There are a total of four players and four missiles. The four missiles may be grouped together and used as a 5th player. These objects are positioned horizontally by 8 horizontal position registers (HPOS (X)). These registers may be reloaded at any time by the processor, allowing an object to be replicated many times across a horizontal TV line.

The shape of a player-missile is determined by the data in its graphics register (GRAF (X)). Players have independent 8 bit graphics registers. The four missiles have 2 bit registers (located within one address). These registers may also be reloaded at any time by the processor, although they are usually changed during horizontal blank time. The data in each graphics register is placed on the display whenever the horizontal sync counter equals the corresponding horizontal position register. The same data will be displayed every line unless the graphic registers are reloaded with new data.

The player-missile graphic registers may be reloaded by the microprocessor (GRAF (X)), or automatically from memory with direct memory access (DMA) (see figure II.3). The programmer must place the object graphics in memory, write the player-missile base address (PMBASE), and enable player-missile DMA (DMACTL, GRACTL). The transfer of object graphics from memory to display is then fully automatic.

PMBASE specifies the most significant byte (MSB) of the address of the player-missile graphics. The location of the graphics for each object is determined by adding an offset to PMBASE *256 (decimal). The bytes between the base address and the missile data are not used by Antic, so they are available to the programmer.

Only the five most significant bits of PMBASE are used with single-line resolution and the six most significant bits are used with two-line resolution. This means that the location of the graphics in memory is restricted to certain page boundaries. Two-line resolution means that each byte of data is repeated for two lines. (see DMACTL, bit 4). 640 (decimal) bytes (5X128) are required for two-line resolution and 1280 bytes (5x256) for one-line resolution.

Each byte in the player graphics area represents eight pixels which are to be displayed on the corresponding line(s) of the TV screen. A 1 indicates that the player's color-lum is to be displayed in that pixel. The graphics may be anything, not just rectangles like the ones in figure II.3. The player graphics may fill the entire height of the screen or they may be only a couple of lines high if the rest of the display data is all 0's. Each byte in the missile display also represents eight pixels, two pixels for each missile. Each pixel may be 1, 2, or 4 color clocks, and is determined by the SIZE registers.

Playfield: Playfield is always generated by DMA. There are four playfields, each identified by its own color-lum register and collision detection. Playfield is generated by two different DMA techniques: memory map and character. Both methods provide lists of instructions in memory, independent of the player-missile generation.

Player-Missile Base Address (PMBASE) = MSB of address.
Resolution is controlled by bit 4 of DMACTL.

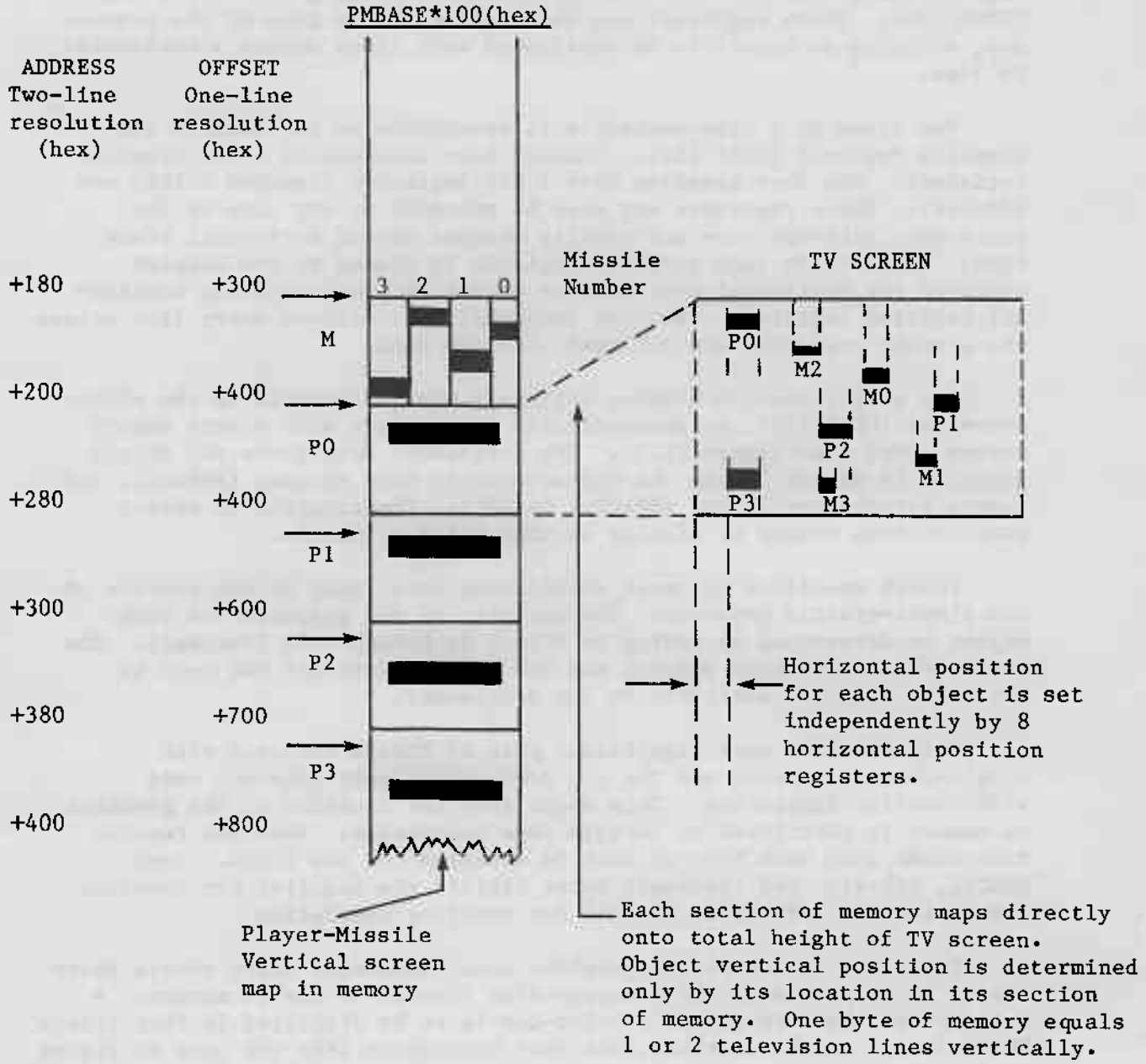


Figure II.2 P L A Y E R - M I S S I L E D M A

Unlike players and missiles, there are no horizontal position registers for playfield. Each player can only have one byte of display per line. Playfield, on the other hand, may require up to 48 bytes per line because it can fill the entire width of the screen.

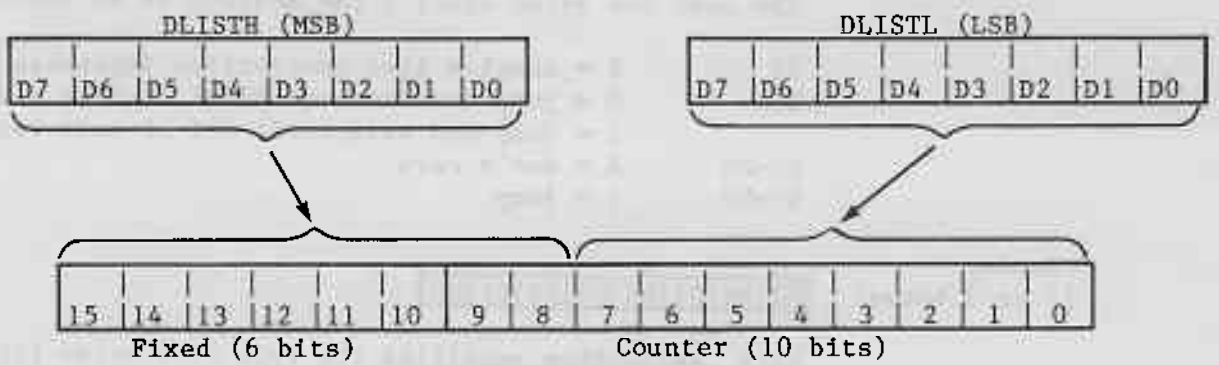
There are three different playfield widths: narrow (128 color clocks), standard (160 color clocks), and wide (192 color clocks). The width is selected by storing into DMACTL. The advantage of a narrower width is that less RAM is required and fewer machine cycles are stolen for DMA. The OS graphics modes use the standard screen width.

Display List: The display list is a sequence of display instructions stored in memory. These instructions are either one (1) byte or three (3) bytes long. The display list can be considered a display program, and the Display List Counter that fetches these instructions can be thought of as a display program counter. (10 bit counter plus 6 bit base register.)

The display list counter can be initialized by writing to DLISTH and DLISTL. (or OS shadow registers SDLSTH and SDLSTL). Once initialized this counter value is used to address the display list, fetch the instruction, display one (1) to sixteen (16) lines of data on the TV screen, increment the Display List Counter, fetch the next display instruction, and so on automatically without microprocessor control (see DLISTL and DLISTH). DLISTL and DLISTH should be altered only during vertical blank or when DMA is disabled (see DMACTL).

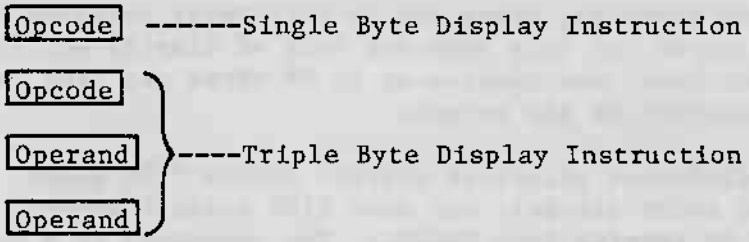
Each instruction defines the type (alpha character or memory map) and the resolution (size of bits on screen) and the location of data in memory to be displayed for a group (1 to 16) of lines. Each group of lines is called a display block.

THE DISPLAY LIST CANNOT CROSS A 1K BYTE MEMORY BOUNDARY UNLESS A JUMP INSTRUCTION IS USED.



DISPLAY LIST COUNTER

Display Instruction Format: Each instruction consists of either an opcode only, or of an opcode followed by two (2) bytes of operand.



The opcode is always fetched first and placed in the Instruction Register. This opcode defines the type of instruction (1 or 3 bytes) and will cause two more bytes to be fetched if needed. If fetched, these next two (2) bytes will be placed in the Memory Scan Counter, or in the Display List Counter (if the instruction is a Jump).

Display Instruction Register (IR): This register is loaded with the opcode of the current display list instruction. It cannot be accessed directly by the programmer. There are three basic types of display list instructions: blank, jump, and display.

Blank
(1-byte)

D7	D6	D5	D4	0	0	0	0

This instruction is used to create 1 to 8 blank lines on the display (blackground color).

D7	1	= display list instruction interrupt
D6 - D4	0-7	= 1-8 blank lines
D3 - D0	0	= blank

Jump
(3-bytes)

D7	D6	X	X	0	0	0	1

This instruction is used to reload the Display List Counter. The next two bytes specify the address to be loaded (LSB first).

D7	1	= display list instruction interrupt
D6	0	= jump (creates one blank line on display)
	1	= jump and wait until end of next vertical blank
D5-D4	X	= don't care
D3-D0	1	= jump

Display
(1 or 3 bytes)

D7	D6	D5	D4	D3	D2	D1	D0

This instruction specifies the type of display for the next display block.

D7	1	= display list instruction interrupt
D6	0	= 1 byte instruction
	1	= 3 byte instruction (reload Memory Scan Counter using address in next two bytes, LSB first).
D5	1	= vertical scroll enable
D4	1	= horizontal scroll enable
D3-D0	2-F	= display mode (memory or character map - see following pages).

HSCROL		XX		XX		XX		XX		XX		XX		XX		XX		Horizontal Scrolling
VSCROL			XX	XX			XX	XX			XX	XX			XX	XX		Vertical Scrolling
LD MEM SCAN					XX	XX	XX	XX					XX	XX	XX	XX		Load memory scan (3 byte)
INST INTERRUPT									XX	XX	XX	XX	XX	XX	XX	XX		Display instruction interrupt
BLK 1	00								80									Blank 1 line
" 2	10								90									Blank 2 lines
" 3-7																		Blank 3 thru 7 lines
" 8	70								F0									Blank 8 lines
JMP	01								81									Jump (3 byte instruction)
JVB	41								C1									Jump & wait for Vert. Blank (also 3 byte)
CHR (40,2,8)	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2		
" (40,2,10)	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3		
" (40,4,8)	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4		Character Mode Instructions
" (40,4,16)	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5		
" (20,5,8)	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6		
" (20,5,16)	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7		
MAP (40,4,8)	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8		
" (80,2,4)	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9		
" (80,4,4)	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA		
" (160,2,2)	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB		Memory Map Mode Instructions
" (160,2,1)	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC		
" (160,4,2)	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD		
" (160,4,1)	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE		
" (320,2,1)	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF		

Number of TV lines per cell
Number of Colors (Background + Playfield types)
Number of Horizontal cells (standard width screen)

Figure II.3 DISPLAY INSTRUCTION OPCODES

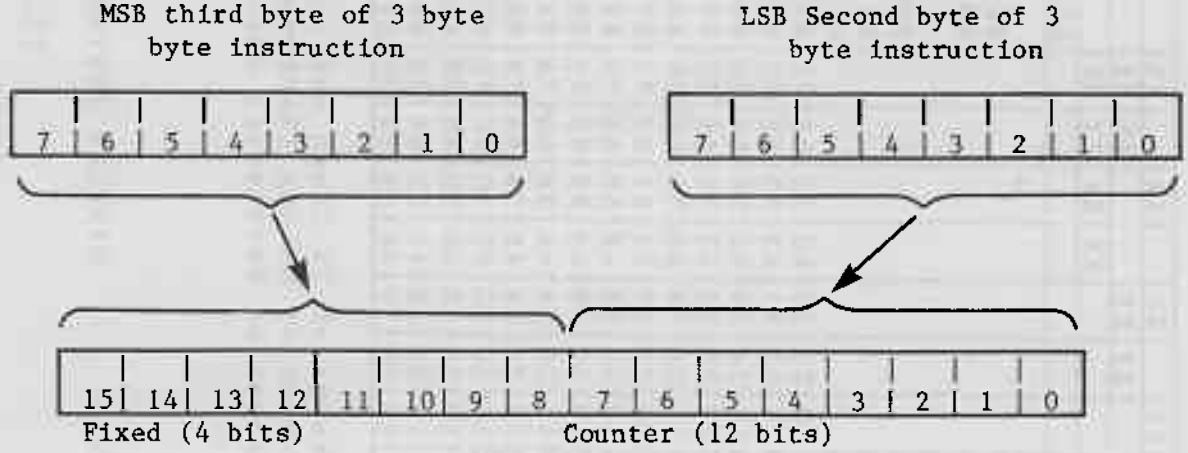
Bit 7 of a display list instruction can be set to create a display list interrupt if bit 7 of NMIEN is set. The display list interrupt code can change the colors or graphics during the middle of the TV display. The type of interrupt is determined by checking NMIST. NMIRES clears NMIST. The current OS will vector through VDSLST (Hex 200 and 201) to the user's display list interrupt routine. See the OS manual for programming details.

