ATARI® 400/800™

ATARI® HOME COMPUTER SYSTEM

HARDWARE MANUAL

ATARI®
A Warner Communications Company

Every effort has been made to ensure that this manual accurately documents this product of the ATARI Home Computer Division. However, due to the ongoing improvement and update of the computer software and hardware, ATARI, INC. cannot guarantee the accuracy of printed material after the date of publication and disclaims liability for changes, errors, or omissions.

TABLE OF CONTENTS

I. INTRODUCTION

The ATARI (R) 800™ and ATARI 400™ Personal Computer Systems contain a 6502 microprocessor, 4 I/O chips, operating system ROM, expandable RAM, and several MSI chips for address decoding and data bus buffering. This manual is primarily intended to describe the 4 I/O chips in sufficent detail to allow experienced programmers to create assembly language programs, such as video games. All four Input/Output chips are controlled by the microprocessor by writing directly into their registers which are decoded to exist in microprocessor memory space just as RAM does. These I/O chips can also be interrogated by the microprocessor by reading similar registers.

Many registers are write only and cannot be read after they are written. In some cases, reading from the same address gives the value contained in a separate read only register. Some write only registers are strobes. No data bits are needed in this case since the presence of the address on the bus is what triggers the requested action. The usual convention is to use the STA (Store Accumulator) instruction for such registers. For example, STA WSYNC performs the wait for Sync function. STX (Store X) or STY (Store Y) would work just as well. In BASIC, a POKE could be used (the data could be anything). Reading a register is accomplished by using any of the load instructions (LDA, LDX etc.). In BASIC a PEEK would be used. When the hardware register names are defined in an equate list, the programmer can refer to the registers by name rather than using the addresses directly.

It is really not necessary for the programmer to know which I/O functions are performed by which of the 4 chips, however it does help in learning these functions.

This manual should be used in conjunction with the Operating System (OS) Manual, a 6502 programming manual, and the ATARI 400/800 Basic Reference Manual.

CHIP NAME	FUNCTION
ANTIC	DMA(Direct Memory Access) control NMI(Non-Maskable Interrupt) control Vertical and Horizontal fine scrolling Light pen position registers Vertical line counter WSYNC(wait for horizontal sync)
CTIA	Priority control (display of overlapping objects) Color-Lumimance control (colors and brightness assigned to all objects including DMA objects from ANTIC) PLAYER-MISSILE objects (4 players and 4 missiles) Graphics registers Size control Horizontal position control Collision detection between all objects Switches and triggers (miscellaneous I/O functions)

CHIP NAME	FUNCTION
POKEY	Keyboard scan and control Serial communications port (bidirectional) Pot scan (digitizes position of 8 independent pots) Audio generation (4 channels) Timers IRQ (maskable interrupt) control from peripherals Random number generator
PIA	Controller (Joystick) jacks read or write Peripheral control and interrupt lines IRQ (maskable) interrupt control from peripherals

Section II describes the hardware in some detail, including the various graphics modes. Section III lists the hardware registers one at a time, describing what each bit is used for. It is organized by functional groups (interrupts, graphics, audio, etc.). Section IV contains a sample display program. Section V contains various figures and block diagrams of the system. Sections VI and VII list the hardware registers in address order and alphabetical order. Section VII includes hex and decimal addresses, the OS shadow registers and the page numbers where more information can be found.

II. DESCRIPTION OF HARDWARE

A. ANTIC AND CTIA

TV Display: The ANTIC and CTIA chips generate the television display at the rate of 60 frames per second on the NTSC (US) system. The PAL (European) system is different and is described in the section on NTSC vs PAL. Each frame consists of 262 horizontal TV lines and each line is made up of 228 color clocks, as shown in figure VI-3. The 6502 microprocessor runs at 1.79 MHz. This rate was chosen so that one machine cycle is equivalent in length to two color clocks. One clock is approximately equal in width to two TV lines.

In any graphics mode, the display is divided up into small squares or rectangles called pixels (picture elements). The highest resolution graphics mode has a pixel size of 1/2 color clock by 1 TV line. A sample display list is given in section IV.

The current TV line may be determined by reading the vertical counter (VCOUNT). This register gives the line count divided by 2. There are 262 lines per frame so VCOUNT runs from 0 to 130 (0 to 155 on the PAL system). The 0 point occurs near the end of vertical blank (see figure VI.5). Vertical blank (VBLANK) is the time during which the electron beam returns back to the top of the screen in preparation for the next frame. The Atari 800 does not do interlacing, so each frame is identical unless the program which is being executed changes the display. Vertical sync (VSYNC) occurs during the fourth through sixth lines of vertical blank (VCOUNT = hex 7D through 7F). This tells the TV set where each frame starts. After VSYNC, there are 16 more lines of VBLANK for a total of 22 lines of VBLANK. The display list jump and wait instruction (to be described later) causes the display list graphics to start at the end of VBLANK.

Operating System (OS): The ATARI 400/800 comes with a 10K Operating System (OS) in ROM. The OS affects some of the hardware registers, so it will be mentioned from time to time in this manual. Refer to the OS manual for more details. The OS descriptions in this manual apply to the version that was being distributed when this manual was written.

The OS supports most of the hardware graphics modes (BASICS, GRAPHICS, PLOT, and DRAWTO commands). The OS always displays 24 background lines after the end of vertical blank. This convention is used at Atari to compensate for television sets which overscan. Most TV's are designed so that the edges of the picture are cut off. This is fine for ordinary broadcasts, but with a computer it is essential for all important information to be displayed on the screen. It is fairly common for four to eight color clocks at the right or left edge of the picture to overscan. A TV set that has excessive overscan may have to readjusted to obtain a satisfactory display.

The OS uses 192 TV lines for its display and devotes the remaining 24 lines to overscan. It uses the standard display width of 160 color clocks. The hardware will allow displays of any length, but it is recommended that the standards be followed. The exception might be a border or other information which is merely decorative and not essential to use of the program.

OS Shadowing: Since many of the hardware registers are write-only and cannot be read the OS has a number of "shadow registers" in RAM. Every TV frame during vertical blank the OS takes the values in some of its shadow registers, and writes them out to the corresponding hardware register. The OS does attract color shifting on all of the color registers if ATRACT (on OS register) is negative. This is to prevent damage to the TV screen phosphors which can occur if the brightness is turned up too high and the same high-luminance display is left on for a long time. The OS also reads the joysticks and other controllers during vertical blank and stores the results in shadow registers, so that user programs do not have to include code to unpack the data. There are a few interrupt-related registers which the OS changes or reads during interrupt processing. Programs usually access the OS shadow registers instead of accessing the hardware directly. However, the OS shadowing can be disabled by changing the vertical blank and interrupt vectors (see OS manual).

WSYNC: In addition to a Vertical Blank Interrupt, which allows the Microprocessor to synchronize to the vertical TV display, this system also provides a Wait for Horizontal Sync (WSYNC) command that allows the microprocessor to synchronize itself to the TV horizontal line rate. This sync takes effect when the processor writes to an I/O location called WSYNC, whenever it desires horizontal synchronization. Writing to this address sets a latch which pulls to zero a pin on the microprocessor called READY. When READY goes to zero the microprocessor stops and waits. The latch is automatically reset (returning READY true) at the beginning of the next horizontal blank interval, releasing the microprocessor to resume program execution.

Object DMA (Direct Memory Access): The primary function of the Antic chip is to fetch data from memory (independent of the microprocessor) for display on the TV screen. It does this with a technique called "Direct Memory Access" or DMA. It requests the use of the memory address and data bus by sending a signal called HALT to the microprocessor, causing the processor to become "TRI-STATE" (open circuit) all during the next computer cycle. The ANTIC chip then takes over the address bus and reads any data it wishes from memory. Another name for this type of DMA is "cycle stealing". Once initiated, this DMA is completely and automatically controlled by the Antic chip without need for futher microprocessor intervention.

There are two types of DMA: Playfield and Player-Missile (see Figure II.2). The playfield DMA control circuit on the Antic chip resembles a small dumb microprocessor. By halting the main microprocessor it can fetch its own instructions from memory (the display list) addressed by its program counter(display list pointer). Each instruction defines the type (alpha character or memory map), and the resolution (size of bits on the screen), and the location of the data in memory which is to be displayed on the next group of lines.

In order to begin this DMA the main microprocessor must store a display list of instructions in memory, store data to be displayed in memory, tell the ANTIC where the display list is (initialize the display list pointer) and enable the DMA control flags on the ANTIC (DMACTL register).

In addition to the playfield DMA described above, the ANTIC chip simultaneously controls another DMA channel. This type of DMA addresses PLAYER-MISSILE graphics data stored in memory and passes the graphics data on to the CTIA chip graphics registers. This type of DMA (if enabled) occurs automatically, interspersed with the playfield DMA described previously. This PLAYER-MISSILE DMA has no display list or instructions, and is therefore much simpler than the PLAYFIELD DMA.

In addition to the two types of display DMA, the ANTIC chip also generates DMA addresses for the refresh of the dynamic memory RAM used in this system. This is also completely automatic and need be considered by the programmer only if he is concerned with real-time programming where an exact count of the computer cycles is important.

Color-luminance: A color-luminance register is used on the CTIA chip for each Player-Missile and Playfield type. Each color-lum register is loaded by the microprocessor with a code representing the desired color and luminance of its corresponding Player-Missile or Playfield type. As the serial data passes through the CTIA chip it is "impressed" with the color and luminance values contained in these registers, before being sent to the TV display. In areas of the screen where there are no objects the background color (COLBK) is displayed. The CTIA also does collision detection (to be described later).

Priority: When moving objects, such as players and missiles, overlap on the TV screen (with each other or with Playfield) a decision must be made as to which object shows in front of the other. Objects which appear to pass in front of others are said to have Priority over them. Priority is assigned to all objects by the CTIA chip before the serial data from each object is combined with the other objects and sent to the TV screen.

The priority of objects can be controlled by the microprocessor by writing into the control register PRIOR. The functions of the bits in this register are given in the table in the PRIOR register description in section III.

Players and Missiles: The players and missiles are small objects which can be moved quickly in the horizontal direction by changing their position registers. They are called players and missiles because they were originally designed to be used in games for objects such as airplanes and bullets. However, there are many other possible applications for them. The four player-missile color registers, in conjunction with the four playfield color registers and the background color register, make it possible to display 9 different colors at the same time.

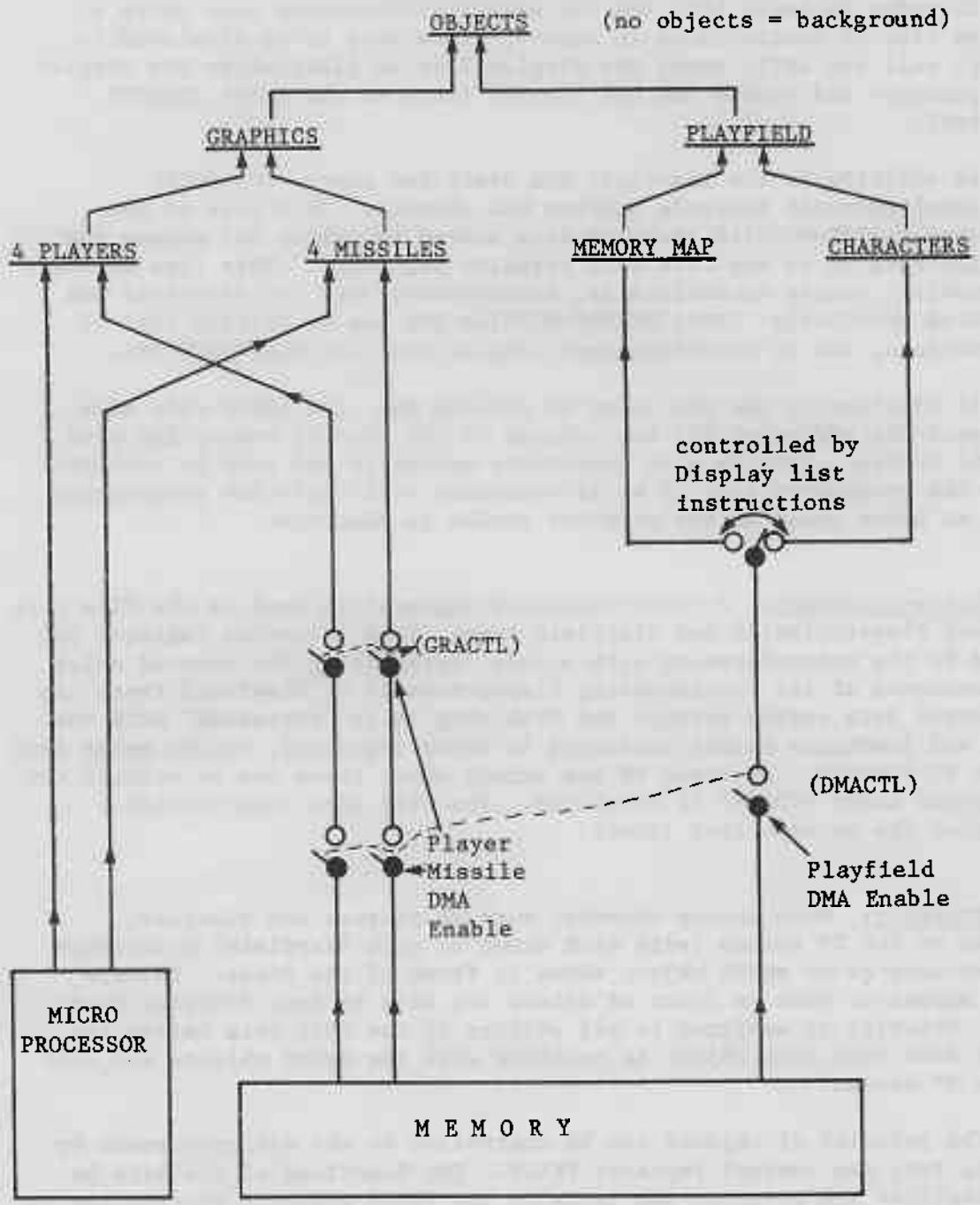


Figure II.2 O B J E C T D I S P L A Y S O U R C E S

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.

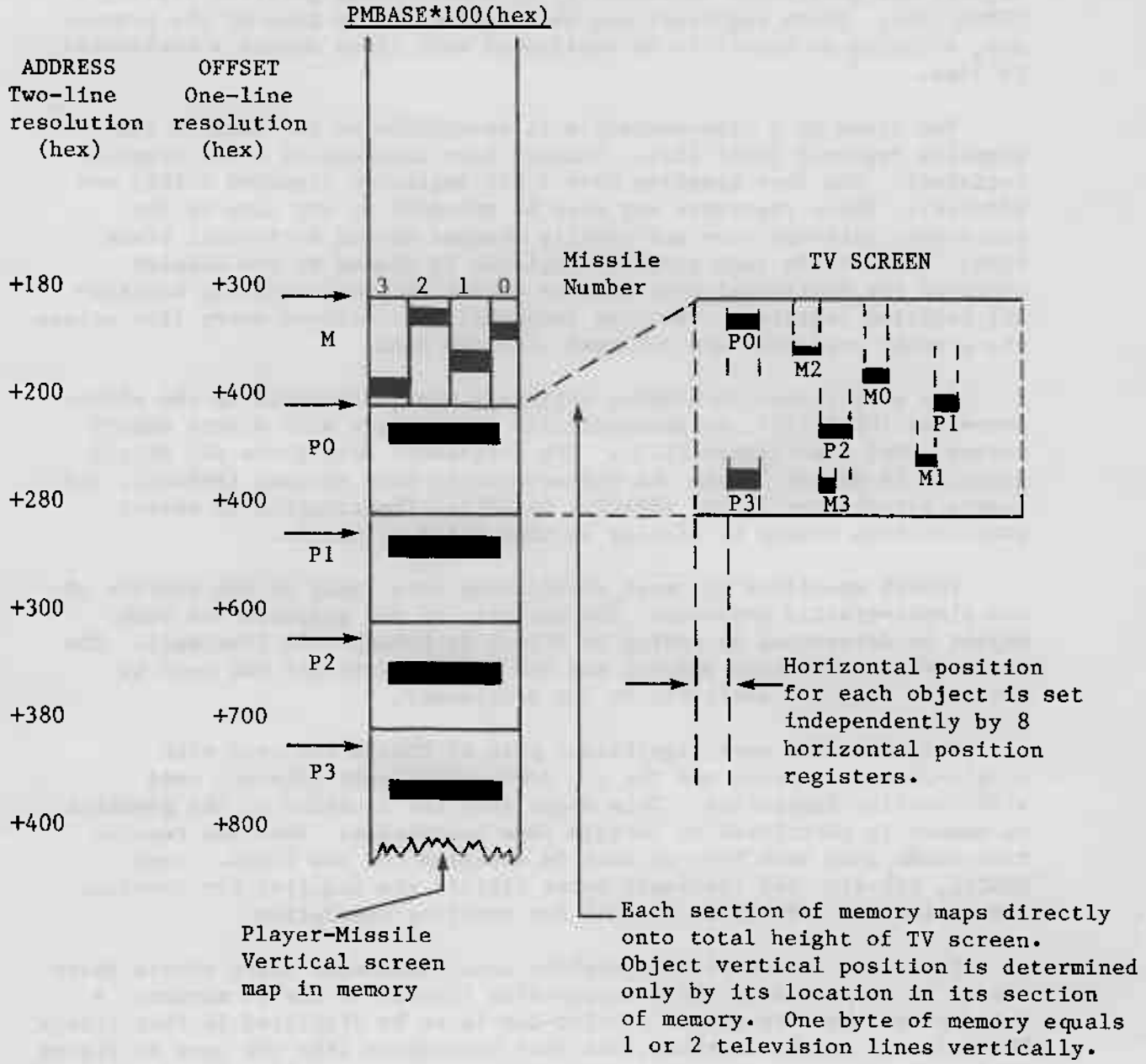


Figure II.2 P L A Y E R - M I S S I L E D M A

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.

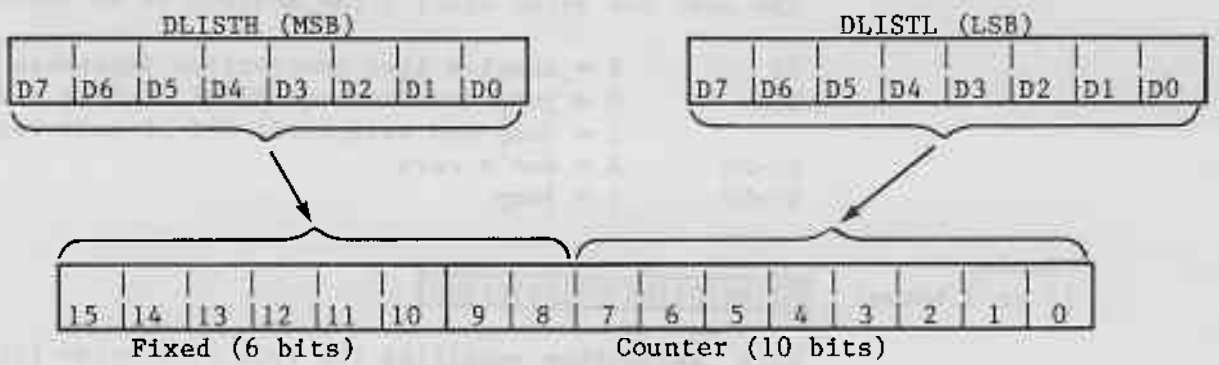
There are three different playfield widths: narrow (128 color clocks), standard (160 color clocks), and wide (192 color clocks). The width is selected by storing into DMACTL. The advantage of a narrower width is that less RAM is required and fewer machine cycles are stolen for DMA. The OS graphics modes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

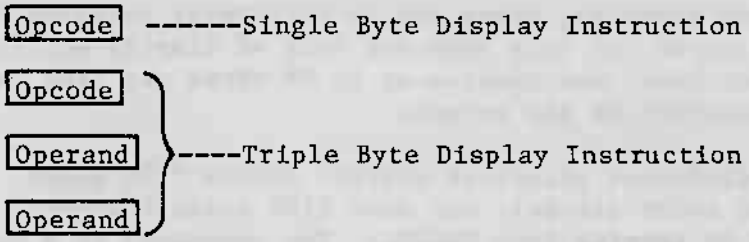
Each instruction defines the type (alpha character or memory map) and the resolution (size of bits on screen) and the location of data in memory to be displayed for a group (1 to 16) of lines. Each group of lines is called a display block.

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

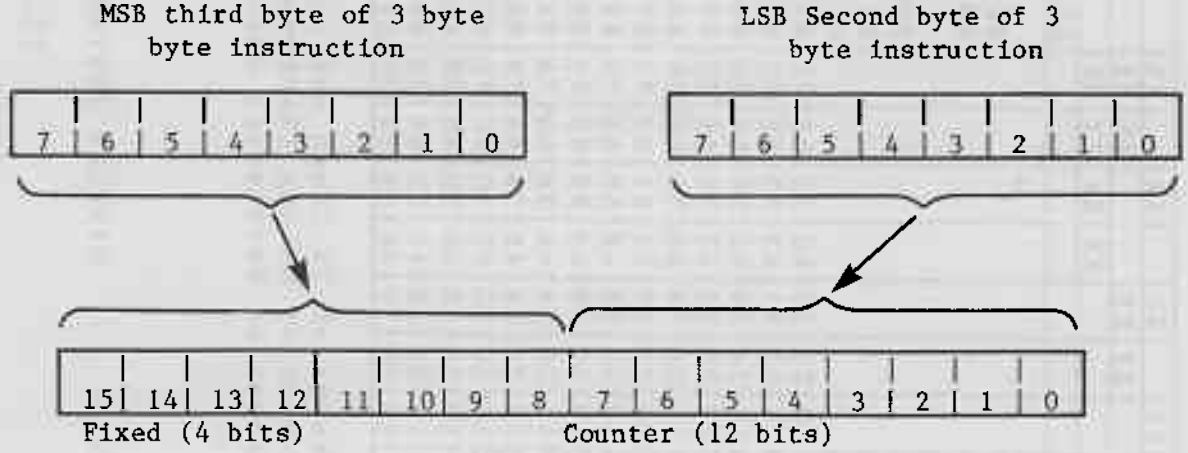
Bit 7 of a display list instruction can be set to create a display list interrupt if bit 7 of NMIEN is set. The display list interrupt code can change the colors or graphics during the middle of the TV display. The type of interrupt is determined by checking NMIST. NMIRES clears NMIST. The current OS 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).

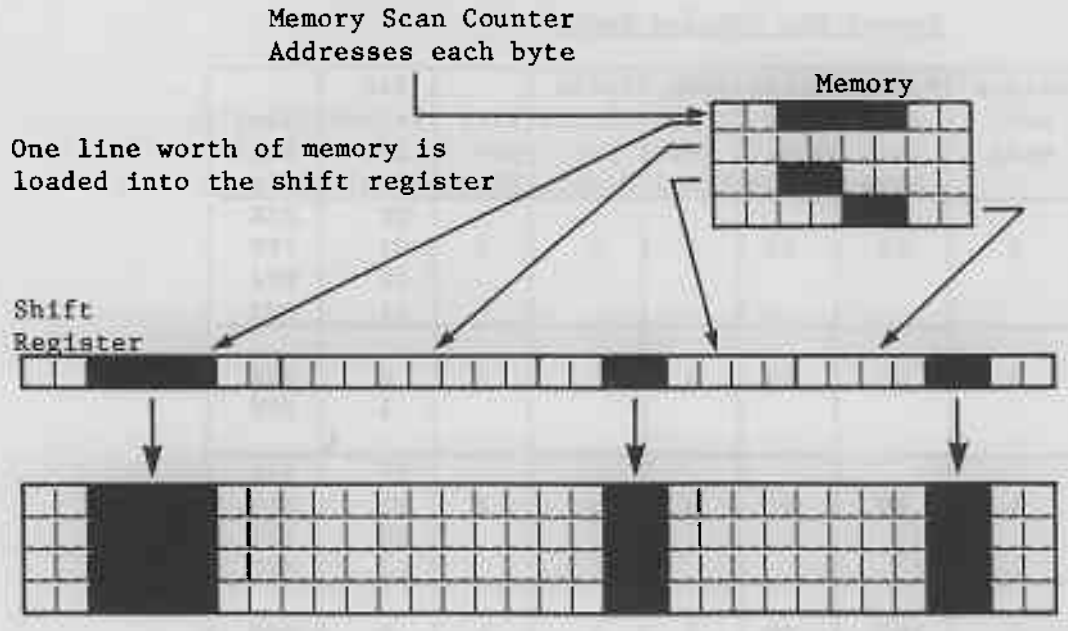
Memory Scan Counter: This counter is not directly accessible by the programmer. It is loaded with the value in the last 2 bytes of a 3 byte (non-Jump) instruction.

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.

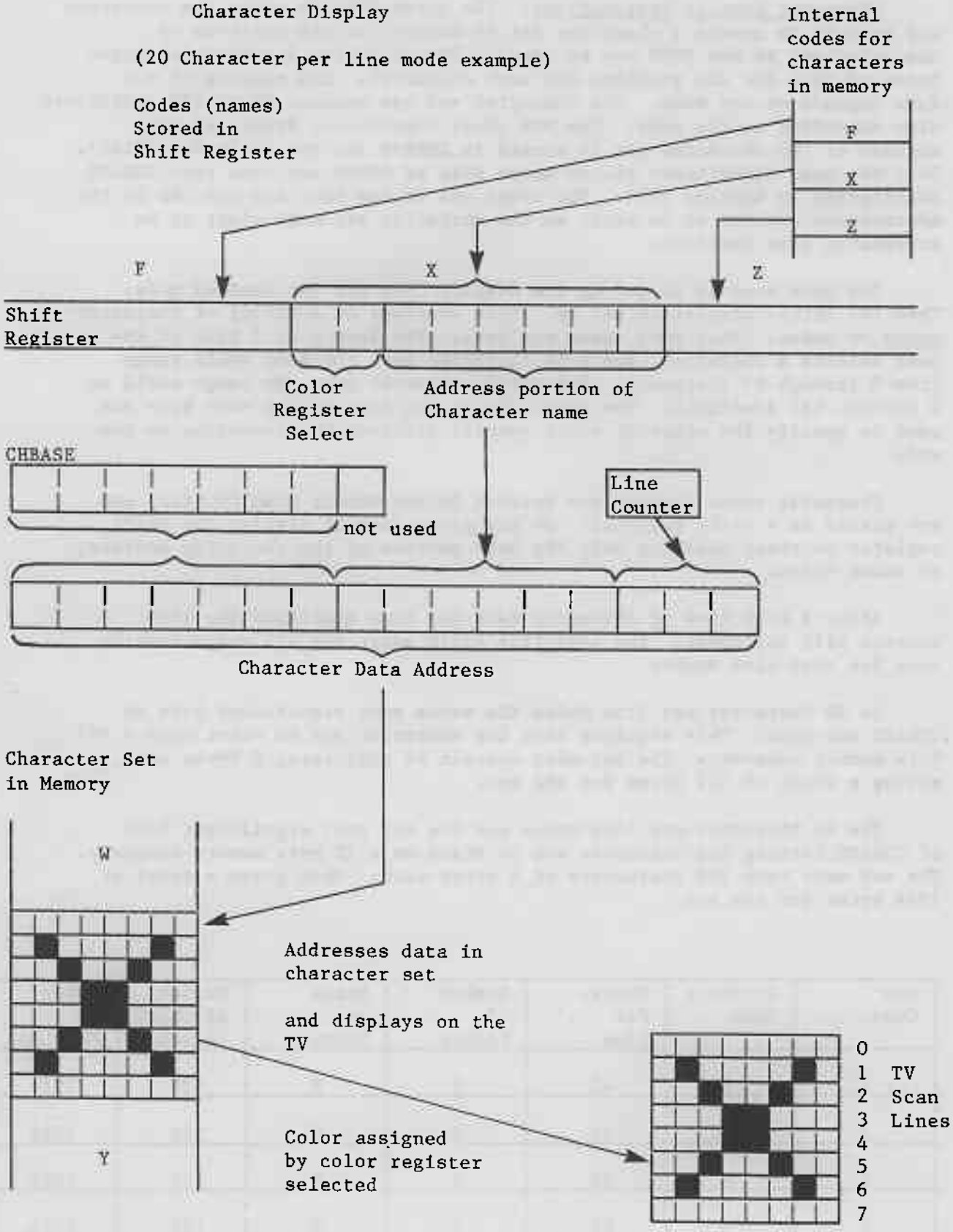
Character names (codes) are fetched by the memory scan counter, and are placed in a shift register. On any given line of display the shift register rotates, changing only the name portion of the character address, as shown below.

