players and move it around the screen
4 REM using the joystick,
5 GOSUB 1000:REM initialize player
20 UP=100;DOWN=200;EAST=300;WEST=400
30 SE=500;NE=550;SW=600;NW=650:REM motion routine locations
40 Z=90:REM no motion, return only
50 REM the following statement checks the joystick, and send control
55 REM to the appropriate subroutine.
60 ON STICK(0) GOSUB Z,Z,Z,SE,NE,EAST,Z,SW,NW,WEST,Z,DOWN,UP,Z
70 IF STRIG(0)=0 THEN GOSUB 800:REM on trigger, change the player color
80 GOTO 60:REM keep checking stick
90 RETURN :REM no motion, keep checking stick
95 REM
99 REM -- motion routines --
100 X=X-1:IF X<0 THEN X=0:REM *** move up
110 POKE PSTART+X,231:POKE PSTART+X+8,0:RETURN
111 REM draw top line, erase bottom line
200 X=X+1:IF X>250 THEN X=250:REM *** move down
210 POKE PSTART+X,231:POKE PSTART+X-8,0:RETURN
211 REM draw bottom line, erase top line
300 H=H+1:IF H>200 THEN H=200:REM *** move east
310 POKE HPoS,H:RETURN
400 H=H-1:IF H<50 THEN H=50:REM *** move west
410 POKE HPoS,H:RETURN
500 GOSUB DOWN:GOSUB EAST:RETURN :REM *** move southeast
550 GOSUB UP:GOSUB EAST:RETURN :REM *** move northeast
600 GOSUB DOWN:GOSUB WEST:RETURN :REM *** move southwest
650 GOSUB UP:GOSUB WEST:RETURN :REM *** move northwest
700 REM
799 REM this subroutine changes the player color
800 C=PEEK(COL):C=C+2
810 IF C>255 THEN C=4:REM skip black
820 POKE COL,C
830 RETURN
990 REM
999 REM this subroutine initializes the player
1000 GRAPHICS 5:PRINT "PLAYER 1,2,3,4...":INPUT PNUM:REM select a player
1005 REM assign start address, color register, horizontal position reg
1010 IF PNUM=1 THEN OFFSET=1024:COL=704;HPoS=53248
1020 IF PNUM=2 THEN OFFSET=1280:COL=705;HPoS=53249
1030 IF PNUM=3 THEN OFFSET=1536:COL=706;HPoS=53250
1040 IF PNUM=4 THEN OFFSET=1792:COL=707;HPoS=53251
1050 POKE 559,62:POKE 53277,3:REM enable players w/single-line resolution
1060 PBMSB=PEEK(106)-24:POKE 54279,PBMBASE:REM step back 24 pages from
1061 REM ramtop to assign player ram area
1070 PSTART=PBMSB*256+OFFSET:REM starting address of player
1080 POKE COL,88;H=50:POKE HPoS,H:REM assign color, horizontal position
1090 FOR I=0 TO 255:POKE PSTART+I,0:NEXT I:REM clear player
1095 PRINT "MOVE STICK DOWN TO MAKE PLAYER APPEAR"
1100 RETURN
1 REM MOVING A PLAYER! VERSION 2
2 REM JB 5/82
3 REM ** select a player and move it with joystick! in this version the
4 REM player is assymetric, so the movement is slower. 7 lines are drawn
5 REM each time instead of 1 line.
6 REM **************************************************************************
10 GOSUB 1000:REM initialize player
20 UP=100:DOWN=200:EAST=300:WEST=400
30 SE=550:NE=550:SW=600:NW=650:REM motion routine locations
40 Z=90:REM no motion, return only
50 REM the following statement checks the joystick, and send control
55 REM to the appropriate subroutine.
60 ON STICK(0) GOSUB Z,Z,Z,SE,NE,EAST,Z,SW,NW,WEST,Z,DOWN,UP,Z
70 IF STRIC(0)=0 THEN GOSUB 800:REM on trigger, change the player color
80 GOTO 60:REM keep checking stick
90 RETURN :REM no motion, keep checking stick
95 REM **************************************************************************
99 REM -- motion routines --
100 X=X-1:IF X<0 THEN X=0:REM *** move up
110 FOR I=0 TO 61:READ B:POKE PSTART+X+I,B:NEXT I
111 POKE PSTART+X+7,0:RESTORE 120:RETURN
120 DATA 126,231,195,219,195,231,126
200 X=X+1:IF X>250 THEN X=250:REM *** move down
210 FOR I=0 TO 61:READ B:POKE PSTART+X+I,B:NEXT I
211 POKE PSTART+X-1,0:RESTORE 120:RETURN
300 H=H+1:IF H>200 THEN H=200:REM *** move east
310 POKE HP0S,H:RETURN
400 H=H-1:IF H<50 THEN H=50:REM *** move west
410 POKE HP0S,H:RETURN
500 GOSUB DOWN:GOSUB EAST:RETURN :REM *** move southeast
550 GOSUB UP:GOSUB EAST:RETURN :REM *** move northeast
600 GOSUB DOWN:GOSUB WEST:RETURN :REM *** move southwest
650 GOSUB UP:GOSUB WEST:RETURN :REM *** move northwest
700 REM **************************************************************************
799 REM this subroutine changes the player color
800 C=PEEK(COL):C=C+2
810 IF C>255 THEN C=4:REM skip black
820 POKE COL,C
830 RETURN
990 REM **************************************************************************
999 REM this subroutine initializes the player
1000 GRAPHICS 5:PRINT "PLAYER 1,2,3,4...":INPUT PNUM:REM select a player
1005 REM assign start address, color register, horizontal position reg
1010 IF PNUM=1 THEN OFFSET=1024:COL=704:HP0S=53248
1020 IF PNUM=2 THEN OFFSET=1280:COL=705:HP0S=53249
1030 IF PNUM=3 THEN OFFSET=1536:COL=706:HP0S=53250
1040 IF PNUM=4 THEN OFFSET=1792:COL=707:HP0S=53251
1050 POKE 559,62:POKE 53277,3:REM enable players w/single-line resolution
1060 PMBASE=PEEK(106)-24:POKE 54279,PMBASE:REM step back 24 pages from
1061 REM rantop to assign player ram area
1070 PSTART=PMBASE*256+OFFSET:REM starting address of player
1080 POKE COL,88:H=50:POKE HP0S,H:REM assign color, horizontal position
1090 FOR I=0 TO 255:POKE PSTART+I,0:NEXT I:REM clear player
1095 PRINT "MOVE STICK DOWN TO MAKE PLAYER APPEAR"
1100 RETURN
PLAYER/MISSILE GRAPHICS
Using the Priority Register
JB 4/82

