ATARI® 400/800™

ATARI® HOME COMPUTER SYSTEM

TECHNICAL REFERENCE NOTES

includes:

Operating System User's Manual
Operating System Source Listing
and
Hardware Manual

ATARI® A Warner Communications Company

C016555 Rev. A

ATARI® 400/800™

ATARI® HOME COMPUTER SYSTEM

OPERATING SYSTEM USER'S 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.

ATARI Home Computer
Operating System USER'S MANUAL

APPENDICES

TABLE OF ILLUSTRATIONS

PREFACE

This manual describes the resident Operating System (OS) for the ATARI@ Home Computer, for readers who are familiar with the internal behavior of the system. It discusses:

- o System functions and utilization techniques
- o Subsystem relationships and organization
- o Characteristics of the ATARI peripheral devices that can be attached to the ATARI400[TM] and ATARI 800[TM] Home Computer
- o Advanced techniques for going beyond the basic OS capabilities
- o The general features of the computer system hardware used by the OS.

It would be helpful to have a familiarity with programming concepts and terminology, assembly language programming in general, the Synertek 6502 in particular, and digital hardware concepts and terminology. you will be provided with the information you need to use the OS resources, without resorting to trial-and-error techniques or the OS listing. Supporting information for tasks that involve OS listing references is also provided.

This manual does not present a comprehensive description of the hardware used to provide OS capabilites. The programmer who needs to go beyond the capabilities described should consult the ATARI Home Computer Hardware Manual.

1 INTRODUCTION

GENERAL DESCRIPTION OF THE ATARI HOME COMPUTER SYSTEM

Operating systems in the ATARI@ 400[TM] and ATARI 800[TM] Home Computer are identical. The primary differences between the two are:

- o Physical packaging
- o The ATARI 400 Computer console has one cartridge slot, the ATARI 800 Computer console has two cartridge slots
- o The ATARI 400 Home Computer contains 16K RAM and cannot be expanded. The ATARI 800 Home Computer can be expanded to a maximum of 48K RAM.
- o The ATARI 800 Computer has a monitor jack; the ATARI 400 Computer does not.

The Hardware Circuitry

- o Produces both character and point graphics for black and white (B/W) or color television.
- o Produces four independent audio channels (frequency controlled) which use the television sound system.
- o Provides one bi-level audio output in the base unit.
- o Interfaces with up to four Joysticks and eight Paddle Controllers.
- o Interfaces with a serial I/O bus for expansion.
- o Contains a built-in keyboard

Figure 1-1 presents a simplified block diagram of the hardware. See the hardware manual for supporting documentation.

```
     +---------+
     |  6502   |
     |processor|
     +----+----+    +---------+
          |         |   OS    |
          +---------+   ROM   |        +---------+
          |         |D800-FFFF|      +---------+ |
          |         +---------+    +---------+ | |
          |                        |   RAM   | | |
          +------------------------+         | |-+
          |                        |0000-xxxx|-+
          |         +---------+    +---------+
          |         |Cartridge|
          +---------+slots A&B|
          |         |8000-BFFF|
processor |         +---------+
external  |                                       +---------+
bus       |         +---------+ joysticks         |control- |
          |         |   PIA   +--------+----------+ler ports|
          +---------+         +--------|-------+  |         |
          |.........|D300-D31F+--------|-----+ |  +---------+
          |   IRQ   +---------+        |     | |
          |                            |     | |  +---------+
          |         +---------+ pots   |     | |  |keyboard |
          |         |  POKEY  +--------+     | |  |data keys|
          +---------+         +--------|-----|-|--+ + BREAK |
          |.........|D200-D21F+--------|---+ | |  +---------+
          |   IRQ   +----+----+ audio  |   | | | on/off
          |              +-------------|-+ | | |  +---------+
          |         +----+----+ trigs  | | | | +--+cassette |
          |         |  CTIA   +--------+ | | |    |interface|
          +---------+         +-----+  | +-|-|----+         |
          |         |D000-D01F+-----|--|-+ | |    +---------+
          |         +----+----+     |  | | | +----+ serial  |
          |              | video    |  | | |      |   I/O   |
          |   DMA   +----+----+ litpen | | +------+   bus   |
          |.........|  ANTIC  +--------+ |        +---------+
          +---------+         |     |    |
          |.........|D400-D41F|     |    |        +---------+
              NMI   +---------+     |    |  from  | console |
                                    |    |  CTIA -+ speaker |
                   composite   +----+    |        |         |
                   audio/video |         |        +---------+
                               |         |
                          +----+----+    |        +---------+
                          |         |    |        | START,  |
                          |    TV   |    +--------+ SELECT, |
                          |         |             | OPTION  |
                          +---------+             +---------+
```

Figure 1-1. ATARI Home Computer Block Diagram

CONVENTIONS USED IN THIS MANUAL

This manual uses the following special notations:

Hexadecimal Numbers

All two-digit numbers preceded by a dollar sign ($) designate hexadecimal numbers. All other numbers (except memory addresses) are in decimal form unless otherwise specified in the supporting text.

Memory Addresses

All references to computer memory and mapped I/O locations are in hexadecimal notation. Memory addresses may or may not be contained in square brackets. (Example: [D20F] and D20F are the same address.)

Kilobytes of Memory

Memory sizes are frequently expressed in units of kilobytes, such as 32K, where a kilobyte is 1024 bytes of memory.

PASCAL As an Algorithm-Specification Language

The PASCAL language (procedure block only) is used as the specification language in the few places where an algorithm is specified in detail. PASCAL syntax is similar to any number of other block-structured languages, and you should have no difficulty following the code presented.

Memory Layouts

Diagrams similar to Figure 1-2 are used whenever pictures of bytes or tables are presented:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!               ! --- This is a single byte.
+-+-+-+-+-+-+-+-+
!               !
+               + --- This is a word (2 bytes).
!               !
+-+-+-+-+-+-+-+-+
!               !
=               = --- This is a block of memory
!               !     of unspecified length.
+-+-+-+-+-+-+-+-+
```

Figure 1-2. Memory Layout Chart

Bit 7 is the most significant bit (MSB) of the byte, and Bit 0 is the least significant bit (LSB).

In tables and figures, memory addresses always increase toward the bottom of the figure.

Backus-Naur Form

A modified version of Backus-Naur Form (BNF) is used to express some syntactic forms, where the following metalinguistic symbols are used:

::=	is the substitution (assignment) operator.
< >	a metasyntactic variable.
¦	separates alternative substitutions.
[ ]	an optional construct.

Anything else is a syntactic literal constant, which stands for itself.

For Example:

<device specification> ::= <device name>[<device number>]:

<device name> ::= C¦D¦E¦K¦P¦R¦S

<device number> ::= 1¦2¦3¦4¦5¦6¦7¦8

A "device specification" consists of a mandatory "device name," followed by an optional "device number," followed by the mandatory colon character. The device name in turn must be one of the characters shown as alternatives. The device number (if it is present) must be a digit 1 through 8.

OS Equate Filenames

Operating System ROM (Read Only Memory) and RAM (Random Access Memory) vector names, RAM database variable names and hardware register names are all referred to by the names assigned in the OS program equate list. When one of these names is used, the memory address is usually provided, such as BOOTAD [0242].

2 OPERATING SYSTEM FUNCTIONAL ORGANIZATION

This section describes the various subsystems of the resident OS in general terms.

Input/Output Subsystem

The Input/Output (I/O) subsystem provides a high-level interface between the programs and the hardware. Most functions are device-independent, such as the reading and writing of character data; yet provisions have been made for device-dependent functions as well. All peripheral devices capable of dealing with character data have individual symbolic names (such as K,D,P, etc). and can be accessed using a Central I/O (CIO) routine.

A RAM data base provides access to controllers (joysticks and paddle controllers), which do not deal with character data. This RAM data base is periodically updated to show the states of these devices.

INTERRUPT PROCESSING

The interrupt system handles all hardware interrupts in a common and consistent manner. By default, all interrupts are fielded by the OS. At your discretion, individual interrupts (or groups of interrupts) can be fielded by the application program.

INITIALIZATION

The system provides two levels of initialization: power up and system reset. The OS performs power-up initialization each time the system power is switched to ON, and system reset initialization is performed each time the [SYSTEM.RESET] key is pressed.

Power-Up

The OS examines and notes the configuration of the unit whenever the system power is switched to ON. The system performs the following tasks at power up:

- o Determines the highest RAM address.
- o Clears all of RAM to zeros.
- o Establishes all RAM interrupt vectors.
- o Formats the device table.
- o Initializes the cartridge(s).
- o Sets up the screen for 24 x 40 text mode.
- o Boots the cassette if directed.
- o Checks cartridge slot(s) for diskette-boot instructions.
- o Boots the diskette if directed to do so and a disk drive unit is attached.
- o Transfers control to the cartridge, diskette-booted program, cassette-booted program, or blackboard program.

[SYSTEM.RESET]

Pressing the [SYSTEM.RESET] key causes the OS to perform these following tasks:

- o Clears the OS portion of RAM.
- o Rechecks top of RAM.
- o Reestablishes all RAM interrupt vectors.
- o Formats the device table.
- o Initializes the cartridge(s).
- o Sets up the screen for 24 x 40 text mode.
- o Transfers control to the cartridge, a diskette-booted program, a cassette-booted program, or the blackboard program.

Note that [SYSTEM.RESET] does not perform all the power-up tasks listed in the power-up section.

FLOATING POINT ARITHMETIC PACKAGE

The OS ROM contains a Floating Point (FP) package that is available to nonresident programs such as ATARI BASIC.
The package is not used by the other parts of the OS itself. The floating point numbers are stored as 10 BCD digits of mantissa, plus a 1-byte exponent. The package contains these routines:

- o ASCII-to-FP and FP-to-ASCII conversion.
- o Integer-to-FP and FP-to-integer conversion.
- o FP add, subtract, multiply and divide.
- o FP log, exp, and polynomial evaluation.
- o FP number clear, load, store, and move.

You set these 2-bytes. They contain information that is used by the OPEN command process and/or is device-dependent.

For OPEN, two bits of ICAX1 are always used to specify the OPEN direction as shown below, where R is set to 1 for input (read) enable and W is set to 1 for output (write) enable.