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).

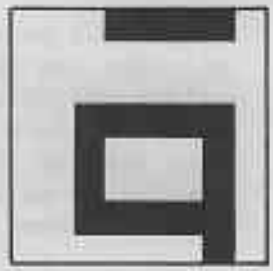
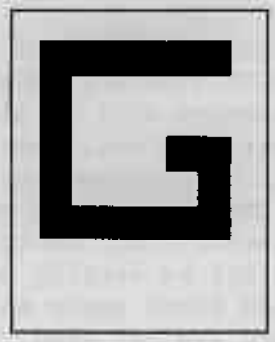
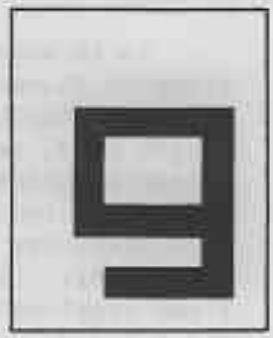
In IR modes 2 and 3, bit 7 of the character name is used for inverse video or blanking. This is controlled by CHACTL (Character Control). If bit 2 of CHACTL is a one then all of the characters will be displayed upside down, regardless of mode. If CHACTL bit 1 is set, then each character which has bit 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.

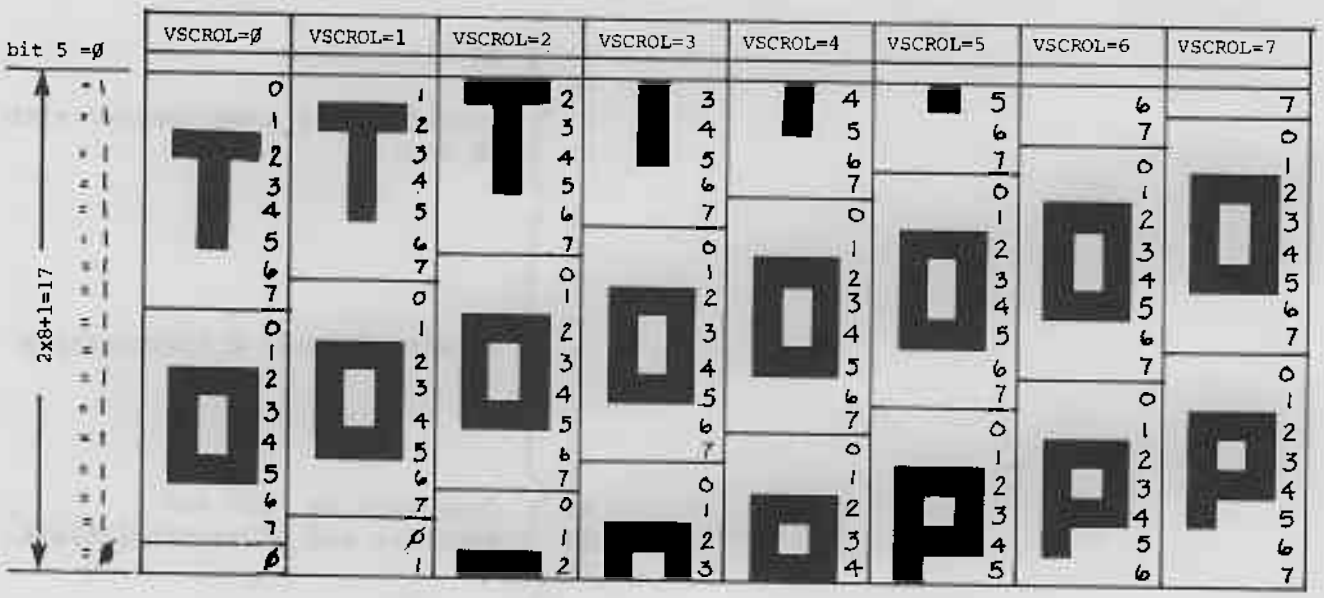
Vertical Scrolling: A zone of playfield on the screen can be scrolled upward by using VSCROL and bit 5 of the display list instruction. The display blocks at the upper and lower boundaries of the zone must have a variable vertical size. In particular, the first display block within that zone must be 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.

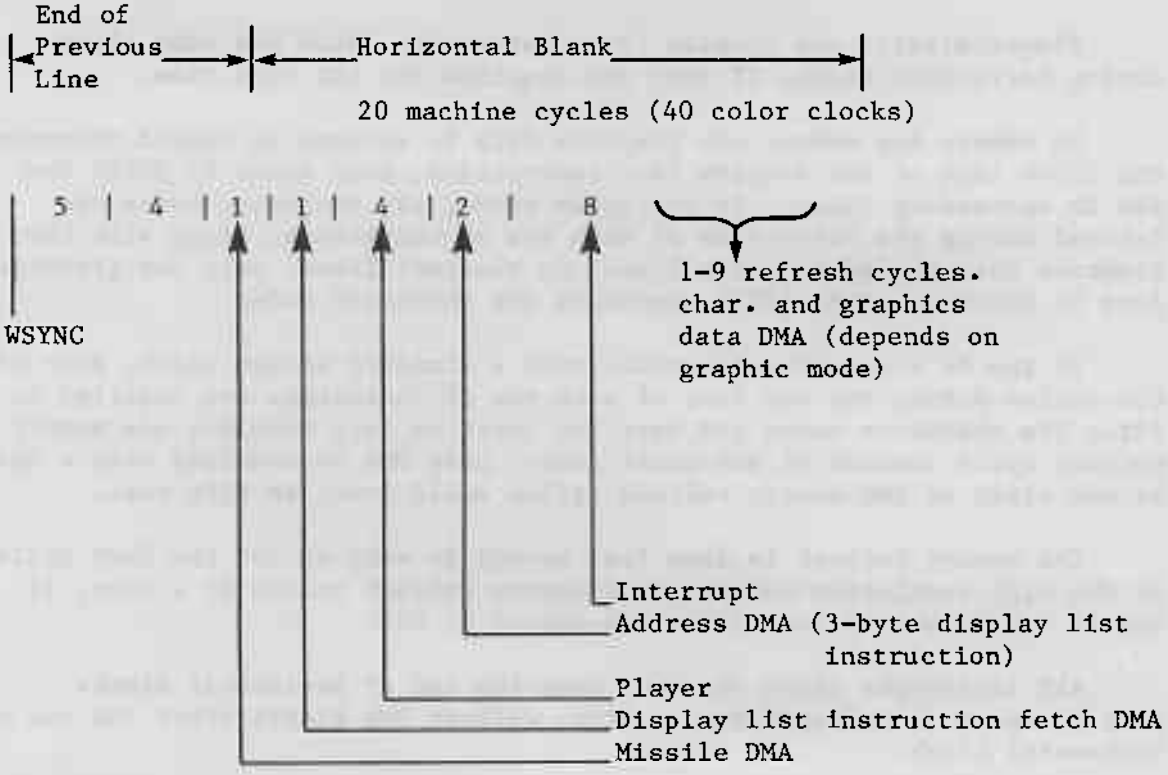
ANTIC does horizontal scrolling of an even number of color clocks by delaying the time at which it DMA's the data. To do an odd number of color clocks (which involves half of a machine cycle), ANTIC has a one color clock internal delay.

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

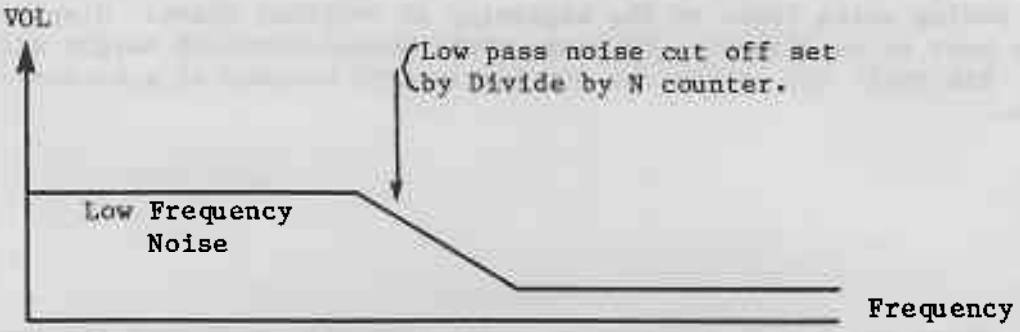
Audio: There are 4 semi-independent audio channels, each with its own frequency, noise, and volume control. Each has an 8 bit "divide by N" frequency divider, controlled by an 8 bit register (AUDFX). (See audio-serial port block diagram.) Each channel also has an 8 bit control register (AUDCX) which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

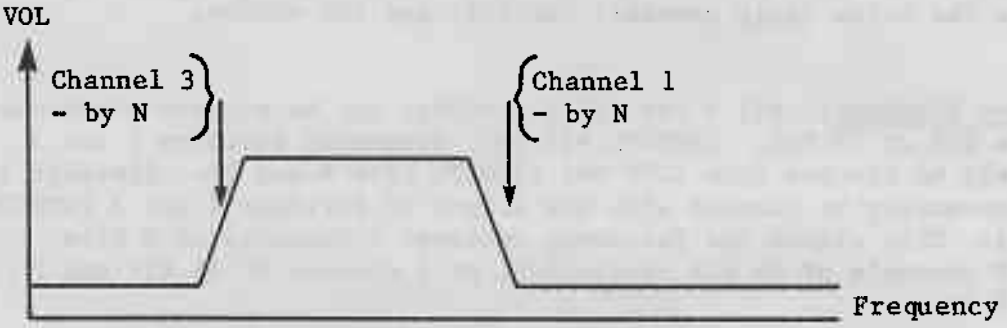
Poly Noise Counters: There are 3 polynomial counters (17 bit, 5 bit and 4 bit) used to generate random noise. The 17 bit poly counter can be reduced to 9 bits (AUDCTL bit 7). These counters are all clocked by 1.79 MHZ. Their outputs, however, can be sampled independently by the four audio channels at 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).

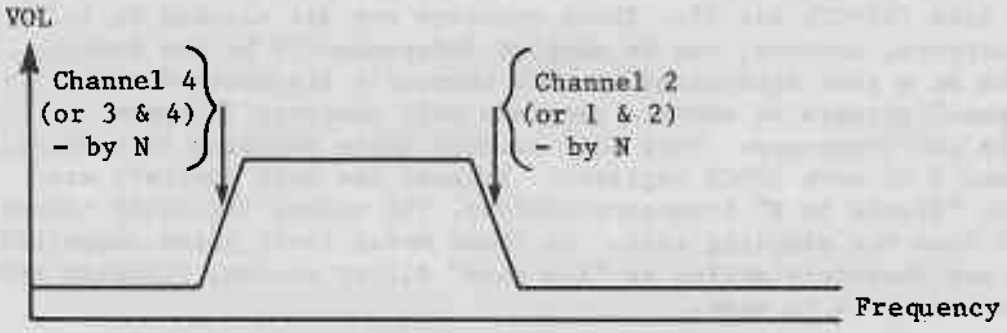
Audio Noise Filters:



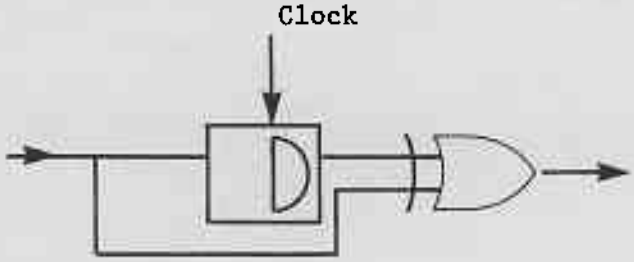
Any channel noise output (without high pass filter)



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

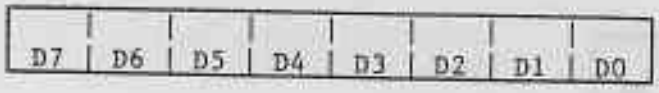
D7 = 0 Break Key Interrupt
D6 = 0 Other Key Interrupt
D5 = 0 Serial Input Data Ready Interrupt
D4 = 0 Serial Output Data Needed Interrupt
D3 = 0 Serial Output (Byte) Transmission Finished Interrupt *
D2 = 0 Timer 4 Interrupt
D1 = 0 Timer 2 Interrupt
D0 = 0 Timer 1 Interrupt

---

* - 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



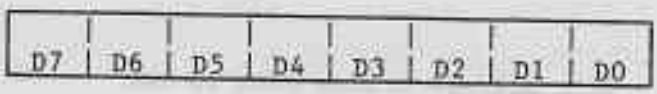
D7 Break Key Interrupt Enable
D6 Other Key Interrupt Enable
D5 Serial Input Data Ready Interrupt Enable
D4 Serial Output Data Needed Interrupt Enable
D3 Serial Out Transmission Finished Interrupt Enable
D2 Timer 4 Interrupt Enable
D1 Timer 2 Interrupt Enable
D0 Timer 1 Interrupt Enable

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

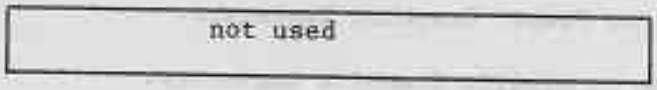
VCOUNT (Vertical Counter)(D40B): This address reads the Vertical TV Line Counter (8 most significant bits).



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)

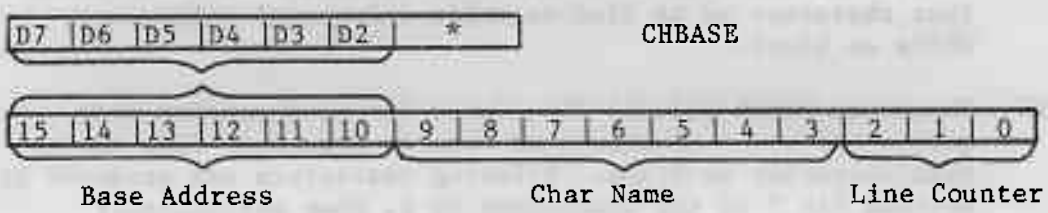
The Display List is a list of display instructions in memory. These instructions are addressed by the Display List Counter. Loading these registers defines the address of the beginning of the Display List. (See sections I and II.)

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.

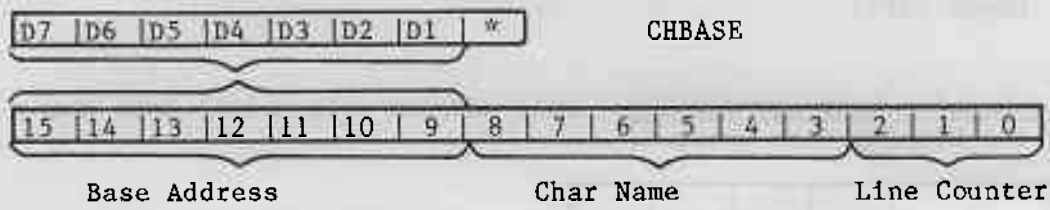
DLISTL and DLISTH should be changed only during vertical blank or with DMA disabled. Otherwise, the screen may roll. Bit 7 of NMIEN must be set in order to receive display list interrupts.

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)

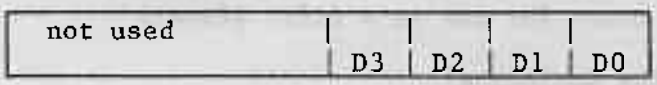
PMBASE (Player-Missile Address Base Register)(D407): This address writes data into the Player-Missile Address Base Register. The data specifies the MSB of the address of the player and missile DMA data (see section II).

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.

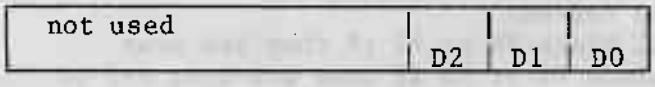


0 to 15 color
clock right shifts

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.

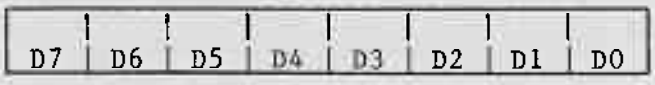


8 line display modes

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.

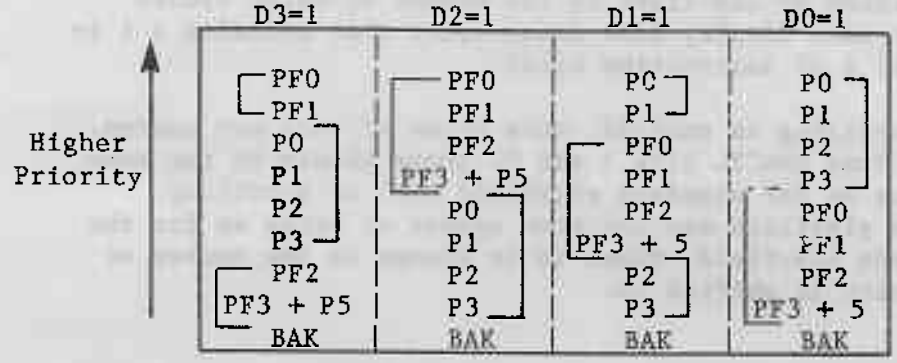


D7-D6 = 0 D5

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)

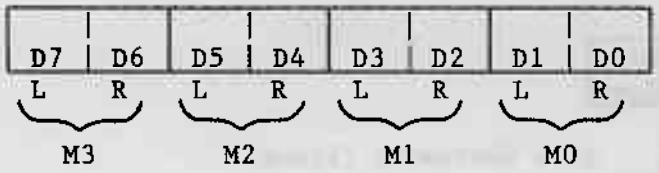
GRAFP0 - GRAFP3 (Player Graphics Registers): (P0 D00D, P1 D00E, P2 D00F, P3 D010): These addresses write data directly into the Player Graphics Registers, independent of DMA. If DMA is enabled then the graphics registers will be loaded automatically from the memory area specified by PMBASE(see page II.3).

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.

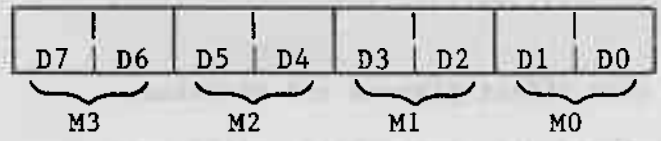


SIZEP0 - SIZEP3 (Player Size)(P0 D008, P1 D009, P2 D00A, P3 D00B): These addresses write data into the Player Size Control Registers.

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)

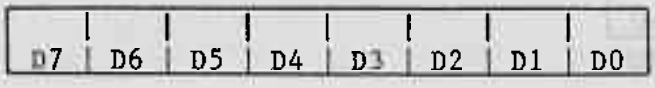
With normal size objects, each bit in the graphics register corresponds to one color clock. For larger objects, each bit is extended over more than one color clock.

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).

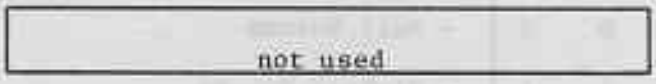
AUDC1, AUDC2, AUDC3, AUDC4 (Audio Channel Control)(D201, D203, D205, D207): These addresses write data into each of the four Audio Control Registers. Each Register controls the noise content and volume of the corresponding Audio Channel.

	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

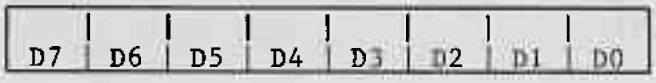
PITCH VALUES FOR THE MUSICAL NOTES-AUDCTL =0, AUDC = hex AX

		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).

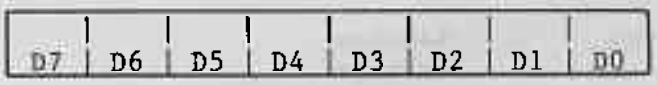
KEYCODE TO ATASCII CONVERSION

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)

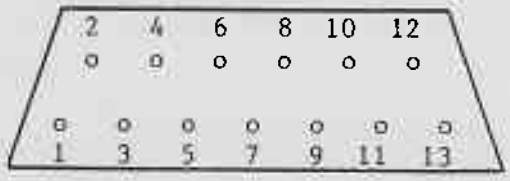
SKRES (Reset above Status Register)(D20A): This write address resets bits 7, 6, and 5 of the Serial Port-Keyboard Status Register to 1.

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

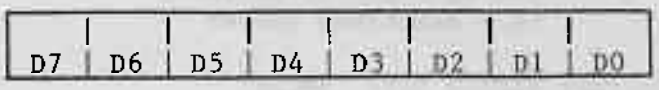
Serial I/O Port Connector Pinout:



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).

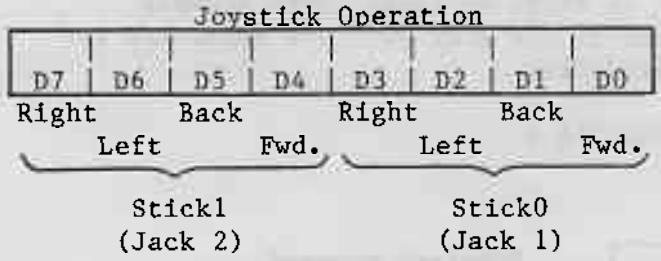


I. CONTROLLER PORTS (front of console)

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

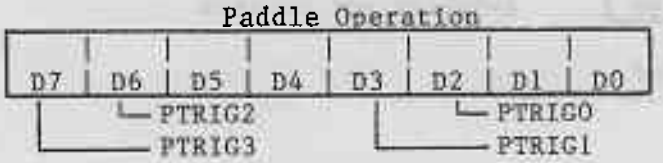
Data Register-Addressed if bit 2 of PACTL is 1.

Joystick Operation



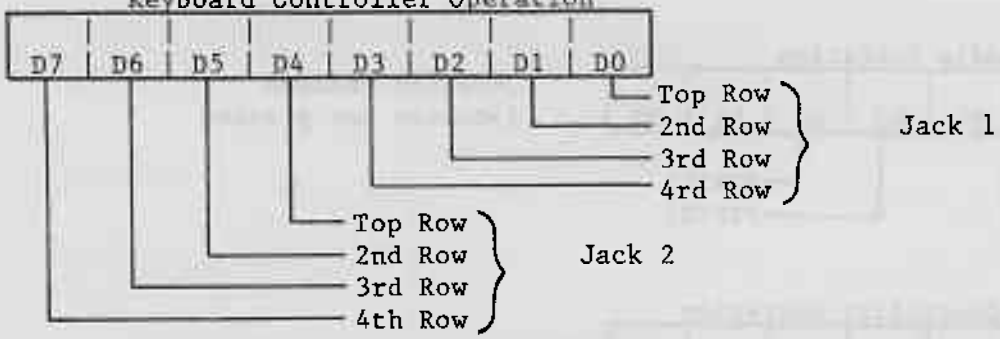
0=Switch pressed
1=Switch not pressed

Paddle Operation

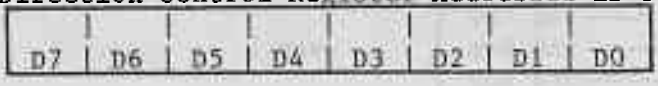


0=Switch pressed
1=Switch not pressed

Keyboard Controller Operation



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

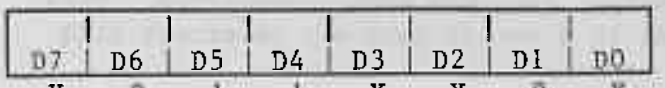


Each bit corresponds to a jack pin

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.

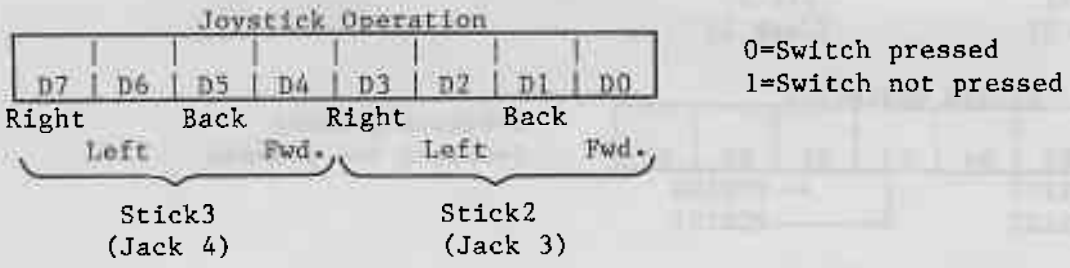


Port A Control Register
Set up register as shown (X = described below)

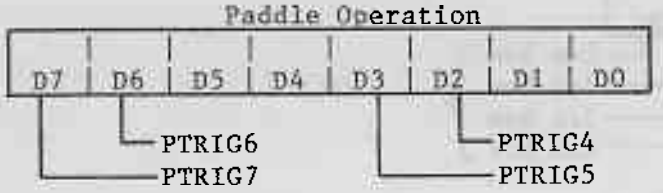
D7 - (Read only) Peripheral A Interrupt Status Bit. Serial bus Proceed line. (Reset by reading Port A Register. Set by Peripheral A Interrupt.)
D3 - Peripheral Motor Control line on serial bus (write). (0 = On 1 = Off)
D2 - Controls Port A addressing described above (write). (1 = Port A Register 0 =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

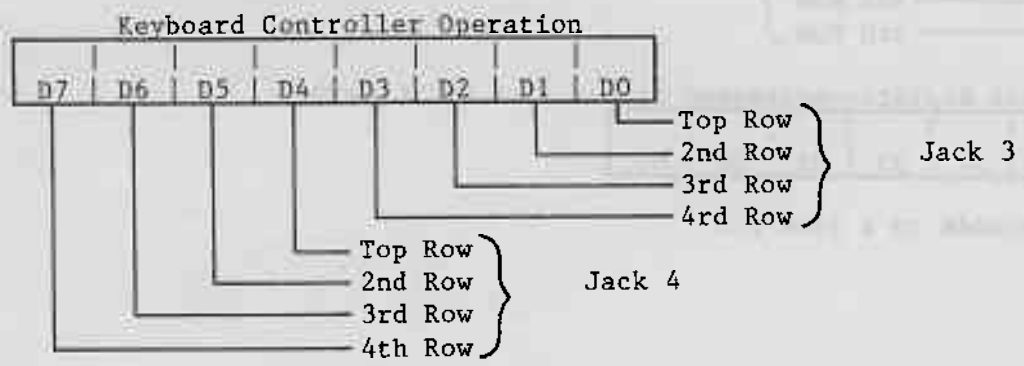
Data Register-Addressed if bit 2 of PBCTL is 1



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).

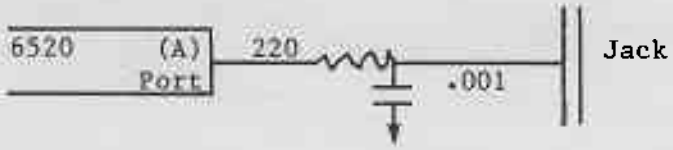
D7	D6	D5	D4	D3	D2	D1	D0	
LP8	7	6	5	4	3	2	1	0 LP0 not read. Two line resolution supplied.