The priority of players and playfield objects can be controlled by setting bits in the priority register, PRIOR, location $D01B. PRIOR has a RAM shadow, GPRIOR, at $26F, or decimal 623. By poking different bits on at this location, you can control whether the player passes in front of or behind a playfield object of a particular color.

There are four types of priority, each of which is selected with one of the four least-significant bits of PRIOR. Bit D0 selects a mode in which all players pass in front of all playfield objects. Bit D1 selects a mode in which players 0 and 1 go in front, and players 2 and 3 go behind the playfield objects. When bit D2 is set, all playfield objects have priority over players, and when bit D3 is set, playfield objects 0 and 1 have priority over all players, which have priority over objects 2 and 3. In all cases, all players and all other playfield types have priority over the background and anything drawn in the background color. There is a chart of these priorities, along with some details on conflicting-priorities, in Tech User Notes, C016555, on page III.8 of the Hardware Manual.

The following program shows priorities in action. A playfield is drawn, using all three colors, each color being a playfield object type. You select which player you want to use, 1-4. The program then asks you to select a priority. The choices, 1,2,4 or 8, are the numbers that can be poked into GPRIOR to turn on the appropriate bit. Once you have selected the priority, move the player across the different playfield objects, using the joystick. Move the joystick down to make the player appear the first time. When you press the trigger, you can select a new priority. To select a different player, press RESET and RUN the program again.
1 REM PRIORITY
2 REM JB 4/82
3 REM -- observe different priorities of players and playfield objects.
4 REM you select a player 1-4 and one of the four priority options,
5 REM then move the player over the different colors and see what happens.
6 REM
7 GOSUB 1000:REM initialize player
8 GOSUB 2000:REM draw playfield
9 UP=100:DOWN=200:EAST=300:WEST=400
10 SE=500:NE=550:SW=600:NW=650:REM motion routine locations
11 Z=90:REM no motion, return only
12 PRINT "PRIORITY 1,2,3,4...";INPUT P:POKE 623,P:REM select priority bit
13 ON STICK(0) GOSUB Z,Z,Z,Z,SE,NE,EAST,Z,SW,NW,WEST,Z,DOWN,UP,Z
14 IF STRIG(0)=0 THEN GOTO 50:REM on trigger, select new priority
15 GOTO 60:REM keep checking stick
16 RETURN :REM no motion, keep checking stick
17 REM
18 REM -- motion routines --
19 X=X-1:IF X<0 THEN X=0:REM *** move up
20 POKE PSTART+X,231:POKE PSTART+X+8,0:RETURN
21 REM draw top line, erase bottom line
22 X=X+1:IF X>250 THEN X=250:REM *** move down
23 POKE PSTART+X,231:POKE PSTART+X-8,0:RETURN
24 REM draw bottom line, erase top line
25 H=H+1:IF H>200 THEN H=200:REM *** move east
26 POKE HPOS,H:RETURN
27 H=H-1:IF H<50 THEN H=50:REM *** move west
28 POKE HPOS,H:RETURN
29 GOSUB DOWN:GOSUB EAST:RETURN :REM *** move southeast
30 GOSUB UP:GOSUB EAST:RETURN :REM *** move northeast
31 GOSUB DOWN:GOSUB WEST:RETURN :REM *** move southwest
32 GOSUB UP:GOSUB WEST:RETURN :REM *** move northwest
33 REM
34 REM this subroutine initializes the player
