Bally/Astrocade
Vector Animation Tutorial

By Lance F. Squire
11-11-2008

This document assumes knowledge of Z80 assembly and basic Bally/Astrocade BIOS functions.

Although the Bally's BIOS is very powerful, *The Nutting Manual*, although concise, doesn't present some of the features in an intuitive way. Leaving new potential programmers wondering how some of the commands relate, and leaving much of the power hidden.

This tutorial will group and explain the 'Vector Animation' commands and structures, so as their true power can be realized more effectively.

Commands:

The complete list of commands follows.

Screen Handler section:

VWRITR	sub versions WRITE, WRITP, WRIT and WRITA
VBLANK	sub version BLANK
VECT	sub version VECTC

Cartridge Conventions (Human Interface) section:
MSKTD

Structures:

Without understanding these structures, the above commands are useless.

Vector Block
Pattern Block

As none of the Vector commands will work without the Vector Block set up, we'll start there.

Structures

Understanding the Vector Block

The Vector Block is a list of bytes that tells the BIOS how you want the graphic to move, and you how the BIOS responded to the last function request.

This is a graphical depiction of the structure:

Vector Block

Byte	Function	HVGLIB Name	Comment
0	Magic Register	VBMR	Do NOT use bit 7
1	Vector Status	VBSTAT	
2	Time Base	VBTIMB	Incremented by User
3	Delta X	VBDXL	
4		VBDXH	
5	X	VBXL	
6		VBXH	
7	X Check Mask	VBXCHK	
8	Delta Y	VBYDL	
9		VBYDH	
10	Y	VBYL	
11		VBYH	
12	Y Check Mask	VBYCHK	
13	Old	VBOAL	Maintained by User
14	Screen Address	VBOAH	(optional)

What does it all mean:

VBMR:
aka Magic Register.

The Magic functions of the Bally chips would take a whole other tutorial to explain. So, in short, it's how your image will be drawn on the screen.

Modes useful for the Vector Block are:

Bit 3	8D 08H	Expand	(MRXPND)	Monochrome stored image -> Colour screen
Bit 4	16D 10H	OR	(MROR)	Draw only bits that are not 0 in image
Bit 5	32D 20H	XOR	(MRXOR)	Flip bits 0-0=0 0-1=1 1-1=0 1-0 =1 screen-image
Bit 6	64D 40H	Flop	(MRFLOP)	Horizontal image flip
All Bits Off		Plop		Draw all bits of image

Flop, (OR or XOR) and Expand can be mixed. That is, you can Expand a monochrome image to the colour screen while XORing it with the screen image in a FLOPped mode.

Translation:

0D	0H	Draw Image, Destroy background.	(PLOP)
8D	08H	Draw Monochrome Image, Destroy Background.	(Expand, PLOP)
16D	0H	Draw Image, Keep Background where image blank (=0)	(OR)
24D	18H	Draw Monochrome Image, Keep Background where image blank	(Expand, OR)
32D	20H	Draw Image, Merge with Background	(XOR)
40D	28H	Draw Monochrome Image, Merge with Background	(Expand, XOR)
64D	40H	Draw Image Horizontally Flipped, Destroy Background	(FLOP, PLOP)
72D	48H	Draw Monochrome Image, Horizontally Flipped, Destroy Background.	(Expand, FLOP, PLOP)
80D	50H	Draw Image Horizontally Flipped, Keep Background	(FLOP, OR)
88D	58H	Draw Monochrome Image, Horizontally Flipped, Keep Background	(Expand, FLOP, OR)
96D	60H	Draw Image Horizontally Flipped, Merge with Background	(FLOP, XOR)
104D	68H	Draw Monochrome Image, Horizontally Flipped, Merge Background	(Expand, FLOP, XOR)

Ok... Got that?

VBSTAT:
(Vector Status)

Bit 7 128D 80H Active(VBSACT) Must be set or Vector routines will ignore.
Bit 6 64D 40H Blank(VBBLNK) 0 to start, used by VBLANK and VWRIT

VBTIMB:
(Time Base)

This byte indicates how many times to apply the Deltas to the Coordinates. That is, 1=1 increment per call, 2=2 increments per call. Dropping the Knob value (0-255) into here gives you variable speed without affecting the direction.