OS SHADOW: LPENV (hex 235)

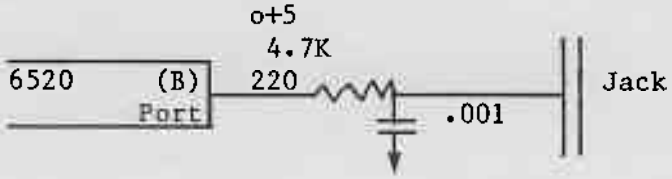
Front Panel (Controller) Jacks as I/O Parts:

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

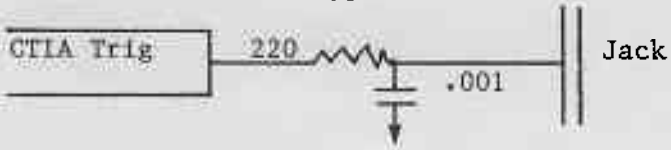
Port A Circuit (typical):



Port B Circuit (typical):



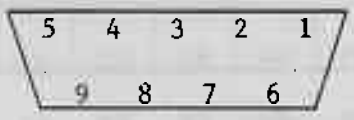
"Trigger" Port Circuit (typical):



Controller Port Pinout:

Male
(console)

Female
(connector)

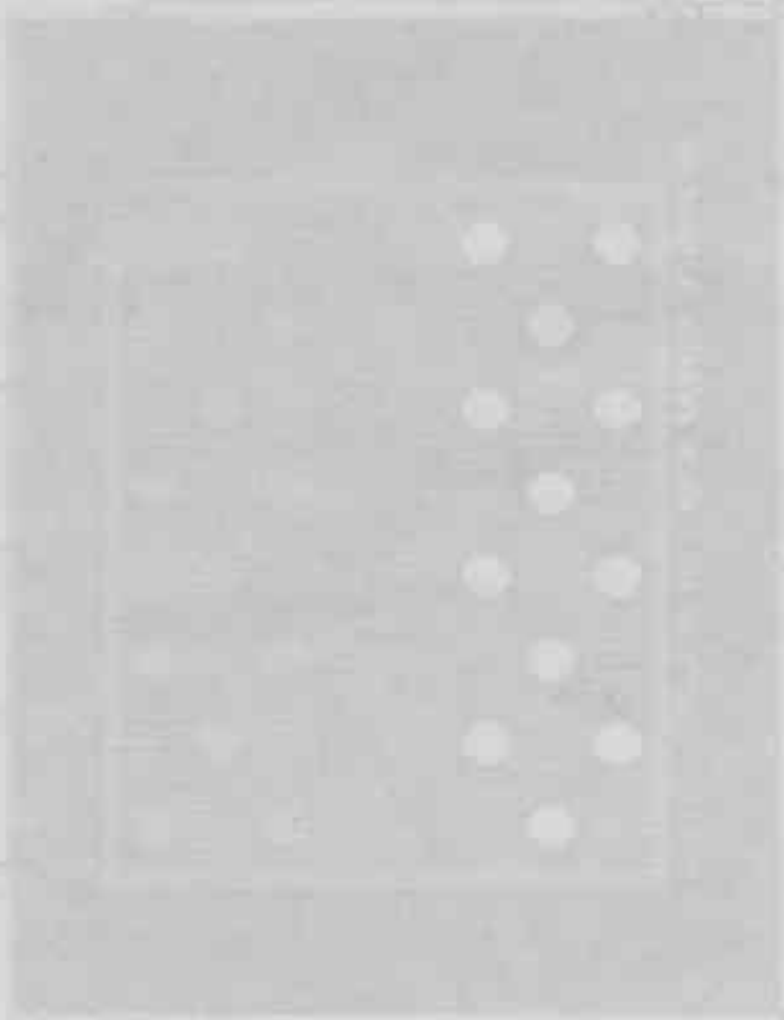


PIN	Controllers JOYSTICK	PADDLE (POT)	KEYBOARD	HARDWARE REGISTERS	OS VARIABLES
1	Forward		Top Row*	Bit 0 or 4**	Bit 0***
2	Back		2nd Row*	Bit 1 or 5**	Bit 1***
3	Left	A(Left)Trigger	3rd Row*	Bit 2 or 6**	PTRIG0,2,4,6 Bit 2***
4	Right	B(Right)Trigger	Bottom Row*	Bit 3 or 7**	PTRIG1,3,5,7 Bit 3***
5		POT 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

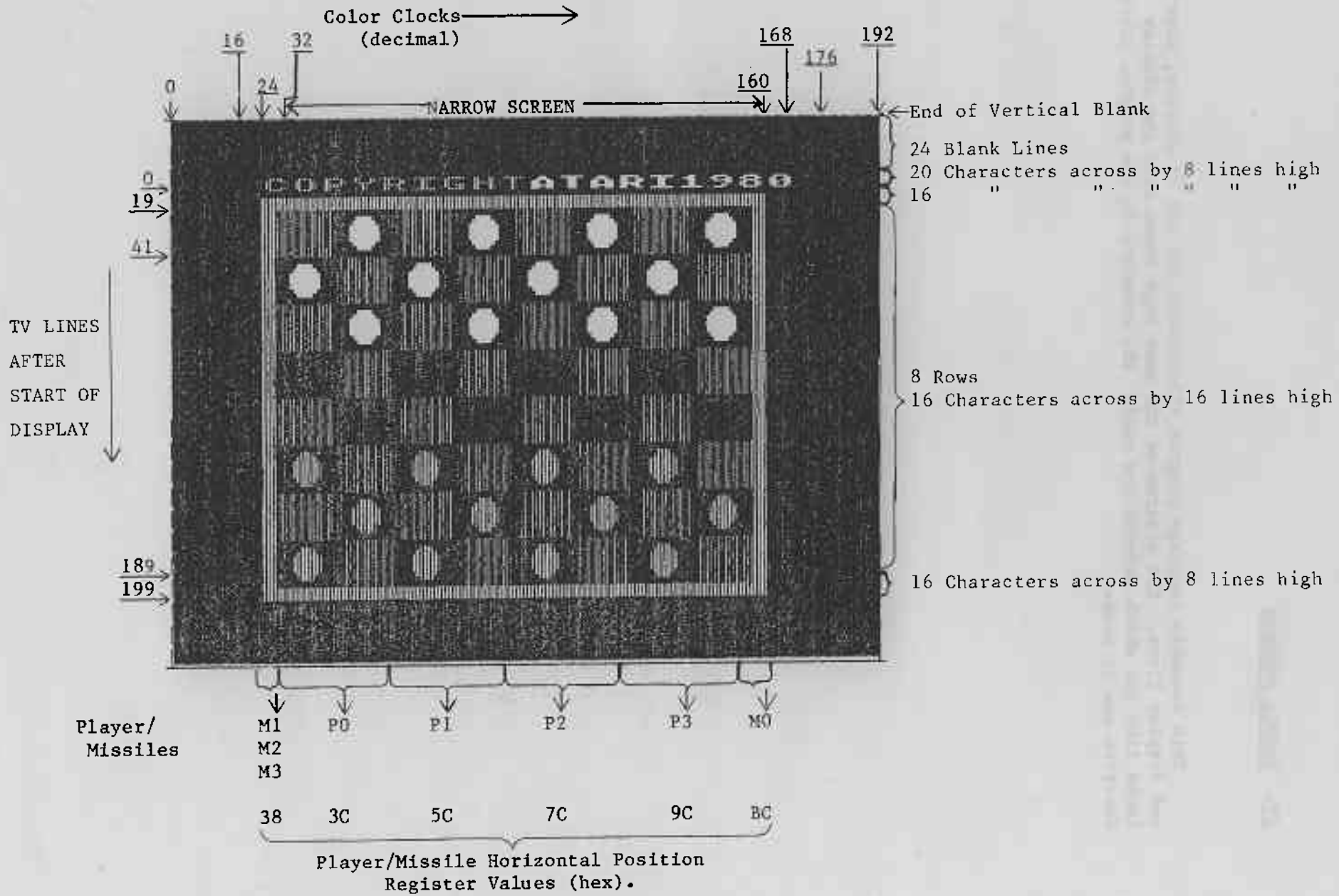
* Write
** PORTA or PORTB
*** STICK 0, 1, 2 or 3

IV. SAMPLE PROGRAM

This assembly language program illustrates the use of players, missiles, and display lists. The diagram on the next page shows what the display looks like and which objects are used. The comments in the program listing describe how it works.



CHECKERS DISPLAY



```
0000                    10          TITLE "ATARI 800 CHECKERS DISPLAY BY C. SHAW 2/31/80"
                        20 ,
                        30 ;COPYRIGHT ATARI 1980
                        40 ;
                        50 ;THIS IS AN EXAMPLE OF A DISPLAY LIST WHICH USES CHARACTER MAPPING TO
                        60 ;PRODUCE THE CHECKERS AND THE TOP AND BOTTOM BORDERS OF THE BOARD.
                        70 ;PLAYERS ARE USED FOR THE RED SQUARES.   THIS GIVES 6 COLORS WITHOUT
                        80 ;CHANGING THE COLOR REGISTERS.
                        90 ;MISSILES ARE USED FOR THE LEFT AND RIGHT BORDERS
                        0100 ;THE PROGRAM STARTS AT THE LOCATION SPECIFIED BY PMB.
                        0110 ;A FEW TRICKS ARE USED TO SAVE RAM, BUT FURTHER OPTIMIZATION IS POSSIBLE
                        0120 ;THIS IS A RAM BASED PROGRAM WHICH RUNS WITH THE ASSEMBLER CARTRIDGE, NOT A
                        0130 ;ROM CARTRIDGE.
                        0140 ;
                        0150 ;COLLEEN (ATARI 800) EQUATES
                        0160 ;
D409                    0170 CHBASE =    $D409
D400                    0180 DMACTL =    $D400
022F                    0190 SDMCTL =    $022F
D000                    0200 HPOSP0 =    $D000
D008                    0210 SIZEP0 =    $D008
02C0                    0220 PCOLR0 =    $02C0
0230                    0230 SDLSTL =    $0230
0231                    0240 SDLSTH =    $0231
D01D                    0250 GRACTL =    $D01D
D407                    0260 PMBASE =    $D407
026F                    0270 GPRIOR =    $026F
0200                    0280 VDSLST =    $0200
D40E                    0290 NMIEN  =    $D40E
                        0300 ;
                        0310 ;DISPLAY LIST EQUATES
                        0320 ;
0080                    0330 INT    =    $80       ;DISPLAY LIST INTERRUPT (BIT 7 OF NMI STATUS)
0041                    0340 JMPWT  =    $41       ;JUMP AND WAIT UNTIL END OF NEXT VERTICAL BLANK (3 BYTES)
0040                    0350 RELOAD =    $40       ;RELOAD MEM SCAN COUNTER (3 BYTES)
0020                    0360 VSC    =    $20       ;VERTICAL SCROLL ENABLE
0010                    0370 HSC    =    $10       ;HORIZONTAL SCROLL ENABLE
0001                    0380 JUMP   =    1         ;JUMP INSTRUCTION (3 BYTES)
0000                    0390 BLANK1 =    0         ;1 BLANK TV LINE
0010                    0400 BLANK2 =    $10       ;2 BLANK LINES
0020                    0410 BLANK3 =    $20       ;3
0030                    0420 BLANK4 =    $30       ;4
0040                    0430 BLANK5 =    $40       ;5
0050                    0440 BLANK6 =    $50       ;6
0060                    0450 BLANK7 =    $60       ;7
0070                    0460 BLANK8 =    $70       ;8 BLANK TV LINES
```

```
ATARI 800 CHECKERS DISPLAY BY C.  SHAW 3/31/80

0000        0470          PAGE
            0480 ;
0020        0490 INTOFF =    $20        ;USED TO GET INTERNAL CODE FOR UPPER CASE ALPHANUMERICS
            0500 ;
            0510 ;INTERNAL CHARACTER CODES
            0520 ;
0000        0530 SPI    =    ' -INTOFF
0021        0540 AI     =    'A-INTOFF
0023        0550 CI     =    'C-INTOFF
0024        0560 DI     =    'D-INTOFF
0025        0570 EI     =    'E-INTOFF
0027        0580 GI     =    'G-INTOFF
0028        0590 HI     =    'H-INTOFF
0029        0600 II     =    'I-INTOFF
002F        0610 OI     =    'O-INTOFF
0030        0620 PI     =    'P-INTOFF
0032        0630 RI     =    'R-INTOFF
0034        0640 TI     =    'T-INTOFF
0039        0650 YI     =    'Y-INTOFF
0011        0660 N1I    =    '1-INTOFF
0018        0670 N8I    =    '8-INTOFF
0019        0680 N9I    =    '9-INTOFF
0010        0690 N0I    =    '0-INTOFF
            0700 ;
            0710 ;CHECKERS EQUATES
            0720 ;
            0730 ;CODES FOR SPECIAL CHECKERS CHARACTER SET
            0740 ;
0000        0750 EMPTY  =    0          ;EMPTY SQUARE
0001        0760 CHECKER=    1          ;ORDINARY CHECKER
0002        0770 KING   =    2
0003        0780 CURS   =    3          ;CURSOR (X)
0004        0790 BORDER =    4          ;USED FOR TOP AND BOTTOM BORDERS OF BOARD
            0800 ;
0000        0810 CLP0   =    0          ;PLAYER 0 (HUMAN)
0080        0820 CLP1   =    $80        ;PLAYER 1 (COMPUTER)
00C0        0830 CLBOR  =    $C0        ;BORDER COLOR (USED TO SET UP 2 MSB'S OF CHAR)
5000        0840 PMB    =    $5000      ;PLAYER MISSILE BASE ADDRESS & PROGRAM LOCATION
```

```
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

0000        0850          PAGE
            0860 ;
            0870 ; PAM VARIABLES
            0880 ;
0000        0890          *=   PMB
5000        0900 BOARD    *=   *+32       ;CHECKER BOARD (ONLY 32 BLACK SQUARES ARE USED)
5020        0910 T0       *=   *+1        ;TEMP FOR MOVING BOARD TO MEM MAP
            0920 ;
            0930 ;PLAYER AND MISSILE GRAPHICS.
            0940 ;PLAYERS ARE USED FOR SQUARES, MISSILES FOR LEFT AND RIGHT BORDERS.
            0950 ;
5021        0960          *=   PMB+$180
5180        0970 GRM03    *=   *+$80      ;MISSILE GRAPHICS
5200        0980 GRP0     *=   *+$80      ;PLAYER 0 GRAPHICS
5280        0990 GRP1     *=   *+$80      ;PLAYER 1
5300        1000 GRP2     *=   *+$80      ;       2
5380        1010 GRP3     *=   *+$80      ;       3
            1020 ;
5400        1030 TITL     *=   *+20       ;TOP LINE OF CHARS -- ATASCII MESSAGE
5414        1040 TOPBRD   *=   *+16       ;TOP BORDER OF BOARD
5424        1050 BRDSP    *=   8*16+*     ;BOARD DISPLAY
54A4        1060 BOTBRD   *=   *+16       ;BOTTOM BORDER
```

```
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80


54B4            1070        .PAGE
                1080 ;
                1090 ;GR -- SPECIAL CHECKERS CHARACTER SET (ONLY CODES 0-4 ARE USED)
                1100 ;
54B4            1110        *=   PMB+$600
                1120 GR
5600 00         1130        .BYTE 0,0,0,0,0,0,0,0 ;BLANK (0)
5601 00
5602 00
5603 00
5604 00
5605 00
5606 00
5607 00
5608 3C         1140        .BYTE $3C,$7E,$FF,$FF,$FF,$FF,$7E,$3C ;CHECKER (1)
5609 7E
560A FF
560B FF
560C FF
560D FF
560E 7E
560F 3C
5610 3C         1150        .BYTE $3C,$7E,$A5,$A5,$C3,$C3,$7E,$3C ;KING (2)
5611 7E
5612 A5
5613 A5
5614 C3
5615 C3
5616 7E
5617 3C
5618 C3         1160        .BYTE $C3,$66,$3C,$18,$18,$3C,$66,$C3 ;CURSOR (3)
5619 66
561A 3C
561B 18
561C 18
561D 3C
561E 66
561F C3
5620 00         1170        .BYTE 0,$FF,$FF,$FF,$FF,$FF,$FF,0 ;BORDER (4)
5621 FF
5622 FF
5623 FF
5624 FF
5625 FF
5626 FF
5627 00
```

```
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

5628            1180        .PAGE
                1190 ;
                1200 ;
                1210 ;DISPLAY LIST
                1220 ;
                1230 DSP
5628 70         1240        BYTE BLANK8   ;24 BLANK LINES
5629 70         1250        BYTE BLANK8
562A 70         1260        BYTE BLANK8
562B 46         1270        BYTE RELOAD+6 ;LINES 0-7.  MESSAGE LINE: 20 ACROSS X 5 COLOR X 1 LINE RESOLUTION CHARACTERS
562C 0054       1280        WORD TITL
562E 80         1290        BYTE INT+BLANK1 ;8.  INTERRUPT TO CHANGE CHARACTER BASE ADDRESS AND CHANGE TO NARROW SCREEN.
562F 06         1300        BYTE 6        ;9-16.  TOP BORDER: 16 X 5 X 1 CHARS (LAST LINE IS TOP OF 1ST ROW OF SQUARES)
5630 10         1310        BYTE BLANK2   ;17-18.  TOP OF FIRST ROW OF SQUARES
                1320 ;                     CHECKERBOARD (8 LINES OF CHARS WITH SPACES INBETWEEN - 22 LINES/SQUARE)
5631 07         1330        BYTE 7        ;19-34.  16X5X2 LINE RESOLUTION CHARS
5632 50         1340        BYTE BLANK6   ;35-40.  FIRST 3 LINES=BOTTOM OF PREVIOUS SQUARE.
5633 07         1350        BYTE 7        ;41-56
5634 50         1360        BYTE BLANK6   ;57-62.  LAST 3 LINES=TOP OF NEXT SQUARE.
5635 07         1370        BYTE 7        ;63-78
5636 50         1380        BYTE BLANK6   ;79-84
5637 07         1390        BYTE 7        ;85-100
5638 50         1400        BYTE BLANK6   ;101-106
5639 07         1410        BYTE 7        ;107-122
563A 50         1420        BYTE BLANK6   ;123-128
563B 07         1430        BYTE 7        ;129-144
563C 50         1440        BYTE BLANK6   ;145-150
563D 07         1450        BYTE 7        ;151-166
563E 50         1460        BYTE BLANK6   ;167-172
563F 07         1470        BYTE 7        ;173-188
                1480 ;                     NEXT THREE LINES ARE BOTTOM OF PREVIOUS SQUARE.
5640 10         1490        BYTE BLANK2   ;189-190.  END OF NORMAL DISPLAY (SHOULD BE ON SCREEN ON ALL TV'S).
5641 06         1500        BYTE 6        ;191-198.  BOTTOM BORDER (MAY OVERSCAN, BUT NOT ESSENTIAL TO GAME PLAY)
5642 41         1510        BYTE JMPWT    ;WAIT FOR NEXT VBLANK, THEN START OVER
5643 2856       1520        WORD DSP
                1530 ;
                1540 ;
                1550 ;DSP -- DISPLAY LIST INTERRUPT HANDLER.
                1560 ;CHANGES CHARACTER BASE AND WIDTH OF DISPLAY FOR SPECIAL CHECKERS GRAPHICS.
                1570 ;THE OS WILL CHANGE CHBASE BACK TO NORMAL DURING VERTICAL BLANK.
                1580 ;
                1590 NCHR
5645 48         1600        PHA
5646 A956       1610        LDA  #GR/256
5648 8D09D4     1620        STA  CHBASE
                1630 ;
                1640 ;INSTRUCTION FETCH DMA ENABLE, P/M 2 LINE RES, P/M DMA ENABLE, NARROW SCREEN (128 CLOCKS)
564B A92D       1650        LDA  #$2D
564D 8D00D4     1660        STA  DMACTL
5650 68         1670        PLA
5651 40         1680        RTI
```

```
5652            1690        PAGE
                1700 ;
                1710 ;INITIALIZATION CODE -- START EXECUTION HERE
                1720 ;
5652            1730        *=   PMB+$700
                1740 ;
                1750 ;INIT OS'S DMACTL VARIABLE
                1760 ;INSTRUCTION FETCH DMA ENABLE, P/M 2 LINE RES, P/M DMA ENABLE, STANDARD SCREEN (160 CLOCKS)
                1770 ;
5700 A92E       1780        LDA  #$2E
5702 8D2F02     1790        STA  SDMCTL
                1800 ;
                1810 ;CLEAR RAM
                1820 ;
5705 A900       1830        LDA  #0
5707 AA         1840        TAX
                1850 INITLP
5708 9D0050     1860        STA  PMB,X
570B 9D0051     1870        STA  PMB+$100,X
570E 9D0052     1880        STA  PMB+$200,X
5711 9D0053     1890        STA  PMB+$300,X
5714 9D0054     1900        STA  PMB+$400,X
5717 E8         1910        INX
5718 D0EE       1920        BNE  INITLP
                1930 ;
                1940 ;INITIALIZE MISSILE GRAPHICS FOR BORDERS
                1950 ;
571A A90E       1960        LDA  #$0E
571C A05E       1970        LDY  #$5E
571E 999451     1980 LOPZ   STA  GRM03+$14,Y
5721 88         1990        DEY
5722 D0FA       2000        BNE  LOPZ
                2010 ;
                2020 ;INITIALIZE TOP AND BOTTOM BORDERS
                2030 ;
5724 A010       2040        LDY  #16
5726 A9C4       2050        LDA  #CLBOR+BORDER
5728 991354     2060 TBLP   STA  TOPBRD-1,Y
572B 99A354     2070        STA  BOTBRD-1,Y
572E 88         2080        DEY
572F D0F7       2090        BNE  TBLP       ;CONTINUE UNTIL Y=0
                2100 ;
                2110 ;INITIALIZE PLAYER GRAPHICS FOR SQUARES (CHECKER BOARD) Y=0
                2120 ;
5731 A9F0       2130        LDA  #$F0
5733 A20A       2140 IN2    LDX  #10
5735 991852     2150 IN3    STA  GRP0+$18,Y
5738 999852     2160        STA  GRP1+$18,Y
573B 991853     2170        STA  GRP2+$18,Y
573E 999853     2180        STA  GRP3+$18,Y
                2190 ;
```

```
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

5741 48      2200        PHA
5742 A90A    2210        LDA  #$0A
5744 999851  2220        STA  GRM03+$18,Y ;REST OF MISSILE GRAPHICS
5747 68      2230        PLA
5748 C8      2240        INY
5749 CA      2250        DEX
574A 10E9    2260        BPL  IN3
574C 49FF    2270        EOR  #$FF       ;FILL IN OPPOSITE SQUARES
574E C058    2280        CPY  #88
5750 90E1    2290        BCC  IN2
5752 A008    2300        LDY  #8
             2310 ;
             2320 ;INITIALIZE PLAYER AND MISSILE POSITIONS AND COLORS
             2330 ;
5754 B9D857  2340 IN4     LDA  ITBL,Y
5757 9900D0  2350        STA  HPOSP0,Y
575A 8A      2360        TXA             ;$FF
575B 9908D0  2370        STA  SIZEP0,Y   ;$03 INDICATES 4 TIMES NORMAL SIZE (REST IS DON'T CARE)
575E B9E057  2380        LDA  ITBL1,Y
5761 99C002  2390        STA  PCOLR0,Y
5764 88      2400        DEY
5765 10ED    2410        BPL  IN4
             2420 ;
             2430 ;OS, ANTIC, POKEY INITIALIZATION
             2440 ;
5767 A928    2450        LDA  #DSP&$FF   ;DISPLAY LIST START ADDRESS (LSB)
5769 8D3002  2460        STA  SDLSTL
576C A956    2470        LDA  #DSP/256   ;MSB OF ADDRESS
576E 8D3102  2480        STA  SDLSTH
5771 A903    2490        LDA  #3         ;ENABLE PLAYER/MISSILE DMA TO GRAPHICS REGS.
5773 8D1DD0  2500        STA  GRACTL
5776 A950    2510        LDA  #PMB/256   ;MSB OF ADDRESS OF PLAYER/MISSILE GRAPHICS
5778 8D07D4  2520        STA  PMBASE
577B A914    2530        LDA  #$14       ;5TH PLAYER ENABLE (USE PF3 FOR MISSILE COLOR), PF TAKES PRIO OVER PLAYERS
577D 8D6F02  2540        STA  GPRIOR     ;OS PRIORITY REG
5780 A945    2550        LDA  #NCHR&$FF ;DISPLAY LIST INTERRUPT VECTOR (LSB)
5782 8D0002  2560        STA  VDSLST
5785 A956    2570        LDA  #NCHR/256
5787 8D0102  2580        STA  VDSLST+1
578A 8E0ED4  2590        STX  NMIEN      ;X=$FF   $C0 ENABLES DISPLAY LIST & VBLANK INTERRUPTS.
             2600 ;
             2610 ;INITILIZE BOARD DISPLAY
             2620 ;
578D A20B    2630        LDX  #11
             2640 BRDLP
578F A981    2650        LDA  #CHECKER+CLR0 ,HUMAN PIECES ON SQUARES 0-11
5791 9D0050  2660        STA  BOARD,X
5794 A981    2670        LDA  #CHECKER+CLR1 ,COMPUTER PIECES ON SQUARES 20-31
5796 9D1450  2680        STA  BOARD+20,X
5799 CA      2690        DEX
579A 10F3    2700        BPL  BRDLP
             2710 ;
```

```
                2720 ;MOVE COPYRIGHT MESSAGE TO MESSAGE DISPLAY LINE
                2730 ;
579C A213       2740        LDX  #19
579E BDE957     2750 IN6    LDA  COPY,X
57A1 9D0054     2760        STA  TITL,X
57A4 CA         2770        DEX
57A5 10F7       2780        BPL  IN6
                2790 ;
                2800 ;LOOP TO MOVE BOARD TO GRAPHICS AREA.
                2810 ;THE CHECKERS PROGRAM LOGIC COULD BE ADDED HERE OR A VBLANK INTERRUPT COULD BE USED.
                2820 ;
                2830 LOOP
57A7 20AD57     2840        JSR  UPCHR
57AA 4CA757     2850        JMP  LOOP
                2860 ;
                2870 ;
                2880 ;
                2890 ;
                2900 ;UPCHR -- SUBROUTINE TO MOVE 32 BYTES OF CHECKER BOARD TO DISPLAY RAM.
                2910 ;
                2920 UPCHR
57AD A21F       2930        LDX  #31        ;SQUARE 31 = UPPER LEFT
57AF A000       2940        LDY  #0
                2950 UPLP1
57B1 A903       2960        LDA  #4-1       ;4 SQUARES/LINE
57B3 8D2050     2970        STA  T0
                2980 UPLP2
57B6 BD0050     2990        LDA  BOARD,X
57B9 992654     3000        STA  BRDSP+2,Y  ;FOR ROWS SHIFTED TO RIGHT
57BC BDFC4F     3010        LDA  BOARD-4,X
57BF 993454     3020        STA  BRDSP+$10,Y ;FOR ROWS SHIFTED TO LEFT
57C2 C8         3030        INY
57C3 C8         3040        INY
57C4 C8         3050        INY
57C5 C8         3060        INY
57C6 CA         3070        DEX
57C7 CE2050     3080        DEC  T0
57CA 10EA       3090        BPL  UPLP2
                3100 ;
57CC 98         3110        TYA
57CD 18         3120        CLC
57CE 6910       3130        ADC  #$10
57D0 A8         3140        TAY
57D1 8A         3150        TXA
57D2 E903       3160        SBC  #4-1       ;CARRY IS CLEAR (SUBTRACT 4)
57D4 AA         3170        TAX
57D5 B0DA       3180        BCS  UPLP1
57D7 60         3190        RTS
                3200 ;
                3210 ;
                3220 ;
                3230 ;
```

```
                3240 ;DATA
                3250 ;HORIZONTAL POSITION OF PLAYERS (SQUARES) AND MISSILES (SIDE BORDERS).
                3260 ;M0=RIGHT BORDER, M1=LEFT BORDER
                3270 ;M2 & M3 ARE PLACED WITH M1.
                3280 ;                P0, P1, P2, P3, M0, M1, M2, M3
                3290 ITBL
57D8 3C         3300        .BYTE $3C,$5C,$7C,$9C,$BC,$38,$38,$38
57D9 5C
57DA 7C
57DB 9C
57DC BC
57DD 38
57DE 38
57DF 38
                3310 ;
                3320 ;COLOR TABLE
                3330 ITBL1
57E0 34         3340        .BYTE $34,$34,$34,$34 ;4 PLAYERS (RED SQUARES)
57E1 34
57E2 34
57E3 34
57E4 36         3350        .BYTE $36       ;PF0: RED CHECKERS AND MESSAGES
57E5 88         3360        .BYTE $88       ;PF1: BLUE CHARACTERS
57E6 0E         3370        .BYTE $0E       ;PF2: WHITE CHECKERS AND MESSAGES
57E7 26         3380        .BYTE $26       ;PF3: YELLOW BORDER (CHARS & MISSILES)
57E8 00         3390        .BYTE 0         ;BK: BLACK BACKGROUND
                3400 ;
                3410 ;"COPYRIGHT ATARI 1980" MESSAGE
                3420 ;
0000            3430 OF     =    $00        ;FOR PF0 COLOR (RED)
0080            3440 OF2    =    $80        ;FOR PF2 COLOR (WHITE)
0040            3450 OF3    =    $40        ;FOR PF1 COLOR (BLUE)
                3460 TGTBL
57E9 00         3470 COPY   .BYTE SPI,CI+OF,OI+OF,PI+OF,YI+OF,RI+OF,II+OF,GI+OF,HI+OF,TI+OF
57EA 23
57EB 2F
57EC 30
57ED 39
57EE 32
57EF 29
57F0 27
57F1 28
57F2 34
57F3 A1         3480        .BYTE AI+OF2,TI+OF2,AI+OF2,RI+OF2,II+OF2,N1I+OF3,N9I+OF3,N8I+OF3,N0I+OF3
57F4 B4
57F5 A1
57F6 B2
57F7 A9
57F8 51
57F9 59
57FA 58
57FB 50
```