Bits 5 and 4 of a display type of display list instructions are used to enable vertical and horizontal scrolling. The amount of scrolling depends on the values in the VSCROL and HSCROL registers (to be described later).

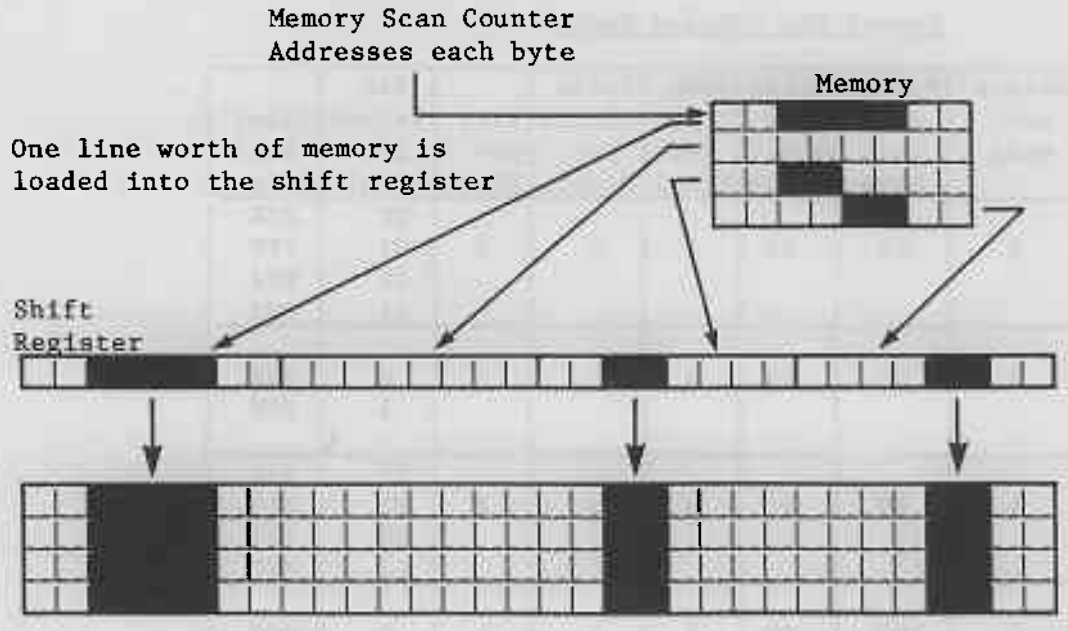
Memory Scan Counter: This counter is not directly accessible by the programmer. It is loaded with the value in the last 2 bytes of a 3 byte (non-Jump) instruction.

This counter points to the location (address) in memory of data to be directly displayed (memory map display) or to the location of character name strings to be indirectly displayed (character display).

A single byte instruction does not reload this counter. This implies a continuation in memory of data to be displayed from that displayed by the previous instruction. Since this counter really consists of 4 bits of register and 12 of actual counter, a continuous memory block cannot cross 4K byte memory boundaries, unless the counter is repositioned with a 3 byte Load Memory Scan Counter instruction.



Memory Map Display Instructions: Data in memory (addressed by the Memory Scan Counter) is displayed directly when executing a memory (bit) map display instruction. As data is being displayed it is also stored in a shift register so that it can be redisplayed for as many TV lines as required by the instruction.



Shift register data is displayed for four TV scan lines in this example.

In Instruction Register (IR) display modes 8 through F, one or two bits of memory are used to specify what is to be displayed on each pixel of the screen. Pixel sizes range from 1/2 clock by 1 TV line to 4 clocks by 8 TV lines. The OS and BASIC support most of these graphics modes (BASIC GRAPHICS command). Two modes, C and E, are not supported by the OS. These modes have rectangular pixels, which are approximately twice as wide as they are high.

In IR mode F, only one color (COLPF2) can be displayed. Two different luminances are available. If a bit is a zero, then the luminance of the corresponding pixel comes from COLPF2. If the bit is a one, them the luminance is determined by the contents of COLPF1 (abbreviated to PF1).

In IR modes 9,B, and C, two different colors can be displayed. A zero indicates background color and a one indicates PF0 color. The difference between the various modes is in the size of the pixels.

In IR modes 8,A,D, and E, two bits are used to specify the color of each pixel. This allows four different colors to be displayed. However, only four pixels can be packed into each byte, instead of eight as in the previous modes. The bit assignments are shown below.

SHIFT REGISTER	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	2 bits form one pixel															

Memory Map Display Modes

OS and BASIC Modes	Inst. Reg. HEX	Colors per Mode	Pixels per Std. Line	Bytes per Std. Line	Scan Lines per Pixel	Color Clocks per Pixel	Bits per Pixel	Bit Values in Pixel	Color Reg. Select
3	8	4	40	10	8	4	2	00 01 10 11	BAK PF0 PF1 PF2
4	9	2	80	10	4	2	1	0 1	BAK PF0
5	A	4	80	20	4	2	2	00 01 10 11	BAK PF0 PF1 PF2
6	B	2	160	20	2	1	1	0 1	BAK PF0
-	C	2	160	20	1	1	1	0 1	BAK PF0
7	D	4	160	40	2	1	2	00 01 10 11	BAK PF0 PF1 PF2
-	E	4	160	40	1	1	2	00 01 10 11	BAK PF0 PF1 PF2
8	F	1 ½	320	40	1	½	1	0 1	PF2 PF1 (LUM)

Character Display Instructions: The first step in using the character map mode is to create a character set in memory (or the built-in OS character set at hex E000 may be used). The character set contains eight bytes of data for the graphics for each character. The meaning of the data depends on the mode. The character set can contain 64 or 128 characters, also depending on the mode. The MSB (Most Significant Byte) of the address of the character set is stored in CHBASE (or the OS Shadow CHBAS). Only the most significant six or seven bits of CHBAS are used (see CHBASE description in section III). The other one or two bits and the LSB of the address are assumed to be zero, so the character set must start at an acceptable page boundary.

The next step is to set up the display list for the desired mode. Then the actual display is set up. This consists of a string of character names or codes. Each name takes one byte. The last 6 or 7 bits of the name selects a character. For a 64 character set, the name would range from 0 through 63 (decimal). For a 128 character set, the range would be 0 through 127 (decimal). The upper one or two bits of the name byte are used to specify the color or other special information, depending on the mode.

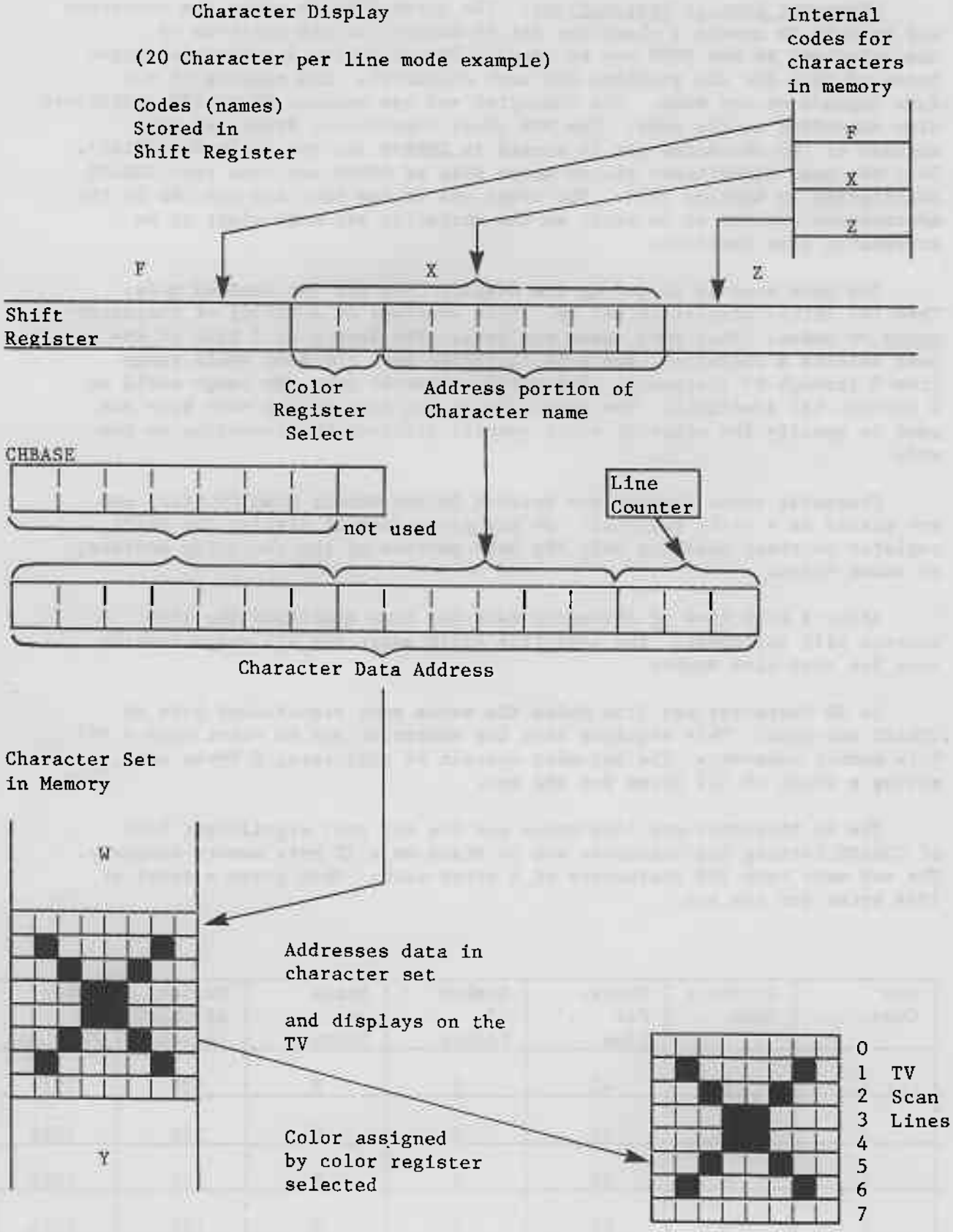
Character names (codes) are fetched by the memory scan counter, and are placed in a shift register. On any given line of display the shift register rotates, changing only the name portion of the character address, as shown below.

After a full line of character data has been displayed the line counter will increment. The next line again addresses all characters by name for that line number.

In 20 character per line modes the seven most significant bits of CHBASE are used. This requires that the character set to start upon a 512 byte memory boundary. The set must contain 64 charcters, 8 bytes each, giving a total of 512 bytes for the set.

The 40 character per line modes use the six most significant bits of CHBASE,forcing the character set to start on a 1K byte memory boundary. The set must have 128 characters of 8 bytes each. This gives a total of 1024 bytes for the set.

Hex Code	Graphics Mode	Chars. Per Line	Number of Colors	Bytes per Char.	Number of Char. in set	Bytes in Char Set
2	0	40	2	8	128	1024
3	-	40	2	8	128	1024
4	-	40	4	8	128	1024
5	-	40	4	8	128	1024
6	1	20	5	8	64	512
7	2	20	5	8	64	512


Character Display
(20 Character per line mode example)
Codes (names)
Stored in
Shift Register
Internal
codes for
characters
in memory
F
X
Z
F
X
Z
Shift
Register
Color
Register
Select
Address portion of
Character name
CHBASE
not used
Line
Counter
Character Data Address
Character Set
in Memory
W
Y
Addresses data in
character set
and displays on the
TV
Color assigned
by color register
selected
0
1
2
3
4
5
6
7
TV
Scan
Lines

There are six charcter map modes, IR modes 2 through 7. Modes 2,6 and 7 are supported by the OS and BASIC (GRAPHICS 0,1 and 2).

In IR modes 6 and 7, the upper two bits of each character name select one of four playfield colors. For each data bit that contains a one, the selected playfield color is displayed. For each zero data bit, the background color is displayed. The four character colors plus the background color gives a total of five different colors. the mode 6 characters are eight lines high and the mode 7 characters are sixteen lines high (each data byte is displayed for two lines).

In IR modes 4 and 5, each character is only four pixels wide instead of eight (as in the other modes). Two bits per pixel of data are used to select one of three playfield colors, or background. Seven name bits are used to select the character. If the most significant name bit is a zero then data of 10 (binary) selects PF1. If the name bit 7 is one, then data bits of 10 select PF2. This makes it possible to display two characters with different colors, using the same data but different name bytes.

In IR modes 2 and 3, each pixel is half of a color clock in width. This makes it possible to have forty eight-pixel-wide characters in a standard width line. These modes are similar to memory mode F in that two luminances can be displayed, but only one color is available at a time. In IR mode 3, each character is 10 lines high. This makes it possible to define lower case characters with descenders. The last fourth of the character set (name bits 5 and 6 equal to one) is lowered. The hardware takes the first two data bytes and moves them to the bottom of the character, displaying two blank lines at the top of the character (see next page).