This byte is always decremented to 0 after calling VECT. Therefor you must always place at least a 1 in here before calling VECT.

Deltas:
A 'Delta' in this case is simply a value indicating a direction or Vector.

If we wanted our graphic to move down 1 pixel but Right 2 pixels, We can simply place 1 in the X Delta and 2 in the Y Delta. When we call VECT the X and Y Co-ordinates will be updated appropriately.

That's fine for very coarse movements, but sometimes you need much finer calculations to reach the target. The Bally allows fractional Deltas by using a 2nd Byte. This allows us to have our graphic move 1 pixel right each call, but only 1 pixel down every 4 calls by indicating Deltas of X=1.0 and Y=0.25.

VBDXL & VBDYL:
(Delta Fractional)

This is the Fractional half of the Delta value. (After the decimal point)

VBDXH & VBDYH:
(Delta)

This is the whole number value of the Delta. (Before the decimal point)

To enter an X Delta of 2.25, set VBDXH=2 and VBDXL=25

VBXCHK & VBYCHK:
(Boundary Checking)

Bit 1	1D 1H	Limit Check (VBCLMT) Set to one (1) for limit checking
Bit 2	2D 2H	Reverse Delta (VBCREV) changes sigh of delta when limit is reached. In other words, if the graphic reaches the limit, it's direction is automatically reversed, giving a Bounce effect.
Bit 4	8D 8H	Limit Attained (VBCLAT) If the desired effect is not an automatic Bounce, The system will set this flag and hold the graphic at the limit position, until the user changes the Delta.

Now the first question here should be, “Where do I set these Limits?!”. It's in the VECT call detailed later.

VBOAL & VBOAH
(Old Screen Address)

These bytes are only used by VBLANK and must be maintained by the User.

This means that if VBLANK is not being used to erase your graphics, then your Vector Block needn’t include them.

However, if VBLANK is being used to erase graphics, you must take the absolute screen address calculated by VWRITR and place it here.

That’s it for the Vector Block. Next we move to the Pattern Block, where your image and related parameters are defined.

The Pattern Block

This is the description of the image to be drawn. For use in the Vector commands, Displacement and Size bytes are included before the actual image.

X Displacement	0
Y Displacement	1
X Size	2
Y Size	3
Image byte1	4
Etc..	...
Image byte +n	N

Displacement:

This value allows you to set the 'control point' somewhere other that the top left corner of the image. That is, if you have a ball image 5 pixels wide and 5 pixels high and want the calculations to relate to the center of the ball, you would set X and Y displacements to 3

Eg:

Normal Displaced

Size:

X size is how many bytes wide the image data is, not how wide it will appear on the screen. For instance, if we're using a monochrome image (1 bit per pixel) our 5x5 ball would look like this in Binary:

01110000
11111000
11111000
11111000
01110000

Equaling 1 byte wide by 5 bytes tall or X=1 and Y=5

If we stored the ball image in colour, as it would appear on the screen it would look like this:

00 10 10 10 00 00 00 00
10 10 10 10 10 00 00 00
10 10 10 10 10 00 00 00
10 10 10 10 10 00 00 00
00 10 10 10 00 00 00 00

00 = Background 01=2nd colour 10=3rd colour and 11=4th colour

This is equivalent to what the monochrome ball could expand to. However this image has a fixed colour. Obviously this image would have an X of 2 and a Y of 5

Or you could make a more colourful image.

00 10 10 10 00 00 00 00
10 01 10 10 10 00 00 00
01 11 01 10 10 00 00 00
11 11 11 01 10 00 00 00
00 11 11 11 00 00 00 00

Commands

VWRITR
(Vector WRITe Relative)

This command tells the Bally's BIOS to draw our graphic to the screen as specified by the Vector Block provided.

Calling Methods:

SYSSUK:
(System loads variables for the call.)

```
SYSSUK    VWRITR
DEFW      (vector block address)
DEFW      (pattern block address)
```

SYSTEM:
(User loads variables before call.)

```
LD        HL, (pattern block address)
LD        IX, (vector block address)
SYSTEM    VWRITR
```

Returned values:

DE = Screen Ram address calculated
A = Magic register value used

The value in DE should be copied into the Vector Block at VBOAL & VBOAH if you are using VBLANK to clear your graphic.