V. HARDWARE REGISTER LISTS

A. ADDRESS ORDER

CTIA ADDRESSES				
	WRITE		READ	
Address	Name	Description	Name	Description
D0FF ↕ D020	REPEAT AS BELOW		7 MORE TIMES	
D01F	CONSOL	Write Consol SW.Port	CONSOL	Read Consol SW. Port
D01E	HITCLR	Collision Clear		
D01D	GRACTL	Graphic Control		
D01C	VDELAY	Vert. Delay		
D91B	PRIOR	Priority Select		
D01A	COLBK	Col-lum Bkgnd		
D019	COLPF3	Color-lum of 3		
D018	COLPF2	Playfield 2		
D017	COLPF1	Playfield 1		
D016	COLPF0	Playfield 0		
D015	COLPM3	Color-lum of 3		
D014	COLPM2	Player-Missile 2	PAL	READ PAL/NTSC bits
D013	COLPM1	Player-Missile 1	TRIG3	Read Joystick
D012	COLPM0	Player-Missile 0	TRIG2	Trigger
D011	GRAFM	Graphics All Missiles	TRIG1	Buttons
D010	GRAFP3	Graphics Player 3	TRIG0	
D00F	GRAFP2	Graphics Player 2	P3PL	Read Player
D00E	GRAFP1	Graphics Player 1	P2PL	to Player
D00D	GRAFP0	Graphics Player 0	P1PL	Collisions
D00C	SIZEM	Size All Missiles	P0PL	
D00B	SIZEP3	Size Player 3	M3PL	Read Missile
D00A	SIZEP2	Size Player 2	M2PL	To Player
D009	SIZEP1	Size Player 1	M1PL	Collisions
D008	SIZEP0	Size Player 0	M0PL	
D007	HPOSM3	Horz. Posit. Missile 3	P3PF	Read Player
D006	HPOSM2	Horz. Posit. Missile 2	P2PF	To Playfield
D005	HPOSM1	Horz. Posit. Missile 1	P1PF	Collisions
D004	HPOSM0	Horz. Posit. Missile 0	P0PF	
D003	HPOSP3	Horz. Posit. Player 3	M3PF	Read Missile
D002	HPOSP2	Horz. Posit. Player 2	M2PF	To Playfield
D001	HPOSP1	Horz. Posit. Player 1	M1PF	Collisions
D000	HPOSP0	Horz. Posit. Player 0	M0PF	

ANTIC ADDRESSES

Address	WRITE		READ	
	Name	Description	Name	Description
D4FF ↕ D410	REPEAT (AS BELOW)		15 MORE TIMES	
D40F	NMIRES	Reset NMI Interrupt Status NMI Interrupt	NMIST	NMI Interrupt Status Register
D40E	NMIEN	ENABLE		
D40D			PENV	Light Pen Register Vertical
D40C			PENH	Light Pen Register Horizontal
D40B			VCOUNT	Vertical Line Counter
D40A	WSYNC	Wait for HBLANK Synchronism		
D409	CHBASE	Character Base Address Red		
D408				
D407	PMBASE	Player-Missile Base Address Register		
D406				
D405	VSCROL	Vertical Scroll Register		
D404	HSCROL	Horizontal Scroll Register		
D403	DLISTH	Display List Pointer (High Byte)		
D402	DLISTL	Display List Pointer (Low Byte)		
D401	CHACTL	Character Control Register		
D400	DMACTL	DMA Control Register		

POKEY ADDRESSES

	WRITE		READ	
	Name	Description	Name	Description
D2FF ↕ D210	REPEAT (AS BELOW)		15 MORE TIMES	
D20F	SKCTLS	Serial Port 4 Key Control	SKSTAT	Serial Port 4 Key Status Register
D20E	IRQEN	IRQ Interrupt Enable	IRQST	IRQ Interrupt Status Register
D20D	SEROUT	Serial Port Output Reg.		Serial Port Input Register
D20C				
D20B	POTGO	Start Pot Scan Sequence		Vertical Line
D20A	SKRES	Reset Status (SKSTAT)	RANDOM	Random Numb Generator
D209	STIMER	Start Timers	KBCODE	Keyboard Code
D208	AUDCTL	Audio Control	ALLPOT	Read 8 Line Pot Port State
D207	AUDC4	Audio Channel 4 Control	POT 7	
D206	AUDF4	Audio Channel 4 Frequency	POT 6	
D205	AUDC3	Audio Channel 4 Control	POT 5	Read the value of each POT
D204	AUDF3	Audio Channel 3 Frequency	POT 4	
D203	AUDC2	Audio Channel 2 Control	POT 3	
D202	AUDF2	Audio Channel 2 Frequency	POT 2	
D201	AUDC1	Audio Channel 1 Control	POT 1	
D200	AUDF1	Audio Channel 1 Frequency	POT 0	

PIA ADDRESSES				
	WRITE		READ	
Address	Name	Description	Name	Description
D3FF D304	Repeat as shown below many times			
D303	PBCTL	PORT B CONTROL	PBCTL	Same as write
D302	PACTL	PORT A CONTROL	PACTL	Same as write
D301	PORTB	Direction Register If PBCTL Bit 2=0 (otherwise)	PORTB	Same as write
	PORTB	Jack 2 & Jack 3 If Direction Bits Are 1 *	PORTB	Jack 2 & Jack 3 If Direction Bits Are 0 *
D300	PORTA	Direction Register If PACTL Bit 2=0 (Otherwise)	PORTA	Same as write
	PORTA	Jack 0 & Jack 1 If Direction Bits Are 1 *	PORTA	Jack 0 & Jack 1 If Direction Bits Are 0 *

* NOTE: Output data is retained in Jack Output Registers. If direction bits are true, a read of the jacks will read old data from these registers.

B. ALPHABETICAL ORDER

Hardware Register				OS Shadow		
		Address			Address	
Name	Description	Hex	Dec	Name	Hex	Dec
ALLPOT	Read 8 line Pot Port State	D208	53768			
AUDC1	Audio Channel 1 Control	D201	53761			
AUDC2	Audio Channel 2 Control	D203	53763			
AUDC3	Audio Channel 3 Control	D205	53765			
AUDC4	Audio Channel 4 Control	D207	53767			
AUDCTL	Audio Control	D208	53768			
AUDF1	Audio Channel 1 Frequency	D200	53760			
AUDF2	Audio Channel 2 Frequency	D202	53762			
AUDF3	Audio Channel 3 Frequency	D204	53764			
AUDF4	Audio Channel 4 Frequency	D206	53766			
CHACTL	Character Control	D401	54273	CHART	2F3	755
CHBASE	Character base address	D409	54281	CHBAS	2F4	756
COLBK	Color-Luminance of Background	D01A	53274	COLOR4	2C8	712
COLPF0	Color Luminance of Playfield 0	D016	53270	COLOR0	2C4	708
COLPF1	Color Luminance of Playfield 1	D017	53271	COLOR1	2C5	709
COLPF2	Color Luminance of Playfield 2	D018	53272	COLOR2	2C6	710
COLPF3	Color Luminance of Playfield 3	D019	53273	COLOR3	2C7	711
COLPM0	Color Luminance of Player-Missile 0	D012	53266	PCOLR0	2C0	704
COLPM1	Color Luminance of Player-Missile 1	D013	53267	PCOLR1	2C1	705
COLPM2	Color Luminance of Player-Missile 2	D014	53268	PCOLR2	2C2	706
COLPM3	Color Luminance of Player-Missile 3	D015	53269	PCOLR3	2C3	707
CONSOL	Console Switch Port	D01F	53279	Set to 8 during VBLANK		
DLISTH	Display List Pointer (high byte)	D403	54275	SDLSTH	231	561
DLISTL	Display List Pointer (low byte)	D402	54274	SDLSTL	230	560
DMACTL	Direct Memory Access (DMA) Control	D400	54272	SDMCTL	22F	559
GRACTL	Graphic Control	D01D	53277			
GRAFM	Graphics for all Missiles	D011	53265			
GRAFP0	Graphics for Player 0	D00D	53261			
GRAFP1	Graphics for Player 1	D00E	53262			
GRAFP2	Graphics for Player 2	D00F	53263			
GRAFP3	Graphics for Player 3	D010	53264			
HITCLR	Colission Clear	D01E	53278			
HPOSM0	Horizontal Position of Missile 0	D004	53252			
HPOSM1	Horizontal Position of Missile 1	D005	53253			
HPOSM2	Horizontal Position of Missile 2	D006	53254			
HPOSM3	Horizontal Position of Missile 3	D007	53255			
HPOSP0	Horizontal Position of Player 0	D000	53248			
HPOSP1	Horizontal Position of Player 1	D001	53249			
HPOSP2	Horizontal Position of Player 2	D002	53250			
HPOSP3	Horizontal Position of Player 3	D003	53251			
HSCROL	Horizontal Scroll	D404	54276			
IRQEN	Interrupt Request (IRQ) Enable	D20E	53774	POKMSK	10	16
IRQST	IRQ Status	D20E	53774			
KBCODE	Keyboard Code	D209	53769	CH	2FC	764
M0PF	Missile 0 to Playfield Collisions	D000	53248			
M0PL	Missile 0 to Player Collisions	D008	53256			
M1PF	Missile 1 to Playfield Collisions	D001	53249			
M1PL	Missile 1 to Player Collisions	D009	53257			
M2PF	Missile 2 to Playfield Collisions	D002	53250			
M2PL	Missile 2 to Player Collisions	D00A	53258			

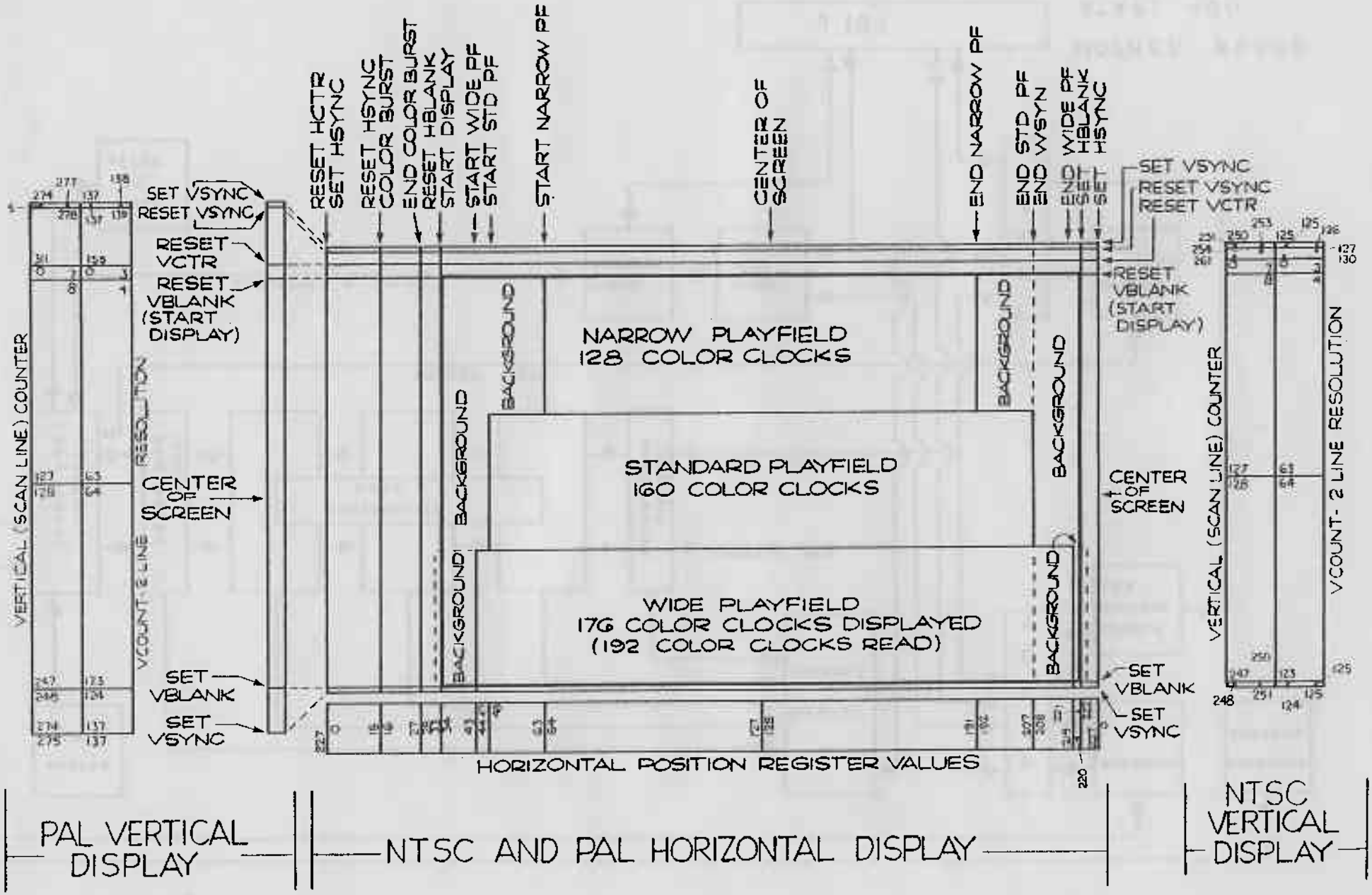
Hardware Register				OS Shadow		
		Address			Address	
Name	Description	Hex	Dec	Name	Hex	Dec
M3PF	Missile 3 to Playfield Collisions	D003	53251			
M3PL	Missile 3 to Player	D00B	53259			
NMIEN	Non-Maskable Interrupt (NMI) Enable	D40E	54286	Set to $40 by IRQ code		
NMIRES	NMI reset	D40F	54287	written to by NMI code		
NMIST	NMI Status	D40F	54287	read by NMI code		
POPF	Player 0 to Playfield Collisions	D004	53252			
POPL	Player 0 to Player Collisions	D00C	53260			
P1PF	Player 1 to Playfield Collisions	D005	53253			
P1PL	Player 1 to Player Collisions	D00D	53261			
P2PF	Player 2 to Playfield Collisions	D006	53254			
P2PL	Player 2 to Player Collisions	D00E	53262			
P3PF	Player 3 to Playfield Collisions	D007	53255			
P3PL	Player 3 to Player Collisions	D00F	53263			
PACTL	Port A Control	D302	54018	Set to $3C by IRQ Code		
PAL	PAL/NTSC indictor	D014	53268			
PBCTL	Port B Control	D303	54019	Set to $3C by IRQ Code		
PENH	Light Pen Horizontal Position	D40C	54284	LPENH	234	564
PENV	Light Pen Vertical Position	D40D	54285	LPENV	235	565
PMBASE	Player Missile Base Address	D407	54279			
PORTA	Port A	D300	54016	STICK0,1	278,279	632,633
PORTB	Port A	D301	54017	STICK2,3	27A,27B	634,635
POT0	Pot 0	D200	53760	PADDL0	270,	624
POT1	Pot 1	D201	53761	PADDL1	271	625
POT2	Pot 2	D202	53762	PADDL2	272	626
POT3	Pot 3	D203	53763	PADDL3	273	627
POT4	Pot 4	D204	53764	PADDL4	274	628
POT5	Pot 5	D205	53765	PADDL5	275	629
POT6	Pot 6	D206	53766	PADDL6	276	630
POT7	Pot 7 (right paddle controller)	D207	53767	PADDL7	277	631
POTGO	Start POT Scan Sequence	D20B	53771	WRITTEN DURING VBLANK		
PRIOR	Priority Select	D01B	53275	GPRIOR	26F	623
RANDOM	Random number generator	D20A	53770			
SERIN	Serial Port Input	D20E	53774			
SEROUT	Serial Port output	D20D	53773			
SIZEM	Sizes for all missiles	D00C	53260			
SIZEP0	Size of Player 0	D008	53256			
SIZEP1	Size of Player 1	D009	53257			
SIZEP2	Size of Player 2	D00A	53258			
SIZEP3	Size of Player 3	D00B	53259			
SKCTL	Serial Port Control	D20F	53775	SSKCTL	232	562
SKREST	Reset Serial Port Status (SKSTAT)	D20A	53770			
SKSTAT	Serial Port Status	D20F	53775			
STIMER	Start Timer	D209	53769			
TRIG0	Joystick Controller Trigger 0	D010	53264	STRIG0	284	644
TRIG1	Joystick Controller Trigger 1	D011	53265	STRIGL	285	645
TRIG2	Joystick Controller Trigger 2	D012	53266	STRIG2	286	646
TRIG3T	Joystick Controller Trigger 3	D013	53267	STRIG3	287	647
VCOUNT	Vertical Line Counter	D40B	54283			
VDELAY	Verical Delay	D01C	54276			
VSCROL	Vertical Scroll	D405	54277			
WSYNC	Wait for Horizontal Sync	D40A	54282	Used by keyboard click routine		

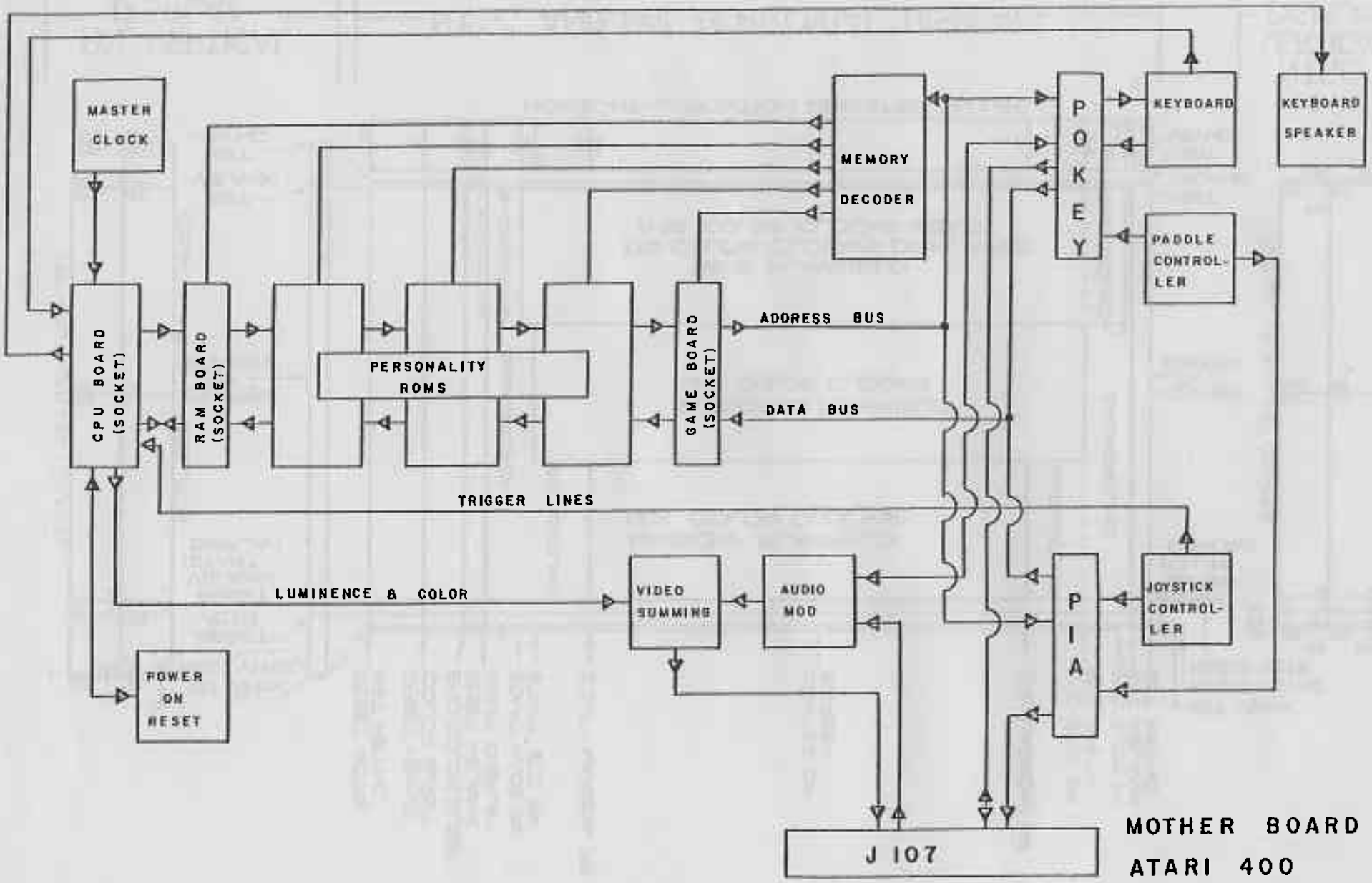
VI. FIGURES

A. MEMORY MAP

ADDRESS	FUNCTION	SIZE
FFFF D800	Operating System And Math Routines	10K
D000-D7FF CFFF	Hardware Addresses Reserved for Future O.S. expansion	2K 4K
BFFF 8000	ROM Cartridge (Colleen left and right slot and Candy single slot all address to this space)	16K
7FFF 2000	RAM Expansion *	
1FFF	RAM initially supplied in the product	8K

* RAM expansion can actually exten to BFFF. However, the ROM cartridges will deselect the RAM. Deselection occurs on 8K boundaries. Atari 400 units are RAM expandable only at the factory. They can accept RAM up to 2FFF (16K) when fully extended.

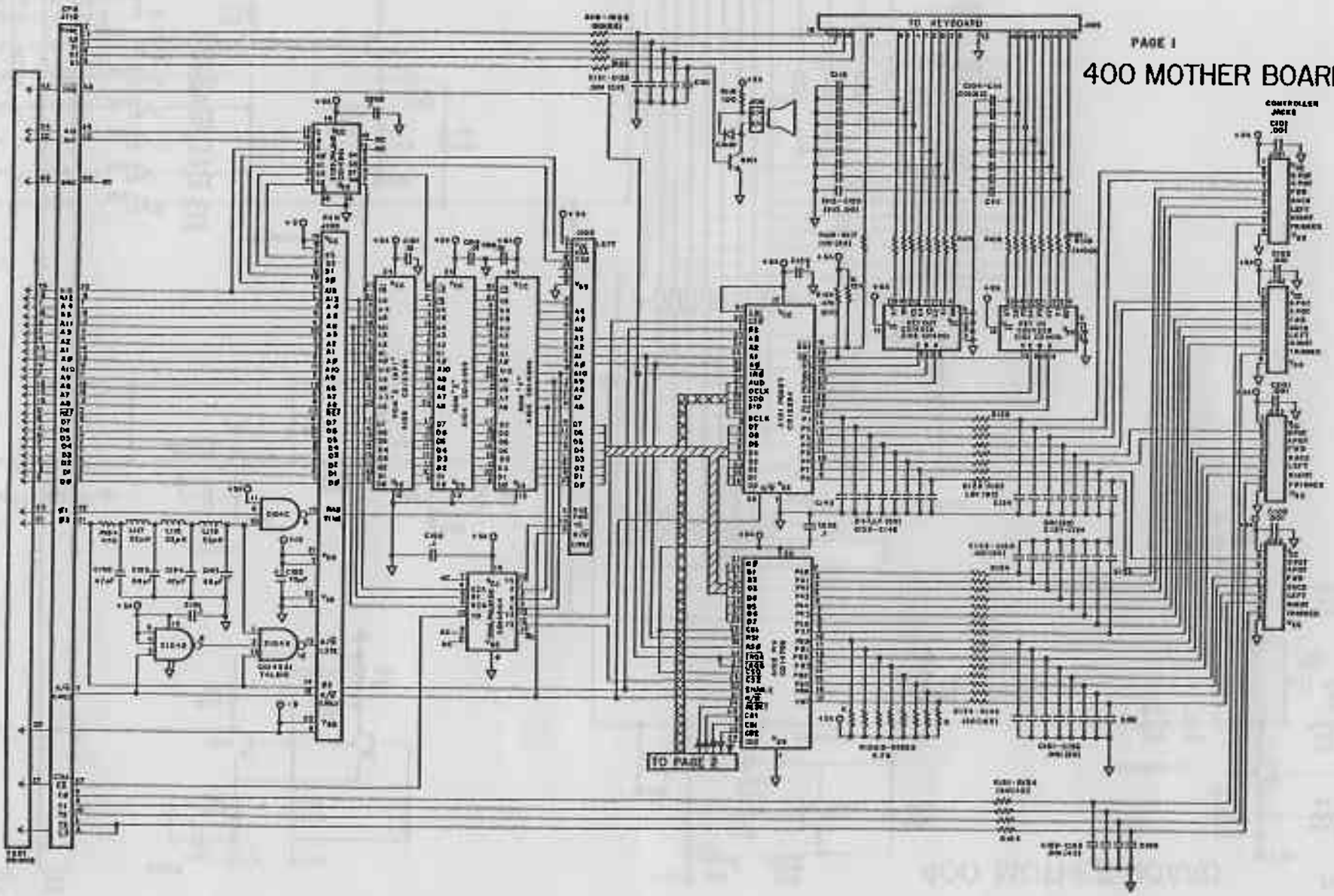

RESET HCTR
SET HSYNC
RESET HSYNC
COLOR BURST
END COLOR BURST
RESET HBLANK
START DISPLAY
START WIDE PF
START STD PF
START NARROW PF
CENTER OF SCREEN
END NARROW PF
END STD PF
END VSYN
WIDE PF HBLANK
SET HSYNC
SET VSYNC
RESET VSYNC
RESET VCTR
RESET VBLANK (START DISPLAY)
SET VSYNC
RESET VSYNC
RESET VCTR
RESET VBLANK (START DISPLAY)
NARROW PLAYFIELD
128 COLOR CLOCKS
STANDARD PLAYFIELD
160 COLOR CLOCKS
WIDE PLAYFIELD
176 COLOR CLOCKS DISPLAYED
(192 COLOR CLOCKS READ)
BACKGROUND
CENTER OF SCREEN
SET VBLANK
SET VSYNC
HORIZONTAL POSITION REGISTER VALUES
VERTICAL (SCAN LINE) COUNTER
VCOUNT-2 LINE RESOLUTION
VERTICAL (SCAN LINE) COUNTER
VCOUNT-2 LINE RESOLUTION
PAL VERTICAL DISPLAY
NTSC AND PAL HORIZONTAL DISPLAY
NTSC VERTICAL DISPLAY