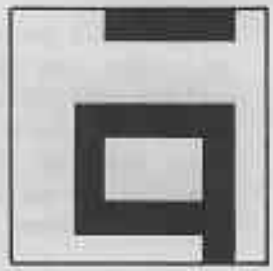
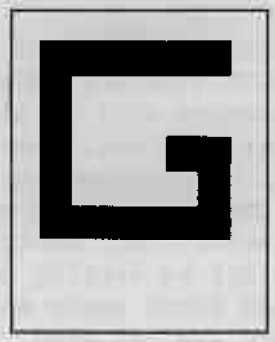
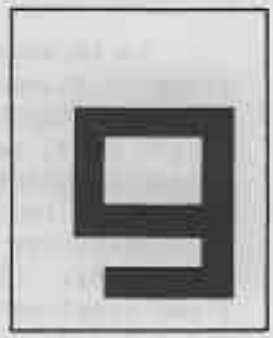
In IR modes 2 and 3, bit 7 of the character name is used for inverse video or blanking. This is controlled by CHACTL (Character Control). If bit 2 of CHACTL is a one then all of the characters will be displayed upside down, regardless of mode. If CHACTL bit 1 is set, then each character which has bit 7 of its name set will be displayed in inverse video (the luminances will be reversed). If CHACTL bit 0 is set, then each character which has bit 7 set will be blanked (only background wil be displayed). Characters can be blinked on and off by setting name bit 7 to 1 and toggling CHACTL bit 0. Inverse video and blank apply only to IR modes 2 and 3. If both inverse video and blank are set then the character will appear as an inverse video blank character (solid square).

Hardware Collision Detection: 60 bits of collision register are provided to detect and store overlap (hits) between players, missiles and playfield. These collisions can be read by the microprocessor from addresses D000 through D00F. There are no bits for missile to missile collisions.

16 bits for Missile to Playfield
16 bits for Player to Playfield
16 bits for Missile to Player
12 bits for Player to Player (P0 to P0 always reads as zero, etc.)

The 1/2 clock memory map mode (IR code 1111) and the 1/2 clock Character mode (IR codes 0011 and 0010) are both playfield type 2 collisions and will be stored in bit 2 of the playfield collision registers.

IR Mode 3-Upper and Lower Case

	Upper Case	Lower Case
Data		
Actual Display		

Character Map Display Modes

OS and BASIC Modes	Inst. Reg. HEX	Colors per Mode	Chars. per Std. Line	Scan Lines per Char.	Color Clocks per Pixel	Data Bits per Pixel	Color Select Bits In Name	Bit Values in Data	Color Reg. Select
0	2	1½	40	8	½	1	-	0 1	PF2 PF1 (LUM)
-	3	1½	40	10	½	1	-	0 1	PF2 PF1 (LUM)
-	4	5	40	8	1	2	Bit 7 = 0 Bit 7 = 1	00 01 10 11 11	BAK PF0 PF1 PF2 PF3
-	5	5	40	16	1	2	Bit 7 = 0 Bit 7 = 1	00 01 10 11 11	BAK PF0 PF1 PF2 PF3
1	6	5	20	8	1	1	- 00 01 10 11	0 1 1 1 1	BAK PF0 PF1 PF2 PF3
2	7	5	20	16	1	1	- 00 01 10 11	0 1 1 1 1	BAK PF0 PF1 PF2 PF3

Vertical and Horizontal Fine Scrolling: Playfield objects are difficult to move smoothly. Memory map playfield can be moved by rewriting sections of memory. However, this is extremely time-consuming if large sections of the screen must be moved smoothly. Character playfield objects can be moved easily in a jerky fashion by changing the memory scan counter. However, this results in a large position jump from one character position to another, not a smooth motion. For this reason hardware registers (VSCROL and HSCROL) and counters are provided to allow smooth horizontal or vertical motion, up to one character width horizontally and up to one character height vertically. After this much smooth motion has been done by increasing the value in these registers, memory is rewritten or the memory scan counter is modified and smooth motion is resumed for another character distance.

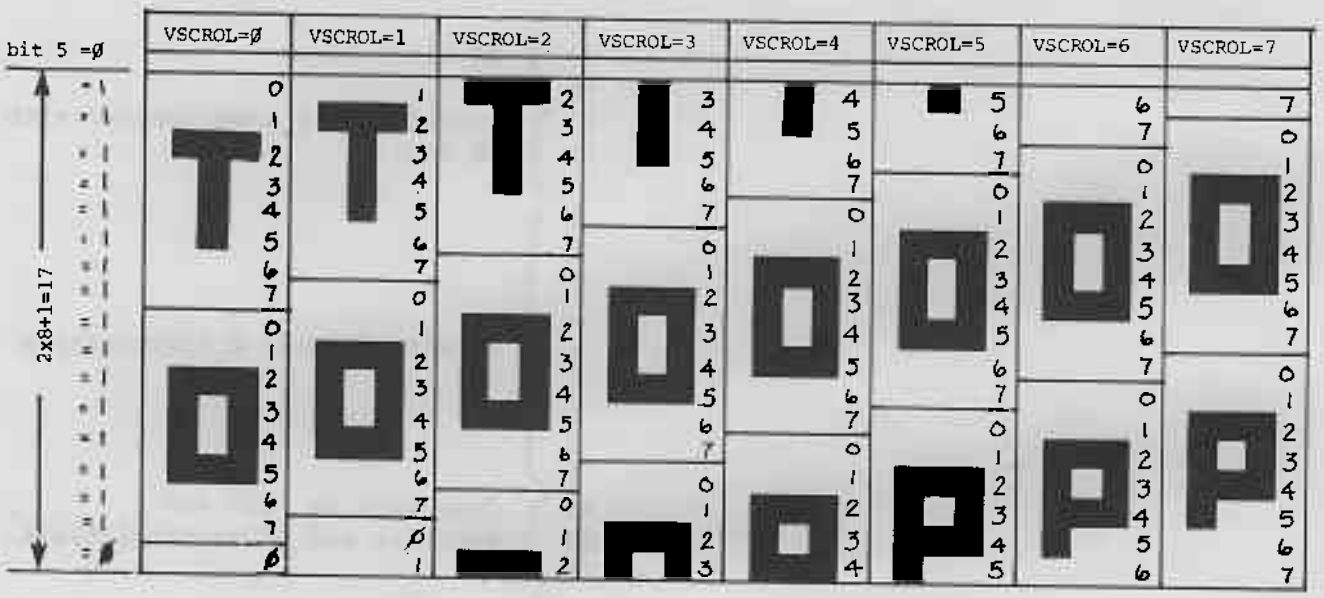
Vertical Scrolling: A zone of playfield on the screen can be scrolled upward by using VSCROL and bit 5 of the display list instruction. The display blocks at the upper and lower boundaries of the zone must have a variable vertical size. In particular, the first display block within that zone must be shortened from the top, and the last display block must be shortened from the bottom (i.e. not all of the top and bottom blocks will be displayed).

The vertical dimension of each display block is controlled by a 4 bit counter within the ANTIC, called the 'Delta Counter' (DCTR). Without vertical scrolling, it starts at 0 on the first line, and counts up to a standard value, determined by the current display instruction. (Ex: for upper and lower case text display, the end value is 9. For 5 color character displays, it is 7 or 15.)

If bit 5 of the instruction remains unchanged between consecutive display blocks, then the second block is displayed in the normal fashion. If bit 5 of the instruction goes from 1 to 0 between two consecutive display blocks, the second block will start with Delta = 0, as usual, but will count up until delta=VSCROL, instead of the standard value. This shortens that display block from the bottom.

To define a vertically scrolled zone, the most direct method is to set bit 5 to 1 in the first display instruction for that zone, and in all consecutive blocks but the last one. If the VSCROL register is not rewrittren on the fly, this results in a total scrolled zone that has a constant number of lines (provided that the VSCROL value does not exceed the standard individual block size). If N is the standard block size, the top block will be N-VSCROL lines (N > VSCROL), and the last block will be VSCROL + 1 lines: (N-VSCROL) + (VSCROL + 1) = N + 1. Shown on the following page is an example of a scrolled zone, top block, for 8 VSCROL values for N = 8.

Horizontal scrolling is described under HSCROL in section III.



Simple Display List Example: BASIC starts out in OS graphics mode 0 which displays 40 characters across by 24 rows. This is IR mode 2 with a standard screen width. The OS sets up the display list near the top of RAM with room for the character names at the top of RAM. On a 32 K-byte machine, the display list would start at hex 7C20. The next three bytes are hex 70's to create 24 blank lines. The next byte is a hex 42. The 4 tells the hardware to reload the memory scan counter with the following address (7C40). This is the address of the data to be displayed. The 2 tells the hardware to display one line of IR mode 2 characters. The next 23 bytes specify 23 more lines of mode 2 characters. Hex 41 is the code for jumping and waiting until the end of the next vertical blank. The address to jump to is 7C20, the start of the display list. The next 960 bytes are the list of characters to be displayed, 40 bytes per line. The OS must set up the display list pointer (DLISTH and DLISTL) to the starting address of the display list (7C20). It also sets CHBASE to the MSB of the address of the character set (E0).

This is a simple example because only one mode is used and the memory scan counter is only loaded at one point. It is possible to have different modes on different lines, change character sets and colors, etc., as shown in the example in Section IV.

OS Mode 0 Display List (40 chars x 24 lines)

Address (hex)	Data (hex)	
7C20	70	24 blank lines
	70	
	70	
	42	reload memory scan counter with 7C40, IR mode 2
	40	
	7C	
	2	23 more IR mode 2 instructions
	2	
	2	
	.	
	.	
	.	
	2	
	2	
	2	
	41	Jump back to 7C20 and wait for end of vertical blank.
	20	
	7C	
7C40		960 bytes of display data (character names)

Cycle Counting: As explained previously, the ATARI 800 6502 microprocessor runs at a rate of 114 machine cycles per TV line (1.79 MHZ). There are 262 lines per TV frame and 60 frames per second on the NTSC (US) system. (The PAL (Europeon) system is different. See the section on NTSC vs. PAL.)

Each machine cycle is equivalent in length to 2 color clocks. There are 228 color clocks on a TV line. The highest resolution graphics modes have a pixel size of 1/2 color clock by 1 TV line. Horizontal blank takes 40 machine cycles. This is when the beam returns to the left edge of the screen in preparation for displaying the next TV line. A wait for Sync (WSYNC) instruction stops the 6502. The processor is restarted exactly 7 machine cycles before the beginning of the next TV line. The program can thus change graphics or colors during horizontal blank in preparation for the next line.

The ANTIC chip steals cycles from the 6502 in order to do memory refresh and fetch graphics data when needed. The general rule to remember is that each byte fetched from memory requires one machine cycle. If a display list memory map instruction extends over several lines then the data is only fetched on the first line. Memory refresh takes 9 cycles out of every line, unless pre-empted by a high-resolution graphics mode. Memory refresh continues during vertical blank.

Missile DMA takes one cycle per line in the one-line resolution mode and one cycle every other line in the two line resolution mode. Missile DMA can be enabled without doing player DMA. However, if player DMA is enabled then missile DMA will also be done (see DMACTL, GRACTL bits). Player DMA requires 4 cycles every one or two lines, depending on the resolution used.

Each fetch of a display list byte takes one cycle, so three cycles are required for a three byte instruction.

Player/missile and display list instruction fetch DMA take place during horizontal blank, if they are required for the next line.

In memory map modes, the graphics data is fetched as needed throughout the first line of the display list instruction, then saved by ANTIC for use in succeeding lines. In character modes, the character codes are fetched during the first line of each row of characters, along with the graphics data needed for that line. On the next lines, only the graphics data is fetched, since ANTIC remembers the character codes.

In the 40 x 24 character mode, with a standard screen width, most of the cycles during the top line of each row of characters are requried to fetch the character codes and data, so there is only time for one memory refresh cycle instead of the usual nine. Less DMA is required with a narrow screen width so two memory refresh cycles would occur in this case.

The memory refresh is done fast enough to make up for the lost cycles in the high resolution modes. Once memory refresh starts on a line, it occurs every four cycles unless pre-empted by DMA.

All interrupts reach the 6502 near the end of horizontal blank. With standard or narrow screen widths, refresh DMA starts after the end of horizontal blank.

The time at which ANTIC does cycle stealing is deterministic, but depends on the graphics mode, screen width and whether or not horizontal scrolling is enabled. Horizontal scrolling requires extra graphics data: see HSCROL.

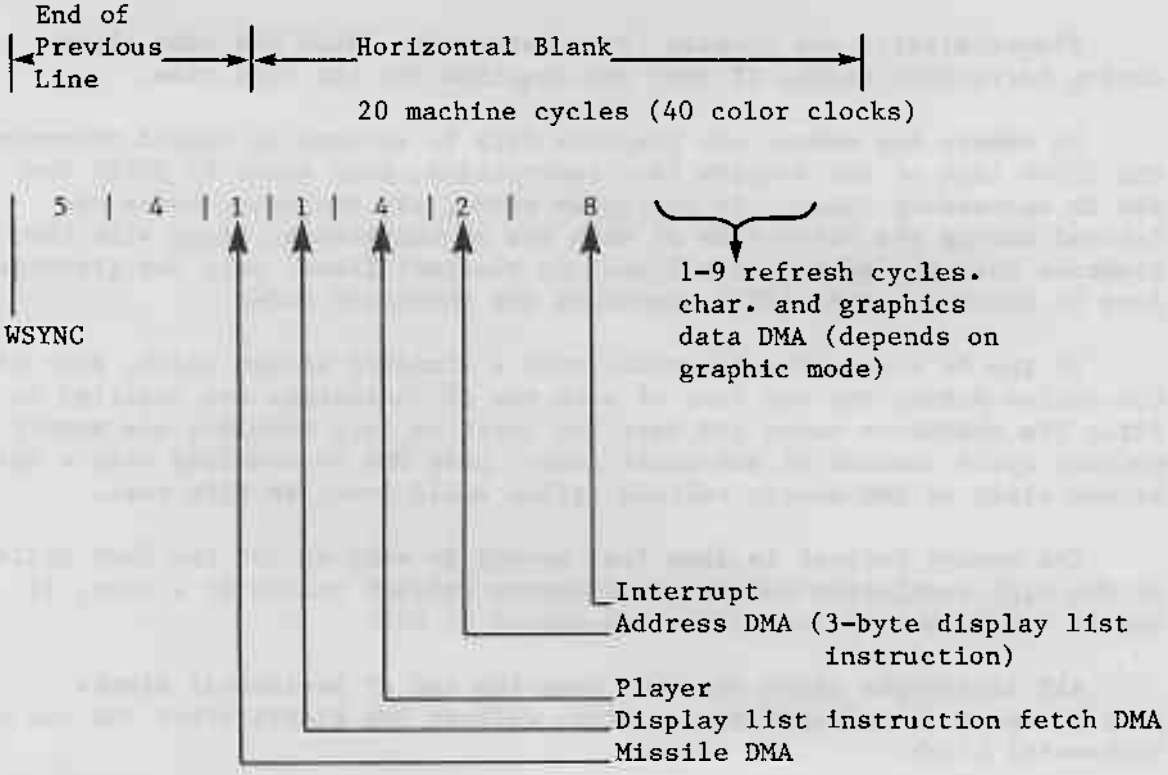
ANTIC does horizontal scrolling of an even number of color clocks by delaying the time at which it DMA's the data. To do an odd number of color clocks (which involves half of a machine cycle), ANTIC has a one color clock internal delay.