35 GRAPHICS 5:PRINT "PLAYER 1,2,3,4...";INPUT PNUM:REM select a player
36 REM assign start address, color register, horizontal position reg
37 IF PNUM=1 THEN OFFSET=1024:COL=704:HPOS=53248
38 IF PNUM=2 THEN OFFSET=1280:COL=705:HPOS=53249
39 IF PNUM=3 THEN OFFSET=1536:COL=706:HPOS=53250
40 IF PNUM=4 THEN OFFSET=1792:COL=707:HPOS=53251
41 POKE 559,62:POKE 53277,3:REM enable players w/single-line resolution
42 PMBASE=PEEK(106)-24:POKE 54279,PMBASE:REM step back 24 pages from
43 REM ram top to assign player ram area
44 PSTART=PMBASE+OFFSET:REM starting address of player
45 POKE COL,80:H=50:POKE HPOS,H:REM assign color, horizontal position
46 FOR I=0 TO 255:POKE PSTART+I,0:NEXT I:REM clear player
47 RETURN
48 REM
49 REM this subroutine draws the playfield-- one bar of each color
50 COLOR 1
51 FOR X=10 TO 20:FOR Y=0 TO 39
52 PLOT X,Y
53 NEXT Y:NEXT X
54 COLOR 2
55 FOR X=30 TO 40:FOR Y=0 TO 39
56 PLOT X,Y
57 NEXT Y:NEXT X
58 COLOR 3
59 FOR X=50 TO 60:FOR Y=0 TO 39
60 PLOT X,Y
61 NEXT Y:NEXT X
62 RETURN
When you are using Player/Missile Graphics, it is possible to detect collisions between players and missiles, players and other players, or between the playfield and either players or missiles. In order to do this, you must check the values at the special collision registers. The numbers reflect the bit patterns which tell you exactly which player, missile or playfield object has been hit.

There are 16 collision registers, and a special register called HITCLR, which clears all of the other registers. HITCLR is write-only, which means you can only POKE it. If you check the PEEK, it does not match what you put there. POKEing anything other than a 0 into HITCLR (decimal location 53278) has the effect of clearing all collision registers.

The collision registers themselves are read-only. You cannot POKE into them. They are cleared by writing to HITCLR; this is the only way to change them. The contents of these registers reflect the state of the screen display. When any object occupies the same coordinates as any other object, the appropriate bit is turned on.

The 16 collision registers are located as follows:

- 53248 Missile 0 to Playfield
- 53249 Missile 1 to Playfield
- 53250 Missile 2 to Playfield
- 53251 Missile 3 to Playfield
- 53252 Player 0 to Playfield
- 53253 Player 1 to Playfield
- 53254 Player 2 to Playfield
- 53255 Player 3 to Playfield
- 53256 Missile 0 to Player
- 53257 Missile 1 to Player
- 53258 Missile 2 to Player
- 53259 Missile 3 to Player
- 53260 Player 0 to Player
- 53261 Player 1 to Player
- 53262 Player 2 to Player
- 53263 Player 3 to Player

The least significant nybble of each register is used to show collisions. The least significant bit, bit D0, is set (contains a 1) when there is a collision with Player or Playfield type 0. The next bit is set on a collision with Player or Playfield type 1, and so on. For example, when Missile 1 collides with Player 3, location 53257 contains the binary number 0000 1000. The decimal equivalent is 8, so PEEK(53257)=8.

If several collisions happen before the registers are cleared, all of the affected bits stay on. The bit for a Player's collision with itself is always 0.