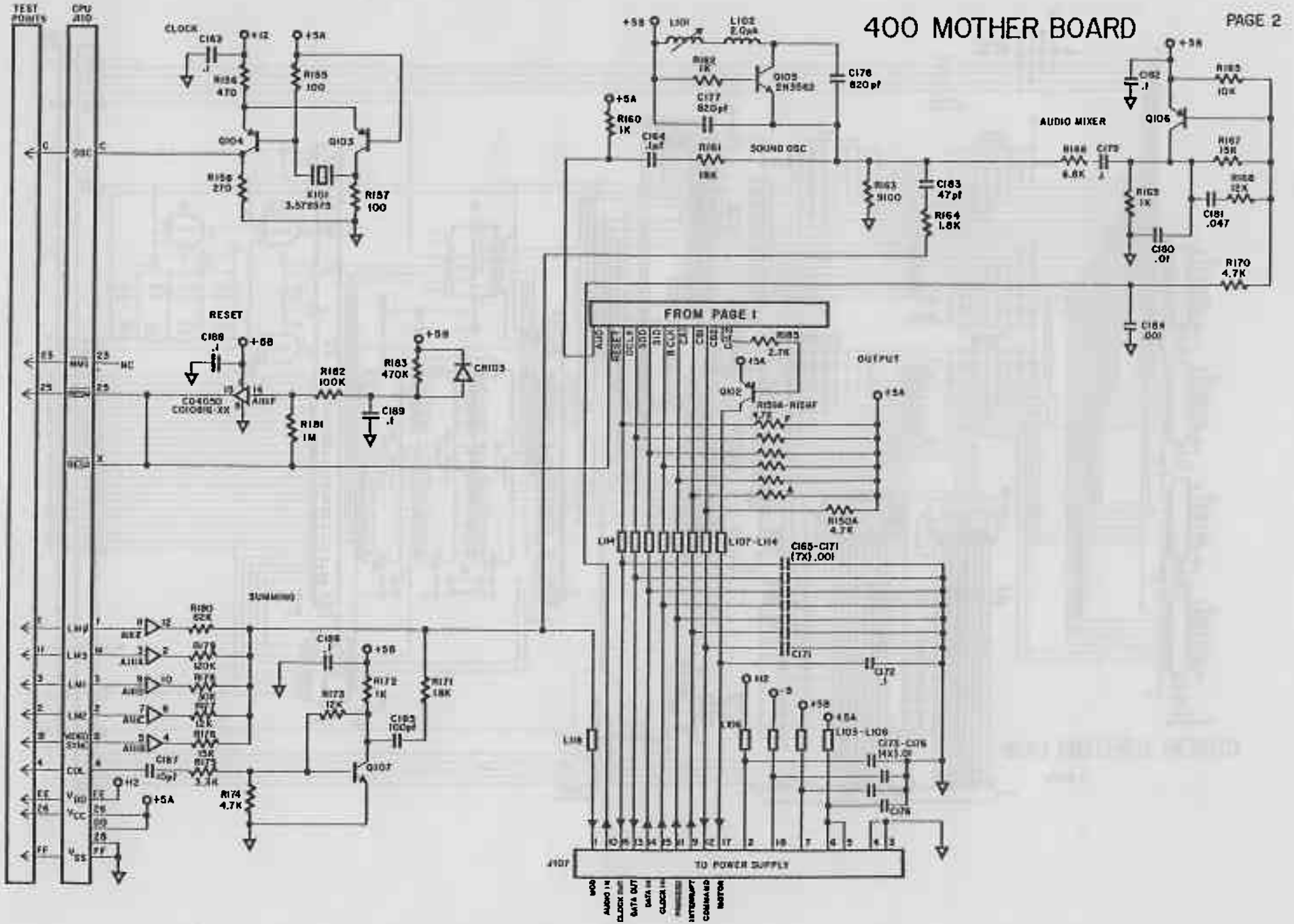
MOTHER BOARD
ATARI 400

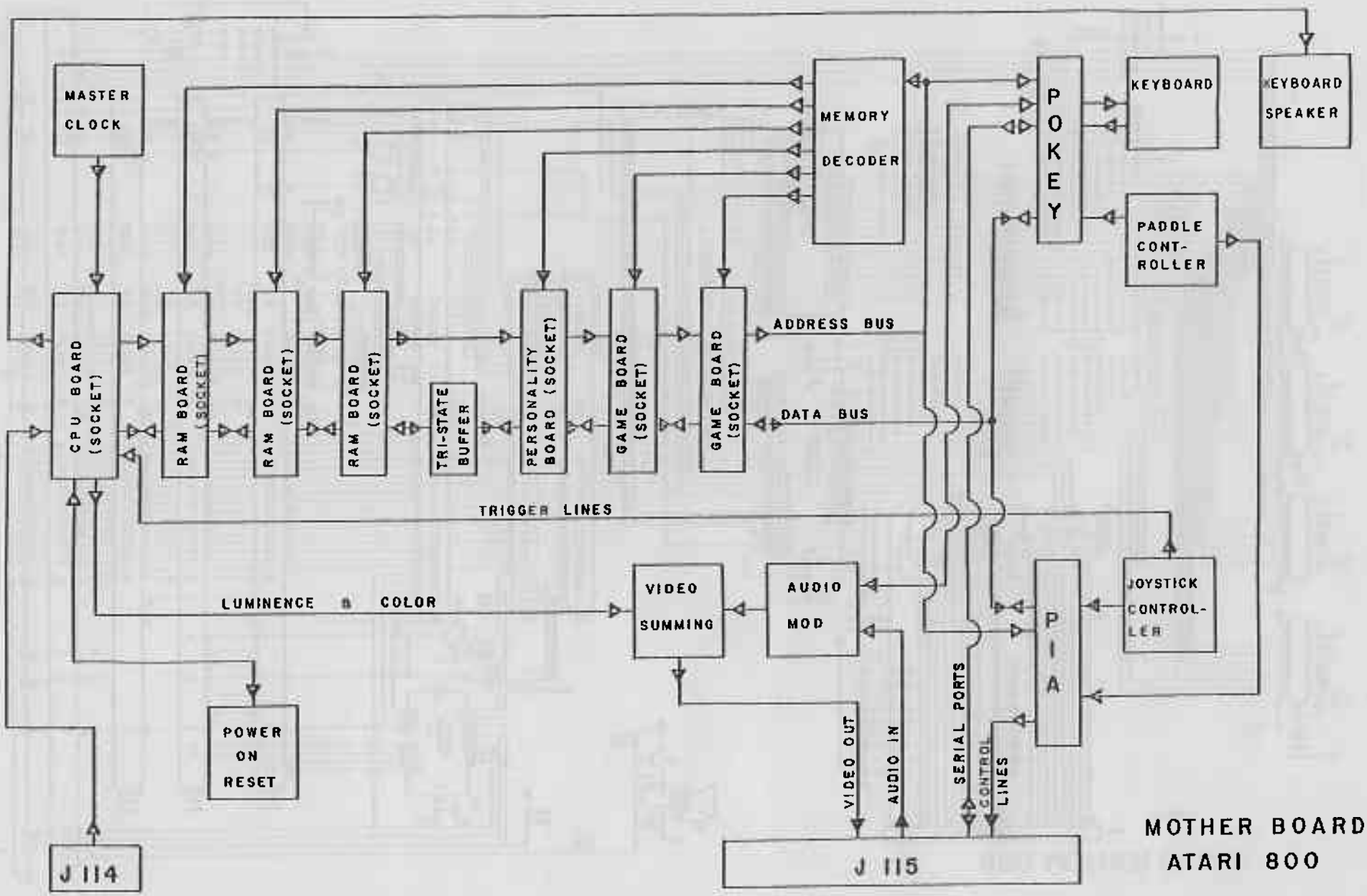
PAGE 1

400 MOTHER BOARD



400 MOTHER BOARD

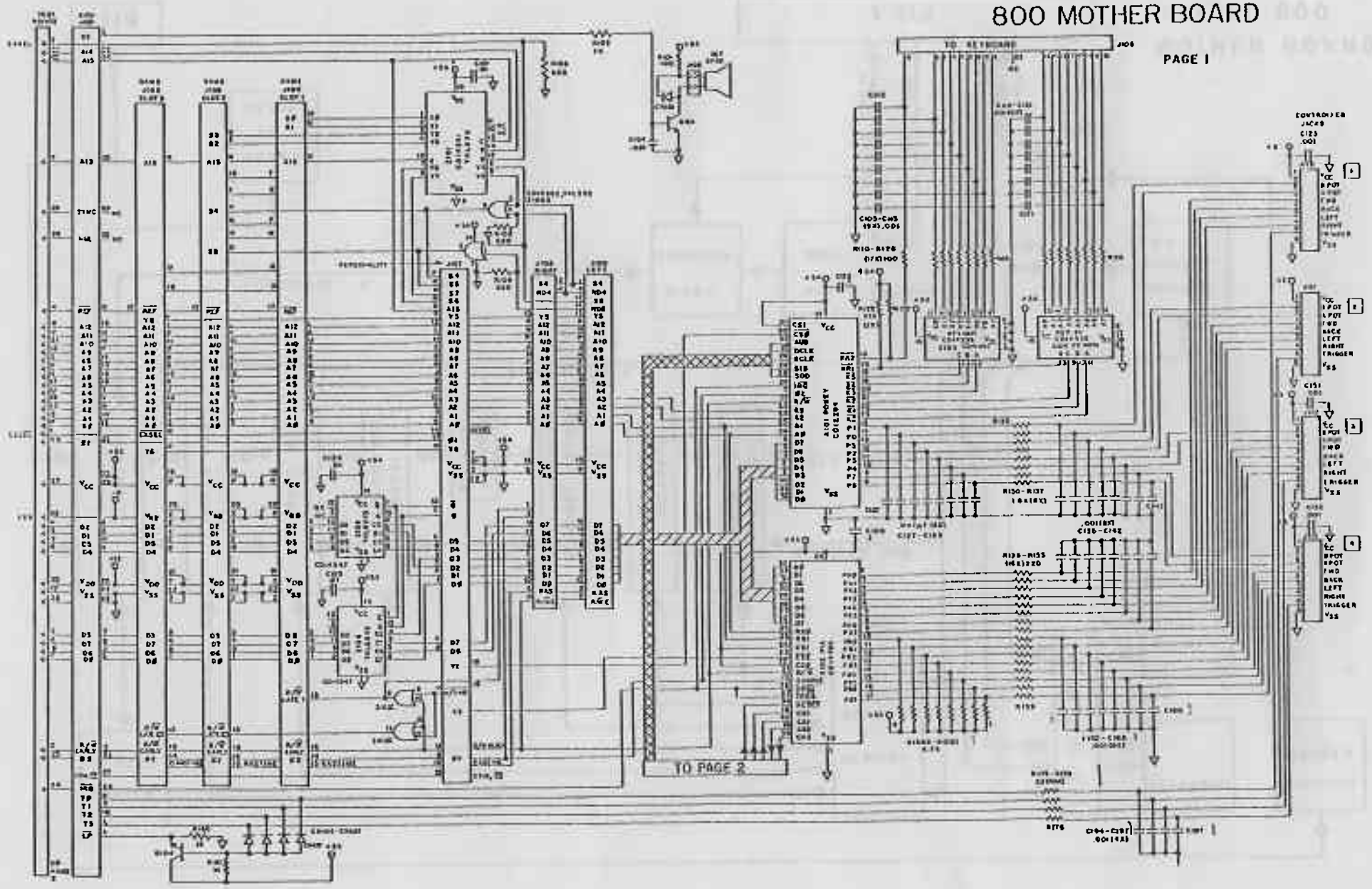




MOTHER BOARD
ATARI 800

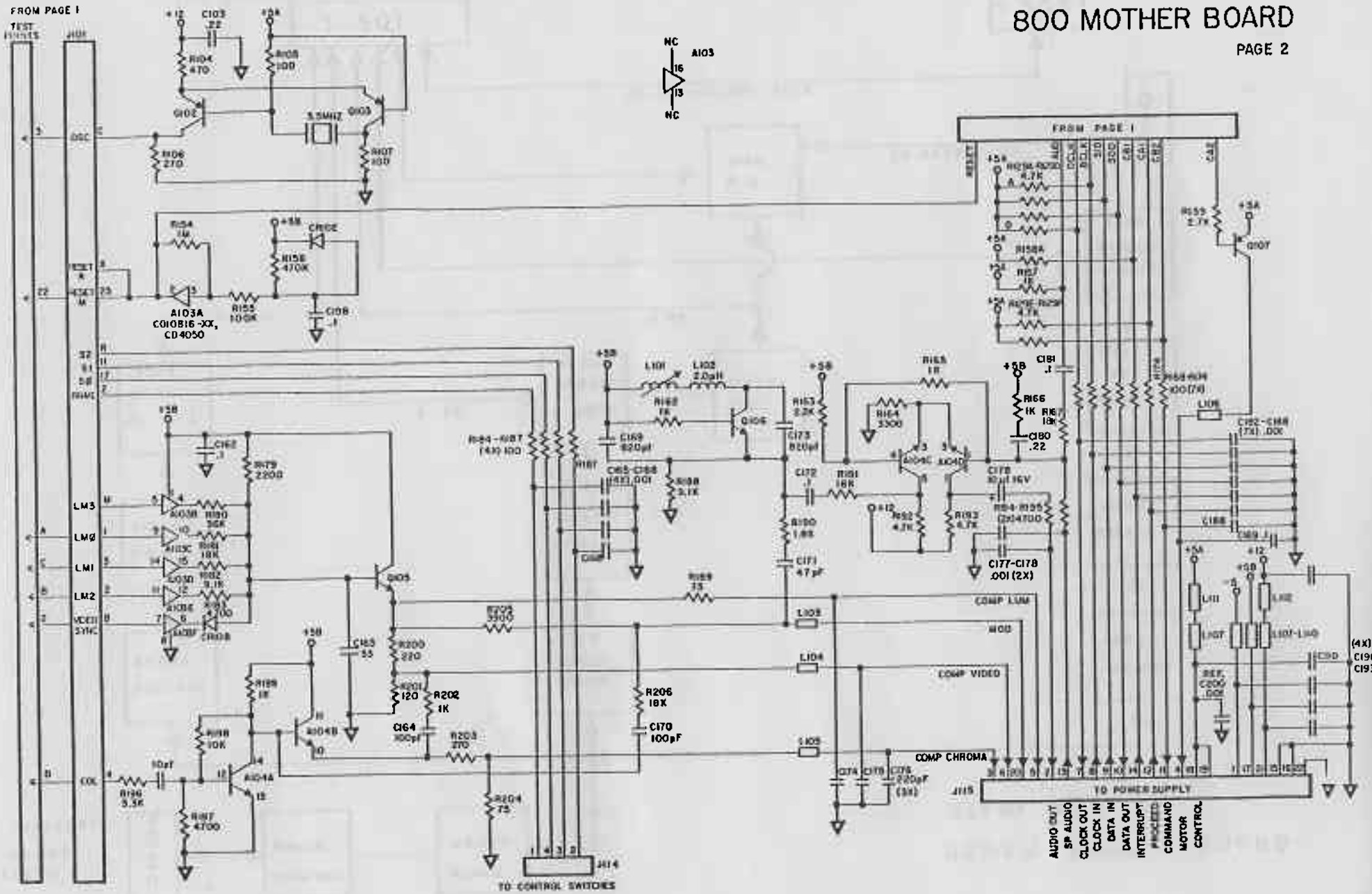
800 MOTHER BOARD

PAGE 1

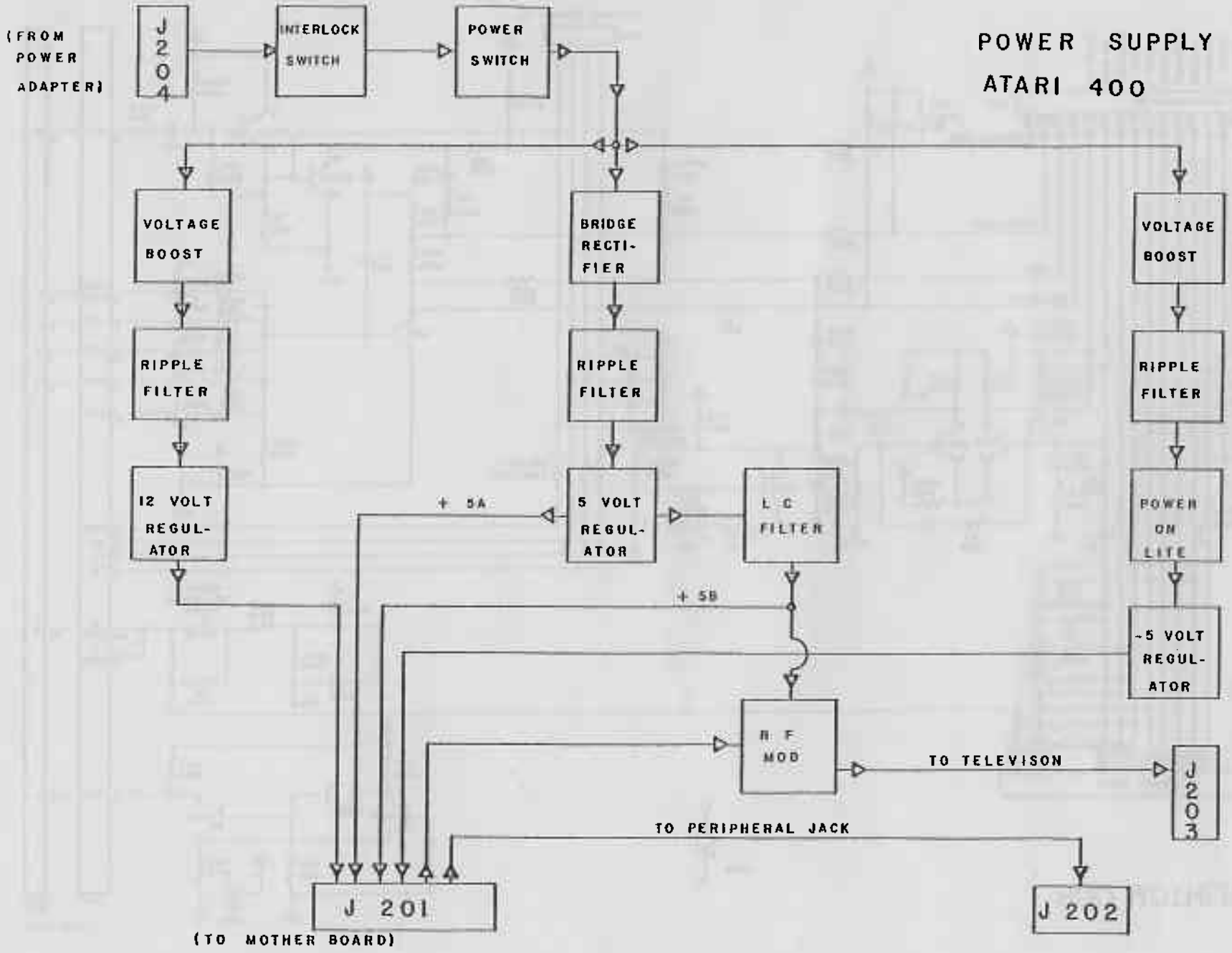


800 MOTHER BOARD

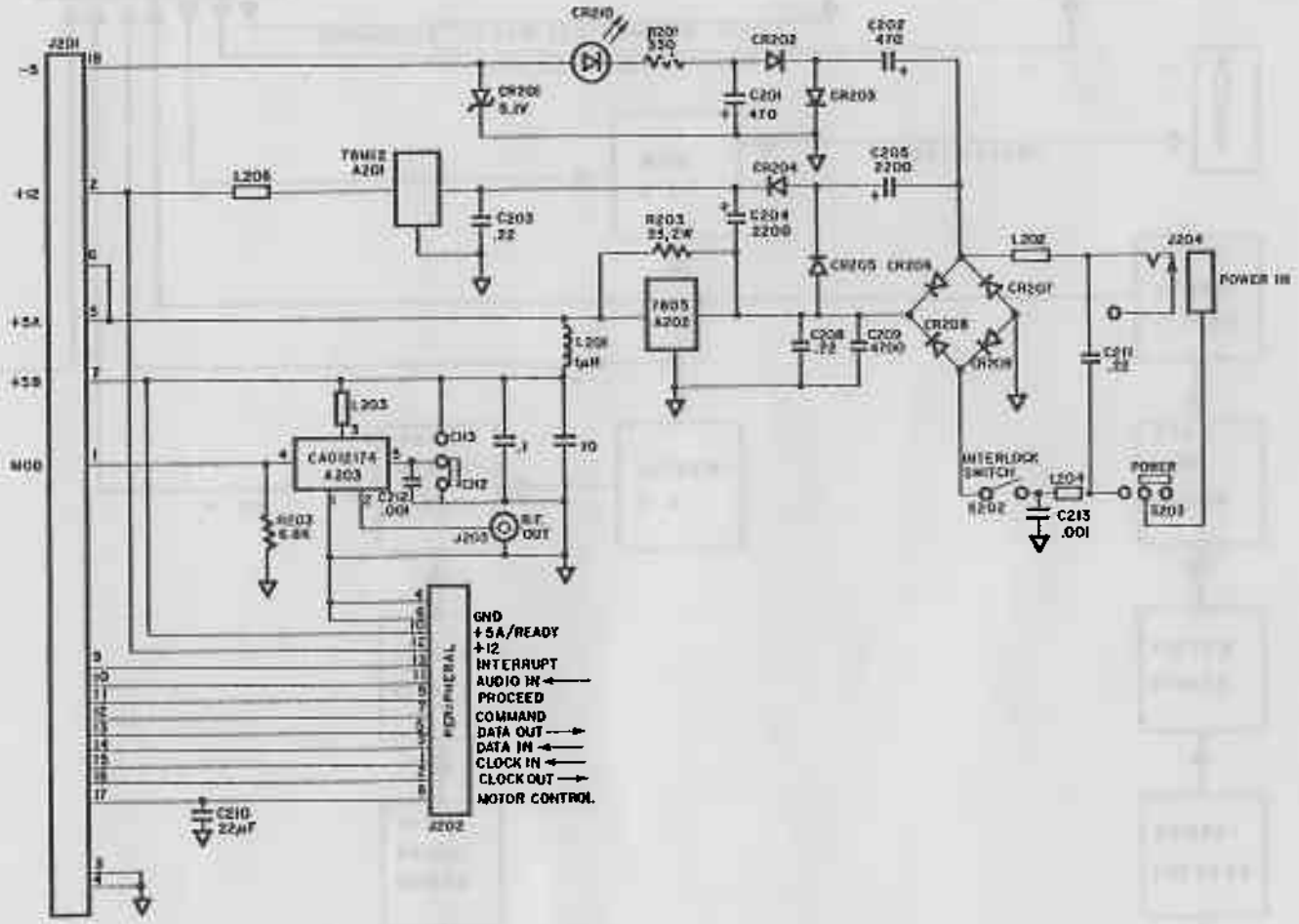
PAGE 2



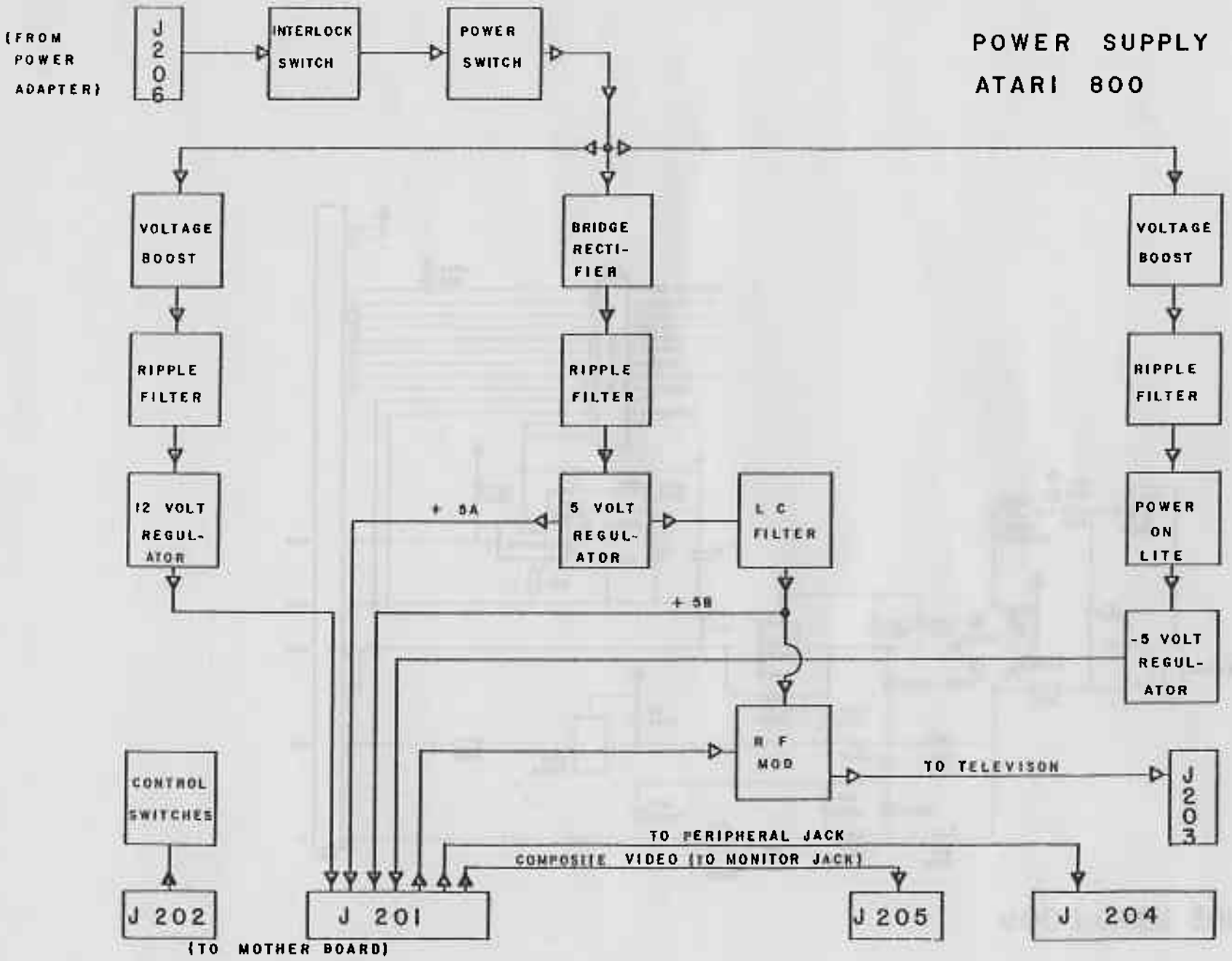
POWER SUPPLY BOARD
ATARI 400



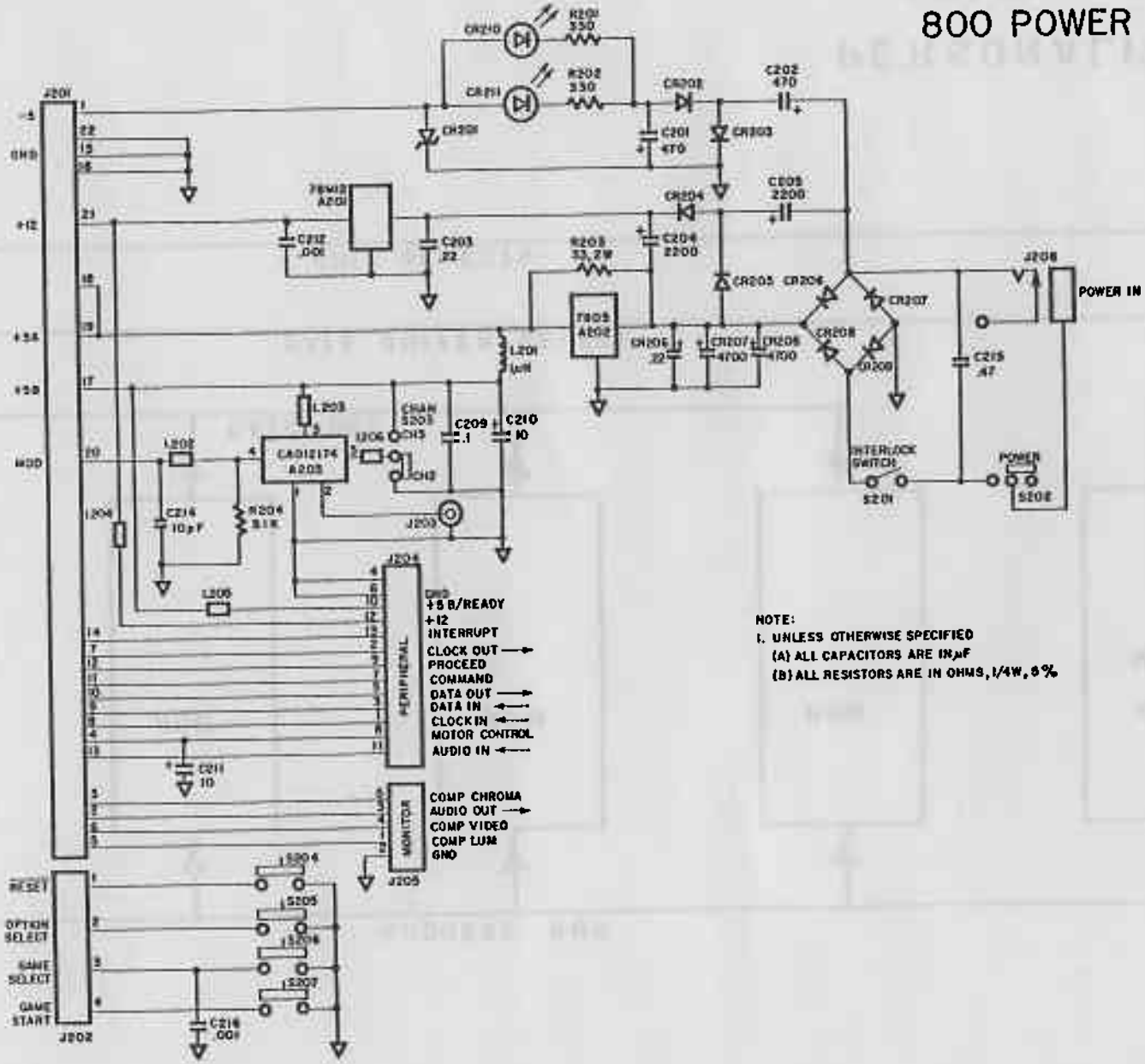
400 POWER SUPPLY



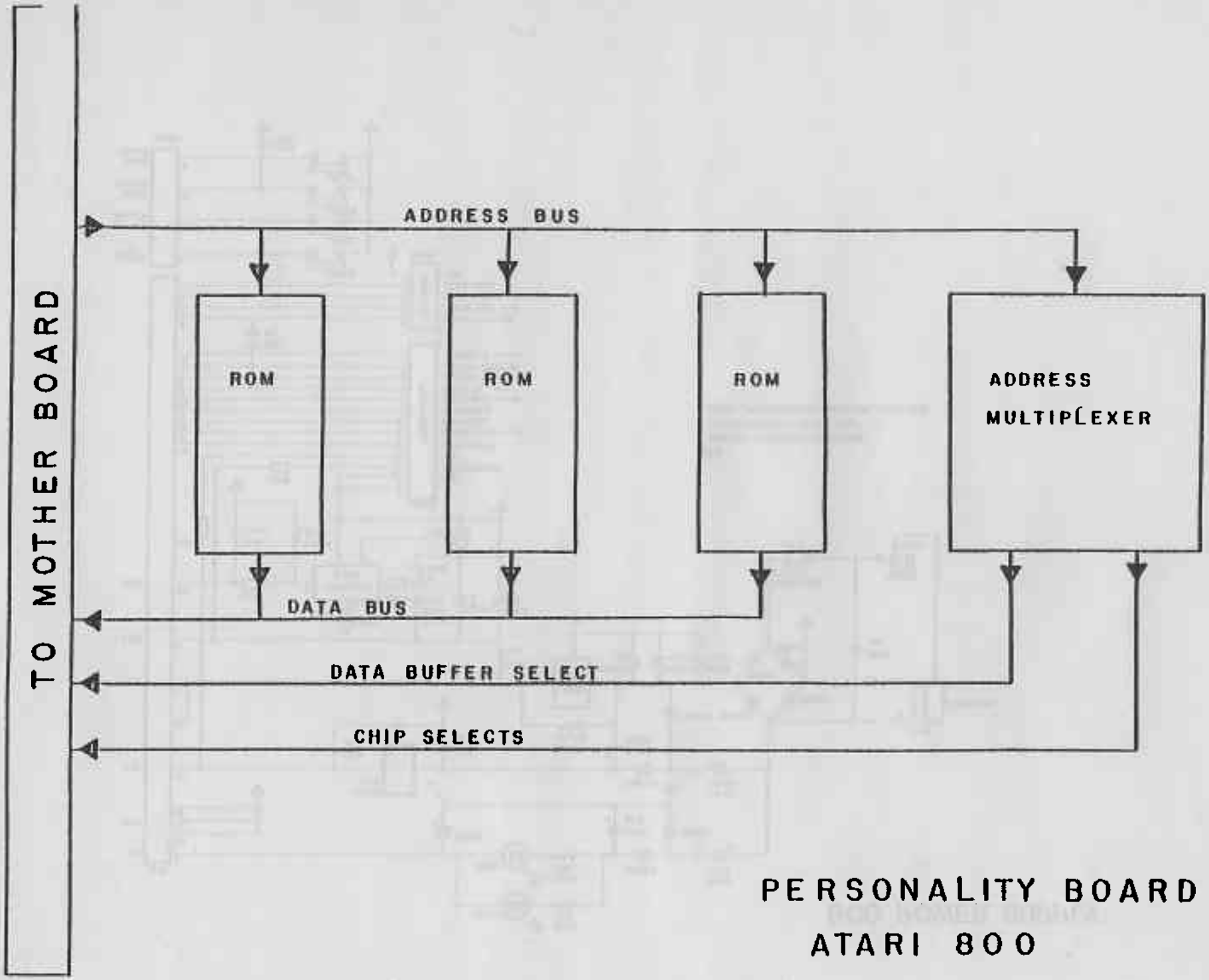
POWER SUPPLY BOARD
ATARI 800



800 POWER SUPPLY



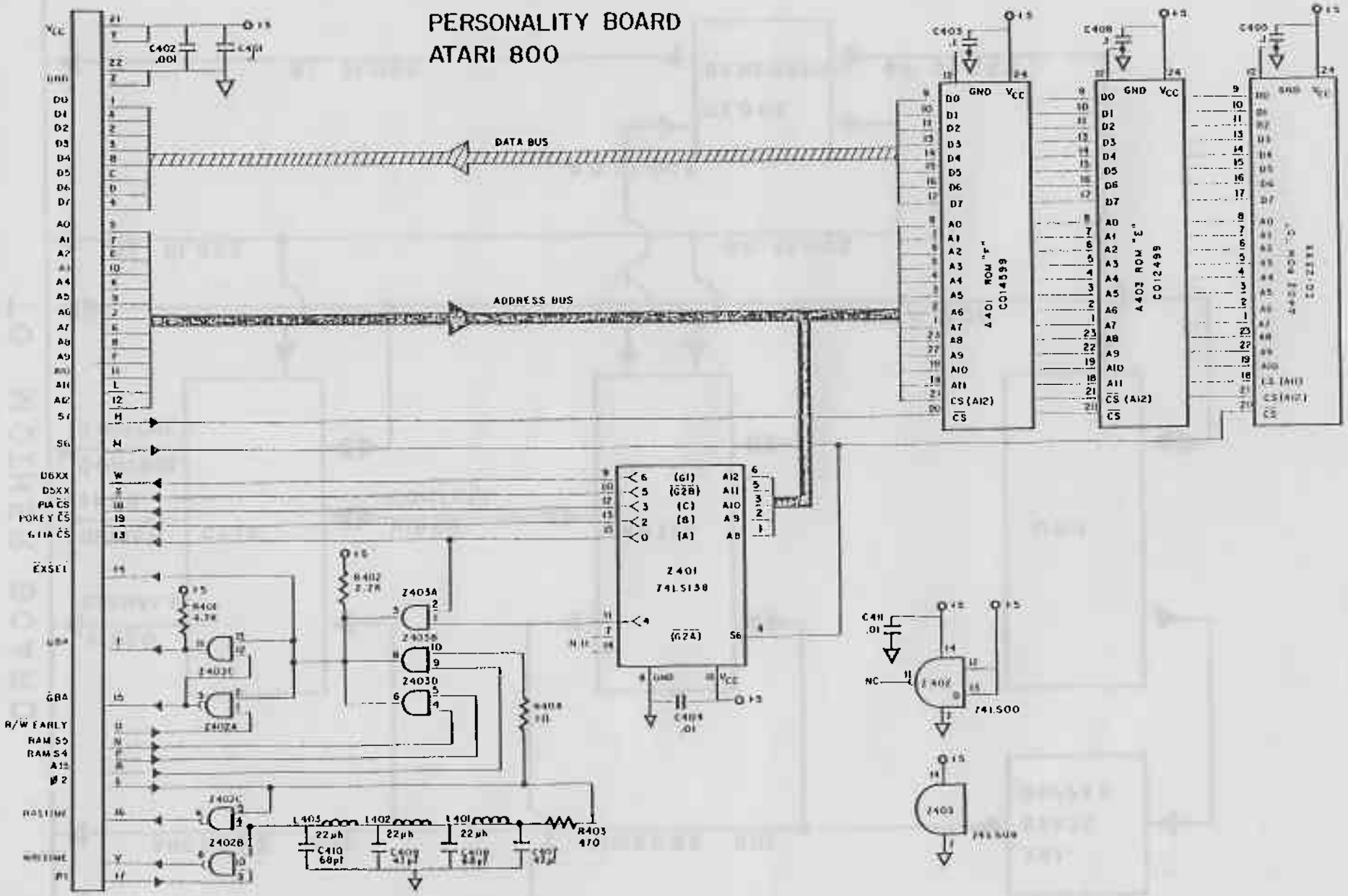
NOTE:
1. UNLESS OTHERWISE SPECIFIED
   (A) ALL CAPACITORS ARE IN μF
   (B) ALL RESISTORS ARE IN OHMS, 1/4W, 5%

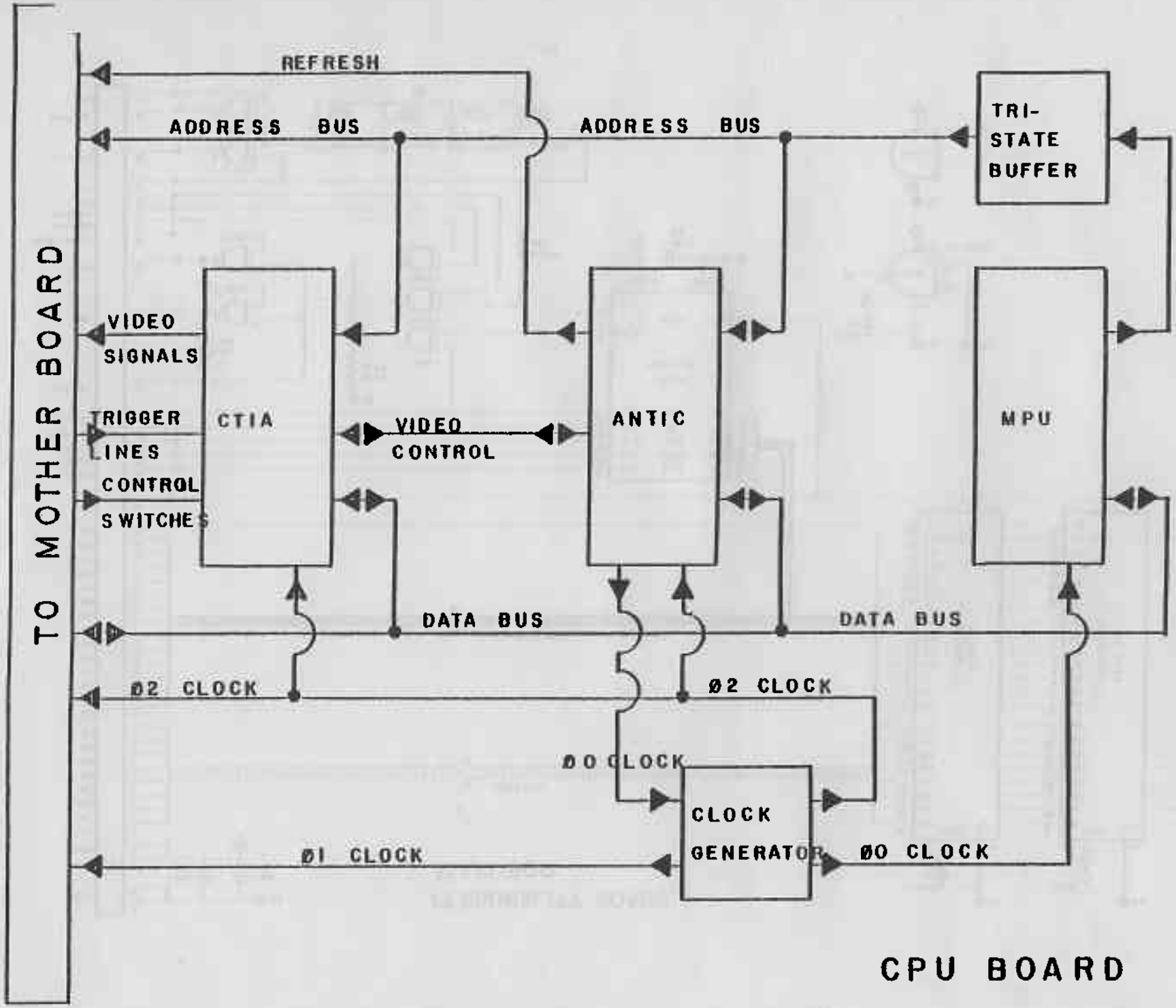


PERSONALITY BOARD