Theoretically, it is possible to write a program which changes graphics or colors "on the fly", i.e. during the middle of a TV line. However, with all the DMA going on, the cycle counting gets to be quite complicated, and is beyond the scope of this manual.

There are a number of delays associated with the display of graphics. These occur in the ANTIC and the CTIA. The ANTIC sends data to the CITA which adds in the color information. Thus the timing for changing colors on the fly is different from that for changing graphics on the fly.

Horizontal Blank DMA Timing

When DMA is enabled, cycles are stolen at the times shown below.



Cycle Counting Example: This example uses the 40 character by 24 line display list given on page II.24. This display list is 32 bytes long so display list DMA takes 32 machine cycles. It takes 960 cycles to DMA the characters and 8*960 to DMA the character data. The refresh DMA takes 9 cycles for each of 262 lines, except for the 24 lines where the characters are read, where only 1 refresh cycle occurs.

DMA description	Machine cycles	
display list		32
characters	40x24	= 960
character data	960x8	=7680
refresh	262x9-24x8	=2166
total		10838

Thus the total DMA per frame is 10838 machine cycles. One frame is 262 lines with 114 machine cycles per line for a total of 29868 machine cycles per frame. Thus 36% of each frame is required for DMA in OS graphics mode 0.

NTSC vs. PAL Systems: There are two versions of the ATARI 800: the NTSC (United States T.V. standard) and PAL (one of the European T.V. standards). The PAL system has been designed so that most programs will run without being modified. However, some differences may be noticeable. There is a hardware register (PAL) which a program can read to determine which type of system it is running on and adjust accordingly.

The PAL T.V. has a slower frame rate (50 Hz. instead of 60 Hz.) so games will be slower unless an adjustment is made. PAL has more T.V. lines per frame (312 instead of 262). The Atari 800 hardware compensates for this by adding extra lines at the beginning of vertical blank. Display lists do not have to be altered. However, their actual vertical height will be shorter. PAL ATARI 800 colors are similar to NTSC because of a hardware modification.

B. POKEY

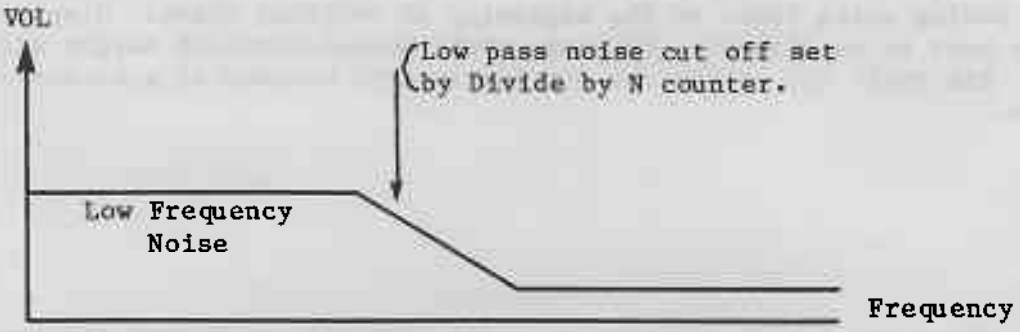
Audio: There are 4 semi-independent audio channels, each with its own frequency, noise, and volume control. Each has an 8 bit "divide by N" frequency divider, controlled by an 8 bit register (AUDFX). (See audio-serial port block diagram.) Each channel also has an 8 bit control register (AUDCX) which selects the noise (poly counter) content, and the volume.

Frequency Dividers: All 4 frequency dividers can be clocked simultaneously from 64 KHZ or 15 KHZ. (AUDCTL bit 0). Frequency dividers 1 and 3 can alternately be clocked from 1.79 MHZ (AUDCTL bits 6 and 5). Dividers 2 and 4 can alternately be clocked with the output of dividers 1 and 3 (AUDCTL bits 4 and 3). This allows the following options: 4 channels of 8 bits resolution, 2 channels of 16 bit resolution, or 1 channel of 16 bit and 2 channels of 8 bit.

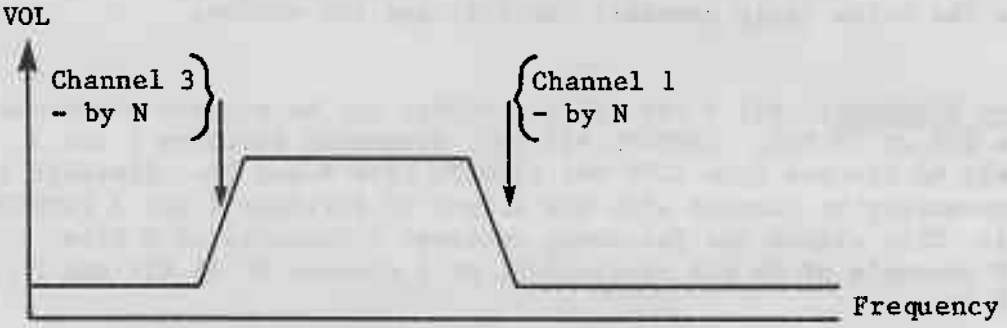
Poly Noise Counters: There are 3 polynomial counters (17 bit, 5 bit and 4 bit) used to generate random noise. The 17 bit poly counter can be reduced to 9 bits (AUDCTL bit 7). These counters are all clocked by 1.79 MHZ. Their outputs, however, can be sampled independently by the four audio channels at a rate determined by each channel's frequency divider. Thus each channel appears to contain separate poly counters (3 types) clocked at its own frequency. This poly counter noise sampling is controlled by bits 5,6 and 7 of each AUDCX register. Because the poly counters are sampled by the "divide by N" frequency divider, the output obviously cannot change faster than the sampling rate. In these modes (poly noise outputted) the dividers are therefore acting as "low pass" filter clocks, allowing only the low frequency noise to pass.

The output of the noise control circuit described above consists of pure tones (square wave type), or polynomial counter noise at a maximum frequency set by the "divide by N" counter (low pass clock). This output can be routed through a high pass filter if desired (AUDCTL bits 1 and 2).

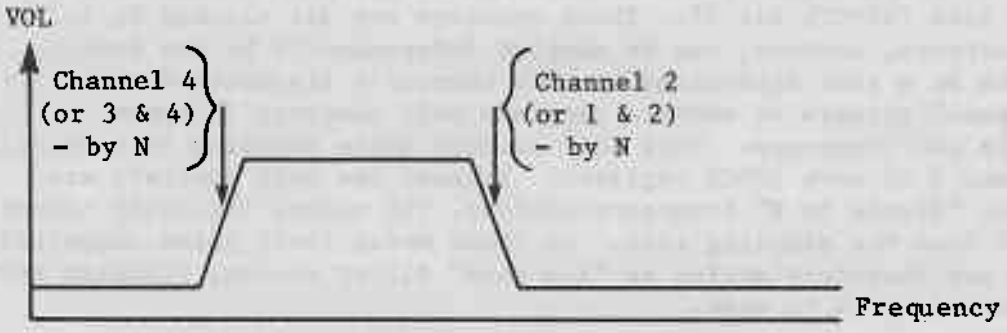
Audio Noise Filters:



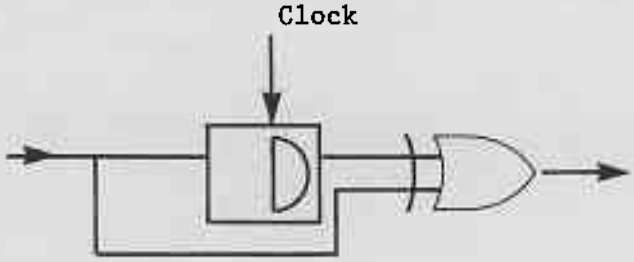
Any channel noise output (without high pass filter)



Channel 1 output (with high pass filter)



Channel 2 output (with high pass filter)



High Pass Filters: The high pass filter consists of a "D" flip flop and an exclusive-OR Gate. The noise control circuit output is sampled by this flip flop at a rate set by the "High Pass" clock. The input and output of the Flip Flop pass through the exclusive-OR Gate. If the flip flop input is changing much faster than the clock rate, the signal will pass easily through the exclusive-OR Gate. However, if it is lower than the clock rate, the flip flop output will tend to follow the input and the two exclusive-OR Gate inputs will mostly be identical (11 or 00) giving very little output. This gives the effect of a crude high pass filter, passing noise whose minimum frequency is set by the high pass clock rate. Only channels 1 and 2 have such a high pass filter. The high pass clock for channel 1 comes from the channel 3 divider. The high pass clock for channel 2 comes from the channel 4 divider. This filter is included only if bit 1 or 2 of AUDCTL is true.

Volume Control: A volume control circuit is placed at the output of each channel. This is a crude 4 bit digital to analog converter that allows selection of one of 16 possible output current levels for a logic true audio input. A logic zero audio input to this volume circuit always gives an open circuit (zero current) output. The volume selection is controlled by bits 0 thru 3 of AUDCX. "Volume Control only" mode can be invoked by forcing this circuit's audio input true with bit 4 of AUDCX. In this mode the dividers, noise counters, and filter circuits are all disconnected from the channel output. Only the volume control bits (0 to 8 of AUDCX) determine the channel output current.

The audio output of any channel can be completely turned off by writing zero to the volume control bits of AUDCX. All ones gives maximum volume.

C. SERIAL PORT

The serial port consists of a serial data output (transmission) line, a serial data input (receiver) line, a serial output clock line, a bi-directional serial data clock line, and other miscellaneous control lines described in the Operating System Manual. Data is transmitted and received as 8 bits of serial data preceded by a logic zero start bit, and succeeded by a logic true stop bit. Input and output clocks are equal to the baud (bit) rate, not 16 times baud rate. Transmitted data changes when the output clock goes true. Received data is sampled when the input clock goes to zero.

Serial Output: The transmission sequence begins when the processor writes 8 bits of parallel data into the serial output register (SEROUT)(see audio and serial port block diagram). When any previous data byte transmission is finished the hardware will automatically transfer new data from (SEROUT) to the output shift register, interrupt the processor to indicate an empty (SEROUT) register (ready to be reloaded with the next byte of data), and automatically serially transmit the shift register contents with start-stop bits attached. If the processor responds to the interrupt, and reloads SEROUT before the shift register is completely transmitted, the serial transmission will be smooth and continuous.

Output data is normally transmitted as logic levels (+4V=true 0V=False). Data can also be transmitted as two tone information. This mode is selected by bit 3 of SKCTL. In this mode audio channel 1 is transmitted in place of logic true, and audio channel 2 in place of logic zero. Channel 2 must be the lower tone of the tone pair.

The processor can force the data output line to zero (or to audio channel 2, if in two tone mode) by setting bit 7 of SKCTL. This is required to force a break (10 zeros) code transmission.

Serial Output Clock: The serial output data always changes when the serial output clock goes true. The clock then returns to zero in the center of the output data bit time.

The baud (bit) rate of the data and clock is determined by audio channel 4 audio channel 2, or by the input clock, depending on the serial mode selected by bits 4, 5, and 6 of SKCTL. (See chart at end of this section.)

Serial Input: The receiving sequence begins when the hardware has received a complete 8 bit serial data word plus start and stop bits. This data is automatically transferred to the 8 bit parallel input register (SERIN), and the processor is interrupted to indicate an input data byte ready to read in SERIN. The processor must respond to this interrupt, and read SERIN, before the next input data word reception is complete, otherwise an input data "over-run" will occur. This over-run will be indicated by bit 5 of SKSTAT (if bit 5 of IRQST is not RESET (true) before next input complete), and means input data has been lost. This bit should be tested whenever SERIN is read. Bit 7 of SKSTAT should also be tested to detect frame errors caused by extra (or missing) data bits.

Direct Serial Input: The serial data input line can be read directly by the microprocessor if desired, ignoring the shift register, by reading bit 4 of SKSTAT.

Bi-Directional Clock: This clock line is used to either receive a clock from an external clock source for clocking transmitted or received data, or is used to supply a clock to external devices indicating the transmit or reception rate. This clock line direction is determined by the serial mode selected by bits 4, 5, and 6 of SKCTL. (See mode chart at the end of this section.) Transmitted data changes on the rising edge of this clock. Received data is sampled on the trailing edge of this clock.

Asynchronous Serial Input: Unclocked serial data (at an approximately known (+5%) rate) can be received in the asynchronous modes. The receive (input) shift register is clocked by audio channel 4. Channels 3 and 4 should be used together (AUDCTL bit 3 = 1) for increased resolution. In asynchronous modes, channels 3 and 4 are reset by each start bit at the beginning of each serial data byte. This allows the serial data rate to be slightly different from the rate set by channels 3 and 4.

Serial Mode Control: There are 6 useful modes (of the possible 8) controlled by bits 4, 5, and 6 of SKCTL. These are described on the next page.

Note that two tone output (bit 3 of SKCTL) may be used in any of these modes except for the bottom pair. This is because channel 2 is used to set the output transmit rate and is therefore not available for one of the two tones.

Note that the output clock rate is identical to the output data rate.