Playfield objects are objects drawn on the screen with regular Display List Graphics, as opposed to Player-Missile Graphics. Anything drawn with PLOT or DRAWTO is a Playfield object.

The type of a Playfield object is determined by which color register it is drawn with. Objects drawn with register 0 are type 0, and collisions show up in bit D0. SETCOLOR numbers are the same as color register numbers.

In modes 2-7, color register 4 contains the background color. In modes 0 and 8 however, register 2 is the background. In these modes, a collision is always indicated between Playfield object 2 and all Players and Missiles that are on the screen.
1 REM COLLISIONS
2 REM JB 5/82
3 REM -- detect collisions between a player and various playfield objects!
4 REM when a collision occurs, the playfield object changes color.
5 REM ************************************************************************
10 GOSUB 1000:REM initialize player
20 GOSUB 2000:REM clear playfield
25 POFF=53252;HITCLR=53278:REM location of collision & hitclear register
30 UP=100:DOWN=200:EAST=300:WEST=400
40 SE=500:NE=550:SW=600:NW=650:REM motion routine locations
50 Z=90:REM no motion, return only
55 GOSUB 3000:REM call collision checking routine
60 ON STICK(0) GOSUB Z,Z,Z,SE,NE,EAST,Z,SW,NW,DOWN,UP,Z
70 GOTO 50:REM keep checking
90 RETURN :REM no motion, keep checking stick
95 REM ************************************************************************
99 REM -- motion routines --
100 X=X-1:IF X<0 THEN X=0:REM *** move up
110 POKE PSTART+X,231:POKE PSTART+X+8,0:RETURN
111 REM draw top line, erase bottom line
200 X=X+1:IF X>250 THEN X=250:REM *** move down
210 POKE PSTART+X,231:POKE PSTART+X-8,0:RETURN
211 REM draw bottom line, erase top line
300 H=H+1:IF H>200 THEN H=200:REM *** move east
310 POKE HPOS,H:RETURN
400 H=H-1:IF H<50 THEN H=50:REM *** move west
410 POKE HPOS,H:RETURN
500 GOSUB DOWN:GOSUB EAST:RETURN :REM *** move southeast
550 GOSUB UP:GOSUB EAST:RETURN :REM *** move southeast
600 GOSUB DOWN:GOSUB WEST:RETURN :REM *** move southwest
650 GOSUB UP:GOSUB WEST:RETURN :REM *** move southwest
700 REM ************************************************************************
999 REM this subroutine initializes the player
1000 GRAPHICS 5+16:REM no text window
1001 REM assign start address, color register, horizontal position register
1010 POKE 559,6:POKE 53277,3:REM enable players w/single-line resolution
1060 PMBASE=PEEK(166)-24:POKE 54279,PMBASE:REM step back 24 pages from
1061 REM raptop to assign player ram area
1070 PSTART=PMBASE*256+OFFSET:REM starting address of player
1080 POKE COL,88:H=50:POKE HPOS,H:REM assign color, horizontal position
1090 FOR I=0 TO 255:POKE PSTART+I,0:NEXT I:REM clear player
1100 RETURN
1998 REM ************************************************************************
1999 REM this subroutine draws the playfield-- one bar of each color
2000 COLOR 1
2001 FOR X=10 TO 201:FOR Y=0 TO 19
2002 PLOT X,Y
2003 NEXT Y:NEXT X
2040 COLOR 2
2050 FOR X=30 TO 40:FOR Y=20 TO 39
2060 PLOT X,Y
2070 NEXT Y:NEXT X
2080 COLOR 3
2090 FOR X=50 TO 60:FOR Y=10 TO 29
2100 PLOT X,Y
2110 NEXT Y:NEXT X
2120 RETURN
2998 REM ************************************************************************
2999 REM check for collisions! if there is one, that object changes color
2998 REM ***************
2999 REM check for collisions: if there is one, that object changes color
3000 C0=PEEK(708):C1=PEEK(709):C2=PEEK(710):REM get colors from registers
3010 IF PEEK(POPF)=1 THEN C0=C0+1:IF C0>255 THEN C0=16:REM check object 0
3020 POKE 708,C0:REM put in new color (or same color if no collision)
3030 IF PEEK(POPF)=2 THEN C1=C1+1:IF C1>255 THEN C1=16:REM check object 1
3040 POKE 709,C1:REM new color if collision
3050 IF PEEK(POPF)=4 THEN C2=C2+1:IF C2>255 THEN C2=16:REM check object 2
3060 POKE 710,C2:REM new color if collision
3070 POKE HITCLR,1:REM clear collision register
3080 RETURN
All of the missiles start at the same offset from PMBASE. The offset is +768 for single-line, and +384 for double-line resolution. The missile area extends to the start of player 0, at +1024 or +512. It is the same length as a player area, 255 bytes in single-line resolution, 127 bytes in double-line resolution.