```
 7     3 2  0
+-+-+-+-+-+-+-+-+
| | | | |W|R| | |
+-+-+-+-+-+-+-+-+
```

ICAX1 is not altered by CIO. You should not alter ICAX1 once the device/file is open.

The remaining bits of ICAX1 and all of ICAX2 contain only device-dependent data and are explained later in this section.

Remaining Bytes (ICAX3-ICAX6)

The handler reserves the four remaining bytes for processing the I/O command for CIO. There is no fixed use for these bytes. They are not user-alterable except as specified by the particular device descriptions. These bytes will be referred to as ICAX3, ICAX4, ICAX5 and ICAX6, although there are no equates for those names in the OS equate file.

CIO Functions

The CIO supports records and blocks and the handlers support single bytes. All of the system handlers support one or more of the eight basic functions subject to restrictions based upon the direction of data transfer (e.g. one cannot read data from the printer). The basic functions are: OPEN, CLOSE, GET CHARACTERS, PUT CHARACTERS, GET RECORD, PUT RECORD, GET STATUS, and SPECIAL.

OPEN -- Assign Device/Filename to IOCB and Ready for Access

A device/file must be opened before it can be accessed. This process links a specific IOCB to the appropriate device handler, initializes the device/file, initializes all CIO control variables, and passes device-specific options to the device handler.

You set up the following IOCB parameters prior to calling CIO for an OPEN operation:

COMMAND BYTE = $03

BUFFER ADDRESS = pointer to a device/filename specification.

AUX1 = OPEN direction bits, plus device-dependent information.

AUX2 = device-dependent information.

After an OPEN operation, CIO will have altered the following IOCB parameters:

HANDLER ID = index to the system device table; this is used only by CIO and must not be altered.

DEVICE NUMBER = device number taken from the device/filename specification and must not be altered.

STATUS = result of OPEN operation; see Appendix B for a list of the possible status codes. In general, a negative status will indicate a failure to open properly.

PUT ADDRESS = pointer to the PUT CHARACTERS routine for the device handler just opened.

It is recommended that this pointer not be used.

CLOSE -- Terminate Access to Device/File and Release IOCB.

You issue a CLOSE command after you are through accessing a given device/file. The CLOSE process completes any pending data writes, goes to the device handler for any device-specific actions, and then releases the IOCB.

You set the following IOCB parameter prior to calling CIO:

COMMAND BYTE = $0C

The CIO alters the following IOCB parameters as a result of the CLOSE operation:

HANDLER ID = $FF

STATUS = Result of CLOSE operation.

PUT ADDRESS = pointer to "IOCB not OPEN" routine.

GET CHARACTERS -- Read n Characters (Byte-Aligned Access)

The specified number of characters are read from the device/file to the user-supplied buffer. EOL characters have no termination features when using this function; there can be no EOL, or many EOL's, in the buffer after operation completion. There is a special case provided that passes a single byte of data in the 6502 A register when the buffer length is set to zero.

You set the following IOCB parameters prior to calling CIO:

COMMAND BYTE = $07

BUFFER ADDRESS = pointer to data buffer.

BUFFER LENGTH = number of bytes to read; if this is zero, the data will be returned in the 6502 A register only.

The CIO alters the following IOCB parameters as a result of the GET CHARACTERS operation:

STATUS = result of GET CHARACTERS operation.

BYTE COUNT/BUFFER LENGTH = number of bytes read to the buffer. The BYTE COUNT will always equal the BUFFER LENGTH except when an error or an end-of-file condition occurs.

PUT CHARACTERS -- Write n Characters (Byte-Aligned Access)

The specified number of characters are written from the user-supplied buffer to the device/file. EOL characters have no buffer terminating properties, although they have their standard meaning to the device/file receiving them; no EOL's are generated by CIO. There is a special case that allows a single character to be passed to CIO in the 6502 A register if the buffer length is zero.

You set the following IOCB parameters prior to initiating the PUT CHARACTERS operation:

COMMAND BYTE = $0B

BUFFER ADDRESS = pointer to data buffer.

BUFFER LENGTH = number of bytes of data in buffer.

The CIO alters the following IOCB parameter as a result of the PUT CHARACTERS operation:

STATUS = result of PUT CHARACTERS operation.

GET RECORD -- Read Up To n Characters (Record-Aligned Access)

Characters are read from the device/file to the user-supplied buffer until either the buffer is full or an EOL character is read and put into the buffer. If the buffer fills before an EOL is read, then the CIO continues reading characters from the device/file until an EOL is read,, and sets the status to indicate that a truncated record was read. No EOL will be put at the end of the buffer.

You set the following IOCB parameters prior to calling CIO:

COMMAND BYTE = $05

BUFFER ADDRESS = pointer to data buffer.

BUFFER LENGTH = maximum number of bytes to read (including the EOL character).

The CIO alters the following IOCB parameters as a result of the GET RECORD operation:

STATUS = result of GET RECORD operation.

BYTE COUNT/BUFFER LENGTH = number of bytes read to data buffer; this can be less than the maximum buffer length.

PUT RECORD -- Write Up To n Characters (Record-Aligned Access)

Characters are written from the user-supplied buffer to the device/file until either the buffer is empty or an EOL character is written. If the buffer is emptied without writing an EOL character to the device/file, then CIO will send an EOL after the last user-supplied character.

You set the following IOCB parameters prior to calling CIO:

COMMAND BYTE = $09

BUFFER ADDRESS = pointer to data buffer.

BUFFER LENGTH = maximum number of bytes in buffer.

The CIO alters the following IOCB parameter as a result of the PUT RECORD operation:

STATUS = result of PUT RECORD operation.

GET STATUS -- Return Device-Dependent Status Bytes

The device controller is sent a STATUS command, and the controller returns four bytes of status information that are stored in DVSTAT [02EA].

You set the following IOCB parameters prior to calling CIO:

COMMAND BYTE = $0D

BUFFER ADDRESS = pointer to a device/filename specification if the IOCB is not already OPEN; see the discussion of the implied OPEN option below.

After a GET STATUS operation, CIO will have altered the following parameters:

STATUS = result of GET STATUS operation; see Appendix B for a list of the possible status codes.

DVSTAT = the four-byte response from the device controller.

SPECIAL -- Special Function

Any command byte value greater than $0D is treated by CIO as a special case. Since CIO does not know what the function is, CIO transfers control to the device handler for complete processing of the operation.

The user sets the following IOCB parameters prior to calling CIO:

COMMAND BYTE > $0D

BUFFER ADDRESS = pointer to a device/filename specification if the IOCB is not already open; see the discussion of the implied OPEN option below.

Other IOCB bytes can be set up, depending upon the specific SPECIAL command being performed.

After a SPECIAL operation, CIO will have altered the following parameters:

STATUS = result of SPECIAL operation; see Appendix B for a list of the possible status codes.

Other bytes can be altered, depending upon the specific SPECIAL command.

Implied OPEN Option

The GET STATUS and SPECIAL commands are treated specially by CIO; they can use an already open IOCB to initiate the process or they can use an unopened IOCB. If the IOCB is unopened, then the buffer address must contain a pointer to a device/filename specification, just as for the OPEN command; CIO will then open that IOCB, perform the specified command and then close the IOCB again.

Device/Filename Specification

As part of the OPEN command, the IOCB buffer address parameter points to a device/filename specification, that is a string of ATASCII characters in the following format:

<specification> ::= <device>[<number>]:[<filename>]<eol>

<device> ::= C|D|E|K|P|R|S
<number> ::= 1|2|3|4|5|6|7|8
<filename> has device-dependent characteristics.
<eol> ::= $9B

The following devices are supported at this writing:

C = Cassette drive
D1 through D8 = Floppy diskette drives *
E = Screen Editor
K = Keyboard
P = 40-column printer
P2 = 80-column printer *
R1 through R4 = RS-232-C interfaces *
S = Screen display

Devices flagged by asterisks (*) are supported by nonresident handlers.

If <number> is not specified, it is assumed to be 1.

The following examples show valid device/filename specifications:

C:	Cassette
D2:BDAT	File "BDAT" on disk drive #2
D:HOLD	File "HOLD" on disk drive #1
K:	Keyboard

I/O Example

The example provided in this section illustrates a simple example of an I/O operation using the CIO routine.