Serial Mode Control (see also register description SKCTL):

Force Break

D7	D6	D5	D4	D3	D2	D1	D0

SKCTL REGISTER

Pot scan and keyboard CTRL

Two Tone Control

Mode Control Bits

A=asynchronous

D6	D5	D4	Out Rate	Out Clock	In Rate	Bi-Dir Clock	Comments
0	0	0	ext	ext	ext	ext input	Trans. & Receive rates set by external clock. Also internal clock phase reset to zero.
0	0	1	ext	ext	chan 4 A	ext input	Trans. rate set by external clock. Receive asynch. (ch. 4) (CH3 and CH4)
0	1	0	chan 4	chan 4	chan 4	chan 4 output	Trans. & Receive rates set by Chan. 4. Chan. 4 output on Bi-Directional clock line.
0	1	1	CH4 A	CH4 A	CH4 A	input	Not Useful
1	0	0	chan 4	chan 4	ext	ext input	Trans. Rate Set by Chan. 4 Receive Rate set by External Clock.
1	0	1	CH4 A	CH4 A	CH4 A	input	Not Useful
1	1	0	Chan 2	Chan 2	Chan 2	Chan 4 Output	Trans. rate set by chan. 2 Recieve rate set by chan. 4 Chan. 4 out on Bi-Direct. Clock line.
1	1	1	Chan 2	Chan 2	Chan 4 A	Input not used	Trans. Rate set by Chan. 2. Receive async. (chan 3&4) Bi-Dir. Clock not used (Tri-state condition)

Two tone (bit3) not useable in these modes

D. INTERRUPT SYSTEM

There are two basic types of interrupts defined on the microprocessor: NMI (non maskable interrupt) and IRQ (interrupt request). It is recommended that a thorough understanding of these interrupt types be acquired by reading all chapters concerning interrupts in the 6502 microprocessor programming and hardware manuals.

In this system NMI interrupts are used for video display and reset. IRQ interrupts are used for serial port communication, peripheral devices, timers, and keyboard inputs.

NMI Interrupts: Even though NMI interrupts are "unmaskable" on the micrprocessor, this system has interrupt enable (mask) bits for NMI function. (Bits 6 and 7 of NMIEN) When these bits are zero NMI interrupts are disabled (masked) and prevented from causing a microprocessor NMI interrupt. (see NMIEN register description) The 3 types of NMI interrupts are:

1. D7 = Instruction Interrupt (during display time any display instruction with bit 7=1 will cause this interrupt to occur (if enabled) at the start of the last video line displayed by that instruction.)

2. D6 = Vertical Blank Interrupt (interrupt occurs (if enabled) at the beginning of the vertical blank time interval.)

3. D5 = Reset Button Interrupt (pushing the SYSTEM RESET button will cause this interrupt to occur.)

Since any of these interrupts will cause the processor to jump to the same NMI address, the system also has NMI status bits which may be examined by the processor to determine which source caused the NMI interrupt. Bits 5, 6, and 7 of NMIST serve this function (see NMIST register description). These status bits are set by the corresponding interrupt function (even if the interrupt is masked from the processor by NMIEN). The status bits may be reset together by writing to the address NMIRES.

Two of the interrupt enable bits (bits 6 and 7 of NMIEN) are cleared automatically during system power turn on and therefore these NMI interrupts are initially disabled (masked), preventing any power turn on service routine from being interrupted before proper initialization of registers and pointers.* They can then be enabled by the processor whenever desired, by writing into bits 6 and 7 of NMIEN. Except for the reset button interrupt, they can also be disabled by the processor by writing a zero into bits 6 or 7 or NMIEN. The reset button cannot be disabled, allowing an unstoppable escape from any possible "hangup" condition.

These NMI interrupt functions are each separated in time (to prevent overlaps) and converted to pulses by the system hardware, in order to supply NMI transitions required by the microprocessor logic.

---

* - NOTE: Bit 5 is never disabled and therefore the Reset Button should not be pressed during power turn on.

IRQ Interrupts: IRQ interrupts are all "maskable" together by one bit of the status register on the microprocessor. This bit is set to the disable condition automatically by power turn on to prevent interrupt of power turn on service routines.** In addition to this processor IRQ mask bit, there are separate system IRQ interrupt enable bits for each IRQ interrupt function (bits 0 thru 7 of IRQEN). These bits are not initialized by power turn on, and must be initialized by the program before enabling the processor IRQ. The 8 types of IRQ interrupts are:

D7 = BREAK KEY (depression of the break key)
D6 = OTHER KEY (depression of any other key)
D5 = SERIAL INPUT READY (Byte of serial data has been received and is ready to be read by the processor in SERIN register).
D4 = SERIAL OUTPUT NEEDED (Byte of serial data is being transmitted and SEROUT is ready to be written to again by the processor).
D3 = TRANSMISSION FINISHED (serial data transmission is finished. Output shift register is empty).
D2 = TIMER #4 (audio divider #4 has counted down to zero)
D1 = TIMER #2 (audio divider #2 has counted down to zero)
D0 = TIMER #1 (audio divider #1 has counted down to zero)

In addition to the above IRQ interrupts (enabled by bits 0 through 7 of IRQEN and identified by status bits 0 thru 7 of IRQST) there are two more system IRQ interrupts which are generated over the serial bus Proceed and Interrupt lines.

D7 of PACTL = peripheral "A" interrupt status bit
D0 of PACTL = peripheral "A" interrupt enable bit
D7 of PBCTL = peripheral "B" interrupt status bit
D0 of PBCTL = peripheral "B" interrupt enable bit

These last two interrupts are automatically disabled by power turn on, and their status bits are reset by reading from port A register and port B register. (See PORTA, PACTL, PORTB, and PBCTL Register descriptions.)

The IRQEN register, like the NMIEN register, enables interrupts when its bits are 1 (logic true). The IRQST however (unlike the NMIST) has interrupt status bits that are normally logic true, and go to zero to indicate an interrupt request. The IRQST status bits are returned to logic true only by writing a zero into the corresponding IRQEN bit. This will disable the interrrupt and simultaneously set the interrupt status bit to one. Bit 3 of IRQST is not a latch and does not get reset by interrupt disable. It is zero when the serial out is empty (out finished) and true when it is not.

** - NOTE: An NMI also disables the I bit.

INTERRUPT SUMMARY

NAME	FUNCTIONS	ENABLE	STATUS	STATUS RESET
NMI INTERRUPTS	Display Instruction Vert. Blank Reset Button	NMIEN (Bits 6 thru 7) Normally Zero (Disabled)	NMIST (Bits 5 thru 7) Normally Zero (no interrupt)	Address NMIRES (Resets all NMI status together)
IRQ INTERRUPTS	KEYS Serial ports Timers	IRQEN (Bits 0 thru 7) zero is (Disabled)*	IRQST (Bits 0 thru 7) Normally True (no interrupt)	Reset (to true) By Zero in Corresponding Bit of IRQEN (except Bit 3)*
	Peripheral A	D0 of PACTL Normally Zero (Disabled)	D7 of PACTL Normally Zero (no interrupt)	Reset by Reading PORT A Register
	Peripheral B	D0 of PBCTL Normally Zero (Disabled)	D7 of PBCTL Normally Zero (no interrupt)	Reset by Reading PORT B Register

E. CONTROLLERS

A variety of controllers can be plugged into the four jacks on the front of the console. This includes joysticks, paddle (pot), twelve-key keyboard, and light pen (when available).

The controller ports are read through the PORTA and PORTB regisers and the POT and TRIG registers. The OS reads these registers during vertical blank and stores into its own RAM locations. These are STICK, PADDL0 through PADDL7, PTRIG'S and STRIG'S. The OS sets up PORTA AND PORTB for input. This is done by setting PACTL or PORTB (Port Control) bit 2 to a 0 (to select the direction control register), then writing all 0's to the desired port. PACTL (PBCTL) bit 2 is then changed back to a 1, allowing the program to read from the port. The ports can also be set up for output by writing 1's instead of 0's while the direction control mode is selected.

Joysticks: The joysticks have four switches, one each for right (R), left (L), back (B) and forward (F).

These switches are read through PORTA and PORTB. A fifth switch is activated by pressing the red trigger button. The trigger buttons are read from TRIG0 through TRIG3. A value of 0 indicates that a button has been pressed and a 1 indicates that it has not been pressed.

The TRIG registers are normally read directly, but they can be used in a latched mode. Writing a zero to bit 2 of GRACTL disables the latches and sets them to 1. Writing a 1 to bit 2 enables the latches. If a joystick trigger button is pushed at any time while bit 2 of GRACTL is 1 the latch value will change to zero and stay that way. A program can use this to determine whether the joystick trigger buttons have ever been pressed during a certain period of time.

Paddles: The paddles come in pairs, so eight paddles can be connected to the four jacks. The paddles are read by storing into POTGO, then reading the POT registers at least 228 lines later. The values range from 0 (with the paddle turned to the right) to 228 (paddle turned counter-clockwise). The value indicates how many TV lines it takes to charge up the capacitor which is the series with the potentiometer. Turning the knob to the right lowers the resistance, so the capacitor charges up quickly. Turning the knob to the left increases the resistance and the charging time. The capacitor dump transistors are used to discharge the capacitors so that a new reading can be mode. The POTGO command clears the counters and turns off the dump transistors to allow the capacitors to charge up. The ALLPOT register contains one bit for each paddle. When the capacitor has charged up to the threshold value the ALLPOT bit changes fron one to zero and the POT register contains the correct readings. Bit 2 of SKCTL (Serial Port Control) enables fast pot scan. In this mode, It takes only two scan lines to charge up the capacitors to the maximum level instead of 228 lines. Bit 2 is first set to 0 to dump the capacitors. Then Bit 2 is set to 1 to start the pot scan. The fast pot scan is not as accurate as the normal scan mode. Bit 2 of SKCTL must be set to 0 to use normal scan mode. Otherwise, the capacitors will never dump. Note that some paddles have a range smaller than 0 to 228 due to differences in the pots. The left and right paddle triggers for each paddle pair are read from the left and right bits for the corresponding joystick (PORTA or PORTB).

Keyboard Controllers: Each keyboard controller has a twelve-key pad and plugs into a joystick controller port. The first step in using the keyboard is to select a row by setting the port direction to output and writing a 0 to the bit in the PORTA or PORTB register which selects the desired row (see PORTA, SECTION III). The other rows should have 1's written to them. Columns are read through the POT and TRIG registers (see controller PORT PINOUT chart in section III). Appendix H of the BASIC Reference Manual contains a Basic program which reads the controllers. The first and second columns of the keyboard use the same pins as the pots for the paddle controllers, so they are read by reading the POT (or PADDL) registers. When a button is pushed, the pot line is grounded, so the pot capacitors never charge up to the threshold level and the reading is 228 (the maximum). When the button in the selected row and column is not pushed the capacitor is connected to +5V through a relatively small resistor, giving a POT value of about 2 (this may vary). Since the reading is not critical, the fast pot scan mode can be used, so that only a 2 line wait is required between selecting the row and reading the POT register. The convention has been adopted of comparing the POT reading with 10 (decimal). If is it greater than 10 then the button has been pressed. The third column is read through the joystick trigger line, so it works just like a joystick trigger (0=button is pressed, 1=not pressed).

Light Pen: A light pen is a device that can detect the electron beam as it sweeps across the TV screen. It is used to point directly at an image on the TV display. Applications include selecting menu items and drawing lines. The ATARI 400/800 hardware was designed so that a light pen can be plugged into any of the joystick controller ports (see end of section III).

When any one of the joystick trigger lines (pin 6) is pulled low, the ANTIC chip takes the current VCOUNT value and stores it in PENV. The horizontal color clock value (0-227 decimal) is stored in PENH. The least significant bit is inaccurate and should be ignored. Since there are a number of delays involved in displaying the data and changing the light pen register, each system must be cal brated. Software which uses the light pen should contain a user-interactive calibration routine. For example, the user could point the light pen at a crosshair in the center of the screen and the program could compute the required horizontal offset. PENH will wrap around from 227 to 0 near the right hand edge of a standard width display because of the delay. The pen will not work if it is pointed at a black area of the screen, since the electron beam is turned off. It is a good idea to read two (or more) values and average them, since the user will probably not hold the pen perfectly steady.

III. HARDWARE REGISTERS

This section lists the hardware registers and Operating System (OS) shadow registers.

In the following descriptions, true always refers to a bit whose value is 1.

A. PAL (D014)

Not Used	D3	D2	D1	Not Used

D3	D2	D1	
1	1	1	NTSC (US TV)
0	0	0	PAL (European TV)

This byte can be read by a program to determine which type of system the program is running on.

B. INTERRUPT CONTROL

NMIEN (Non Maskable Interrupt Enable)(D40E): This address writes data to the NMI interrupt enable bits.

0 = disabled (masked)
1 = enabled

D7	D6	Not Used

D7 Display List Instruction Interrupt Enable. This bit is cleared by Power Reset, and may be set or cleared by the processor.

D6 Vertical Blank Interrupt Enable. This bit is cleared by Power Reset, and may be set or cleared by the processor.

SYSTEM RESET Button Interrupt

This interrupt is always enabled. The SYSTEM RESET button should not be pressed during power turn on.

(Set to hex 40 by OS IRQ code.)

NMIST (Non Maskable Interrupt Status)(D40F): This address read the NMI Status Register (Read by OS NMI code).

0 = no interrupt
1 = interrupt

D7	D6	D5	Not Used

D7 This bit identifies an NMI interrupt caused by bit 7 of a Display List Instruction.

D6 This bit identifies an NMI interrupt caused by the beginning of vertical blank.

D5 This bit identifies an NMI interrupt caused by the SYSTEM RESET button.

NMIRES (NMI Status Register Reset)(D40F): This write address resets the Non Maskable Interrupt Status Register (NMIST).

Not Used

(Written by OS NMI code.)

IRQST (IRQ Interrupt Status)(D20E): This address reads the data from the IRQ Interrupt Status Register.

0 = Interrupt
1 = No Interrupt

D7	D6	D5	D4	D3	D2	D1	D0

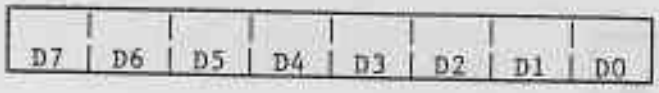
D7 = 0 Break Key Interrupt
D6 = 0 Other Key Interrupt
D5 = 0 Serial Input Data Ready Interrupt
D4 = 0 Serial Output Data Needed Interrupt
D3 = 0 Serial Output (Byte) Transmission Finished Interrupt *
D2 = 0 Timer 4 Interrupt
D1 = 0 Timer 2 Interrupt
D0 = 0 Timer 1 Interrupt

---

* - NOTE: Used for generation of 2 stop bits. See IRQ description in section II (no direct reset on bit 3).