The missiles are very much like a fifth player. The difference is that the missile area is controllable two bits at a time. The horizontal position register for missile 0 controls the horizontal position of the lowest two bits of the missile area. Missile 0 gets its color from player 0.

To turn on a missile, you must enable Player-Missile Graphics and define the start of the missile area at the correct offset from PMBASE. Select a location on the screen by adjusting the offset from the missile starting address. Once you have figured out this location, turn on the missile by poking in data. The data you put there controls which missile is turned on.

The data for a missile is the number which turns on the associated bits. For example, the lower two bits are missile 0. To turn on missile 0, you need the binary number 0000 0011. This is a decimal 3. If you POKE MISSILESTART+OFFSET,3 missile 0 appears on the screen. If you want both missile 0 and missile 3, you need the binary number 1100 0011. This is decimal 195 (3+192). To turn on both of these missiles in the same vertical position, POKE MISSILESTART+OFFSET,195.

The bits are associated with the missiles as follows:

- 0000 0000: all missiles off (0)
- 0000 0011: missile 0 on (3)
- 0000 1100: missile 1 on (12)
- 0011 0000: missile 2 on (48)
- 1100 0000: missile 3 on (192)
- 1111 1111: all missiles on (255)

Like players, the vertical position of a missile is changed by changing the offset from the starting address. Zero the missile bits at the old offset, to erase the previous image, and poke the data at the new offset. Remember to erase only the missile that moves. You cannot just POKE in a zero, you must zero the bits that belong to that missile.

The size of a missile can be set in the size register, 53260. Missiles, like players, can be single, double or quadruple width. For double size, turn on the lower, or right-hand bit of the appropriate missile. For quadruple size, turn on both bits.

The following program turns on three missiles. All three are different colors. Two of them move vertically up the screen, at different horizontal positions. The third is quadruple size, and moves horizontally across the screen.