If you are using the EXPAND function of the Magic Register you must specify what colours the image is to appear, by placing the combined value into the 'Expansion port'.

Like so,

```
LD    A,0CH         ; COLOURS TO EXPAND TO (1100)
OUT   (XPAND),A     ; INTO EXPANSION PORT
```

The XPAND port takes the 1 and 0 bits of a monochrome image and maps them to the specified pairs on the port. The value above tells the port that 1 bits should be colour value 3 (11 binary) and that 0 bits should be color value 0 (00 binary)

This allows us not only to change the intended colour of the image by mapping the 1 bits to any non background value (01, 10, 11) but to also produce inverse images by making the 0 bits the non background value. You can also mix as you please.

VBLANK
(Vector BLANK)

This command zeros out any area indicated.

It first checks the Vector Block to see if the VBBLNK bit is set. This should have been done by VWRITR. If not set, operation is ignored. If set VBBLNK is cleared along with the area of screen indicated.

Calling Methods:

SYSSUK:

```
SYSSUK    VBLANK
DEFW      (vector block address)
DEFB      (X size in bytes *)
DEFB      (Y size)
```

SYSTEM:

```
LD        IX, (vector block address)
LD        D, (Y size)
LD        E, (X size in bytes *)
SYSTEM    VBLANK
```

* You should remember that any monochrome image expanded to the screen now occupies twice as many bytes horizontally. This means that a monochrome image with X size of 2 in the Pattern Block will need an X size of 4 to blank properly! Also VBLANK ignores the Magic Register and Shift bits. You must add 1 to the X width to blank the pixels shifted right from the saved address.

EG:

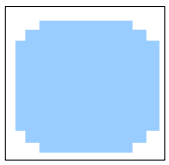

Image in
BLANK box.

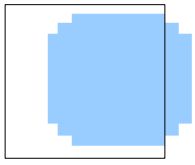

Image shifted 2 pixels right
BLANK box doesn't shift.

VECT
(VECTor)

This command calculates the next position of the graphic, and indicates if any actual (whole number) motion took place.

Calling Methods:

SYSSUK:

```
SYSSUK      VECT
DEFW (vector block address)
DEFW (Limit Table)
```

SYSTEM:

```
LD          HL, (Limit Table)
LD          IX, (vector block address)
SYSTEM      VECT
```

Returned values:

C = Time base used (number of times vectors added)
Z = True, if no whole number values changed. (no drawable motion)

Limit Table:

Not shown anywhere in *The Nutting Manual*.

This is a list of bytes indicating the boundaries of your moving graphic.

X LOWER LIMIT	LEFT EDGE
X UPPER LIMIT	RIGHT EDGE
Y LOWER LIMIT	TOP EDGE
Y UPPER LIMIT	BOTTOM EDGE

Remember to account for the width of your image. If we wanted our 5 pixel ball to bounce off the right edge of the screen, (presuming no offset) we must move X UPPER 5 pixels from the right. On a 160 screen that would be 155. If we don't do this the right of the ball will shoot off the right of the screen and wrap around to the left. Not very professional looking... :)

MAKTD
(joystick MaSK To Deltas)

This command uses the value returned by a Joystick and returns positive, negative or 0 values depending on the 'Delta' values we give it.

This allows us to drop or add the returned values into the Vector Block without further consideration for control.

Calling Methods:

SYSSUK:

```
        LD      B, (joystick mask)
        SYSSUK  MSKTD
        DEFW    (X Delta Positive value)
        DEFB    (Flop flag)
        DEFW    (Y Delta Positive value)
```

SYSTEM:

```
        LD      B, (joystick mask)
        LD      C, (Flop flag)
        LD      DE, (X Delta Positive value)
        LD      HL, (Y Delta Positive value)
        SYSTEM  MSKTD
```

Returned values:

DE = X Delta
HL = Y Delta

Note: B is not 'sucked' in and must be loaded manually or from a previous system call.

The Delta values entered are what we would like to use if the joystick is held Right (Left if Flopped) or Down. Negative values will be returned if the joystick is Left (Right if Flopped) or Up.