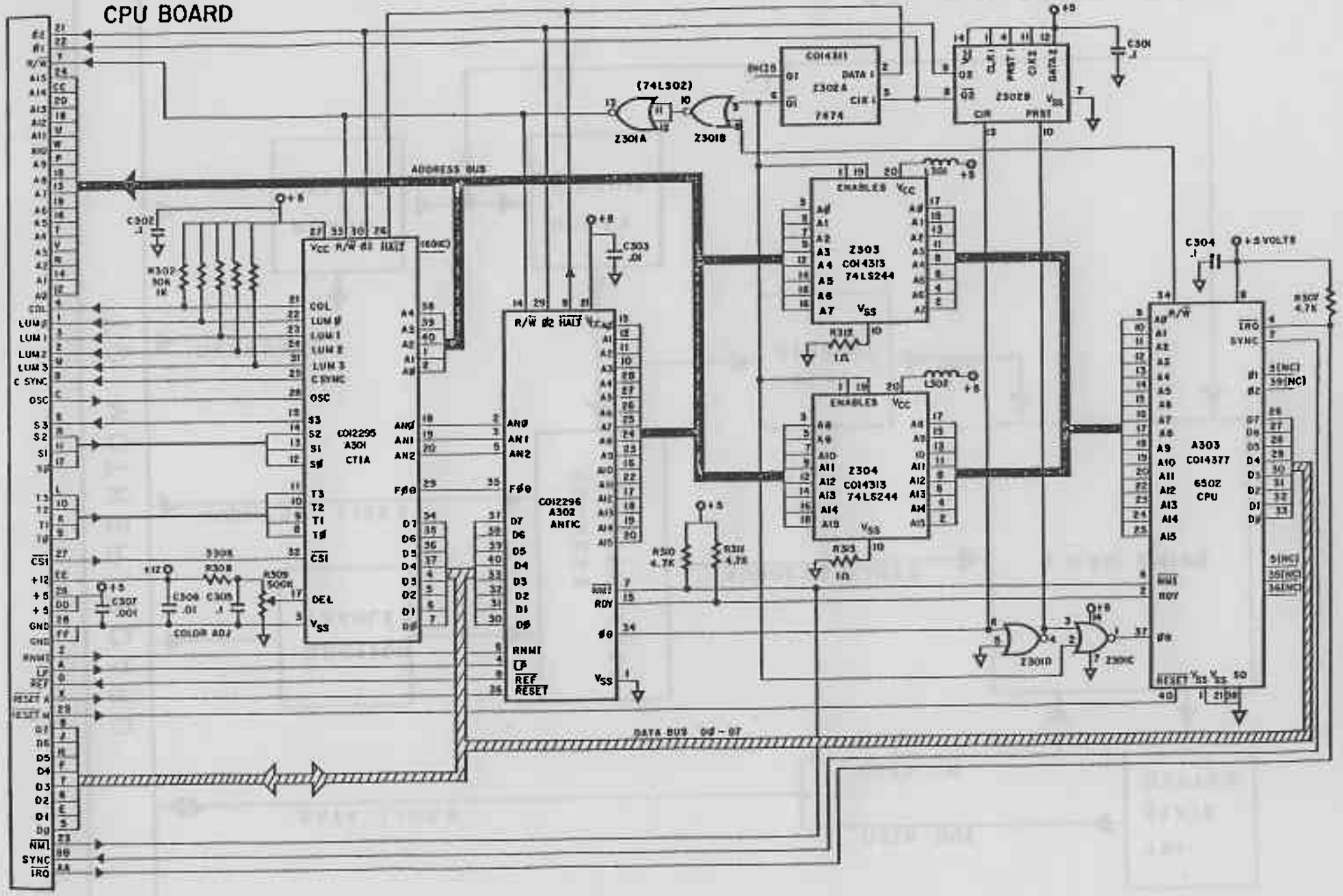
ATARI 800

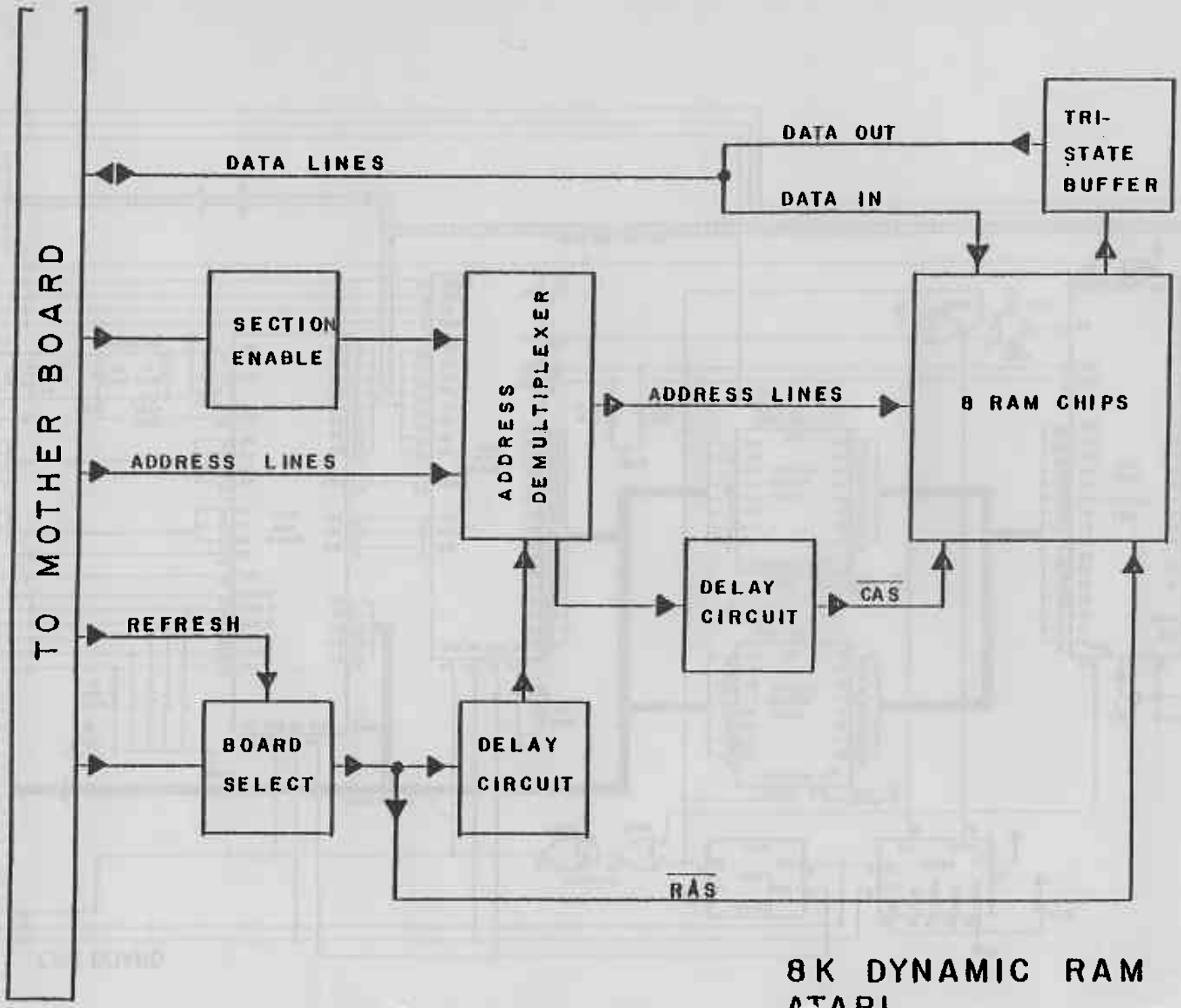
PERSONALITY BOARD
ATARI 800




TO MOTHER BOARD
REFRESH
ADDRESS BUS
ADDRESS BUS
TRI-
STATE
BUFFER
VIDEO
SIGNALS
TRIGGER
LINES
CONTROL
SWITCHES
CTIA
VIDEO
CONTROL
ANTIC
MPU
DATA BUS
DATA BUS
Ø2 CLOCK
Ø2 CLOCK
Ø0 CLOCK
CLOCK
GENERATOR
Ø1 CLOCK
Ø0 CLOCK

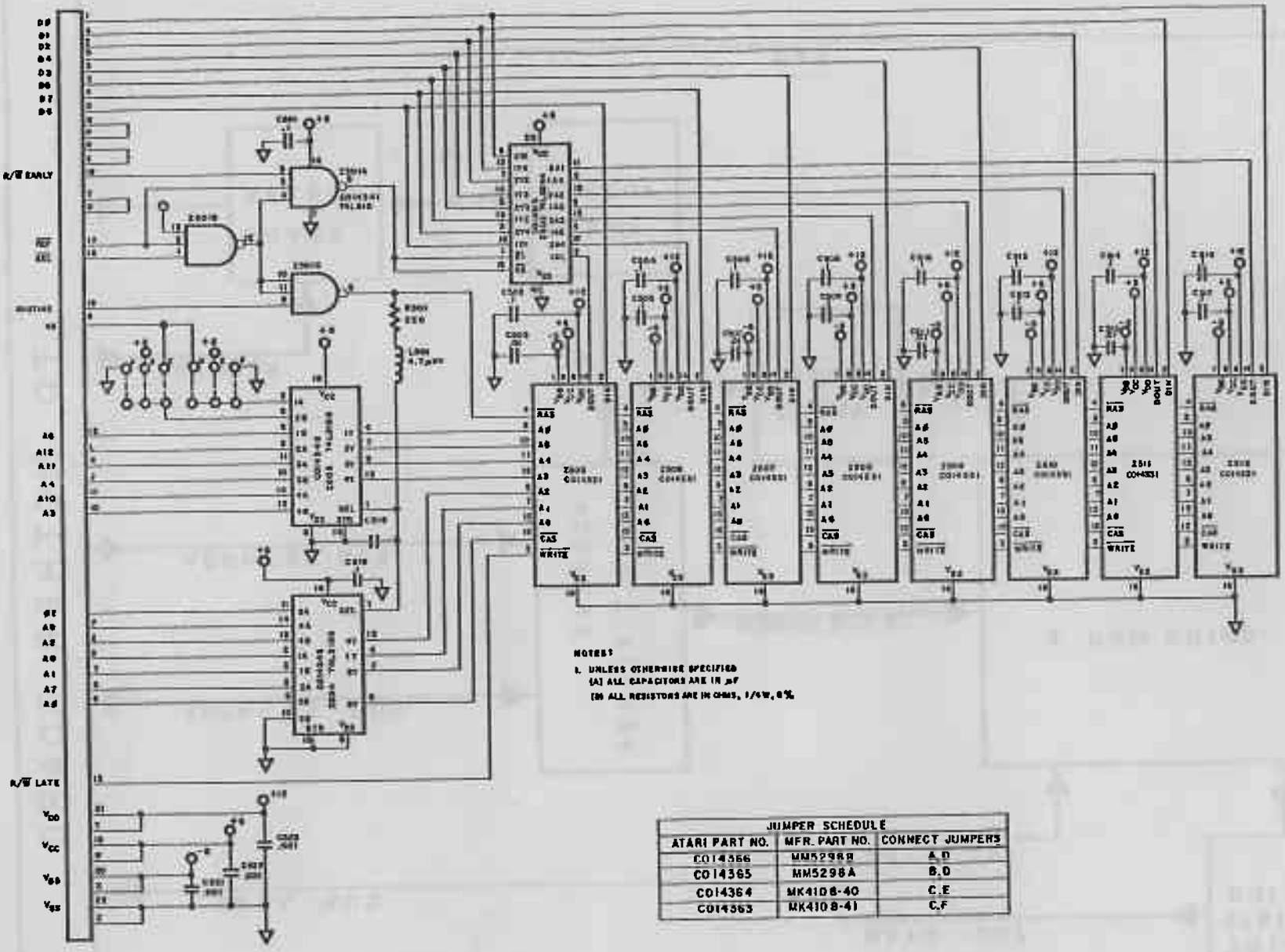
CPU BOARD


CPU BOARD
ADDRESS BUS
DATA BUS D0 - D7
(74LS02)
Z301A
Z301B
Z303
CO14313
74LS244
Z304
CO14313
74LS244
A303
CO14377
6502
CPU
CO12295
A301
CTIA
CO12296
A302
ANTIC
Z301D
Z301C
R302
10K
1M
R308
330R
R309
500R
COLOR ADJ
C302
.1
C303
.01
C304
.1
C305
.1
C306
.01
C307
.001
R310
10
R311
10
R312
10
R313
10
R314
4.7K
R315
4.7K
+5 VOLTS
COL
LUM 0
LUM 1
LUM 2
LUM 3
C SYNC
OSC
S3
S2
S1
S0
T3
T2
T1
T0
CSI
+12
+5
+5
GND
RNMI
LP
REF
RESET A
RESET M
D7
D6
D5
D4
D3
D2
D1
D0
NMI
SYNC
IRQ