To get a feeling for missiles, you can try putting in the fourth missile, or changing the various parameters in this simple program, such as size, horizontal position, color, or direction of movement.
1 REM MISSILES
2 REM JB 5/82
3 REM demonstrate the use of missiles in player-missile graphics
4 REM ******************************************************************************************
10 M0=3:M1=12:M2=48:REM data for each missile
20 GOSUB 1000:REM set up p/m graphics
30 POKE SIZEM,M2:REM missile 2 is quadruple size
40 H=50:POKE HPOSMM2,H:REM horizontal position of missile 2
50 POKE MSTART+50,M2:POKE 706,88:REM color and initial position, m2
60 POKE HPOSMM0,100:POKE 704,62:REM color and horizontal position, M0
70 POKE HPOSMM1,120:POKE 705,191:REM color and horizontal position, M1
80 FOR I=127 TO 1 STEP -1:REM move up from bottom of screen
90 POKE MSTART+I,M0+M1:POKE MSTART+I+1,0:REM poke in new image, erase old
100 IF I=50 THEN POKE MSTART+I,M0+M1+M2:REM when the paths cross
110 IF I<50 THEN POKE MSTART+50,M2:REM keep m3 turned on
120 H=H+1:POKE HPOSMM2,H:REM move m3 horizontally
130 NEXT I:REM until m0 and m1 go off the screen
140 H=H+1:IF H<250 THEN POKE HPOSMM2,H:GOTO 140:REM move m3 rest of way
150 PRINT "THERE THEY WENT...":END
999 REM ******************************************************************************************
1000 PRINT "HERE THEY COME...
1010 POKE 559,46:POKE 53277,3:REM enable p/m graphics, double-line resolution
1020 I=PEEK(106)-16:POKE 54279,I:REM set up pmbase
1030 MSTART=I*256+384:REM start of missile data area
1040 SIZEM=53250:REM size register for missiles
1050 HPOSMM0=53252:HPOSMM1=53253:HPOSMM2=53254:REM horizontal positions
1060 FOR I=0 TO 127:POKE MSTART+I,0:NEXT I:REM clear missiles
1070 RETURN
1 REM : STRING-PLAYER
2 REM : EZ/JB 11/81
4 REM : make BASIC think the player/missile area is a string:
5 REM : player movement is then accomplished by string-assignment
9 REM ##################################################################
100 DIM P$(1),D$(22)
108 REM player/string of control characters
109 REM contains spaces on ends to erase previous image
110 D$="◇◇◇<<<<<□□<<<<□□"
111 REM to define your own control string, use line 110 GOSUB 1000 instead.
119 REM assign location of variable value table, and string-array area
120 VTAB=PEEK(139)+256*PEEK(135)
130 ATAB=PEEK(140)+256*PEEK(141)
200 GRAPHICS 8
210 POKE 559,62:REM set resolution
230 POKE 704,80:REM set color
240 PMBASE=PEEK(106)-8:REM step back from RAMTOP
250 POKE 54279,PMBASE:REM to set PMBASE
260 POKE 53277,3:REM enable players
270 POKE 53256,3:REM at quadruple size
340 X=110:POKE 53248,X:REM set horizontal position
500 OFFSET=256*PMBASE+1024-ATAB:REM figure offset to player 0
510 V3=INT(OFFSET/256):REM hi-byte
520 V2=OFFSET-256*V3:REM lo-byte
530 POKE VTAB+2,V2:REM displacement of player (string) from STARP
540 POKE VTAB+3,V3:REM hi-byte
550 POKE VTAB+4,20:REM string length (266 bytes)
560 POKE VTAB+5,1:REM hi-byte
570 POKE VTAB+6,20:REM dimension length (266 bytes)
580 POKE VTAB+7,1:REM hi-byte
590 Y=110:POKE(Y,Y+21)=D$:REM initialize string-player in middle of screen
600 FOR EVER=0 TO 0 STEP 0:REM check stick
610 IF STRING(0)=0 THEN 800:REM use trigger to exit
620 SVVAL=STICK(0):IF SVVAL=15 THEN 690
641 IF SVVAL>9 AND SVVAL<8 THEN X=X+1
642 IF SVVAL>8 AND SVVAL<12 THEN X=X-1
644 IF SVVAL=5 OR SVVAL=9 OR SVVAL=13 THEN Y=Y+2
647 IF SVVAL=6 OR SVVAL=10 OR SVVAL=14 THEN Y=Y-2
670 POKE 53248,X:POKE(Y,Y+21)=D$:REM set horizontal and vertical position
690 NEXT EVER
800 POKE 53248,1:REM horizontal position off screen for exit
810 POKE 53277,0:REM disable Player/Missile DMA
899 REM ##################################################################
900 REM the following subroutine can be used to define
910 REM the string of control characters which contains the player shape.
1000 D$="◇◇"
1005 ? "300 TO STOP"
1010 ? "BIT PATTERN ";INPUT N
1020 IF N=300 THEN 1050
1030 D$(LEN(D$)+1)=CHR$(N)
1040 GOTO 1010
1050 D$(LEN(D$)+1)="◇◇"
1060 RETURN
COLOR ARTIFACTS
Extra Colors in Mode 8
JB 2/82

Mode 8 is the highest resolution graphics mode available. The individual pixels are very small, half a color clock wide. Only one color register is available, although any of the 16 hues can be put into that register. The foreground is a bright luminance of that hue, and the background uses a low luminance.

A color clock is the smallest unit of horizontal measurement in which all of the colors can be displayed. Since each mode 8 pixel is only half a color clock wide, you cannot get every color in every pixel. If you hit one side of the color clock, you get one color, and if you hit the other side, you get the other color. The foreground color which shows up is a combination of the two artifacts, which actually appear in individual pixels.

Artifacts can sometimes work for you. If you wish to separate the colors, simply turn on only even or odd pixels. Since the resolution is so fine, the resulting color areas still appear solid. In this way you can get 4 colors at a time in a 2-color mode, without resorting to machine language subroutines. The 4 colors are the two artifacts, the foreground (a combination of the artifacts), and the background.

The following program demonstrates artifact colors by drawing a bar of even-numbered pixels, a bar of odd-numbered pixels, and a solid bar, with both even and odd pixels. The program then cycles through the 16 colors, with the highest luminance in the foreground register and the lowest luminance in the background. You will notice that the artifact colors are not the same as the usual 16 colors. With the GTIA chip, both the usual colors and the artifact colors are slightly different than with CTIA.