Flopped values will make more sense after reading the 'Flopping around' tutorial.

If we give it an X Delta of 1.0 for single pixel motion, we will get -1.0 for Left, 0.0 for middle and 1.0 for Right. Dropping these values directly into the X Delta bytes of the Vector Block, doing a VECT and VWRITR will move our graphic appropriately for direct motion control.

If we give it a Delta of 0.25 and added the resulting values to the Vector Block Deltas, we could simulate the inertial motion of a space ship or hover craft.

Now the BIG question, What IS the Joystick Mask!?

JOYSTICK MASK

This is actually the byte value returned by the joystick port. That is, reading the appropriate port for a joystick (10H for joystick 0) and placing the value here works. Alternatively, you can call the 'SENTRY' command, pick up the joystick changes through 'DOIT' and the value will already be in the B register.

See 'SENTRY Tutorial' for more detail on SENTRY and DOIT.

Putting it all together!

Now that we have an idea of what these commands do, lets put them to work.

Our first demo will be a simple bouncing ball. The assembly code looks like this:

```
;*************************
;* Set-up for Bally Cart *
;*************************
INCLUDE  HVGLIB.H               ; Bally Library

B1VEC EQU 4F00H                 ; VECTOR BLOCK FOR BALL TEST

         ORG    2000H           ; Start of Cart memory

         DEFB   55H             ; NORMAL (MENUED) CART 'SENTENAL'
         DEFW   MENUST          ; (START OF ONBOARD MENU)
         DEFW   LABEL           ; ADDRESS OF MENU TEXT
         DEFW   PROG1           ; WHERE TO GO IF SELECTED.

LABEL    DEFB   'BALL TEST',0   ; ZERO DELIMITED STRING


;*****************
;* Bouncing ball *
;*****************

PROG1    SYSSUK FILL            ; CLEAR SCREEN
         DEFW   4000H           ; START OF SCREEN RAM
         DEFW   95*40           ; 95 LINES (40 BYTES PER LINE)
         DEFB   0               ; ZERO VALUE

         LD     IX,B1VEC        ; BEGIN SETUP OF VECTOR BLOCK
         LD     (IX+VBMR),8     ; MAGIC: EXPAND
         LD     (IX+VBSTAT),192 ; STATUS: ACTIVE+BLANKING
         LD     (IX+VBTIMB),1   ; TIMES: 1 (NUMBER OF TIMES DELTA IS ADDED TO POSITION)
         LD     (IX+VBDXL),0    ; DELTA (DIRECTION)X LOW ORDER BYTE (FRACTIONS)
         LD     (IX+VBDXH),1    ; 1 PIXEL PER CALL MOTION (RIGHT)
         LD     (IX+VBXL),0     ; INITIAL POSITION LOW ORDER BYTE (FRACTIONS)
         LD     (IX+VBXH),0     ; ACTUAL X POSITION ON SCREEN
         LD     (IX+VBXCHK),3   ; X LIMIT CHECKING: ON + REVERSING (BOUNCE)
         LD     (IX+VBDYL),0    ; DELTA (DIRECTION)Y LOW ORDER BYTE (FRACTION)
         LD     (IX+VBDYH),1    ; 1 PIXEL PER CALL MOTION (DOWN)
         LD     (IX+VBYL),0     ; INITIAL POSSITION LOW ORDER BYTE (FRACTION)
         LD     (IX+VBYH),0     ; ACTUAL Y POSITION ON SCREEN
         LD     (IX+VBYCHK),3   ; Y LIMIT CHECKING: ON + REVERSING (BOUNCE)
         LD     (IX+VBOAL),0    ; OLD ADDRESS: CLEARED FOR NOW
```

```
        LD      (IX+VBOAH),0    ; END VECTOR BLOCK SETUP

        LD      A,0CH           ; COLOURS TO EXPAND TO
        OUT     (XPAND),A       ; INTO EXPANSION PORT

LOOP    SYSSUK VBLANK           ; ERASE BALL
        DEFW    B1VEC           ; VECTOR BLOCK
        DEFB    3               ; X SIZE IN BYTES PLUS 1
        DEFB    6               ; Y SIZE

        SYSSUK VWRITR           ; DRAW BALL
        DEFW    B1VEC           ; VECTOR BLOCK TO USE
        DEFW    BALL1           ; IMAGE TO DRAW

        LD      IX,B1VEC
        LD      (IX+VBOAL),E    ; COPY SCREEN ADDRESS TO VECTORBLOCK
        LD      (IX+VBOAH),D

        SYSSUK PAWS             ; PAUSE FOR
        DEFB    1               ; ONE CYCLE


        SYSSUK VECT             ; MOVE BALL
        DEFW    B1VEC           ; VECTOR BLOCK TO USE
        DEFW    BLMT            ; LIMITS TABLE TO USE

        LD      IX,B1VEC
        LD      (IX+VBTIMB),1   ; RE ACTIVATE?
        JR      LOOP            ; BACK TO DRAW BALL

;********************
;* BALL IMAGE BLOCK *
;********************

BALL1   DEFB    0               ; NO X OFFSET
        DEFB    0               ; NO Y OFFSET
        DEFB    1               ; 1 BYTE WIDE
        DEFB    6               ; 6 BYTES TALL
        DEFB    01111000B       ;
        DEFB    11011100B       ; IMAGE DATA
        DEFB    10111100B
        DEFB    10111100B
        DEFB    11111100B
        DEFB    01111000B

;***************
;* BALL LIMITS *
;***************

BLMT    DEFB    0     ; X LOWER LIMIT (LEFT EDGE)
        DEFB    153   ; X UPPER LIMIT (RIGHT EDGE)
                      ; 159-8 FOR 8 PIXEL WIDE BALL
        DEFB    0     ; Y LOWER LIMIT (TOP EDGE)
        DEFB    89    ; Y UPPER LIMIT (BOTTOM EDGE)
                      ; 95-6 FOR 6 PIXEL HIGH BALL
```