8K DYNAMIC RAM
ATARI

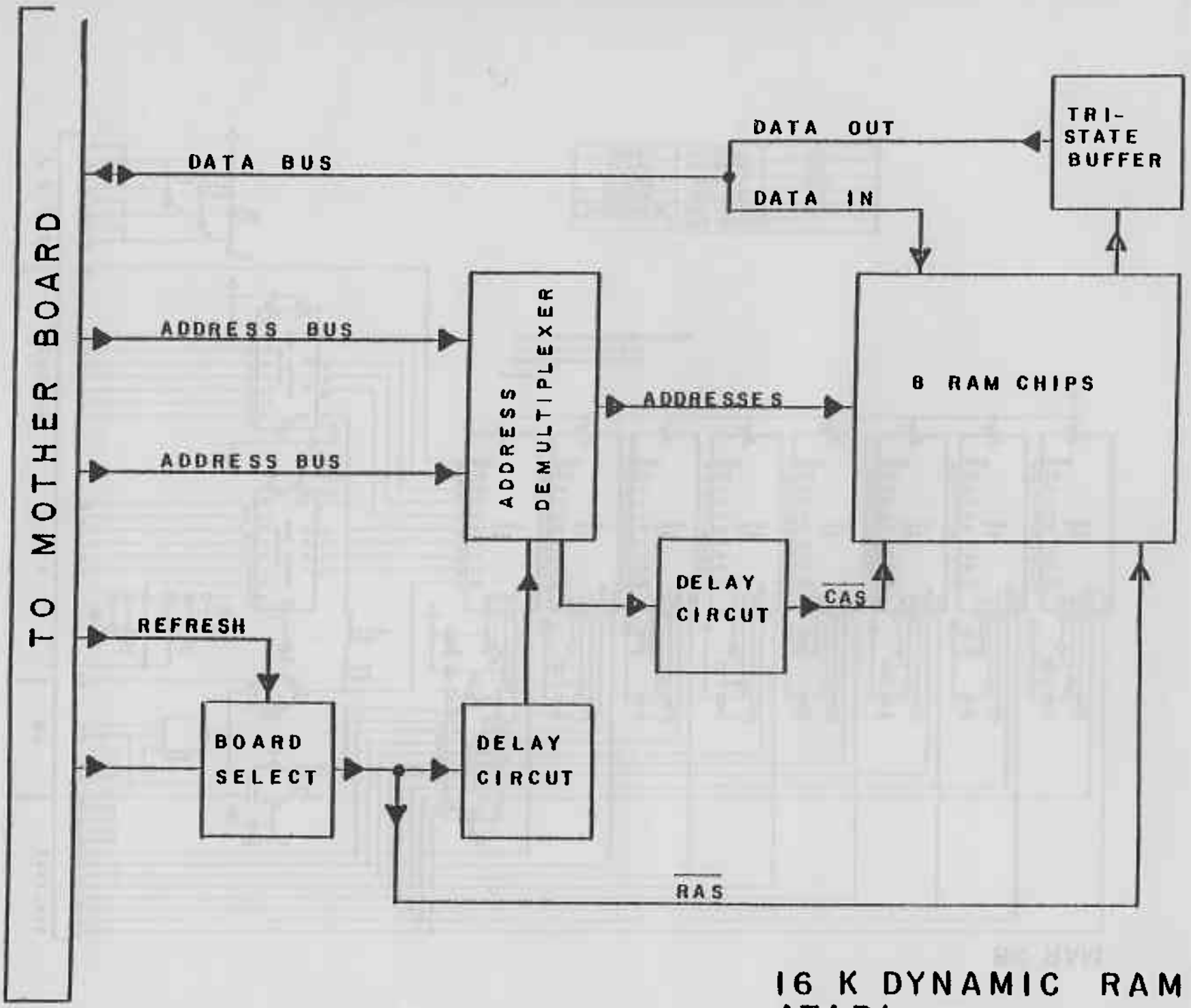
8K RAM



NOTES:

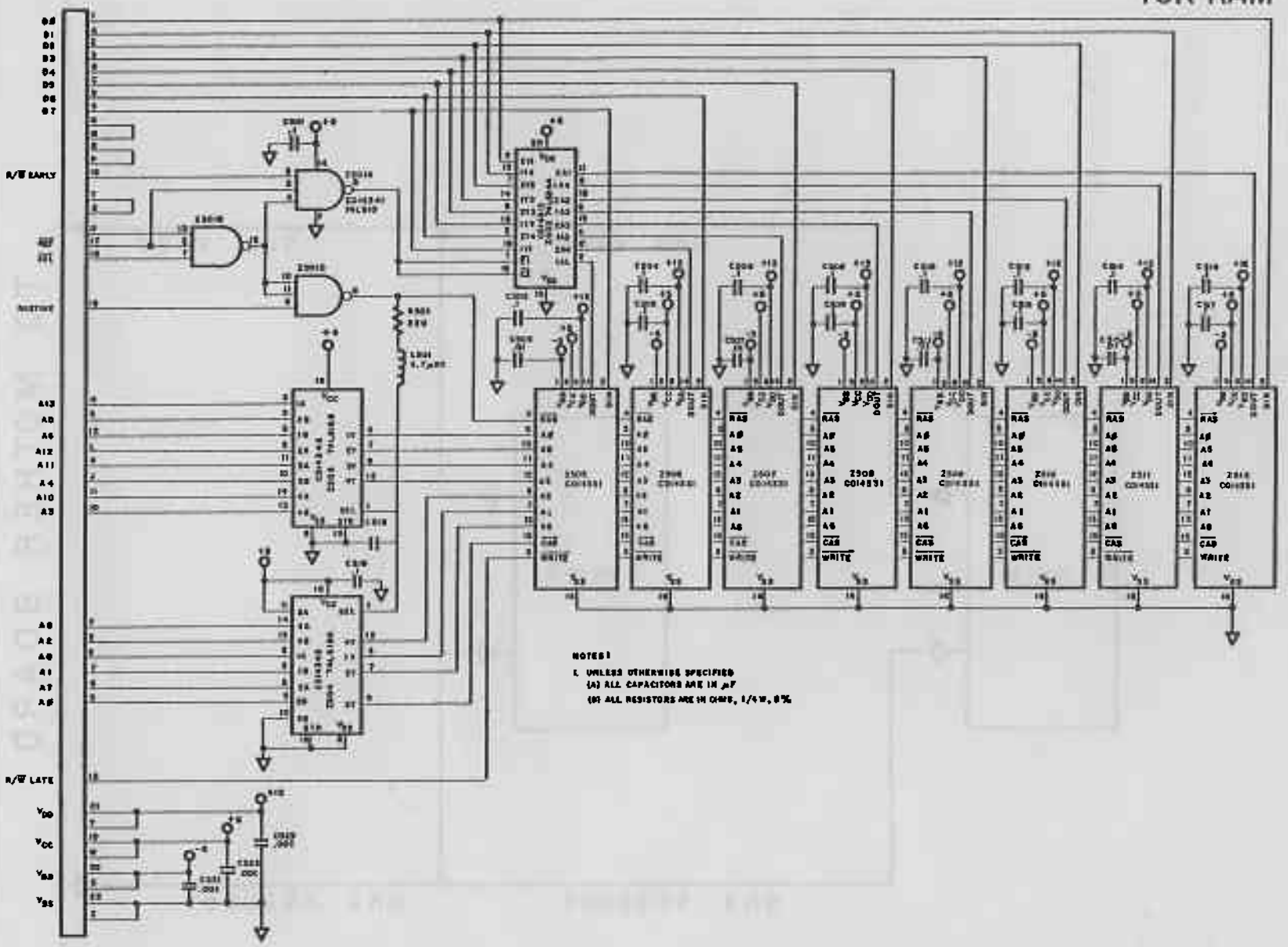
1. UNLESS OTHERWISE SPECIFIED
   (A) ALL CAPACITORS ARE IN μF
   (B) ALL RESISTORS ARE IN OHMS, 1/4W, 5%

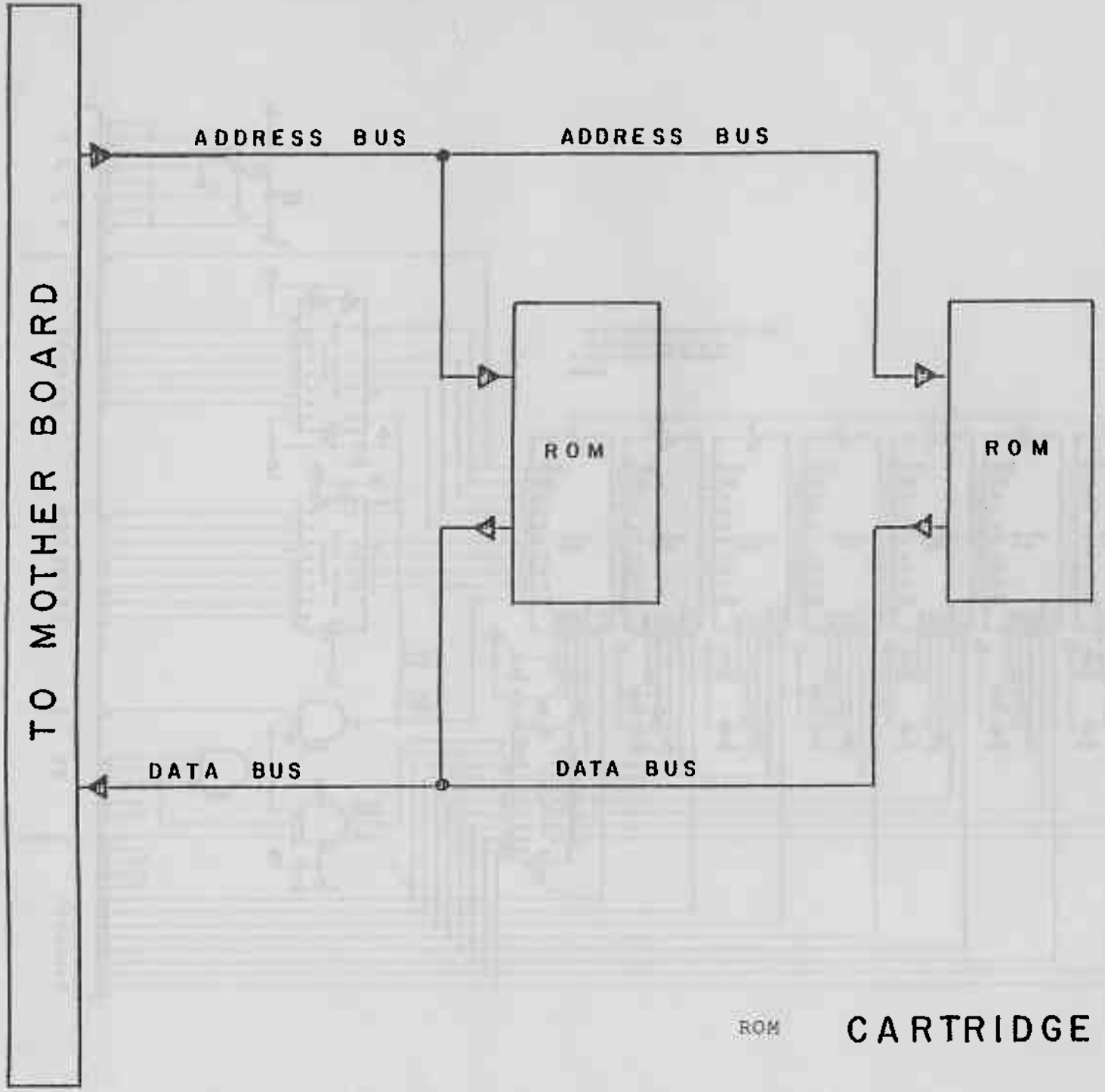
JUMPER SCHEDULE		
ATARI PART NO.	MFR. PART NO.	CONNECT JUMPERS
C014366	MM5298B	A,D
C014365	MM5298A	B,D
C014364	MK4108-40	C,E
C014363	MK4108-41	C,F



16 K DYNAMIC RAM
ATARI

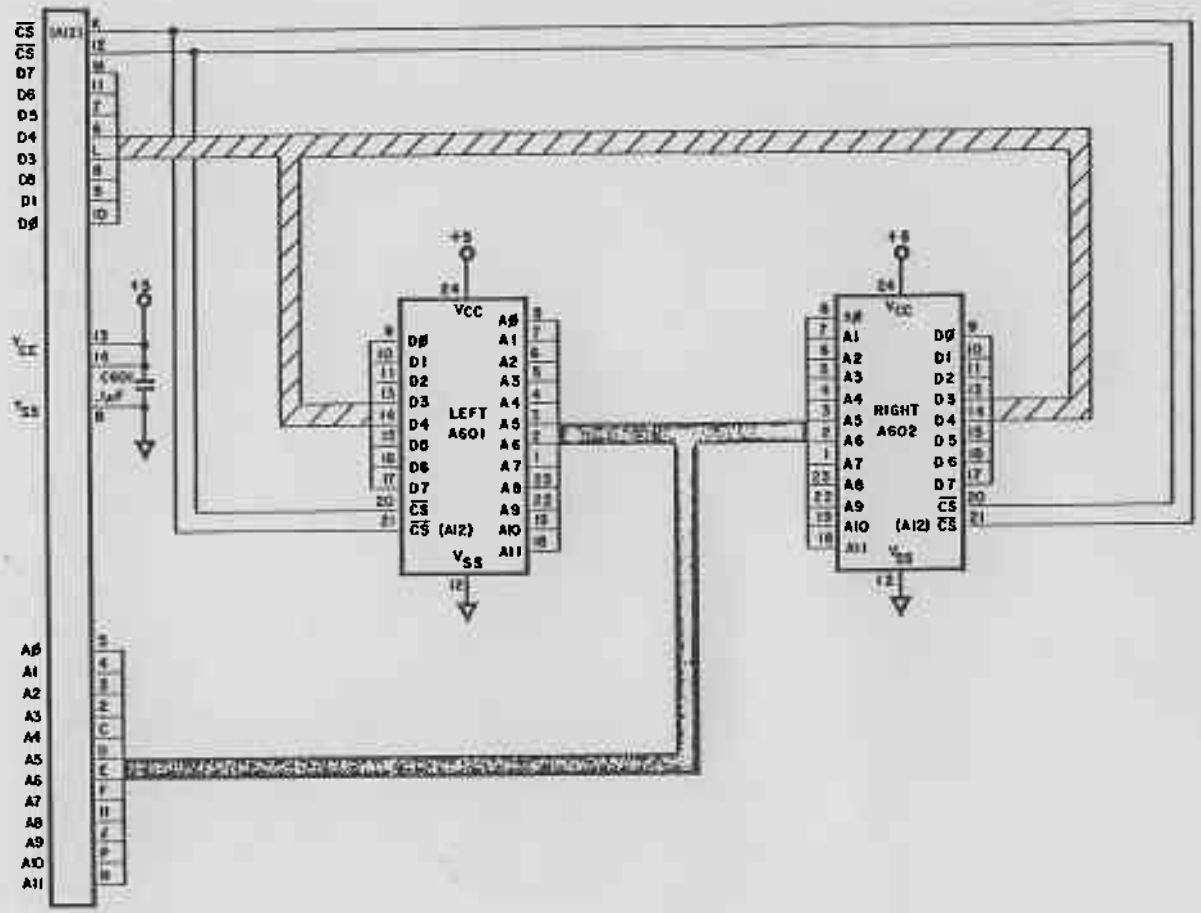
16K RAM

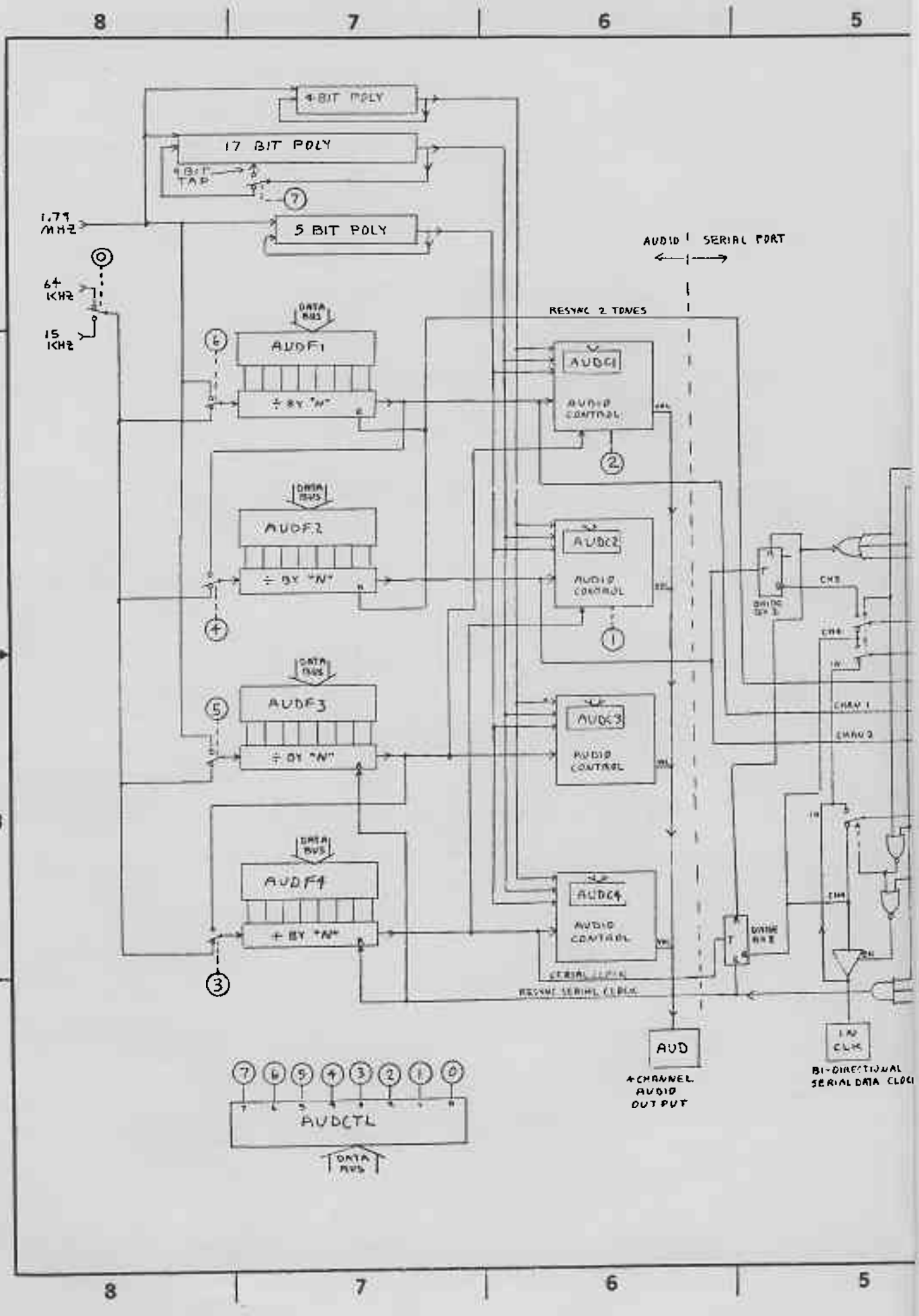

D0
D1
D2
D3
D4
D5
D6
D7
R/W EARLY
R/W LATE
A13
A9
A6
A12
A11
A4
A10
A3
A8
A2
A0
A1
A7
A5
V_{DD}
V_{CC}
V_{BB}
V_{SS}
RAS
CAS
WRITE
NOTES:
1. UNLESS OTHERWISE SPECIFIED
(A) ALL CAPACITORS ARE IN μF
(B) ALL RESISTORS ARE IN OHMS, 1/4 W, 5%

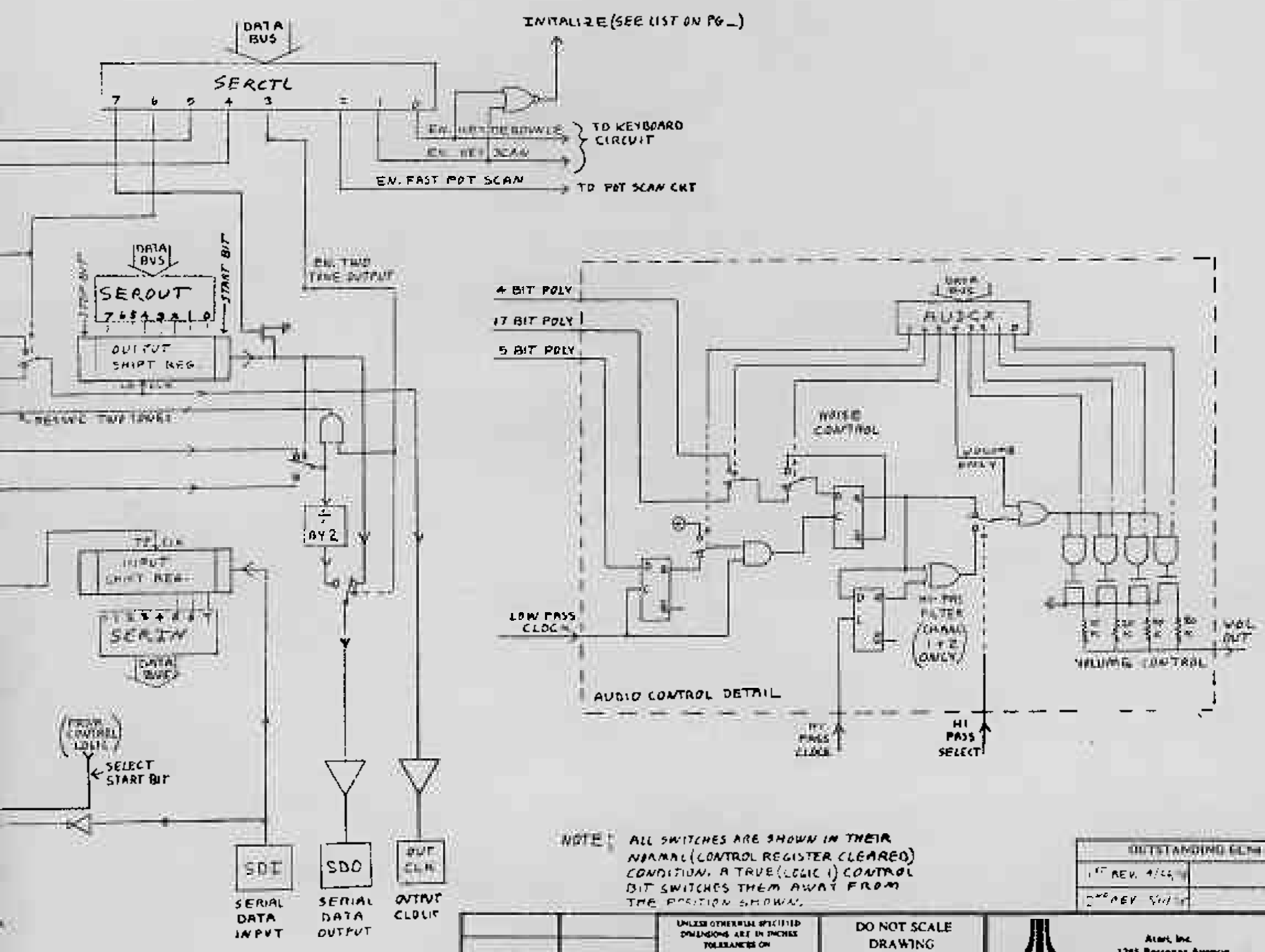


ROM CARTRIDGE BOARD

SCHEMATIC ROM CARTRIDGE




4 BIT POLY
17 BIT POLY
4 BIT TAP
5 BIT POLY
1.79 MHZ
64 KHZ
15 KHZ
AUDIO SERIAL PORT
RESYNC 2 TONES
DATA BUS
AUDF1
÷ BY "N"
AUDF2
AUDF3
AUDF4
AUDC1
AUDC2
AUDC3
AUDC4
AUDIO CONTROL
AUDCTL
AUD
4 CHANNEL AUDIO OUT PUT
BI-DIRECTIONAL SERIAL DATA CLOCK



NOTE: ALL SWITCHES ARE SHOWN IN THEIR NORMAL (CONTROL REGISTER CLEARED) CONDITION. A TRUE (LOGIC 1) CONTROL BIT SWITCHES THEM AWAY FROM THE POSITION SHOWN.

OUTSTANDING ECN
1ST REV. 4/
2ND REV.

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ON

DO NOT SCALE DRAWING

Atari, Inc.
1265 Borregas Avenue
Sunnyvale, Calif 94086
A Warner Communications Company

TITLE COLLEEN MANUAL
AUDIO AND SERIAL PORT
BLOCK DIAGRAM

SIZE D

APPENDIX A

USE OF PLAYER/MISSILE GRAPHICS

WITH BASIC

The ATARI® 400/800™ Hardware Manual should be read first to understand the details of the Player/Missile Graphics.

To enable the P/M Graphics from BASIC the following procedure can be used:*

1. Generate the playfield, either with a GRAPHICS call or build a custom display list with a series of POKE statements.

2. Enable P/M DMA control by a POKE 559 with either a 62 for single line resolution players or a 46 for double line resolution players.

3. There are four players and four missiles (or five players if the four missiles are combined into one player). Each of these has a horizontal position register that controls its horizontal position on the screen. The registers and their locations are as follows:

ADDRESS	HORIZONTAL POSITION OF
53248	Player 0
53249	Player 1
53250	Player 2
53251	Player 3
53252	Missile 1
53253	Missile 2
53254	Missile 3
53255	Missile 4

The horizontal positions can range on the playfield between 41 and 200. So POKE 53249,120 will move Player 1 to the middle of the screen.

---

*NOTE: All number references are decimal.

4. Each player (and its missile) has a color register which determines its color. These registers can be controlled by poking to the following locations:

ADDRESS	COLOR OF
704	P/M 0
705	P/M 1
706	P/M 2
707	P/M 3
711	fifth player (if enabled)

Thus a POKE 706,200 will color player 2 green.

5. The P/M bit information (those bytes which actually describe the shape of the player) must be stored in an area where it will not interfere with BASIC or the operating system. It must also start at a 2K memory boundary if single line resolution players are used, or a 1K boundary for double line resolution players.

6. The page number (i.e. number of 256 byte sections of memory) for the starting address of the P/M information obtained in step 5 is poked into location 54279.

7. Enable the P/M DMA by a POKE 53277,3.

8. The starting address of each player is obtained by multiplying the number obtained in step 6 by 256 and then adding the offset indicated in P/M memory configuration table.