1 REM EZ ARTIFACTS
2 REM EZ/JB 2/82
3 REM ******************************************************************************
10 GRAPHICS 8:POKE 752,1:REM disable cursor
20 SETCOLOR 1,0,14:REM brightest luminance for foreground
30 SETCOLOR 2,0,0:REM lowest luminance for background
40 COLOR 1:REM select foreground register
50 REM *** draw horizontal bar, using only odd-numbered pixels ***
55 FOR I=1 TO 319 STEP 2:PLLOT I,0:DRAWTO I,40:NEXT I
60 REM *** draw horizontal bar, using only even-numbered pixels ***
65 FOR I=0 TO 318 STEP 2:PLLOT I,41:DRAWTO I,80:NEXT I
70 REM *** draw horizontal bar, using all pixels ***
75 FOR I=0 TO 319:PLLOT I,81:DRAWTO I,120:NEXT I
80 REM ******************************************************************************
90 REM *** cycle through colors to observe all artifact combinations ***
100 FOR C=0 TO 15
110 SETCOLOR 1,C,14:SETCOLOR 2,C,0
120 PRINT "C=";C
130 FOR WAIT=0 TO 400:NEXT WAIT
140 NEXT C
150 GOTO 100
1 REM CHARACTER IN MODE 8
2 REM ME/JB 4/82
3 REM put mode 0 characters on a mode 8 hi-res graphic screen
4 REM -- the program converts each ATASCII character to internal code,
5 REM finds that character in the ROM character set, and pokes the data
6 REM for that character directly into the screen data area in RAM.
7 REM -- note that this is only possible with mode 0 characters and mode 8
8 REM graphics, because the pixel size happens to be the same.
9 REM
10 DIM STRING$(5),X$(1)
11 STRING$="ATARI"
12 X=15:Y=80:REM some test coordinates (alters placement on screen)
13 GRAPHICS 8
14 SCREEN=PEEK(88)+256*PEEK(89):REM starting address of screen RAM
15 LOC=SCREEN+Y*40+X:REM location on screen (offset from starting adr)
16 FOR CHAR=1 TO LEN(STRING$):REM for each character in string
17 X$=STRING$(CHAR,CHAR):REM individual character
18 X=ASC(X$):REM get ATASCII code
19 IF X>127 THEN X=X-128:REM turn off inverse video
20 IF X>31 AND X<96 THEN X=X-32
21 IF X<32 THEN X=X+64:REM turn ATASCII into internal display code
22 CHARLOC=57344+X*8:REM location of character in ROM character set
23 FOR BYTE=0 TO 7:REM character data is 8 bytes long
24 POKE LOC+BYTE*40,PEEK(CHARLOC+BYTE):REM get from ROM, put on screen
25 NEXT BYTE:REM next byte of character
26 REM note that each byte is below the previous one (1 line-length apart)
27 LOC=LOC+1:REM next character is one space to the right
28 NEXT CHAR:REM get the next character in the string
29 REM
30 REM ** the following routine draws an ATARI logo with mode 8 graphics
31 N=0:COLOR 1:FOR X=100 TO 150
32 IF X<132 THEN PLOT 120,X;DRAWTO 130,X
33 IF X>=132 THEN N=N+1:PLOT 120-N,X;DRAWTO 130-N,X
34 PLOT 135,X;DRAWTO 145,X
35 IF X<132 THEN PLOT 150,X;DRAWTO 160,X
36 IF X>=132 THEN PLOT 150+N,X;DRAWTO 160+N,X
37 NEXT X
REM VBBLANK PLAYER MOVE
1 REM LW/JP 8/82
2 REM a machine language routine to move a player during vertical blank
3 REM*****************************************************************************
4 REM*****************************************************************************
5 REM set up vbblank routine on page 6 (listing follows) ******
6 REM*****************************************************************************
7 FOR I=1356 TO 1656
8 READ X:POKE I,X:NEXT I
9 REM The following numbers are the decimal equivalents of the hex object
10 REM code in the machine language program on the next page.
11 DATA 173,120,2,41,2,208,3,32,43,6,173,120,2,41,2,208,3,32,67,6,173,120
12 DATA 2,41,4,208,3,32,91,6,173,120,2,41,8,208,3,32,106,6,76,98,228,160
13 DATA 8,174,240,6,202,224,33,144,13,142,240,6,189,0,60,157,255,59,232,136
14 DATA 16,246,96,160,8,174,240,6,232,224,219,176,295,142,240,6,189,5,60
15 DATA 157,6,60,202,136,16,246,96,174,241,6,202,224,48,144,223,142,241,6
16 DATA 142,0,208,96,174,241,6,232,224,201,176,208,142,241,6,142,0,208,96
17 REM*****************************************************************************
18 REM name locations ******
19 SDMCTL=559:PMBASE=54279:GRACCTL=53277:NMIEN=54286:VVELKD=548
20 COLP0=704:HPOSP0=53248
21 REM*****************************************************************************
22 REM set up player ******
23 POKE SDMCTL,62:REM , single-line resolution
24 POKE PMBASE,14*1024/256:REM , set up player data on page 14 (hi-byte)
25 POKE GRACCTL,3:REM , enable players
26 POKE COLP0,88:POKE HPOSP0,100:REM , set color and initial horizontal pos.
27 PSTART=15*1024:REM , starting address of player 0
28 FOR I=0 TO 7:REM , create 8-line player shape
29 READ X:POKE PSTART+100+I,X
30 NEXT I
31 DATA 255,126,60,24,24,60,126,255
32 REM*****************************************************************************
33 REM set up vertical blank vector ******
34 REM*****************************************************************************
35 POKE 1776,101:POKE 1777,100:REM , init hpos and vpos in VBBLANK routine
36 POKE NMIEN,0:REM , disable DMA
37 POKE VVELKD,0:POKE VVELKDP+1,6:REM , point vector to page 6 routine
38 POKE NMIEN,64:REM , reenable DMA (P/M, standard playfield)
39 END:REM ,
40 REM ,
41 REM V blank routine is in now in place, and functions regardless of BASIC prg.
* PMOVE: A VBLANK ROUTINE TO READ JOYSTICK0 AND MOVE PLAYER