```
; This code segment illustrates the simple example of reading
; text lines (records) from a diskette file named TESTER on disk
; drive #1. All symbols used are equated within the program
; although many of the symbols are in the OS equate file.

; The program performs the following steps:
;
;      1. Opens the file 'D1:TESTER' using IOCB #3.
;      2. Reads records until an error or EOF is reached.
;      3. Closes the file.

; I/O EQUATES

EOL=    $9B                     ; END OF LINE CHARACTER.
IOCB3=  $30                     ; IOCB #3 OFFSET (FROM IOCB #0).

ICHID=  $0340                   ; (HANDLER ID -- SET BY CIO).
ICDNO=  ICHID+1                 ; (DEVICE # -- SET BY CIO).
ICCOM=  ICDNO+1                 ; COMMAND BYTE.
ICSTA=  ICCOM+1                 ; STATUS BYTE -- SET BY CIO.
ICBAL=  ICSTA+1                 ; BUFFER ADDRESS (LOW).
ICBAH=  ICBAL+1                 ; BUFFER ADDRESS (HIGH).
ICPTL=  ICBAH+1
ICPTH=  ICPTL+1
ICBLL=  ICPTH+1                 ; BUFFER LENGTH (LOW).
ICBLH=  ICBLL+1                 ; BUFFER LENGTH (HIGH).
ICAX1=  ICBLH+1                 ; AUX 1.
ICAX2=  ICAX1+1                 ; AUX 2.

OPEN=   $03                     ; OPEN COMMAND.
GETREC= $05                     ; GET RECORD COMMAND.
CLOSE=  $0C                     ; CLOSE COMMAND.

OREAD=  $04                     ; OPEN DIRECTION = READ.
OWRIT=  $08                     ; OPEN DIRECTION = WRITE.

EOF=    $88                     ; END OF FILE STATUS VALUE.

CIOV=   $E456                   ; CIO ENTRY VECTOR ADDRESS.


;
; FIRST INITIALIZE THE IOCB FOR FILE "OPEN".
;

        LDX     #IOCB3          ; SETUP TO ACCESS IOCB #3.
```

```
        LDA     #OPEN           ; SETUP OPEN COMMAND.
        STA     ICCOM,X

        LDA     #NAME           ; SETUP BUFFER POINTER TO ...
        STA     ICBAL,X         ; ... POINT TO FILENAME.
        LDA     #NAME/256
        STA     ICBAH,X


        LDA     #OREAD          ; SETUP FOR OPEN READ.
        STA     ICAX1,X

        LDA     #0              ; CLEAR AUX 2.
        STA     ICAX2,X


;
; "OPEN" THE FILE.
;

        JSR     CIOV            ; PERFORM "OPEN" OPERATION.
        BPL     TP10            ; STATUS WAS POSITIVE -- OK.

        JMP     ERROR           ; NO -- "OPEN" PROBLEM.


;
; SETUP TO READ A RECORD.
;

TP10    LDA     #GETREC         ; SETUP "GET RECORD" COMMAND.
        STA     ICCOM,X

        LDA     #BUFF           ; SETUP DATA BUFFER POINTER.
        STA     ICBAL,X
        LDA     #BUFF/256
        STA     ICBAH,X


;
; READ RECORDS.
;


LOOP    LDA     #BUFFSZ         ; SETUP MAX RECORD SIZE ...
        STA     ICBLL,X         ; ... PRIOR TO EVERY READ.
        LDA     #BUFFSZ/256
        STA     ICBLH,X

        JSR     CIOV            ; READ A RECORD.
        BMI     TP20            ; MAY BE END OF FILE.

;
; A RECORD IS NOW IN THE DATA BUFFER "BUFF".  IT IS TERMINATED BY
```

```
; AN EOL CHARACTER, AND THE RECORD LENGTH IS IN "ICBLL" and "ICBLH".
; THIS EXAMPLE WILL DO NOTHING WITH THE RECORD JUST READ.
;

        JMP     LOOP            ; READ NEXT RECORD.


;
; NEGATIVE STATUS ON READ -- CHECK FOR END OF FILE.
;

TP20    CPY     #EOF            ; END OF FILE STATUS?
        BNE     ERROR           ; NO -- ERROR.

        LDA     #CLOSE          ; YES -- CLOSE FILE.
        STA     ICCOM,X

        JSR     CIOV            ; CLOSE THE FILE.

        JMP     *               ; *** END OF PROGRAM ***


;
; DATA REGION OF EXAMPLE PROGRAM
;

NAME    .BYTE   "D1:TESTER",EOL

BUFFSZ= 80                      ; 80 CHARACTER RECORD MAX
                                                (INCLUDES EOL).

BUFF=   *                       ; READ BUFFER.
*=      *+BUFFSZ
        .END
```

Figure 5-3   An I/O Example

Device-Specific Information

This section provides device-specific information regarding the device handlers that interface to CIO.

Keyboard Handler (K:)

The keyboard device is a read only device with a handler that supports the following CIO functions:

OPEN
CLOSE
GET CHARACTERS
GET RECORD
GET STATUS (null function)

The Keyboard Handler can produce the following error statuses:

$80 -- [BREAK] key abort.
$88 -- end-of-file (produced by pressing [CTRL] 3).

The Keyboard Handler is one of the resident handlers. It has a set of device vectors starting at location E420.

The keyboard can produce any of the 256 codes in the ATASCII character set (see Appendix F). Note that a few of the keyboard keys do not generate data at the Keyboard Handler level. These keys are described below:

[/|\] - The ATARI key toggles a flag that enables/disables the inversion of bit 7 of each data character read. The Screen Editor editing keys are exempted from such inversion, however.

CAPS - The [CAPS/LOWR] key provides three functions:

[SHIFT][CAPS/LOWR] -- Alpha caps lock.
[CNTRL][CAPS/LOWR] -- Alpha [CTRL] lock.
[CAPS/LOWR] -- Alpha unlock.

The system powers up and will system reset to the alpha caps lock option.

Some key combinations are ignored by the handler, such as [CTRL] 4 through [CTRL] 9, [CTRL] 0, [CTRL] 1, [CTRL] /, and all key combinations in that the [SHIFT] and [CTRL] keys are depressed simultaneously.

The [CTRL] 3 key generates an EOL character and returns EOF status.

The [BREAK] key generates an EOL character and returns BREAK status.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions (described earlier in this section) are detailed below:

OPEN

The device name is K, and the handler ignores any device number and filename specification, if included.

There are no device-dependent option bits in AUX1 or AUX2.

CLOSE

No special handler actions.

GET CHARACTERS and GET RECORD

The handler returns the ATASCII key codes to CIO as they are entered, with no facility for editing.

GET STATUS

The handler does nothing but set the status to $01.

Theory of Operation

Pressing a keyboard key generates an IRQ interrupt and vectors to the Keyboard Handler's interrupt service routine (see Section 6). The key code for the key pressed is then read and stored in data base variable CH [02FC]. This occurs whether or not there is an active read request to the Keyboard Handler, and effects a one-byte FIFO for keyboard entry. See Appendix L (E8) for a discussion of the auto repeat feature.

The Keyboard Handler monitors the CH variable for not containing the value $FF (empty state) whenever there is an active read request for the handler. When CH shows nonempty, the handler takes the key code from CH and sets CH to $FF again. The key code byte obtained from CH is not an ATASCII code and has the following form:

```
 7             0
+-+-+-+-+-+-+-+-+
|C|S| key code  |
+-+-+-+-+-+-+-+-+
```

Where: C = 1 if the [CTRL] key is pressed.
S = 1 if the [SHIFT] key is pressed.

The remaining six bits are the hardware key code.

The key code obtained is then converted to ATASCII using the first of the following rules that applies:

1. Ignore the code if the C and S bits are both set.
2. If the C bit is set, process the key as a [CTRL] code.
3. If the S bit is set, process the key as a [SHIFT] code.
4. If [CTRL] lock is in effect, process alpha characters as CTRL codes, all others as lowercase.
5. IF [SHIFT] lock is in effect, process alpha characters as SHIFT codes, all others as lowercase.
6. Else, process as lowercase character.

Then: If the resultant code is not a Screen Editor control code, and if the video inverse flag is set, then set bit 7 of the ATASCII code (will cause inverse video when displayed).

KEY CODE TO ATASCII CONVERSION TABLE

Key Code	Key Cap	Lwr. Case	[SHIFT]	[CTRL]	Key Code	Key Cap	Lwr. Case	SHIFT	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	9B*	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

* [CTRL] 3 returns EOF status.

A complement of this table (ATASCII to keystroke) is given in Appendix F.

Figure 5-4   Keycode to ATASCII Conversion Table

Display Handler (S:)

The display device is a read/write device with a handler that supports the following CIO functions:

- OPEN
- CLOSE
- GET CHARACTERS
- GET RECORD
- PUT CHARACTERS
- PUT RECORD
- GET STATUS (null function)
- DRAW
- FILL

The Display Handler can produce the following error statuses:

- $84 -- Invalid special command.
- $8D -- Cursor out-of-range.
- $91 -- Screen mode > 11.
- $93 -- Not enough memory for screen mode selected.

The Display Handler is one of the resident handlers, and therefore has a set of device vectors starting at location E410.

Screen Modes

You can operate the display screen in any of 20 configurations (modes 1 through 8, with or without split screen; plus mode 0, and modes 9 through 11 without split screen). Mode 0 is the text displaying mode. Modes 1 through 11 are all graphics modes (although modes 2 and 3 do display a subset of the ATASCII character set). Modes 9 through 11 require a GTIA chip to be installed in place of the standard CTIA chip.

TEXT MODE 0

In text mode 0 the screen is comprised of 24 lines of 40 characters per line. Program alterable left and right margins limit the display area. They default to 2 and 39 (of a possible 0 and 39).

A program-controllable cursor shows the destination of the next character to be output onto the screen. The cursor is visible as the inverse video representation of the current character at the destination position.

The text screen data is internally organized as variable length logical lines. The internal representation is 24 lines when the screen is cleared. Each EOL marks the end of a logical line as text is sent to the screen. If more than 3 physical lines of text are sent, a logical line will be formed every 3 physical lines. The number of physical lines used to comprise a logical line (1 to 3) is always the minimum required to hold the data for that logical line.

The text screen "scrolls" upward whenever a text line at the bottom row of the screen extends past the right margin, or a text line at the bottom row is terminated by an EOL. Scrolling removes the entire logical line that starts at the top of the screen, and then moves all subsequent lines upward to fill in the void. The cursor also moves upward, if the logical line deleted exceeds one physical line.

All data going to or coming from the text screen is represented in 8-bit ATASCII code as shown in Appendix E.

TEXT MODES 1 AND 2

In text modes 1 and 2 the screen comprises either 24 lines of 20 characters (mode 1), or 12 lines of 20 characters (mode 2). The left and right margins are of no consequence in these modes and there is no visible cursor. There are no logical lines associated with the data and in all regards these modes are treated as graphics modes by the handler.

Data going to or coming from the screen is in the form shown below:

```
7               0
+-+-+-+-+-+-+-+-+
| C |     D     |
+-+-+-+-+-+-+-+-+
```

Where:C is the color/character-set select field

C Value	Color (default)	Color Register (see Appendix H)	Character Set CHBAS=$E0	Character Set CHBAS=$E2
0	green	(PF1)	! - ?	[HEART] [ARROW]
1	gold	(PF0)	! - ?	[HEART] [ARROW]
2	gold	(PF0)	@ - _	[DIAMOND][TRIANGLE]
3	green	(PF1)	@ - _	[DIAMOND][TRIANGLE]
4	red	(PF3)	! - ?	[HEART] [ARROW]
5	blue	(PF2)	! - ?	[HEART] [ARROW]
6	blue	(PF2)	@ - _	[DIAMOND][TRIANGLE]
7	red	(PF3)	@ - _	[DIAMOND][TRIANGLE]

D is a 5-bit truncated ATASCII code that selects the specific character within the set selected by the C field. See Appendix E for the graphics representations of the characters.

Data base variable CHBAS [02F4] allows for the selection of either of two data sets. The default value of $E0 provides the capital letters, numbers and punctuation characters; the alternate value of $E2 provides lowercase letters and the special character graphics set.

Figure 5-5 Text Modes 1 and 2 Data Form

GRAPHICS MODES (Modes 3 Through 11)

The screen has varying physical characteristics for each of the graphics modes as shown in Appendix H. Depending upon the mode, a 1 to 16 color selection is available for each pixel and the screen size varies from 20 by 12 (lowest resolution) to 320 by 192 (highest resolution) pixels.

There is no visible cursor for the graphics mode output.

Data going to or coming from the graphics screen is represented as 1 to 8-bit codes as shown in Appendix H and in the GET/PUT diagrams following.

SPLIT-SCREEN CONFIGURATIONS

In split-screen configurations, the bottom of the screen is reserved for four lines of mode 0 text. The text region is controlled by the Screen Editor, and the graphics region is controlled by the Display handler. Two cursors are maintained in this configuration so that the screen segments can be managed independently.

To operate in split-screen mode, the Screen Editor must first be opened and then the Display Handler must be opened using a separate IOCB (with the split-screen option bit set in AUX1).

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions (described earlier in this section) are detailed below:

OPEN

The device name is S, and the handler ignores any device number an filename specification, if included.

The handler supports the following options:

```
                 7             0
                +-+-+-+-+-+-+-+-+
       AUX1     |   |C|S|W|R|   |
                +-+-+-+-+-+-+-+-+
```

Where: C = 1 indicates to inhibit screen clear on OPEN.
S = 1 indicates to set up a split-screen configuration (for modes 1 through 8 only).
R and W are the direction bits (read and write).

```
                 7             0
                +-+-+-+-+-+-+-+-+
       AUX2     |       | mode  |
                +-+-+-+-+-+-+-+-+
```

Where: mode is the screen mode (0 through 11).

Note: If the screen mode selected is 0, then the AUX1 C and S options are assumed to be 0.

You share memory utilization with the Display Handler information. Sharing is necessary because the Display Handler dynamically allocates high address memory for use in generating the screen display, and because different amounts of memory are needed for the different screen modes. Prior to initiating an OPEN command the variable APPMHI [000E] should contain the highest address of RAM you need. The Screen handler will open the screen only if no RAM is needed at or below that address.

Upon return from a screen OPEN, the variable MEMTOP [02E5] will contain the address of the last free byte at the end of RAM memory prior to the screen-required memory.

As a result of every OPEN command, the following screen variables are altered:

The text cursor is enabled (CRSINH = 0). The tabs are set to the default settings (2 and 39). The color registers are set to the default values (shown in Appendix H).

Tabs are set at positions 7,15,23,31,39, 47,55,63,71,79,87,95,103,111,119.

CLOSE

No special handler actions.

GET CHARACTERS and GET RECORD

Returns data in the following screen mode dependent forms, where each byte contains the data for one cursor position (pixel); there is no facility for having the handler return packed graphics data.