IRQEN (IRQ Interrupt Enable)(D20E): This address writes data to the IRQ Interrupt Enable bits.

0 = disable, corresponding IRQST bit is set to 1
1 = enable



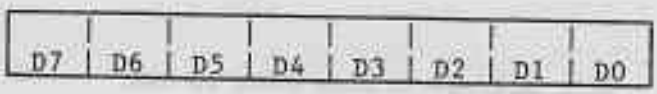
D7 Break Key Interrupt Enable
D6 Other Key Interrupt Enable
D5 Serial Input Data Ready Interrupt Enable
D4 Serial Output Data Needed Interrupt Enable
D3 Serial Out Transmission Finished Interrupt Enable
D2 Timer 4 Interrupt Enable
D1 Timer 2 Interrupt Enable
D0 Timer 1 Interrupt Enable

OS SHADOW: POKMSK (hex 10)

Use AND's and OR's to change one bit in POKMSK without affecting the others. Store the desired value in both IRQEN and POKMSK.

C. TV LINE CONTROL

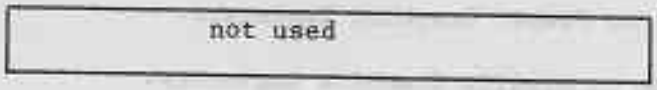
VCOUNT (Vertical Counter)(D40B): This address reads the Vertical TV Line Counter (8 most significant bits).



V8 V7 V6 V5 V4 V3 V2 V1 V0

V0 not read. Two line resolution supplied.

WSYNC (Wait for Horizontal Blank Synchronism - i.e. wait until start of next TV line.)(D40A):



This address sets a latch that pulls down on the RDY line to the microprocessor, causing it to wait until this latch is automatically reset by the beginning of horizontal blank. Display list interrupts may be delayed by 1 line if WSYNC is used. (Used by OS keyboard click routine.)

D. GRAPHICS CONTROL

DMACTL (Direct Memory Access Control)(D400): This address writes data into the DMA Control Register.

Not Used	D5	D4	D3	D2	D1	D0

D5 = 1 Enable instruction fetch DMA

D4 = 1 1 Line P/M resolution

D4 = 0 2 line P/M resolution

D3 = 1 Enable Player DMA

D2 = 1 Enable Missile DMA

D1,D0 = 0 0 No Playfield DMA

= 0 1 Narrow Playfield DMA (128 Color Clocks)

= 1 0 Standard Playfield DMA (160 Color Clocks)

= 1 1 Wide Playfield DMA (192 Color Clocks)

See GRACTL. OS Shadow: SDMCTL (22F) default value hex 22

GRACTL (Graphics Control)(D01D): This address writes data to the Graphic Control Register.

Not Used	D2	D1	D0

D2 = 1 Enable latches on TRIG0 - TRIG3 inputs (latches are cleared and TRIG0 - TRIG3 act as normal inputs when this control bit is zero).

D1 = 1 Enable Player DMA to Player Graphics Registers.

D0 = 1 Enable Missile DMA to Missile Graphics Registers.

DMA is enabled by setting bits in both DMACTL and GRACTL. Setting DMACTL only will result in cycles being stolen but no display will be generated.

CHACTL (Character Control)(D401): This address writes data into the Character Control Register.

Not Used	D2	D1	D0

D2 Character Vertical Reflect Bit. This bit is sampled at the beginning of each line of characters. If true it causes the line of characters to reflect (invert) vertically (for upside down characters).

D1 Character Video Invert Flag (used for 40 Character Mode only). If bit 7 of character code is true this flag causes that character to be blue on white (if normal colors are white on blue).

D0 Character Blank (Blink) Flag (used for 40 Character Mode only). If bit 7 of character code is true this flag causes that character to blank. Blinking characters are produced by setting bit 7 of the characters to 1, then periodically changing D0 of CHACTL.

OS SHADOW: CHACT (2F3)

DLISTL(Display List Low)(D402): This address writes data into the low byte of the Display List Counter.

D7	D6	D5	D4	D3	D2	D1	D0
7	6	5	4	3	2	1	0

{ Display List Counter Bit Position.

OS SHADOW: SDLSTL (hex 230)

DLISTH (Display List High)(D403): This address writes data into the high byte of the Display List Counter.

D7	D6	D5	D4	D3	D2	D1	D0
15	14	13	12	11	10	9	8

{ Display List Counter Bit Position.

OS SHADOW: SDLSTH (HEX 231)

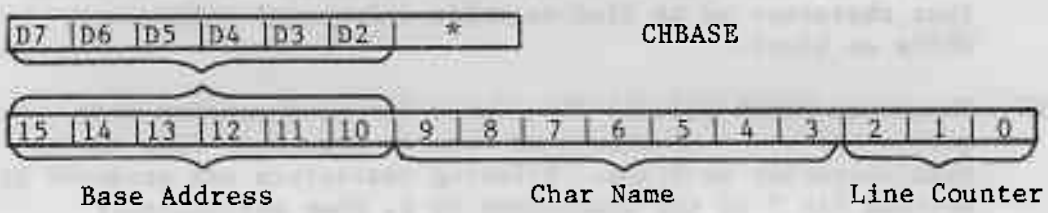
The Display List is a list of display instructions in memory. These instructions are addressed by the Display List Counter. Loading these registers defines the address of the beginning of the Display List. (See sections I and II.)

Note: The top 6 bits are latches only and have no count capability, therefore the display list can not cross a 1K byte memory boundary unless a jump instruction is used.

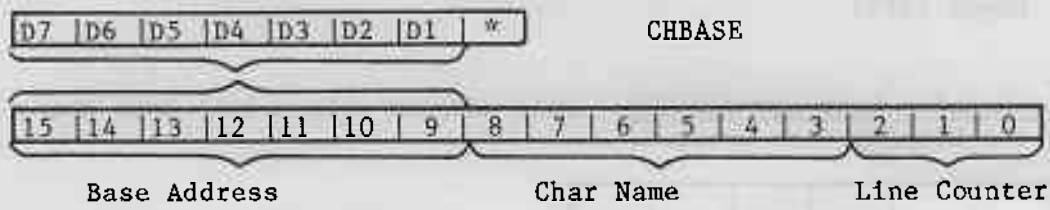
DLISTL and DLISTH should be changed only during vertical blank or with DMA disabled. Otherwise, the screen may roll. Bit 7 of NMIEN must be set in order to receive display list interrupts.

CHBASE (Character Address Base Register)(D409): This address writes data into the Character Address Base Register. The data specifies the most significant byte (MSB) of the address of the desired character set (see section II). Note that the last 1 or 2 bits are assumed to be 0.

40 Character Modes



20 Character Modes



OS SHADOW: CHBAS (2F4)

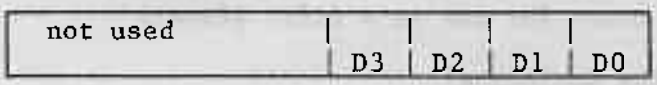
PMBASE (Player-Missile Address Base Register)(D407): This address writes data into the Player-Missile Address Base Register. The data specifies the MSB of the address of the player and missile DMA data (see section II).

One Line Resolution

Two Line Resolution

* = Not Used

HSCROL (Horizontal Scroll Register)(D404): This address writes data into the Horizontal Scroll Register. Only playfield is scrolled, not players and missiles.

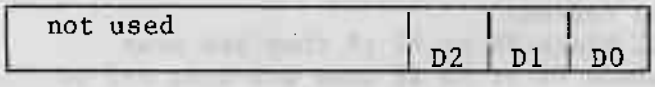


0 to 15 color
clock right shifts

The display is shifted to the right by the number of color clocks specified by HSCROL for each display list instruction that contains a 1 in its HSCROL Flag bit (bit 4 of instruction byte).

When horizontal scrolling is enabled, more bytes of data are needed. For a narrow playfield (see DMACTL bits 1 and 0) there should be the same number of bytes per line as for standard playfield with no scrolling. Similarly, for standard playfield use the same number of bytes as for the wide playfield. For wide playfield, there is no change in the number of bytes and background color is shifted in.

VSCROL (Vertical Scroll Register)(D405): This address writes data into the Vertical Scroll Register.

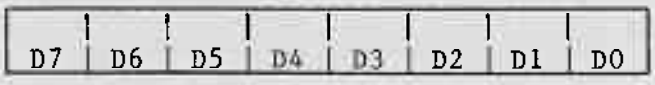


8 line display modes

16 line display modes

The display is scrolled upward by the number of lines specified in the VSCROL register for each display list instruction that contains a 1 in its VSCROL Flag bit (bit 5 of instruction byte). The scrolled area will terminate with the first instruction having a zero in bit 5. (see section II for more details).

PRIOR (Priority)(D01B): This address writes data into the Priority Control Register.

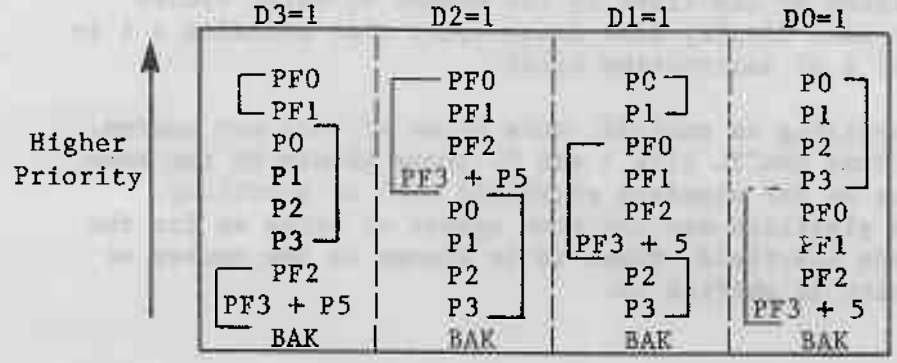


D7-D6 = 0 D5

Multiple Color Player Enable.
This bit causes the logical "or" function of the bits of the colors of Player 0 with Player 1, and also of Player 2 with Player 3. This permits overlapping the position of 2 players with a choice of 3 colors in the overlapped region.

D4 Fifth Player Enable.
This bit causes all missiles to assume the color of Playfield Type 3. (COLPF3). This allows missiles to be positioned together with a common color for use as a fifth player.

D3, D2, D1, & D0 Priority Select (Mutually Exclusive).
These bits select one of 4 types of priority. Objects with higher priority will appear to move in front of objects with lower priority.



NOTE: The use of Priority bits in a "non-exclusive" mode (more than 1 bit true) will result in objects (whose priorities are in conflict) turning BLACK in the overlap region.

EXAMPLE: PRIOR code = 1010 This will black P0 or P1 if they are over PF0 or PF1. It will also black P2 or P3 if they are over PF2 or PF3. In the one-color 40 character modes, the luminance of a pixel in a character is determined by COLPF1, regardless of the priority. If a higher priority player or missile overlaps the character then the color is determined by the player's color.

OS SHADOW: GPRIOR (26F)

COLPF0 - COLPF3 (Playfield Color)(D016, D017, D018, D019): These addresses write data to the Playfield Color-Lum Registers.

D7	D6	D5	D4	D3	D2	D1	D0

(see COLBK for bit assignment)

OS SHADOWS: COLOR0 - 3 (2C4-2C7)

COLBK (Background Color)(D01A): This address writes data to the Background Color-Lum Register.

Color				Luminance			
D7	D6	D5	D4	D3	D2	D1	Not Used
X	X	X	X	0	0	0	Zero Luminance (black)
				0	0	1	
				ETC.			
				1	1	1	Max. Luminance(white)
0	0	0	0	Grey			
0	0	0	1	Gold			
0	0	1	0	Orange			
0	0	1	1	Red-Orange			
0	1	0	0	Pink			
0	1	0	1	Purple			
0	1	1	0	Purple-Blue			
0	1	1	1	Blue			
1	0	0	0	Blue			
1	0	0	1	Light-Blue			
1	0	1	0	Turquoise			
1	0	1	1	Green-Blue			
1	1	0	0	Green			
1	1	0	1	Yellow-Green			
1	1	1	0	Orange-Green			
1	1	1	1	Light-Orange			

OS SHADOW: COLOR4 (2C8)

E. PLAYERS AND MISSILES

DMACTL, GRACTL, PMBASE and PRIOR also affect players and missiles.

COLPM0 - COLPM3 (Player-Missile Color)(D012, D013, D014, D015): These addresses write to the Player-Missile Color-Lum Registers. Missiles have the same color-lum as their player unless missiles are used as a 5th player (see bit 4 of PRIOR). A 5th player missile gets its color from COLPF3.

D7	D6	D5	D4	D3	D2	D1	D0

(see COLBK for bit assignments)

OS SHADOWS: PCOLR0 - 3 (2C0-2C3)

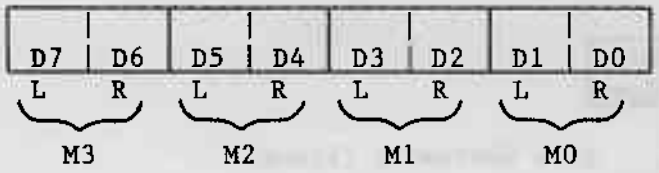
GRAFP0 - GRAFP3 (Player Graphics Registers): (P0 D00D, P1 D00E, P2 D00F, P3 D010): These addresses write data directly into the Player Graphics Registers, independent of DMA. If DMA is enabled then the graphics registers will be loaded automatically from the memory area specified by PMBASE(see page II.3).

D7	D6	D5	D4	D3	D2	D1	D0

Left Right

Player on TV Screen

GRAFM (Missile Graphics Registers)(D011): This address writes data directly into the Missile Graphics Register, independent of DMA.

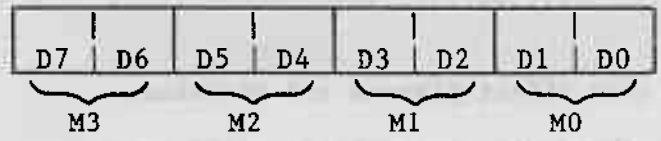


SIZEP0 - SIZEP3 (Player Size)(P0 D008, P1 D009, P2 D00A, P3 D00B): These addresses write data into the Player Size Control Registers.