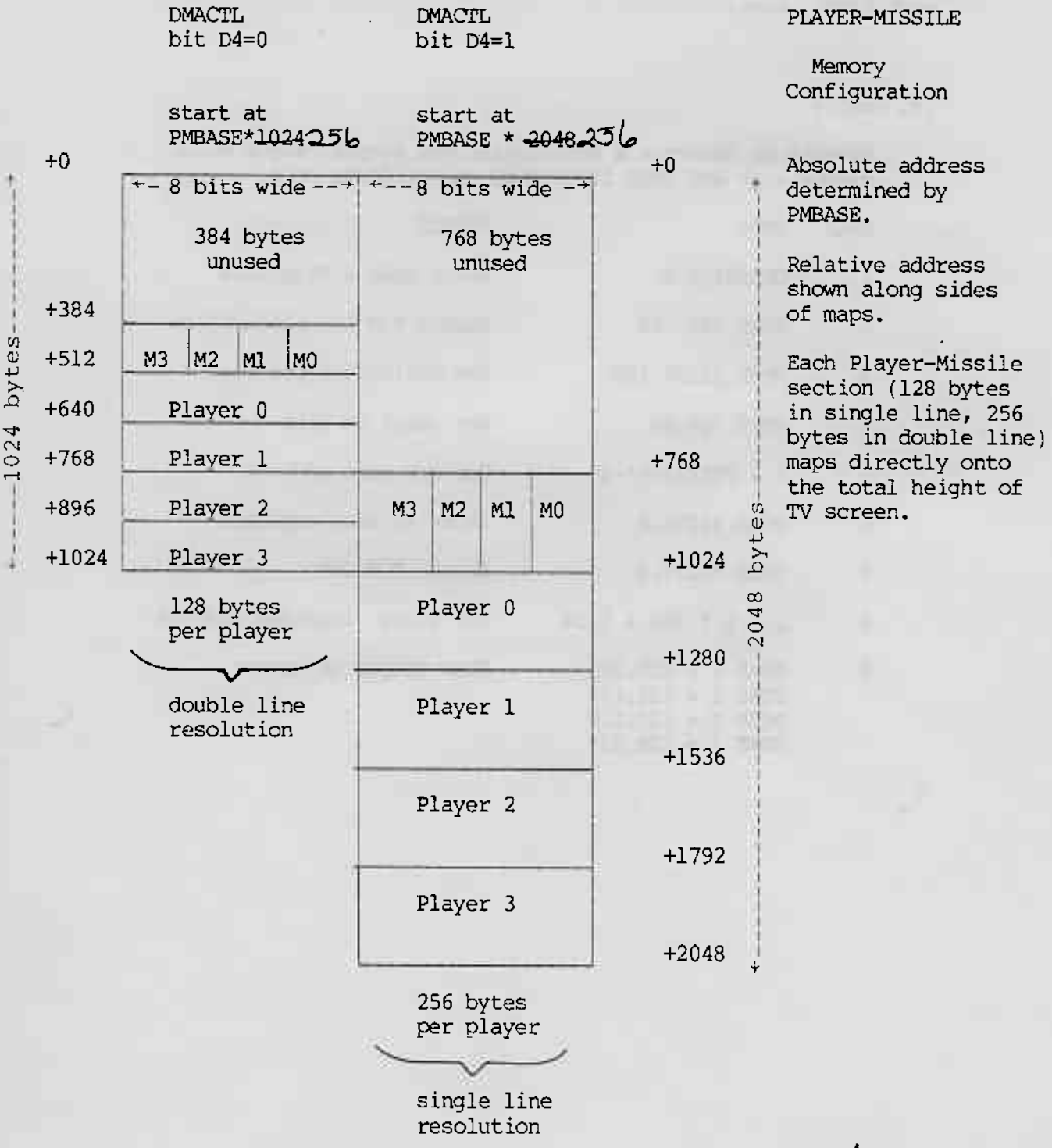
9. The vertical position of the player is determined by its location in memory. After the initial offset is obtained in step 8, its height may be defined. Its range on the playfield is from 32 to 223 in single line resolution and from 16 to 111 in double line resolution. By adding the desired height to the initial offset, the absolute address of each player is found. The appropriate bit information for the player can now be poked into this address.

(9, cont.)

Example to Generate a rectangular box player, eight color clocks wide and four lines high in immediate mode.

STEP	TYPE	RESULT
1	GRAPHICS 8	Setup Mode 8 Playfield
2	POKE 559, 62	Enable P/M DMA single line
3	POKE 53248,120	Set horizontal position
4	POKE 704,88	Set color to pink
5	I = PEEK(106)-8	Get P/M base address
6	POKE 54279,I	Store in base register
7	POKE 53277,3	Enable P/M DMA
8	J = I * 256 + 1024	Get player starting address
9	POKE J + 125,255 POKE J + 126,129 POKE J + 127,129 POKE J + 128,255	Draw player on screen

Use of Player/Missile Graphics
with BASIC, cont.



Absolute address determined by PMBASE.

Relative address shown along sides of maps.

Each Player-Missile section (128 bytes in single line, 256 bytes in double line) maps directly onto the total height of TV screen.

APPENDIX B

MIXING GRAPHICS MODES

I. GENERAL

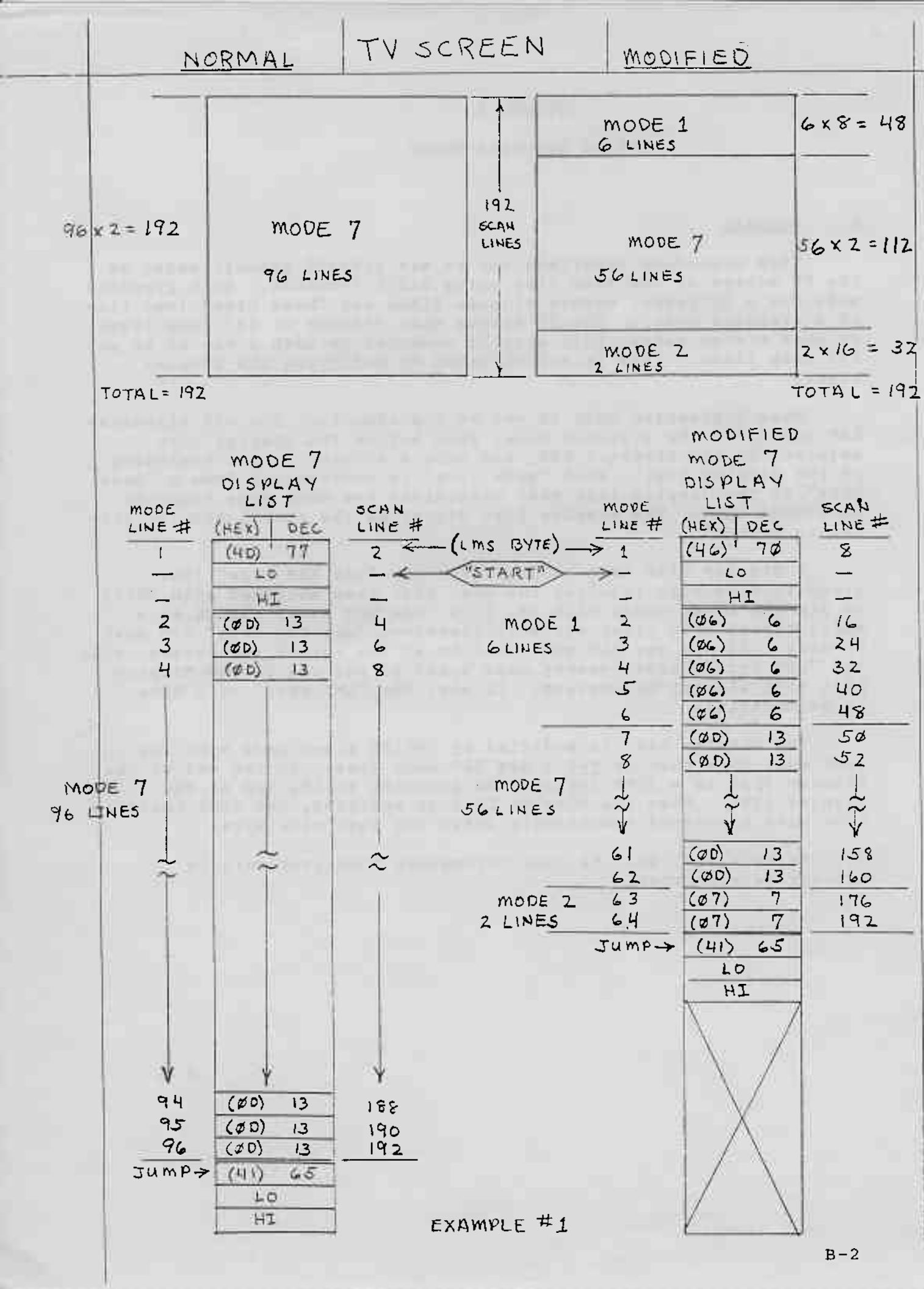
This procedure describes how to mix several graphic modes on the TV screen at the same time using BASIC commands. Each graphics mode has a different number of scan lines per "Mode Line" (one line of a graphics mode). The TV screen must consist of 192 scan lines, so when mixing modes, they must be combined in such a way as to get 192 scan lines. This is accomplished by modifying the Display List.

When a graphics mode is set on the computer, the O/S allocates RAM space for the graphics mode, then builds the display list adjacent to the graphics RAM, and sets a pointer to the beginning of the display list. Each "mode line" is constructed from a "mode byte" in the display list that determines how many scan lines in each mode line. The display list describes the screen display from top to bottom.

A Display List must be built for the "max RAM mode" (the graphics mode that requires the most RAM) then modified with POKES to mix the other modes with it. This "max RAM mode" cannot be a split screen mode (text window), therefore "max RAM mode" +16 must be used. If the max RAM mode will be at the top of the screen, then the "LMS byte" (load memory scan byte) at the top of the Display List will already be correct. If not, the "LMS byte" will have to be modified.

The Display List is modified by POKING a new mode byte for each mode line that is <u>not</u> a max RAM mode line. At the end of the display list is a JUMP instruction pointing to the top of the Display list. When the Display List is modified, the JUMP instruction must be placed immediately after the last mode byte.

Example #1 will be used throughout this procedure to illustrate each step.


NORMAL
TV SCREEN
MODIFIED
96 x 2 = 192
MODE 7
96 LINES
192 SCAN LINES
MODE 1
6 LINES
6 x 8 = 48
MODE 7
56 LINES
56 x 2 = 112
MODE 2
2 LINES
2 x 16 = 32
TOTAL = 192
TOTAL = 192
MODE 7 DISPLAY LIST
MODE LINE #
(HEX) | DEC
SCAN LINE #
(LMS BYTE)
"START"
MODIFIED MODE 7 DISPLAY LIST
MODE 7 96 LINES
MODE 1 6 LINES
MODE 7 56 LINES
MODE 2 2 LINES
Jump
LO
HI
EXAMPLE #1
B-2

II. PROCEDURE TO SET UP SCREEN IN MIXED MODES:

1. Select modes desired, then look up which mode is the max RAM mode from table #2.

   example: modes selected - mode 1, mode 7, mode 2

   mode 7 = max RAM mode

2. Use table #1 to calculate the number of mode lines such that the total number of scan lines = 192.

   example:

mode	# mode line	scan lines per mode line	scan lines
1	6	8	48
7	56	2	112
2	2	16	32
			192 TOTAL

3. If the max RAM mode is at the top of the screen, then skip this step: Calculate the LMS byte by setting the left nibble to 4, then use table #1 to find the right nibble for the graphics mode at the top of the screen.

   example: 1. left nibble = 4
   2. right nibble for mode 1 = 6
   3. LMS byte = 46 (HEX)

4. Calculate the mode byte for each mode. Set the left nibble to 0, use table #1 to find the right nibble for each mode.

   example:

Mode	Left Nibble	Right Nibble	Mode Byte (HEX)
1	0	6	06
7	0	D	0D
2	0	7	07

II. PROCEDURE TO SET UP SCREEN IN MIXED MODES, cont.:

5. Convert all bytes to decimal.

example:

Byte	(HEX)	DEC
LMS	46	70
Mode 1	06	6
Mode 7	0D	13
Mode 2	07	7

6. Execute a graphics call on the computer using the max RAM mode (+16).

   example: GRAPHICS 7 + 16

7. PEEK the Display List pointer and use it to calculate a variable labelled "START".

   example: START = PEEK(560) + PEEK(561) * 256 + 4

8. If the max RAM mode is at the top of the screen, then skip this step: Poke the LMS byte to location START-1.

   example: POKE START-1,70

9. Every mode line requires a mode byte in the Display List in the same order as the mode lines appear on the screen. The mode bytes must be POKED into the Display List at location START + offset, where offset = mode line #.

Example:	MODE LINE #	POKE INSTRUCTION
MODE 1	2	POKE START + 2,6
	3	POKE START + 3,6
	4	POKE START + 4,6
	5	POKE START + 5,6
	6	POKE START + 6,6
MODE 7	see note for mode 7 (max RAM mode)	
MODE 2	63	POKE START + 63,7
	64	POKE START + 64,7

NOTE: The Display List will already be correct for the max RAM mode, therefore its mode bytes do not need to be POKEd.

II. PROCEDURE TO SET UP SCREEN IN MIXED MODES, cont.:

10. POKE the JUMP instruction followed by the LO byte, then the HI byte into the Display List. The offset for the JUMP POKE is the last mode line # + 1, for LO byte it is + 2, for HI byte it is + 3.

example: (last mode line # was 64)

REMARK	POKE INSTRUCTION
JUMP	POKE START + 65,65
LO BYTE	POKE START + 66, PEEK(560)
HI BYTE	POKE START + 67, PEEK(561)

III. PROCEDURE TO PRINT AND PLOT IN MIXED MODES

1. If the mode line #'s of a mode on the screen fall within the range of that mode's normal mode line #'s then use the following procedure:
   a. POKE 87 with the mode #
   b. Determine the Y coordinate by counting the # of mode lines from the top of the screen to the current position.
   c. Determine the X position in the normal manner for that mode.
   d. Depending on the mode, either PLOT and DRAWTO, or POSITION and PRINT.
   e. These steps must be done for each mode on the screen that meets the condition in step 1.

```
example: MODE 1   POKE 87,1
                  POSITION 2,1:PRINT #6;"TEXT"

         MODE 7   POKE 87,7
                  COLOR 1:PLOT 20,20:DRAWTO 30,30

         MODE 2   See step 2
```

III. PROCEDURE TO PRINT AND PLOT IN MIXED MODES, cont.

2. Some modes may have mode line #'s outside of their normal range.

example: Mode 2 normally has mode line #'s 1 through 12 (full screen). These are modified to #63 and #64 in example #1.

To prevent the computer from giving a "cursor out of range" error message the following procedure can be used:

a. Set a variable labelled "MEMST" to be the display memory start pointer.

```
MEMST = PEEK(START) + PEEK(START + 1) * 256
```

b. Set a variable labelled CHRPOS to position characters to be printed on the target line.

$$CHRPOS = MEMST + [(M_1 - 1)*R - M_2*(R-20) - M_3*(R-10)] + X$$

Where:

X = horizontal position of character on the target line.
R = the RAM per line of the Max RAM Mode (table #1).
M_1 = the Mode Line # of the target line.
M_2 = the number of mode lines of 20 bytes of RAM per line above the target line.
M_3 = the number of mode lines of 10 bytes of RAM per line above the target line.

Example: calculate CHRPOS for Mode Line #64 (the last line of the Mode 2 area) at horizontal position 5.

X = 5
R = 40
M_1 = 64
M_2 = 7 (6 from Mode 1 area, 1 from Mode 2 area).
M_3 = 0

```
CHRPOS = MEMST + [(64-1)*40-7*(40-20)-0*(40-10)]+5
CHRPOS = MEMST + [(63)*40-7*(20)-0*(30)]+5
CHRPOS = MEMST + [2520 - 140] + 5
CHRPOS = MEMST + [2380] + 5
CHRPOS = MEMST + 2385
```

III. PROCEDURE TO PRINT AND PLOT IN MIXED MODES, cont.

2. cont.

c. If few characters will be printed, then each character's internal value may be looked up in the Internal Character Set Table (Table 9.6), in the new BASIC Reference Manual. This value is then POKED into CHRPOS.

d. If strings are to be output, and if the ATASCII values of all the characters lie within one of the ranges shown in the table below, then do the following:

1) Obtain the appropriate ATASCII value range for the characters
2) Do the OPERATION the table indicates on the ATASCII value of each character.
3) POKE this value into CHRPOS.

ATASCII VALUE RANGE	OPERATION
0-31	Value + 64
32-95	Value - 32
96-127	NONE
128-159	Value + 64
160-223	Value - 32
224-255	NONE

Example: 1) assume we want to print the word "TEXT" in the mode 2 area of example #1 using the CHRPOS calculated previously.

2) these characters are in the ATASCII VALUE RANGE of "32 - 95".

3) the OPERATION for this range is "Value-32", so 32 must be subtracted from each ATASCII value.

4) the program statements would now look like this:

```
T$(1,4) = "TEXT"
CHRPOS  = MEMST + 2385

FOR X = 1 TO LEN(T$)
POKE CHRPOS + X - 1, ASC[T$(X,X)] - 32

NEXT X
```

(OPERATION: value - 32)

5) the FOR/NEXT loop POKEs the first character of T$, ASC[T$(X,X)]-32, into CHRPOS + 0.

6) the next iteration POKEs the next character of T$ into the next CHRPOS, and so on.

Mixing Graphics Modes, cont.

TABLE #1

REMARK	MODE BYTE: LEFT NIBBLE (HEX)		MODE BYTE: RIGHT NIBBLE (HEX)	C.C. PER PIXEL	SCAN LINES PER MODE LINE	# COLORS	MODE	RAM PER LINE
			2	½	8	1½ ③	0	40
①	4	CHAR	3	½	10	1½	-	40
			4	1	8	4	-	40
		MODES	5	1	16	4	-	40
②	0		6	1	8	5	1	20
			7	1	16	5	2	20
			8	4	8	4	3	10
			9	2	4	2	4	10
①	4	GRAPHIC	A	2	4	4	5	20
			B	1	2	2	6	20
		MODES	C	1	1	2	-	20
			D	1	2	4	7	40
②	0		E	1	1	4	-	40
			F	½	1	1½	8	40
BLANK	0-7	④	0	BLANK	-	-	-	-
JUMP	4	SPECIAL	1	JUMP	-	-	-	-

(1.) When the max RAM mode is not at the top of the screen, the left nibble of the LMS byte must be changed to a 4.

(2.) Left nibble for all mode bytes after the LMS byte.

(3.) Color & Lum for the field is controlled by Setcolor 2, and Lum for charactors or graphics from Setcolor 1.

(4.) JUMP - used to end the display list and return to the beginning.

BLANK - to output selected number of background lines.

Mixing Graphics Modes, cont.

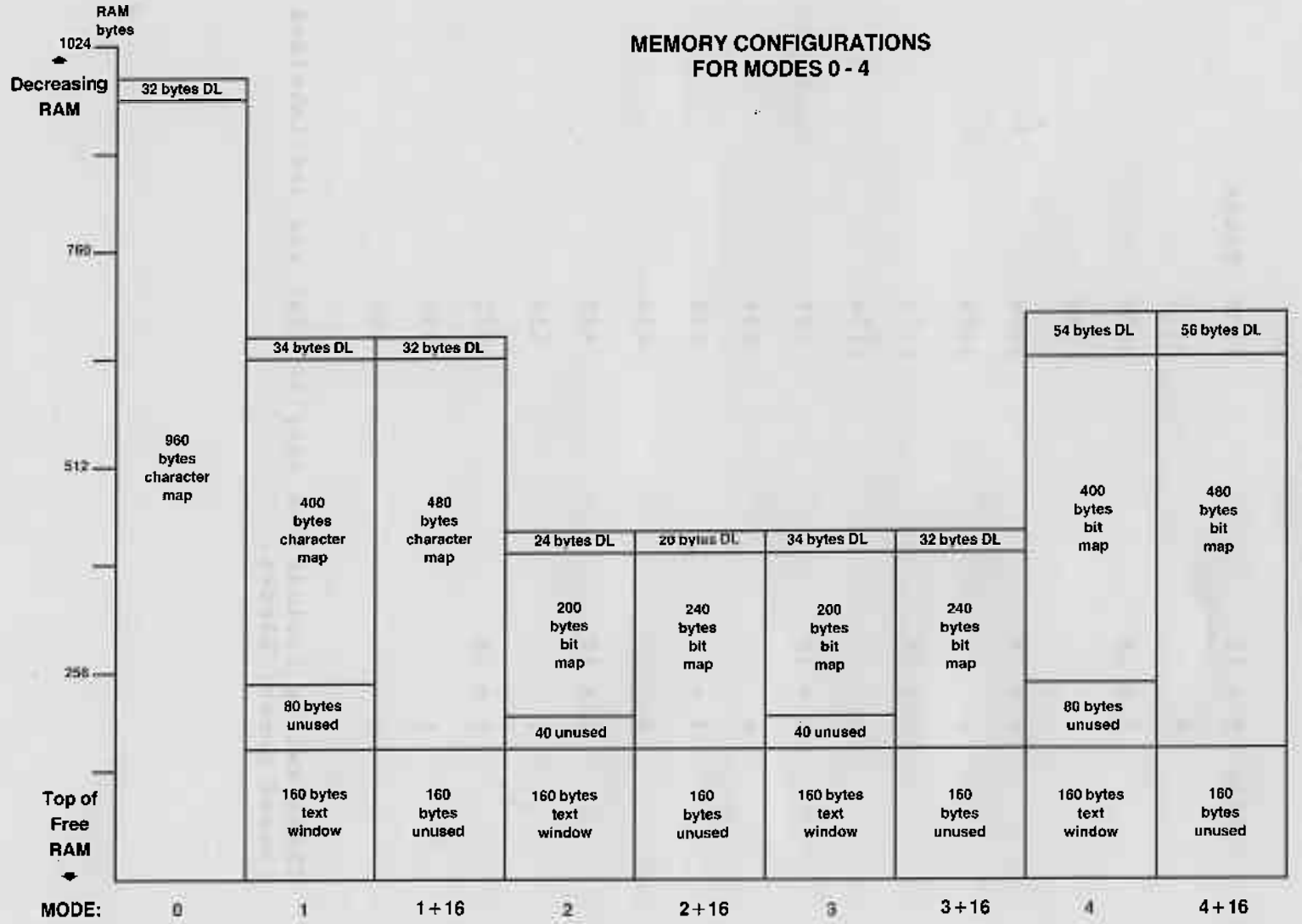
TABLE #2

GRAPHICS MODES
RAM REQUIREMENTS

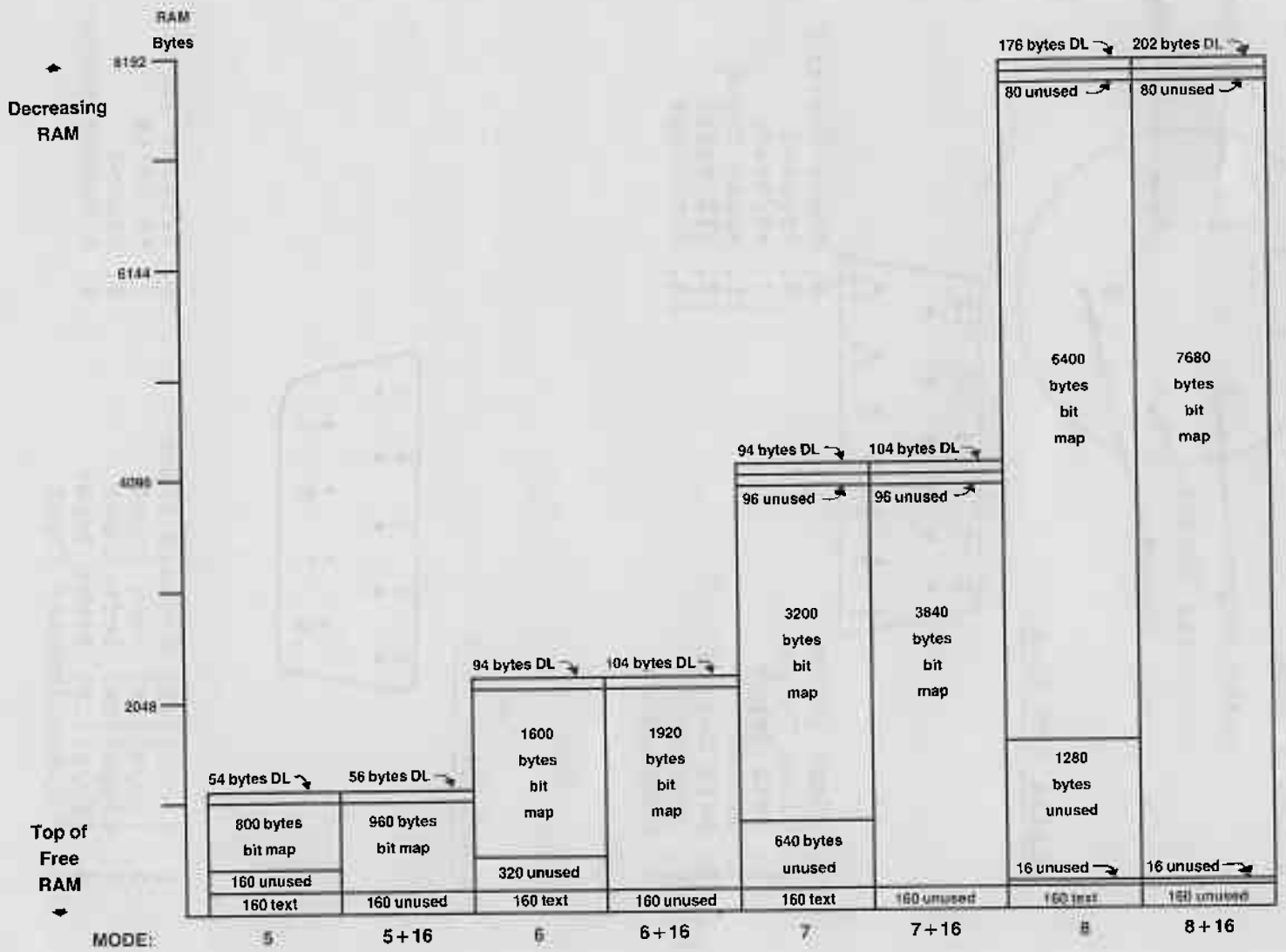
Mode		
	8 + 16	8138 Bytes
	8	8112
	7 + 16	4200
	7	4190
	6 + 16	2184
	6	2174
	5 + 16	1176
	5	1174
	4 + 16	696
	4	694
	3 + 16	432
	3	434
	2 + 16	420
	2	424
	1 + 16	672
	1	674
	0	992

These values include the display list and any imbedded unused memory blocks.

MEMORY CONFIGURATIONS FOR MODES 0 - 4

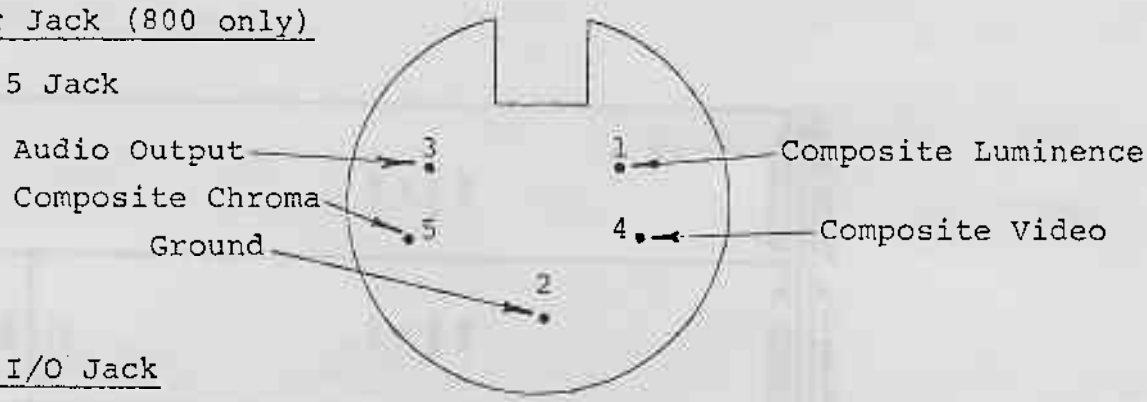


MEMORY CONFIGURATIONS FOR MODES 5 - 8

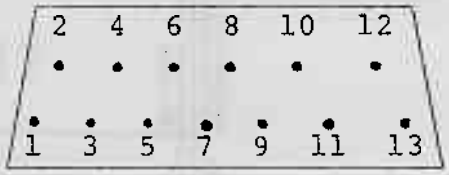


Monitor Jack (800 only)

D.I.N. 5 Jack

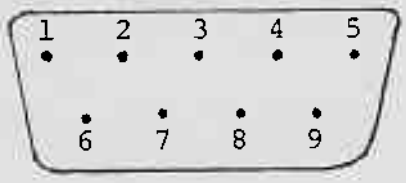


Serial I/O Jack



1. Clock Input
2. Clock Output
3. Data Input
4. Ground
5. Data Output
6. Ground
7. Command
8. Motor Control
9. Proceed
10. +5/Ready
11. Audio Input
12. +12 volts
13. Interrupt

Controller Jack



1. (Joystick) Forward Input
2. (Joystick) Back Input
3. (Joystick) Left Input
4. (Joystick) Right Input
5. B Potentiometer Input
6. Trigger Input
7. +5 volts
8. Ground
9. A Potentiometer Input