```
 7             0
+-+-+-+-+-+-+-+-+
!    ATASCII    !      Mode 0
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
! C !     D     !      Modes 1,2 -- C = color/data
+-+-+-+-+-+-+-+-+                       set.

                       D = truncated ATASCII.

+-+-+-+-+-+-+-+-+
!   zero    ! D !      Modes 3,5,7 -- D = color.
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
!     zero    !D!      Modes 4,6,8 -- D = color.
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
! zero  !   D   !      Modes 9,10,11 -- D = data.
+-+-+-+-+-+-+-+-+
```

Figure 5-6   Graphics Mode 3-11 GET Data Form

The cursor moves to the next position as each data byte is returned. For mode 0, the cursor will stay within the specified margins; for all other modes, the cursor ignores the margins.

PUT CHARACTERS and PUT RECORD

The handler accepts display data in the following screen mode dependent forms; there is no facility for the handler to receive graphics data in packed form.

```
 7              0
+-+-+-+-+-+-+-+-+
!    ATASCII    !        Mode 0
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
!  C  !   D     !        Modes 1,2 -- C = color/data
+-+-+-+-+-+-+-+-+                           set,
                         D = truncated ATASCII.

+-+-+-+-+-+-+-+-+
!     ?     ! D !        Modes 3,5,7 -- D = color.
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
!      ?      !D!        Modes 4,6,8 -- D = color.
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
!   ?   !   D   !        Modes 9,10,11 -- D = data.
+-+-+-+-+-+-+-+-+
```

Figure 5-7   Graphics Mode 3-11 PUT Data Form

NOTE: For all modes, if the output data byte equals $9B (EOL), that byte will be treated as an EOL character; and if the output data byte equals $7D (CLEAR) that byte will be treated as a screen-clear character.

The cursor moves to the next cursor position as each data byte is written. For mode 0, the cursor will stay within the specified margins; for all other modes, the cursor ignores the margins.

While outputting, the Display Handler monitors the keyboard to detect the pressing of the [CTRL] 1 key combination. When this occurs, the handler loops internally until that key combination is pressed again: This effects a stop/start function that freezes the screen display. Note that there is no ATASCII code associated with either the [CTRL] 1 key combination or the start/stop function. The stop/start function can be controlled only from the keyboard (or by altering database variable CH as discussed in Appendix L, E4).

GET STATUS

No handler action except to set the status to $01.

DRAW

This special command draws a simulated "straight" line from the current cursor position to the location specified in ROWCRS [0054] and COLCRS [0055]. The color of the line is taken from the last character processed by the Display Handler or Screen Editor. To force the color, store the desired value in ATACHR [02FB]. At the completion of the command, the cursor will be at the location specified by ROWCRS and COLCRS.

The value for the command byte for DRAW is $11.

FILL

This special command fills an area of the screen defined by two lines with a specified color. The command is set up the same as in DRAW, but as each point of the line is drawn, the routine scans to the right performing the procedure shown below (in PASCAL notation):

```
WHILE PIXEL [ROW,COL] = 0 DO
   BEGIN
      PIXEL [ROW,COL] := FILDAT;
      COL := COL + 1;
      IF COL > Screen right edge THEN COL := 0
   END;
```

An example of a FILL operation is shown below:

```
                                      + 1
                                     +
+----------------------------------+
 +--------------------------------+
4 +------------------------------+
                                +
                               + 2
```

Where: '-' represents the fill operation.
'+' are the line points, with '+' for the endpoints.

1 -- set cursor and plot point.
2 -- set cursor and DRAW line.
3 -- set cursor and plot point.
4 -- set fill data value, set cursor, and FILL.

FILDAT [02FD] contains the fill data, and ROWCRS and COLCRS contain the cursor coordinates of the line endpoint. The value in ATACHR [02FB] will be used to draw the line; ATACHR always contains the last data read or written, so if the steps above are followed exactly, ATACHR will not have to be modified.

The value for the command byte for FILL is $12.

User-Alterable Data Base Variables

Certain functions of the Display Handler require you to examine and/or alter variables in the OS database. The following describes some of the more commonly used handler variables. (see Appendix L, B1-55, for additional descriptions).

Cursor Position

Two variables maintain the cursor position for the graphics screen or mode 0 text screen. ROWCRS [0054] maintains the display row number; and COLCRS [0055] maintains the display column number. Both numbers range from 0 to the maximum number of rows/columns, - 1. The cursor can be set outside of the defined text margins with no ill effect. You can read and write this region. The home position (0,0) for both text and graphics is the upper left corner of the screen.

ROWCRS is a single byte. COLCRS is maintained at 2-bytes, with the least significant byte being at the lower address.

When you alter these variables, the screen representation of the cursor will not move until the next I/O operation involving the display is performed.

Inhibit/Enable Visible Cursor Display

You can inhibit the display of the text cursor on the screen by setting the variable CRSINH [02F0] to any nonzero value. Subsequent I/O will not generate a visible cursor.

You can enable the display of the text cursor by setting CRSINH to zero. Subsequent I/O will then generate a visible cursor.

Text Margins

The text screen has user-alterable left and right margins. The OS sets these margins to 2 and 39. The variable LMARGN [0052] defines the left margin, and the variable RMARGN [0053] defines the right margin. The leftmost margin value is 0 and the rightmost margin value is 39.

The margin values inclusively define the useable portion of the screen for all operations in that you do not explicitly alter the cursor location variables as described prior to this paragraph.

Color Control

The OS updates hardware color registers using data from the OS data base as part of normal Stage 2 VBLANK processing (see Section 6). Shown below are the data base variable names, the hardware register names, and the function of each register. See Appendix H for the mode dependent uses for the registers.

Data Base	Hardware	Function
COLOR0	COLPF0	PF0 -- Playfield 0.
COLOR1	COLPF1	PF1 -- Playfield 1.
COLOR2	COLPF2	PF2 -- Playfield 2.
COLOR3	COLPF3	PF3 -- Playfield 3.
COLOR4	COLBK	BAK -- Playfield background.
PCOLR0	COLPM0	PM0 -- Player/missile 0.
PCOLR1	COLPM1	PM1 -- Player/missile 1.
PCOLR2	COLPM2	PM2 -- Player/missile 2.
PCOLR3	COLPM3	PM3 -- Player/missile 3.

Theory of Operation

The Display Handler automatically sets up all memory resources required to create and maintain the screen display at OPEN time. The screen generation hardware requires that two distinct data areas exist for graphics modes: 1) a display list and 2) a screen data region. A third data area must exist for text modes. This data area defines the screen representation for each of the text characters. Consult the ATARI Home Computer Hardware Manual for a complete understanding of the material that is to follow.

The simplified block diagram below shows the relationships between the memory and hardware registers used to set up a screen display (without player/missile graphics) by the OS Note that the hardware registers allow for many other possibilities.

```
                      DATA BASE                   HARDWARE
                      VARIABLE                    REGISTER
                                                  (Updated every
                                                   VBLANK)

                     +----------+
                     ¦ MEMTOP   ¦
                     +          +
                     ¦          ¦
                     +---+------+
        +----------------+
        ¦ +----------------------------------------+
        ¦ ¦                                        ¦
    +---+-+--------+ +----------+     +----------+ ¦
    ¦    Display   ¦ ¦ SDLSTL   ¦     ¦  DLISTL  ¦ ¦
    ¦    List      ¦ +          +---->+          +-+
    =              = ¦ SDLSTH   ¦     ¦  DLISTH  ¦
    ¦              ¦ ¦          ¦     ¦          ¦
    +--------------+ +----------+     +----------+
    ¦              ¦
    ¦              ¦  +----------+
    ¦ Screen Data  ¦<-- SAVMSC   ¦
    =              =  +          +
    ¦ Graphics     ¦  ¦          ¦
    ¦  and/or      ¦  +----------+
    ¦   Text       ¦
    +--------------+
End of RAM memory
           +---------------------------------------+
           ¦      +-----------+    +---------+     ¦
           ¦      ¦ CHBAS=E0  ¦--->¦ CHBASE  +-----+
           ¦      +-----------+    +---------+
    +------+-------+
    ¦ Specials and¦ E000
    ¦ Numbers      ¦
    +--------------+
    ¦ Capital      ¦ E100
    ¦ Letters      ¦
    +--------------+
    ¦ Special      ¦ E200
    ¦ Graphics     ¦
    +--------------+
    ¦ Lowercase    ¦ E300
    ¦ Letters      ¦
    +--------------+
```