Not Used	D1	D0	Horizontal Size Register (Player)
	0	0	Normal Size (8 color clocks wide)
	0	1	Twice Normal Size (16 color clocks wide)
	1	0	Normal Size
	1	1	4 Times Normal Size (32 color clocks wide)

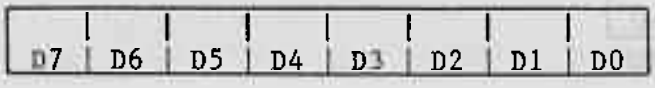
With normal size objects, each bit in the graphics register corresponds to one color clock. For larger objects, each bit is extended over more than one color clock.

SIZEM (Missile Size)(D00C): This address writes data into the Missile Size Control Register.



D1	D0	Horizontal Size Register (Missile)
0	0	Normal Size (2 color clocks wide)
0	1	Twice Normal Size (4 color clocks wide)
1	0	Normal Size
1	1	4 Times Normal Size (8 color clocks wide)

HPOSP0 - HPOSP3 (Player Horizontal Position)(P0 D000, P1 D001, P2 D002, P3 D003): These addresses write data into the Player Horizontal Position Register (see display diagram in section IV). The horizontal positon value determines the color clock location of the left edge of the object. Hex 30 is the left edge of a standard width screen. Hex D0 is the right edge of a standard screen.



HPOSM0 - HPOSM3 (Missile Horizontal Position)(M0 D004, M1 D005, M2 D006, M3 D007): These addresses write data into the Missile Horizontal Position Registers (see HPOSP0 description).

D7	D6	D5	D4	D3	D2	D1	D0

VDELAY (Vertical Delay)(D01C): This address writes data into the Vertical Delay Register.

D7	D6	D5	D4	D3	D2	D1	D0
P3	P2	P1	P0	M3	M2	M1	M0

VDELAY is used to give one-line resolution in the vertical positioning of an object when the 2-line resolution display is enabled. Setting a bit in VDELAY to 1 moves the corresponding object down by one TV line.

If player-missile DMA is enabled then changing the vertical location of an object by more than one line is accomplished by moving bits around in the memory map. If DMA is disabled then the vertical location can be set up by assembly language code which stores data into the graphics registers at the desired line.

M0PF, M1PF, M2PF, M3PF (Missile to Playfield Collisions)(D000, D001, D002, D003): These addresses read Missile to Playfield Collisions. A 1 bit means that a collision has been detected since the last HITCLR.

Not Used (zero forced)	D3	D2	D1	D0
	3	2	1	0

Playfield Type

P0PF, P1PF, P2PF, P3PF (Player to Playfield Collisions)(D004, D005, D006, D007): These addresses read Player to Playfield Collisions.

Not Used (zero forced)	D3	D2	D1	D0
	3	2	1	0

Playfield Type

M0PL, M1PL, M2PL, M3PL (Missile to Player Collision)(D008, D009, D00A, D00B): These addresses read Missile to Player Collisions.

Not Used (zero forced)	D3	D2	D1	D0
	3	2	1	0

Player Number

P0PL, P1PL, P"PL, P3PL (Player to Player Collisions)(D00C, D00D, D00E, D00F): These addresses read Player to Player Collisions

Not Used (zero forced)	D3	D2	D1	D0
	3	2	1	0

Player Number

(Player 0 against Player 0 is always a zero). Etc.

HITCLR (Collision "HIT" Clear) D01E

This write address clears all collision bits described above.

Not Used

F. AUDIO

AUDCTL (Audio Control)(D208): This address writes data into the Audio Mode Control Register. (Also see SKCTL two-tone bit 3 and notes).

D7	D6	D5	D4	D3	D2	D1	D0

D7 Change 17 bit poly into a 9 bit below poly.
D6 Clock Channel 1 with 1.79 MHZ, instead of 64 KHZ.
D5 Clock Channel 3 with 1.79 MHZ, instead of 64 KHZ.
D4 Clock Channel 2 with Channel 1, instead of 64 KHZ (16 BIT).
D3 Clock Channel 4 with Channel 3, instead of 64 KHZ (16 BIT).
D2 Insert Hi Pass Filter in Channel 1, clocked by Channel 3. (See section II.)
D1 Insert Hi Pass Filter in Channel 2, clocked by Channel 4.
D0 Change Normal 64 KHZ frequency, into 15 KHZ.

Exact Frequencies: The frequencies given above are approximate. The Exact Frequency (fin) that clocks the divide by N counters is given below (NTSC only, PAL different).

FIN (Approximate)	FIN (Exact)	
1.79 MHZ	1.78979 MHZ	- Use modified formula for fout
64 KHZ	63.9210 KHZ	- Use normal formula for fout
15 KHZ	15.6999 KHZ	

The Normal Formula for output frequency is:

$$Fout = Fin/2N$$

Where N = The binary number in the frequency register (AUDF), plus 1 (N=AUDF+1). The MODIFIED FORMULA should be used when Fin = 1.79 MHZ and a more exact result is desired:

$$Fout = \frac{Fin}{2(AUDF + M)}$$

Where: M = 4 if 8 bit counter (AUDCTL bit 3 or 4 = 0)
M = 7 if 16 bit counter (AUDCTL bit 3 or 4 = 1)

AUDF1, AUDF2, AUDF3, AUDF4 (Audio Frequency) (D200, D202, D204, D206)
These addresses write data into each of the four Audio Frequency Control Registers. Each register controls a divide by "N" counter.

D7	D6	D5	D4	D3	D2	D1	D0	"N"
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	2
				ETC.				
1	1	1	1	1	1	1	1	256

Note: "N" is one greater than the binary number in Audio Frequency Register AUDF(X).

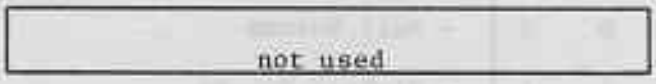
AUDC1, AUDC2, AUDC3, AUDC4 (Audio Channel Control)(D201, D203, D205, D207): These addresses write data into each of the four Audio Control Registers. Each Register controls the noise content and volume of the corresponding Audio Channel.

	Noise Content or Distortion				Volume				
HEX	D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	0	0					Divisor "N" set by audio frequency register. - 17 BIT poly - 5 BIT poly - N
2	0	0	1	0					- 5 BIT poly - N - 2
4	0	1	0	0					- 4 BIT poly - 5 BIT poly - N
6	0	1	1	0					- 5 BIT poly - N - 2
8	1	0	0	0					- 17 BIT poly - N
A	1	X	1	0					- Pure Tone - N - 2
C	1	1	0	0					- 4 BIT poly - N
1	X	X	X	1					- Force Output (Volume only)
0					0	0	0	0	- Lowest Volume (Off)
8					1	0	0	0	- Half Volume
F					1	1	1	1	- Highest Volume

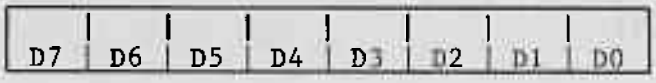
PITCH VALUES FOR THE MUSICAL NOTES-AUDCTL =0, AUDC = hex AX

		AUDF	
		Hex	Dec
HIGH	C	1D	29
NOTES	B	1F	31
	A# or Bb	21	33
	A	23	35
	G# or Ab	25	37
	G	28	40
	F# or Gb	2A	42
	F	2D	45
	E	2F	47
	D# or Eb	32	50
	D	35	53
	C# or Db	39	57
	C	3C	60
	B	40	64
	A# or Bb	44	68
	A	48	72
	G# or Ab	4C	76
	G	51	81
	F# or Gb	55	85
	F	5B	91
	E	60	96
	D# or Eb	66	102
	D	6C	108
	C# or Db	72	114
MIDDLE C	C	79	121
	B	80	128
	A# or Bb	88	136
	A	90	144
	G# or Ab	99	153
	G	A2	162
	F# or Gb	AD	173
	F	B6	182
	E	C1	193
	D# or Eb	CC	204
	D	D9	217
LOW	C# or Db	E6	230
NOTES	C	F3	243

STIMER (Start Timer)(D209): This write address resets all audio frequency dividers to their "AUDF" value. These dividers generate timer interrupts when they count down to zero (if enabled by IRQEN). (also see IRQST)



RANDOM (Random Number Generator)(D20A): This address reads the high order 8 bits of a 17 bit polynomial counter (9 bit, if bit 7 of AUDCTL=1).



G. KEYBOARD AND SPEAKER

CONSOL (Console Switch Port)(D01F): This address reads or writes data from the console switches and indicators. (Set to 8 by OS Vertical Blank code.)

Not Used (zero forced)	D3	D2	D1	D0

Hex 08 should be written to this address before reading the switches.

Ones written will pull down on the switch line.

CONSOL Bit Assignment:

D0	Game Start	- 0 means switch pressed.
D1	Game Select	
D2	Option Select	
D3	Loudspeaker	- should be held at 1 except when writing 0 momentarily. OS writes a 1 during vertical blank.

KBCODE (Keyboard Code)(D209): This address reads the Keyboard Code, and is usually read in response to a Keyboard Interrupt (IRQ and bits 6 or 7 of IRQST). See IRQEN for information on enabling keyboard interrupts. See SKCTL bits 1 and 0 for key scan and debounce enable.

D7	D6	D5	D4	D3	D2	D1	D0

D7 = Control Key
D6 = Shift Key

Read by OS into shadow CH when key is hit. The OS has a get character function which converts the keycode to ATASCII (Atari ASCII).

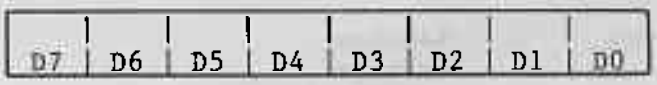
KEYCODE TO ATASCII CONVERSION

KEY CODE	KEY CAP	L.C.	U.C.	CTRL		KEY CODE	KEY CAP	L.C.	U.C.	CTRL
00	L	6C	4C	0C		20	,	2C	5B	00
01	J	6A	4A	0A		21	SPACE	20	20	20
02	;	3B	3A	7B		22	.	2E	5D	60
03						23	N	6E	4E	0E
04						24				
05	K	6B	4B	0B		25	M	6D	4D	0D
06	+	2B	5C	1E		26	/	2F	3F	
07	*	2A	5E	1F		27	^	*	*	*
08	O	6F	4F	0F		28	R	72	52	12
09						29				
0A	P	70	50	10		2A	E	65	45	05
0B	U	75	55	15		2B	Y	79	59	19
0C	RET	9B	9B	9B		2C	TAB	7F	9F	9E
0D	I	69	49	09		2D	T	74	54	14
0E	-	2D	5F	1C		2E	W	77	57	17
0F	=	3D	7C	1D		2F	Q	71	51	11
10	V	76	56	16		30	9	39	28	
11						31				
12	C	63	43	03		32	0	30	29	
13						33	7	37	27	
14						34	BACKS	7E	9C	FE
15	B	62	42	02		35	8	38	40	
16	X	78	58	18		36	<	3C	7D	7D
17	Z	7A	5A	1A		37	>	3E	9D	FF
18	4	34	24			38	F	66	46	06
19						39	H	68	48	08
1A	3	33	23	*		3A	D	64	44	04
1B	6	36	26			3B				
1C	ESC	1B	1B	1B		3C	CAPS	*	*	*
1D	5	35	25			3D	G	67	47	07
1E	2	32	22	FD		3E	S	73	53	13
1F	1	31	21	*		3F	A	61	41	01

* = special handling

H. SERIAL PORT (see peripheral connector on console)

SKCTL (Serial Port control)(D20F): This address writes data into the register that controls the configuation of the serial port, and also the Fast Pot Scan and Keyboard Enable.



(Bits are normally zero and perform the functions shown below when true.)

D7 Force Break (force serial output to zero (space))*

D6, D5, D4 Serial Port Mode Control (see mode chart at end of Serial port description, page II.34).

D3 Two Tone (Serial output transmitted as two tone signal instead of logic true/false.)

D2 Fast Pot (Fast Pot Scan. The Pot Scan Counter completes its sequence in two TV line times instead of one frame time. The capacitor dump transistors are completely disabled.)

D1 Enable Key Scan (Enables Keyboard Scanning circuit)

D0 Enable Debounce (Enables Keyboard Debounce circuits)

D0-D1 (Both Zero) Initialize (State used for testing and initializing chip) **

OS SHADOW: SSKCTL (hex 232)

The OS enables key scan and debounce and may change the other bits for different I/O operations. In particular, an aborted cassette operation may leave the two tone bit in the true state, causing undesirable audio signals. This may be corrected by writing hex 13 to both SKCTL and SSKCTL after doing I/O and/or before modifying the audio registers.

* NOTE: When powered on, serial port output may stay low even if this bit is cleared. To get S.P. high (mark), send a byte out (recommend 00 or FF).

**NOTE: There is no original power on state. Pokey has no reset pin.

SKSTAT (Serial Port-Keyboard Status)(D20F): This address reads the status register giving information about the serial port and keyboard.

D7	D6	D5	D4	D3	D2	D1	D0

(Bits are normally true and provide the following information when zero.)

D7 = 0 = Serial Data Input Frame Error

D6 = 0 = Serial Data Input Over-run

D5 = 0 = Keyboard Over-run

D4 = 0 = Direct from Serial Input Port

D3 = 0 = Shift Key Depressed

D2 = 0 = Last Key is Still Depressed

D1 = 0 = Serial Input Shift Register Busy

D0 = 1 Not Used (Logic True)

Latches must be reset = 1 (SKRES)

(D5 and D6 are set to zero when new data and same bit of IRQST is zero)

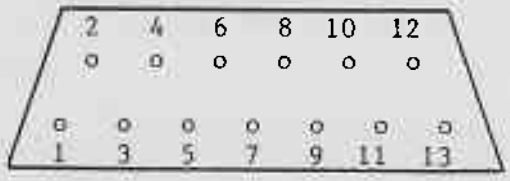
SKRES (Reset above Status Register)(D20A): This write address resets bits 7, 6, and 5 of the Serial Port-Keyboard Status Register to 1.

not used

SERIN (Serial Input Data)(D20D): This address reads the 8 bit parallel holding register that is loaded when a full byte of serial input data has been received. This address is usually read in response to a serial data in interrupt (IRQ and bit 5 of IRQST). Also see IRQEN.