Bouncing Ball Take apart.

Lines 1-17
First we load the Bally Library, 'HVGLIB.H'.
This allows us to use the system commands by name.

Next is the location for the Vector Block. As the only RAM in the Bally/Astrocade IS screen ram, it has to hide there. 4F00H is near the bottom of the screen, and already hidden by the Bally Menu screen set-up.

In line 8 we declare where in memory this program is to reside. 2000H is the start of Cartridge ROM.
The rest of this section is the standard set-up for a Bally Cartridge using a menu.

Lines 18-46
First we clear the screen of the Bally Menu, using the FILL command.

Then we initialize our Vector Block.

Finally, we load the 'Expand port' with the colour pixel values for our Ball image.

Lines 47-51
Start of Main Loop!

OH, oh, we called VBLANK before we drew the Ball or got the screen address!!
Not to worry, the VBBLNK bit isn't set, so this will be ignored the first time through.

Note: It is always best to not erase your images until immediately before you re-draw them. This reduces flickering/blinking to the absolute minimum. If done right, it'll happen during the vblank of the TV and never be seen at all.

Lines 52-55
VWRITR draws our Ball image as specified by the Vector Block and Image Block.

Lines 56-59
Copies calculated screen address into Vector Block for use in VBLANK call.

Lines 60-63
Pauses execution for 1/60 of a second. (One screen draw on the TV)
It would be better to tie our draw routines into the vblank timing, but this is a quick kluge.

Lines 64-67
VECT calculates the next position of our Ball, using the Vector Block and the Ball Limit Table.

Lines 68-71
Reset Vector Block Timebase to 1 (it was decremented to 0 in VECT call)
Jump to start of loop.

Lines 72-86
Define Ball Image block.

Lines 87-96
Define Ball Limit Table.

Version 1.0 - 2-13-2004
- First Release (to A.T.)