```
+----------+   +----------+
| COLOR 0 |    | COLPF0   |
=         =-->| COLPF1   |
| COLOR 1 |    | COLPF2   |
| COLOR 2 |    | COLPF3   |
| COLOR 3 |    | COLBK    |
| COLOR 4 |    +----------+
+----------+
```

Figure 5-8 Screen Display Block Diagram

The following relationships are present in the preceding diagram:

1. Data base variables SDLSTL/SDLSTH contain the address of the current display list. This address is stored in the hardware display list address registers DLISTL and DLISTH as part of the VBLANK process.

2. The display list itself defines the characteristics of the screen to be displayed and points to the memory containing the data to be displayed.

3. Data base variable CHBAS contains the MSB of the base address of the character representations for the character data (text modes only).

   The default value for this variable is $E0. This variable declares that the character representations start at memory address E000 (the character set provided by the OS in ROM). Each character is defined as an 8X8 bit matrix, requiring 8 bytes per character. 1024 bytes are required to define the largest set, since a character code contains up to 7 significant bits (set of 128 characters). The OS ROM contains the default set in the region from E000 to E3FF.

   All character codes are converted by the handler from ATASCII to an internal code (and vice versa), as shown below:

ATASCII CODE	INTERNAL CODE
00-1F	40-5F
20-3F	00-1F
40-5F	20-3F
60-7F	60-7F
80-9F	C0-DF
A0-BF	80-9F
C0-DF	A0-BF
E0-FF	E0-FF

The character set in ROM is ordered by internal code order. Three considerations differentiate the internal code from the external (ATASCII) code:

ATASCII codes for all but the special graphics characters were to be similar to ASCII. The alphabetic, numeric, and punctuation character codes are identical to ASCII.

In text modes 1 and 2 it was desired that one character subset include capital letters, numbers, and punctuation and the other character subset include lowercase letters and special graphics characters.

The codes for the capital and lowercase letters were to be identical in text modes 1 and 2.

Database variables COLOR0 through COLOR4 contain the current color register assignments. Hardware color registers receive these values as part of the stage 1 VBLANK process, thus providing synchronized color changes (see Appendix H).

Database variable SAVMSC points to the lowest memory address of the screen data region. It corresponds to the data displayed at the upper left corner of the display.

When the Display Handler receives an open command, it first determines the screen mode from the OPEN IOCB. Then it allocates memory from the end of RAM downward (as specified by data base variable RAMTOP), first for the screen data and then for the display list. The screen data region is cleared and the display list is created if sufficient memory is available. The display list address is stored to the database.

Screen Editor (E:)

The Screen Editor is a read/write handler that uses the Keyboard Handler and the Display Handler to provide "line-at-a-time" input with interactive editing functions, as well as formatted output.

The Screen Editor supports the following CIO functions:

- OPEN
- CLOSE
- GET CHARACTERS
- GET RECORD
- PUT CHARACTERS
- PUT RECORD
- GET STATUS (null function)

See Keyboard Handler and Display Handler Sections for a discussion of Screen Editor error statuses.

The Screen Editor is one of the resident handlers, and therefore has a set of device vectors starting at location E400.

The Screen Editor is a program that reads key data from the Keyboard Handler and sends each character to the Display Handler for immediate display. The Screen Editor also accepts data from you to send to the Display Handler, and reads data from the Display Handler (not the Keyboard Handler) for you. In fact, the Keyboard Handler, Display Handler, and the Screen Editor are all contained in one monolithic hunk of code.

Most of the behaviors already defined for the Keyboard Handler and the Display Handler apply as well to the Screen Editor: The discussions in this Section will be limited to deviations from those behaviors, or to additional features that are part of the Screen Editor only. The Screen Editor deals only with text data (screen mode 0). This Section also explains the split-screen configuration feature.

The Screen Editor uses the Display Handler to read data from graphics and text screens on demand. You use the Screen Editor to determine when the program will read Screen data, and where upon the screen the data will be read from. You first locates the cursor on the screen to determine the screen area to be read; you then press the [RETURN] key to determine when the program will begin to read the data indicated.

When the [RETURN] key is pressed, the entire logical line within that the cursor resides is then made available to the calling program: Trailing blanks in a logical line are never returned as data, however. After all of the data in the line has been sent to the caller (this can entail multiple READ CHARACTERS functions if desired), an EOL character is returned and the cursor is positioned to the beginning of the logical line following the one just read.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN

The device name is E, and the Screen Editor ignores any device number and filename specification, if included.

The Screen Editor supports the following option:

```
              7               0
             +-+-+-+-+-+-+-+-+
      AUX1   |       |W|R|   |F|
             +-+-+-+-+-+-+-+-+
```

Where: R and W are the direction bits (read and write).
F = 1 indicates that a "forced read" is desired (see GET CHARACTER and GET RECORD for more information).

CLOSE

No special handler actions.

GET CHARACTER and GET RECORD

Normally the Screen Editor will return data only when you press the [RETURN] key at the keyboard. However, the "forced read" OPEN option allows you to read text data without intervention. When you command a READ operation, the Screen Editor will return data from the start of the logical line in which the text cursor is located, and then move the cursor to the beginning of the following logical line. A read of the last logical line on the screen will cause the screen data to scroll.

A special case occurs when characters are output without a terminating EOL, and then additional characters are appended to that logical line from the keyboard. When the [RETURN] key is pressed, only the keyboard entered characters are sent to the caller, unless the cursor has been moved out of and then back into the logical line, in that case all of the logical line will be sent.

PUT CHARACTER and PUT RECORD

The Handler accepts ATASCII characters as one character per byte. Sixteen of the 256 ATASCII characters are control codes; the EOL code has universal meaning, but most of the other control codes have special meaning only to a display or print device. The Screen Editor processing of the ATASCII control codes is explained below:

CLEAR ($7D) -- The Screen Editor clears the current display of all data and the cursor is placed at the home position (upper left corner of the screen).

CURSOR UP ($1C) -- The cursor moves up by one physical line. The cursor will wrap from the top line of the display to the bottom line.

CURSOR DOWN ($1D) -- The cursor moves down by one physical line. The cursor will wrap from the bottom line of the display to the top line.

CURSOR LEFT ($1E) -- The cursor moves left by one column. The cursor will wrap from the left margin of a line to the right margin of the same line.

CURSOR RIGHT ($1F) -- The cursor moves right by one column. The cursor will wrap from the right margin of a line to the left margin of the same line.

BACKSPACE ($7E) -- The cursor moves left by one column (but never past the beginning of a logical line), and the character at that new position is changed to a blank ($20).

SET TAB ($9F) -- The Screen Editor establishes a tab point at the logical line position at that the cursor is residing. The logical line tab position is not synonymous with the physical line column position since the logical line can be up to 3 physical lines in length. For example, tabs can be set at the 15th, 30th, 45th, 60th and 75th character positions of a logical line as shown below:

```
0 2      9         19        29        39  Screen column #.
--L------+---------+---------+---------R   L/R = margins.

xx--------------T--------------T--------   A logical line.
xx------T--------------T--------------T-   x = inaccesible
xx--------------------------------------       columns.
```

Note the effect of the left margin in defining the limits of the logical line.

The Handler default tab settings are shown below:

```
0 2      9         19        29        39  Screen column #.
--L------+---------+---------+---------R   L/R = margins.