* DEFINITIONS

= 0278 STICK0 = $0278
= D000 HPOSF0 = $D000
= 3C00 PSTART = $3C00
= 06F0 VPOS = $6F0
= 06F1 HPOS = $6F1
= E462 XITVBU = $E462

0000 = 0600 ORG $600

* READ JOYSTICK

0600 AD7802 LDA STICK0
0603 2901 AND $1 ;CHECK FIRST BIT
0605 D003 ^060A BNE S1 ;BIT SET MEANS NO
0607 202E06 JSR UP ;IF CLEAR, MOVE UP
060A AD7802 S1 LDA STICK0
060D 2902 AND $2 ;CHECK NEXT BIT
060F D003 ^0614 BNE S2
0611 204306 JSR DOWN ;IF CLEAR, MOVE DOWN
0614 AD7802 S2 LDA STICK0
0617 2904 AND $4 ;CHECK NEXT BIT
0619 D003 ^061E BNE S3
061B 205E06 JSR LEFT ;IF CLEAR, MOVE LEFT
061E AD7802 S3 LDA STICK0
0621 2908 AND $8 ;CHECK LAST BIT
0623 D003 ^0628 BNE EXIT
0625 206A06 JSR RIGHT ;IF CLEAR, MOVE RIGHT
0628 4C62E4 EXIT JMP XITVBU ;THAT'S ALL

* MOVE ROUTINES

* MOVE UP

062B A008 UP LDY $8 ;INIT LINE COUNTER
062D AEF006 LDX VPOS ;GET TEMP VERTICAL POSITION
0630 CA DEX ;MOVE UP ONE
0631 E021 CPX $33 ;TOO HIGH?
0633 90D ^0642 ECC RETURN ;YES, FORGET IT
0635 BEF006 STX VPOS
0638 BD003C UPLOOP LDA PSTART,X ;MOVE IMAGE UP
063B 9DF3E STA PSTART-1,X
063E 8B INX
063F 88 DEY ;DO NINE LINES
0640 10F6 ^0638 BPL UPLOOP
0642 60 RETURN RTS

* MOVE DOWN

0643 A008 DOWN LDY $8 ;INIT LINE COUNTER
0645 AEF006 LDX VPOS
0648 8B INX ;MOVE DOWN ONE
0649 E0DA CPX $218 ;TOO LOW?
064B BF5F ^0642 BCS RETURN ;YES, FORGET IT
064D BEF006 STX VPOS
0650 BD053C DNLOOP LDA PSTART+5,X ;MOVE IMAGE DOWN
STA POSTART+6,X
DEX
DEY ;DO NINE LINES
BPL DLIN
RTS

* MOVE LEFT *

LDX HPOS ;GET TEMP HORIZONTAL POSITION
DEX
CPX $A8 ;TOO FAR?
ECC RETURN ;YES, FORGET IT
STX HPOS
STX HPOSP0
RTS

* MOVE RIGHT *

LDX HPOS
INX
CPX $201 ;TOO FAR?
ECS RETURN ;YES, FORGET IT
STX HPOS
STX HPOSP0
RTS

no ERRORS, 17 Labels, $4A0E free.