Version 1.01 – May 7, 2004 (Editing by A.T.)
- Corrected spelling errors
- Removed tabs from source code; replaced with spaces
- Changed source code to courier font
- Changed tables so that all text is visible and centered
- Changed table/listing font sizes so that they wouldn't wrap
- Added Assembly listing of Ball example as an appendix

Version 1.02 – November 11, 2008 (Editing by Lance referencing comments by Richard Degler)
- Corrected missed spelling
- Added full colour ball example
- Changed 'BLANK' to ' VBBLNK' where appropriate
- Changed 'SENTENAL' to 'SENTRY' where appropriate
- Some pageing adjustments

Version 1.03 - November 23,2008 (Editing by Lance referencing comments by Richard Degler)
- Added 1 to VBLANK X size, with explanation
- Added HVGLIB names for Magic Register bits

Appendix 1: Ball Example Assembly Listing

```
   1:              ;*************************
   2:              ;* Set-up for Bally Cart *
   3:              ;*************************
   4:              ;
   5:              ; zmac -i -m -o exam1.bin -x exam1.lst exam1.asm
   6:              ;
   7:
   8:              INCLUDE  HVGLIB.H               ; Bally Library
**** HVGLIB.H ****
**** exam1.asm ****
   9:
  10:  4F00               B1VEC EQU 4F00H                 ; VECTOR BLOCK FOR BALL TEST
  11:
  12:  2000                       ORG    2000H           ; Start of Cart memory
  13:
  14:  2000  55                   DEFB   55H             ; NORMAL (MENUED) CART 'SENTENAL'
  15:  2001  1802                 DEFW   MENUST          ; (START OF ONBOARD MENU)
  16:  2003  0720                 DEFW   LABEL           ; ADDRESS OF MENU TEXT
  17:  2005  1120                 DEFW   PROG1           ; WHERE TO GO IF SELECTED.
  18:
  19:  2007  42414C4C     LABEL   DEFB   'BALL TEST',0   ; ZERO DELIMITED STRING
             20544553
             5400
  20:
  21:
  22:              ;*****************
  23:              ;* Bouncing ball *
  24:              ;*****************
  25:
  26:  2011               PROG1    SYSSUK FILL           ; CLEAR SCREEN
  26:  2011  FF                   RST    56
  26:  2012  1B                   DB     FILL+1
  26:  0000                       IF     FILL = INTPC
  26:                       ENDIF
  26:  2013                       ENDM
  26:
  27:  2013  0040                  DEFW   4000H           ; START OF SCREEN RAM
  28:  2015  D80E                  DEFW   95*40           ; 95 LINES (40 BYTES)
  29:  2017  00                    DEFB   0               ; ZERO VALUE
  30:
  31:  2018  DD21004F              LD     IX,B1VEC        ; BEGIN SETUP OF VECTOR BLOCK
  32:  201C  DD360008              LD     (IX+VBMR),8     ; MAGIC: EXPAND
  33:  2020  DD3601C0              LD     (IX+VBSTAT),192 ; STATUS: ACTIVE+BLANKING
  34:  2024  DD360201              LD     (IX+VBTIMB),1   ; TIMES: 1 (NUMBER OF TIMES DELTA IS
ADDED TO POSITION)
  35:  2028  DD360300              LD     (IX+VBDXL),0    ; DELTA (DIRECTION)X LOW ORDER BYTE
(FRACTIONS)
  36:  202C  DD360401              LD     (IX+VBDXH),1    ; 1 PIXEL PER CALL MOTION (RIGHT)
  37:  2030  DD360500              LD     (IX+VBXL),0     ; INITIAL POSITION LOW ORDER BYTE
(FRACTIONS)
  38:  2034  DD360600              LD     (IX+VBXH),0     ; ACTUAL X POSITION ON SCREEN
  39:  2038  DD360703              LD     (IX+VBXCHK),3   ; X LIMIT CHECKING: ON + REVERSING
(BOUNCE)
  40:  203C  DD360800              LD     (IX+VBDYL),0    ; DELTA (DIRECTION)Y LOW ORDER BYTE
(FRACTION)
  41:  2040  DD360901              LD     (IX+VBDYH),1    ; 1 PIXEL PER CALL MOTION (DOWN)
  42:  2044  DD360A00              LD     (IX+VBYL),0     ; INITIAL POSSITION LOW ORDER BYTE
(FRACTION)
  43:  2048  DD360B00              LD     (IX+VBYH),0     ; ACTUAL Y POSITION ON SCREEN
  44:  204C  DD360C03              LD     (IX+VBYCHK),3   ; Y LIMIT CHECKING: ON + REVERSING