D7	D6	D5	D4	D3	D2	D1	D0

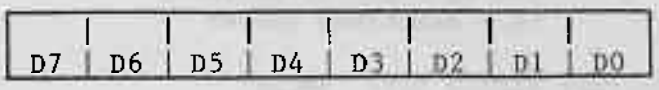
Serial I/O Port Connector Pinout:



1.	Clock In	2.	Clock Out
3.	Data In to computer	4.	GND
5.	Data Out of Computer	6.	GND
7.	Command	8.	Motor Control
9.	Proceed	10.	+5 / Ready
11.	Audio In	12.	+12
13.	Interrupt		

See serial port description in OS manual for more details.

SEROUT (Serial Output Data)(D20D): This address writes to the 8 bit parallel holding register that is transferred to the output serial shift register when a full byte of serial output data has been transmitted. This address is usually written in response to a serial data out interrupt (IRQ and bit 4 of IRQST).

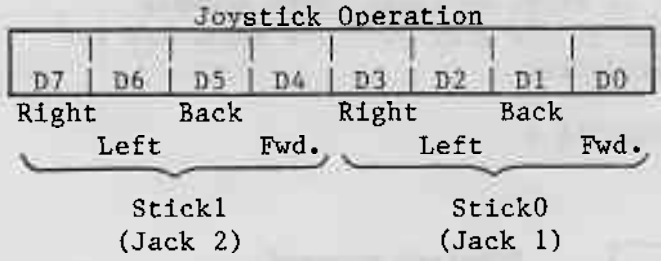


I. CONTROLLER PORTS (front of console)

PORTA (Port A)(D300): This address reads or writes data from Player 0 and Player 1 controller jacks if bit 2 of PACTL is true. This address writes to the direction control register if bit 2 of PACTL is zero. I/O for both ports (A and B) goes through a 6520/6820

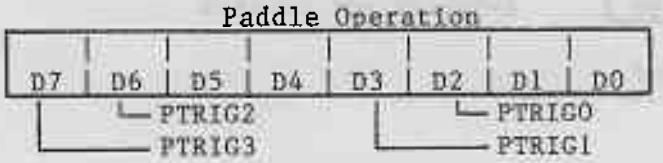
Data Register-Addressed if bit 2 of PACTL is 1.

Joystick Operation



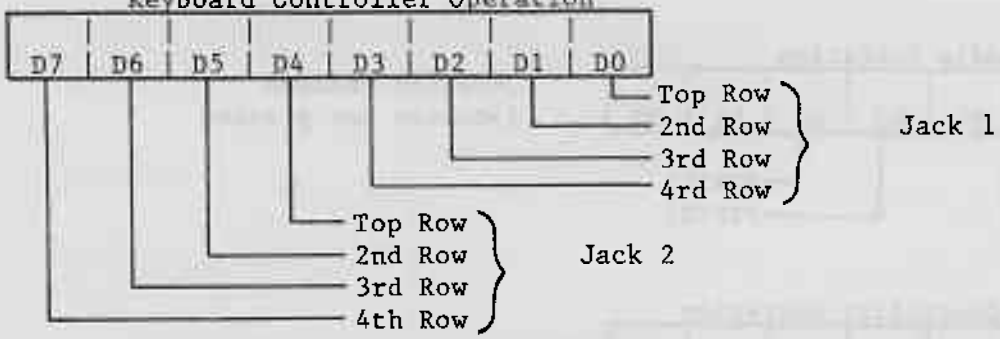
0=Switch pressed
1=Switch not pressed

Paddle Operation

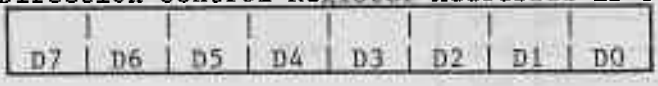


0=Switch pressed
1=Switch not pressed

Keyboard Controller Operation



Direction Control Register-Addressed if bit 2 of PBCTL is 0

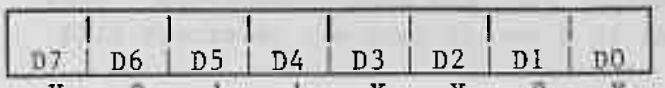


Each bit corresponds to a jack pin

0=input
1=output

OS SHADOWS: STICK0 (hex 278), STICK1 (279), PTRIG0-3 (27C-27F

PACTL (Port A Control)(D302): This address writes or reads data from the Port A Control Register.

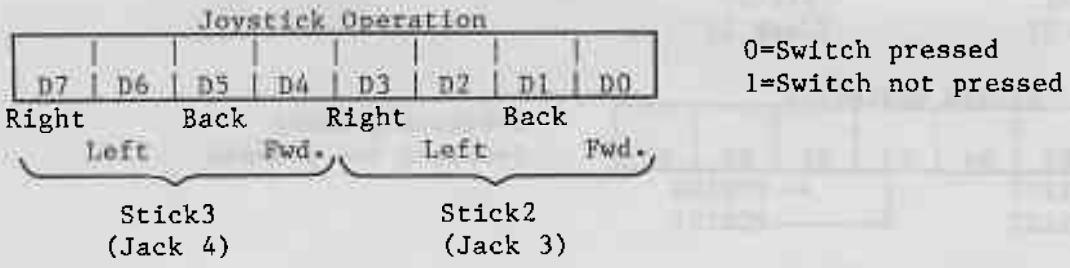


Port A Control Register
Set up register as shown (X = described below)

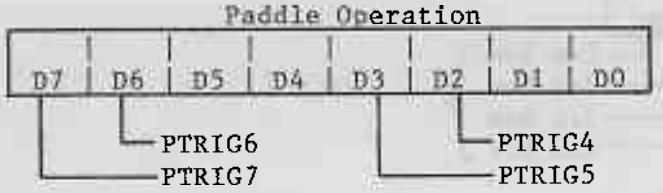
D7 - (Read only) Peripheral A Interrupt Status Bit. Serial bus Proceed line. (Reset by reading Port A Register. Set by Peripheral A Interrupt.)
D3 - Peripheral Motor Control line on serial bus (write). (0 = On 1 = Off)
D2 - Controls Port A addressing described above (write). (1 = Port A Register 0 = Direction Control Register).
D0 - Peripheral A Interrupt Enable Bit. (Write) 1 = Enable. Reset by power turn-on or processor. Set by Processor.

PORTB (Port B)(D301): This address reads or writes data from Player 2 and Player 3 controller jacks if bit 2 of PBCTL is true. This address writes to the direction control register if bit 2 of PBCTL is zero. I/O for both ports (A and B) goes through a 6520/6820.

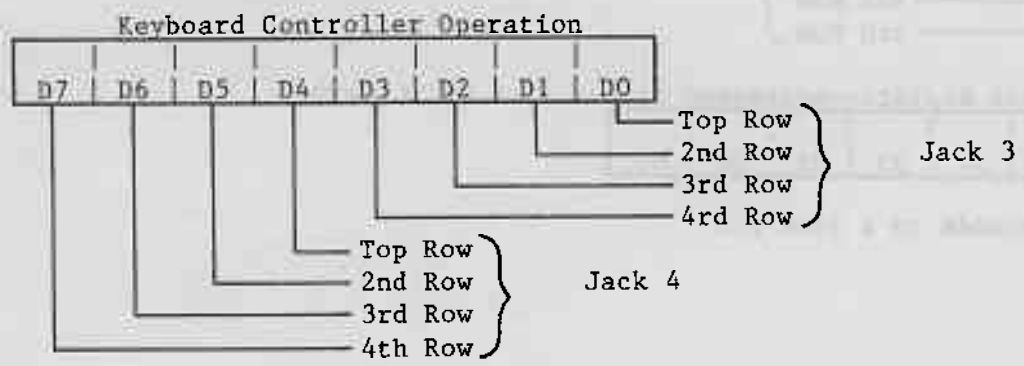
Data Register-Addressed if bit 2 of PBCTL is 1



0=Switch pressed
1=Switch not pressed



0=Switch pressed
1=Switch not pressed



Direction Control Register-Addressed if bit 2 of PBCTL is 0

D7	D6	D5	D4	D3	D2	D1	D0

Each bit corresponds to a jack pin

0=input
1=output

OS SHADOWS: STICK2 (hex 27A), STICK3 (27B), PTRIG4-7 (280-283)

PBCTL (Port B Control)(D303): This address writes or reads data from the Port B Control Register.

Read Only

D7	D6	D5	D4	D3	D2	D1	D0
x	0	1	1	X	X	0	X

Port B Control Register
Set up register as shown (X=Described below)

D7 (Read only) Peripheral B Interrupt Status Bit. Serial bus Interrupt line. Reset by Reading Port B Register. Set by Peripheral B Interrupt.

D3 Peripheral Command Identification. Serial bus Command Line.

D2 Controls Port B addressing described above. (1= Port B Register 0 = Direction Control Register)

D0 Peripheral B Interrupt Enable Bit. 1 = Enable. Reset by power turn-on or processor. Set by processor. (Set to hex 3C by OS IRQ code)

POT0 - POT7 (Pot Values)(D200-D207): These addresses read the value (0 to 228) of 8 pots (paddle controllers) connected to the 8 lines pot port. The paddle controllers are numbered from left to right when facing the console keyboard. Turning the paddle knob clockwise results in decreasing pot values. The values are valid only after 228 TV lines following the "POTGO" command described below or after ALLPOT changes.

D7	D6	D5	D4	D3	D2	D1	D0

Each Pot Value (0-228)

OS SHADOWS: PADDL0 - 7 (hex 270-277)

ALLPOT (All Pot Lines Simultaneously)(D208): This address reads the present state of the 8 line pot port.

Capacitor dump transitors must be turned off by either going to fast pot scan mode (bit 2 of SKCTL) or starting pot scan (POTGO).

D7	D6	D5	D4	D3	D2	D1	D0

Pot number:
7 6 5 4 3 2 1 0

0 = Pot register value is valid.
1 = Pot register value is not valid.

8 Pot Line States

POTGO (Start Pot Scan)(D20B):

No Data Bits Used

This write address starts the pot scan sequence. The pot values (POT0 - POT7) should be read first. This write strobe is then used causing the following sequence.

1. Scan Counter cleared to zero.
2. Capacitor dump transistors turned off.
3. Scan Counter begins counting.
4. Counter value captured in each of 8 registers (POT0 - POT7) as each pot line crosses trigger voltage.
5. Counter reaches 228, capacitor dump transistors turned on.

(Written to by OS vertical blank code)

TRIG0, TRIG1, TRIG2, TRIG3 (Trigger Ports)(0 D010, 1 D011, 2 D012, 3 D013): These addresses read port pins normally connected to the joystick controller trigger buttons.

Not Used (Zero Forced)	D0

0 = button pressed
1 = button not pressed

OS SHADOWS: STRIG0-3 (hex 284-287)

NOTE: TRIG0 thru TRIG3 are normally read directly by the microprocessor. However, if bit 2 of GRACTL is 1, these inputs are latched whenever they go to logic zero. These latches are reset (true) when bit 2 of GRACTL is set to 0.

PENH (Light Pen Horizontal Color Clock Position)(D40C): This address reads the Horizontal Light Pen Register (based on the horizontal color clock counter in hardware). The values range from 0 to decimal 227. Wraparound occurs when the pen if near the right edge of a standard-width screen. PENH and PENV are modified when any of the joystick trigger lines is pulled low.

D7	D6	D5	D4	D3	D2	D1	D0
H7	H6	H5	H4	H3	H2	H1	H0

OS SHADOW: LPENH (hex 234)

PENV (Light Pen Vertical TV Line Position)(D40D): This address reads the Vertical Light Pen Register (8 most significant bits, same as VCOUNT).

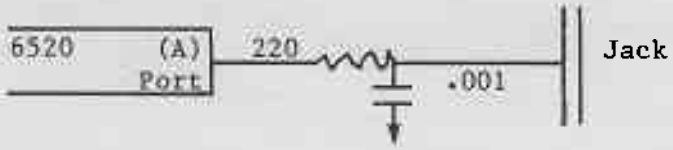
D7	D6	D5	D4	D3	D2	D1	D0	
LP8	7	6	5	4	3	2	1	0 LP0 not read. Two line resolution supplied.

OS SHADOW: LPENV (hex 235)

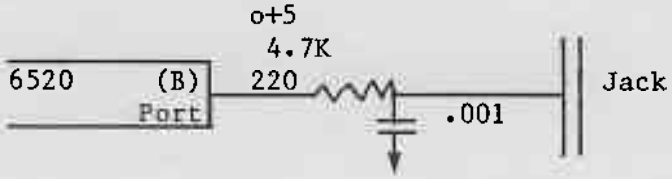
Front Panel (Controller) Jacks as I/O Parts:

PIA (6520/6820)
Out: TTL levels, 1 load
In : TTL levels, 1 load

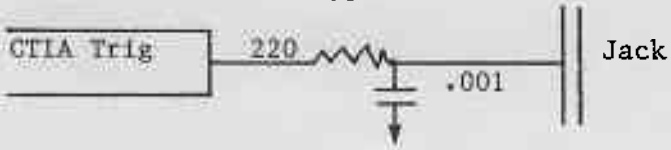
Port A Circuit (typical):



Port B Circuit (typical):



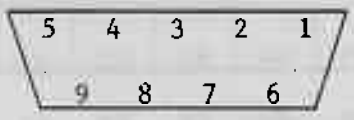
"Trigger" Port Circuit (typical):



Controller Port Pinout:

Male
(console)

Female
(connector)



PIN	Controllers JOYSTICK	PADDLE (POT)	KEYBOARD	HARDWARE REGISTERS	OS VARIABLES
1	Forward		Top Row*	Bit 0 or 4**	Bit 0***
2	Back		2nd Row*	Bit 1 or 5**	Bit 1***
3	Left	A(Left)Trigger	3rd Row*	Bit 2 or 6**	PTRIG0,2,4,6 Bit 2***
4	Right	B(Right)Trigger	Bottom Row*	Bit 3 or 7**	PTRIG1,3,5,7 Bit 3***
5		POT B(Right)	1st Column	POT 1,3,5,7	PADDL1,3,5,7
6	Trigger Button		3rd Column	TRIG0,1,2,3	STRIG0,1,2,3
7		+5	+5		
8	GND	GND			
9		POTA (Left)	2nd Column	POT 0,2,4,6	PADDL0,2,4,6

* Write
** PORTA or PORTB
*** STICK 0, 1, 2 or 3