(BOUNCE)
```

```
45:  2050  DD360D00              LD     (IX+VBOAL),0    ; OLD ADDRESS: CLEARED FOR NOW
46:  2054  DD360E00              LD     (IX+VBOAH),0    ; END VECTOR BLOCK SETUP
47:
48:  2058  3E0C                  LD     A,0CH           ; COLOURS TO EXPAND TO
49:  205A  D319                  OUT    (XPAND),A       ; INTO EXPANSION PORT
50:
51:  205C               LOOP     SYSSUK VBLANK          ; ERASE BALL
51:  205C  FF                    RST    56
51:  205D  29                    DB     VBLANK+1
51:  0000                        IF     VBLANK = INTPC
51:                       ENDIF
51:  205E                        ENDM
51:
52:  205E  004F                  DEFW   B1VEC           ; VECTOR BLOCK
53:  2060  03                    DEFB   3               ; X SIZE IN BYTES
54:  2061  06                    DEFB   6               ; Y SIZE
55:
56:  2062                        SYSSUK VWRITR          ; DRAW BALL
56:  2062  FF                    RST    56
56:  2063  1F                    DB     VWRITR+1
56:  0000                        IF     VWRITR = INTPC
56:                       ENDIF
56:  2064                        ENDM
56:
57:  2064  004F                  DEFW   B1VEC           ; VECTOR BLOCK TO USE
58:  2066  8520                  DEFW   BALL1           ; IMAGE TO DRAW
59:
60:  2068  DD21004F              LD     IX,B1VEC
61:  206C  DD730D                LD     (IX+VBOAL),E    ; COPY SCREEN ADDRESS TO VECTORBLOCK
62:  206F  DD720E                LD     (IX+VBOAH),D
63:
64:  2072                        SYSSUK PAWS            ; PAUSE FOR
64:  2072  FF                    RST    56
64:  2073  51                    DB     PAWS+1
64:  0000                        IF     PAWS = INTPC
64:                       ENDIF
64:  2074                        ENDM
64:
65:  2074  01                    DEFB   1               ; ONE CYCLE
66:
67:
68:  2075                        SYSSUK VECT            ; MOVE BALL
68:  2075  FF                    RST    56
68:  2076  3F                    DB     VECT+1
68:  0000                        IF     VECT = INTPC
68:                       ENDIF
68:  2077                        ENDM
68:
69:  2077  004F                  DEFW   B1VEC           ; VECTOR BLOCK TO USE
70:  2079  8F20                  DEFW   BLMT            ; LIMITS TABLE TO USE
71:
72:  207B  DD21004F              LD     IX,B1VEC
73:  207F  DD360201              LD     (IX+VBTIMB),1   ; RE ACTIVATE?
74:  2083  18D7                  JR     LOOP            ; BACK TO DRAW BALL
75:
76:                ;********************
77:                ;* BALL IMAGE BLOCK *
78:                ;********************
79:
80:  2085  00            BALL1   DEFB   0               ; NO X OFFSET
81:  2086  00                    DEFB   0               ; NO Y OFFSET
82:  2087  01                    DEFB   1               ; 1 BYTE WIDE
83:  2088  06                    DEFB   6               ; 6 BYTES TALL
84:  2089  78                    DEFB   01111000B       ;
85:  208A  DC                    DEFB   11011100B       ; IMAGE DATA
```

```
 86:  208B  BC                   DEFB   10111100B
 87:  208C  BC                   DEFB   10111100B
 88:  208D  FC                   DEFB   11111100B
 89:  208E  78                   DEFB   01111000B
 90:
 91:              ;***************
 92:              ;* BALL LIMITS *
 93:              ;***************
 94:
 95:  208F  00           BLMT    DEFB   0     ; X LOWER LIMIT (LEFT EDGE)
 96:  2090  99                   DEFB   153   ; X UPPER LIMIT (RIGHT EDGE)
 97:                                        ; 159-8 FOR 8 PIXEL WIDE BALL
 98:  2091  00                   DEFB   0     ; Y LOWER LIMIT (TOP EDGE)
 99:  2092  59                   DEFB   89    ; Y UPPER LIMIT (BOTTOM EDGE)
100:                                        ; 95-6 FOR 6 PIXEL HIGH BALL
```