xxT----T-------T-------T-------T-------T   A logical line.
xx-----T-------T-------T-------T-------T   x = inaccesible
xx-----T-------T-------T-------T-------T       columns.
```

CLEAR TAB ($9E) -- The Screen Editor clears the current cursor position within the logical line from being a tab point. There is no "clear all tab points" facility provided by the Handler.

TAB ($7F) -- The cursor moves to the next tab point in the current logical line, or to the beginning of the next line if no tab point is found. This function will not increase the logical line length to accommodate a tab point outside the current length (e.g. the logical line length is 38 characters and there is a tab point at position 50).

INSERT LINE ($9D) -- All physical lines at and below the physical line in that the cursor resides, are moved down by one physical line. The last logical line on the display can be truncated as a result. The blank physical line at the insert point becomes the beginning of a new logical line. A logical line can be split into two logical lines by this process, the last half of the original logical line being concatenated with the blank physical line formed at the insert point.

DELETE LINE ($9C) -- The logical line in that the cursor resides is deleted and all data below that line is moved upward to fill the void. Empty logical lines are created at the bottom of the display.

INSERT CHARACTER ($FF) -- All physical characters at and behind the cursor position on a logical line are moved one position to the right. The character at the cursor position is set to blank. The last character of the logical line will be lost when the logical line is full and a character is inserted. The number of physical lines comprising a logical line can increase as a result of this function.

DELETE CHARACTER ($FE) -- The character on which the cursor resides is removed, and the remainder of the logical line to the right of the deleted character is moved to the left by one position. The number of physical lines composing a logical line can decrease as a result of this function.

ESCAPE ($1B) -- The next non-EOL character following this code is displayed as data, even if it would normally be treated as a control code. The sequence [ESC][ESC] will cause the second [ESC] character to be displayed.

BELL ($FD) -- An audible tone is generated; the display is not modified.

END OF LINE ($9B) -- In addition to its record termination function, the EOL causes the cursor to advance to the beginning of the next logical line. When the cursor reaches the bottom line of the screen, the receipt of an EOL will cause the screen data to scroll upward by one logical line.

GET STATUS

The Handler takes no action other than to set the status to $01.

User-Alterable Data Base Variables

Also see the Display Handler data base variable discussion.

Cursor Position

When in a split-screen configuration, ROWCRS and COLCRS are associated with the graphics portion of the display and two other variables, TXTROW [0290] and TXTCOL [0291], are associated with the text window. TXTROW is a single byte, and TXTCOL is 2-bytes with the least significant byte being at the lower address. Note that the most significant byte of TXTCOL should always be zero.

The home position (0,0) for the text window is the upper left corner of the window.

Enable/Inhibit of Control Codes in Text

Normally all text mode control codes are operated upon as received, but sometimes it is desirable to have the control codes displayed as if they were data characters. This is done by setting the variable DSPFLG [02FE] to any nonzero value before outputting the data containing control codes. Setting DSPFLG to zero restores normal processing of text control codes.

Cassette Handler (C:)

The Cassette device is a read or write device with a Handler that supports the following CIO functions:

OPEN
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS (null function)

The Cassette Handler can produce the following error statuses:

$80 -- [BREAK] key abort.
$84 -- Invalid AUX1 byte on OPEN.
$88 -- end-of-file.
$8A-90 -- SIO error set (see Appendix C).

The Cassette Handler is one of the resident handlers, and therefore has a set of device vectors starting at location E440.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN

The device name is C, and the Handler ignores any device number and filename specification, if included.

The Handler supports the following option:

```
              7             0
             +-+-+-+-+-+-+-+-+
       AUX2  |C|             |
             +-+-+-+-+-+-+-+-+
```

Where: C = 1 indicates that the cassette is to be read/written without stop/start between records (continuous mode).

Opening the cassette for input generates a single audible tone, as a prompt for you to verify that the cassette player is set up for reading (power on; Serial Bus cable connected; tape cued to start of file; and PLAY button depressed). When the cassette is ready, you can press any keyboard key (except [BREAK]) to initiate tape reading.

Opening the cassette for output generates two closely spaced audible tones, as a prompt for you to verify that the cassette player is set up for writing (as above, plus RECORD button depressed). When the cassette is ready, you can press any keyboard key (except [BREAK]) to begin tape writing. There is no way for the computer to verify that the RECORD or PLAY button is depressed. It is possible for the file not to be written, with no immediate indication of this fact.

There is a potential problem with the cassette in that when the cassette is opened for writing, the motor keeps running until the first record (128 data bytes) is written. If 128 data bytes are written or the cassette is closed within about 30 seconds of the OPEN, and no other serial bus I/O is performed, then there is no problem. However, if those conditions are not met, some noise will be written to the tape prior to the first record and an error will occur when that tape file is read later. If lengthy delays are anticipated between the time the cassette file is opened and the time that the first cassette record (128 data bytes) is written, then a dummy record should be written as part of the file; typically 128 bytes of some innocuous data would be written, such as all zeros, all $FFs, or all blanks ($20).

The system sometimes emits whistling noises after cassette I/O has occurred. The sound can be eliminated by storing $03 to SKCTL [D20F], thus bring POKEY out of the two-tone (FSK) mode.

CLOSE

The CLOSE of a tape read stops the cassette motor.

The CLOSE of a tape write does the following:

    Writes any remaining user data in the buffer to tape.
    Writes an end-of-file record.
    Stops the cassette motor.

GET CHARACTERS and GET RECORD

The Handler returns data in the following format:

```
 7             0
+-+-+-+-+-+-+-+-+
!   data byte   !
+-+-+-+-+-+-+-+-+
```

PUT CHARACTERS and PUT RECORD

The Handler accepts data in the following format:

```
 7             0
+-+-+-+-+-+-+-+-+
!   data byte   !
+-+-+-+-+-+-+-+-+
```

The Handler attaches no significance to the data bytes written, a value of $9B (EOL) causes no special action.

GET STATUS

The Handler does no more than set the status to $01.

Theory of Operation

The Cassette Handler writes and reads all data in fixed-length records of the format shown below:

```
+-+-+-+-+-+-+-+-+
!0 1 0 1 0 1 0 1!      Speed measurement bytes.
+-+-+-+-+-+-+-+-+
!0 1 0 1 0 1 0 1!
+-+-+-+-+-+-+-+-+
! control byte  !
+-+-+-+-+-+-+-+-+
!      128      !
=      data     =
!      bytes    !
+-+-+-+-+-+-+-+-+
!   checksum    !      (Managed by SIO, not the
+-+-+-+-+-+-+-+-+       Handler.)
```

Figure 5-9   Cassette Handler Record Format

The control byte contains one of three values:

- $FC indicates the record is a full data record (128 bytes).

- $FA indicates the record is a partially full data record; you supplied fewer than 128 bytes to the record. This case can occur only in the record prior to the end-of-file. The number of user-supplied data bytes in the record is contained in the byte prior to the checksum.

- $FE indicates the record is an End-of file record; the data portion is all zeroes for an end-of-file record.

The SIO routine generates and checks the checksum. It is part of the tape record, but it is not contained in the Handler's record buffer CASBUF [03FD].

The processing of the speed-measurement bytes during cassette reading is discussed in Appendix L, D1-D7.

File Structure

The Cassette Handler writes a file to the cassette device with a file structure that is totally imposed by the Handler (soft format). A file consists of the following three elements:

- A 20-second leader of mark tone.

- Any number of data-record frames.

- An end-of-file frame.

The cassette-data record frames are formatted as shown below:

```
frame = pre-record write tone (PRWT),
        + data record,
        + post record gap (PRG)
```

The nondata portions of a frame have characteristics that are dependent upon the write OPEN mode, i.e. continuous or start/stop.

```
Stop/start PRWT = 3 seconds of mark tone.
Continuous PRWT = .25 second of mark tone.

Stop/start PRG = up to 1 second of unknown tones.
Continuous PRG = from 0 to n seconds of unknown tones, where
                 n is dependent upon your program timing.
```

The inter-record gap (IRG) between any two records consists of the PRG of the first record followed by the PRWT of the second record.

Printer Handler (P:)

The Printer device is a write-only device with a Handler that supports the following CIO functions:

- OPEN
- CLOSE
- PUT CHARACTERS
- PUT RECORD
- GET STATUS

The Printer Handler can produce the following error statuses:

$8A-90 -- SIO error set (see Appendix C).

The Printer Handler is one of the resident handlers, and therefore has a set of device vectors starting at location E430.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN

The device name is P. The Handler ignores any device number and filename specification, if included.

CLOSE

The Handler writes any data remaining in its buffer to the printer device, with trailing blanks to fill out the line.

PUT CHARACTERS and PUT RECORD

The Handler accepts print data in the following format:

```
 7             0
+-+-+-+-+-+-+-+-+
|    ATASCII    |
+-+-+-+-+-+-+-+-+
```

The only ATASCII control code of any significance to the Handler is the EOL character. The printer device ignores bit 7 of every data byte and prints a sub set of the remaining 128 codes. (see Appendix G for the printer character set).

The Handler supports the following print option:

```
                7             0
                +-+-+-+-+-+-+-+-+
       AUX2     !  print mode   !
                +-+-+-+-+-+-+-+-+
```

Where: $4E (N) selects normal printing (40 characters per line).
$53 (S) selects sideways printing (29 characters per line).
$57 (W) selects wide printing (not supported by printer device).

Any other value (including 00) is treated as a normal (N) print select, without producing an error status.

GET STATUS

The Handler obtains a 4-byte status from the printer controller and puts it in system location DVSTAT [02EA]. The format of the status bytes is shown below:

```
+-+-+-+-+-+-+-+-+
! command stat. !      DVSTAT + 0
+-+-+-+-+-+-+-+-+
! AUX2 of prev. !             + 1
+-+-+-+-+-+-+-+-+
!    timeout    !             + 2
+-+-+-+-+-+-+-+-+
!    (unused)   !             + 3
+-+-+-+-+-+-+-+-+
```

The command status contains the following status bits and condition indications:

bit 0: an invalid command frame was received.
bit 1: an invalid data frame was received.
bit 7: an intelligent controller (normally = 0).

The next byte contains the AUX2 value from the previous operation.

The timeout byte contains a controller provided maximum timeout value (in seconds).

Theory of Operation

The ATARI 820[TM] 40-Column Printer is a line-at-a-time printer rather than a character-at-a-time printer, so your data must be buffered by the Handler and sent to the device in records corresponding to one print line (40 characters for normal, 29 characters for sideways).

The printer device does not attach any significance to the EOL character, so the Handler does the appropriate blank fill whenever it sees an EOL.

Disk File Manager (D:)

The OS supports four unique File Management Subsystems at the time of this writing. Version IA is the original version. Version IB is a slightly modified version of IA and is the one described in this document. Most of this discussion applies as well to Version II, that handles a double-density diskette (720 256-byte sectors) in addition to the single-density diskette (720 128-byte sectors). Version III has all new file/directory/map structures and can possibly contain changes to your interface as well.

The File Management Subsystem includes a disk-bootable (RAM-resident) Disk File Manager (DFM) that maintains a collection of named files on diskettes. Up to 4 disk drives (D1: through D4:) can be accessed, and up to 64 files per diskette can be accessed. The system diskettes supplied by ATARI allow a single disk drive (D1) and up to 3 OPEN files, but you can alter these numbers as described later in this section.

The Disk File Manager supports the following CIO functions:

OPEN FILE
OPEN DIRECTORY
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS

NOTE
POINT
LOCK
UNLOCK
DELETE
RENAME
FORMAT

The Disk File Manager can produce the following error statuses:

- $03 -- Last data from file (EOF on next read).
- $88 -- end-of-file.
- $8A-90 -- SIO error set (see Appendix C).
- $A0 -- Drive number specification error.
- $A1 -- No sector buffer available (too many open files).
- $A2 -- Disk full.
- $A3 -- Fatal I/O error in directory or bitmap.
- $A4 -- Internal file # mismatch (structural problem).
- $A5 -- File name specification error.
- $A6 -- Point information in error.
- $A7 -- File locked to this operation.
- $A8 -- Special command invalid.
- $A9 -- Directory full (64 files).
- $AA -- File not found.
- $AB -- Point invalid (file not OPENed for update).

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN FILE

The device name is D. Up to four disk drives can be accessed (D1 through D4). The disk filename can be from 1 to 8 characters in length with an optional 1- to 3-character extension.

The OPEN FILE command supports the following options:

```
                7             0
               +-+-+-+-+-+-+-+-+
      AUX1     |       |W|R| |A|
               +-+-+-+-+-+-+-+-+
```

Where: W and R are the direction bits.
WR = 00 is invalid
01 indicates OPEN for read only.
10 indicates OPEN for write only.
11 indicates OPEN for read/write (update).

A = 1 indicates appended output when W = 1.

You may use these following valid AUX1 options:

OPEN Input (AUX1 = $04)

The indicated file is opened for input. Any wild-card characters are used to search for the first match. If the file is not found, an error status is returned, and no file will be opened.

OPEN Output (AUX1 = $08)

The indicated file is opened for output starting with the first byte of the file, if the file is not locked. Any wild-card characters are used to search for the first match. If the file already exists, the existing file will be deleted before opening the named file as a new file. If the file does not already exist, it will be created.

A file opened for output will not appear in the directory until it has been closed. If an output file is not properly closed, some or all of the sectors that were acquired for it can be lost until the disk is reformatted.

A file that is opened for output can not be opened concurrently for any other access.

OPEN Append (AUX1 = $09)

The indicated file is opened for output starting with the byte after the last byte of the existing file (that must already exist), if the file is not locked. Any wild-card characters are used to search for the first match.

If a file opened for append is not properly closed, the appended data will be lost. The existing file will remain unmodified and some or all of the sectors that were acquired for the appended portion can be lost until the diskette is reformatted.

OPEN Update (AUX1 = $0C)

The indicated file (that must already exist) will be opened for update provided it is not locked. Any wild-card characters are used to search for the first match.

The GET, PUT, NOTE and POINT operations are all valid, and can be intermixed as desired.

If a file opened for update is not properly closed, a sector's worth of information can be lost to the file. A file opened for update can not be extended.

Device/Filename Specification

The Handler expects to find a device/filename specification of the following form:

```
D[<number>]:<filename><EOL>
```

where:

```
<number> ::= 1|2|3|4
<filename> ::= [<primary>][.[<extension>]]<terminator>
<primary> ::= an uppercase alpha character followed by 0 to 7
              alphanumeric characters. If the primary name is
              less than 8 characters, it will be padded with
              blanks; if it is greater than 8 characters, the
              extra characters will be ignored.

<extension> ::= Zero to 3 alphanumeric characters. If the
              extension name is missing or less than 3
              characters, it will be padded with blanks; if
              it is greater than 3 characters, the extra
              characters will be ignored.

<terminator> ::= <EOL>|<blank>
```

Figure 5-10   Device/Filename Syntax

The following are all valid device/filenames for the diskette:

```
D1:GAME.SRC
D:MANUAL6
D:.WHY
D3:FILE.
D4:BRIDGE.002
```

Filename Wildcarding

The filename specification can be further generalized to include the use of the "wild-card" characters * and ?. These wildcard characters allow portions of the primary and/or extension to be abbreviated as follows:

The ? character in the specification allows any filename character at that position to produce a "match." For example, WH? will match files named WHO, WHY, WH4, etc., but not a file named WHAT.

The * character causes the remainder of the primary or extension field in that it is used to be effectively padded with ? characters. For example, WH* will match WHO, WHEN, WHATEVER, etc.

Some valid uses of wild-card specifications are shown below:

*.SRC	Files having an extension of SRC.
BASIC.*	Files named BASIC with any extension.
.	All files.
H*.?	Files beginning with H and having a 0 or 1 character extension.

If wildcarding is used with an OPEN FILE command, the first file found (if any) that meets the specification will be the one (and only one) opened.

OPEN DIRECTORY

The OPEN DIRECTORY command allows you read directory information for the selected filename(s), using normal GET CHARACTERS or GET RECORD commands. The information read will be formatted as ATASCII records, suitable for printing, as shown below. Wildcarding can be used to obtain information for multiple files or the entire diskette.

The OPEN DIRECTORY command uses the same CIO parameters as a standard OPEN FILE command:

COMMAND BYTE = $03

BUFFER ADDRESS = pointer to device/filename specification.

AUX1 = $06

After the directory is opened, a record will be returned to the caller for each file that matches the OPEN specification. The record, that contains only ATASCII characters, is formatted as shown below:

```
                     1
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|s|b| primary name   | ext |b|count|e|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Where: s = * or ' ', with * indicating file locked.
       b = blank.
       primary name = left-justified name with blank fill.
       ext = left-justified extension with blank fill.
       b = blank.
       count = number of sectors comprising the file.
       e = EOL ($9B).
```

After the last filename match record is returned, an additional record is returned. This record indicates the number of unused sectors available on the diskette. The format for this record is shown below:

```
                    1
 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!count!  F R E E   S E C T O R S!e!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
Where: count = the number of unused sectors on the diskette.
       e = EOL ($9B).
```

The EOF statuses ($03 and $88) are returned as in a normal data file when the last directory record is read.

The opening of another diskette file while the directory read is open will cause subsequent directory reads to malfunction, so care must be taken to avoid this situation.

CLOSE

Upon closing a file read, the Handler releases all internal resources being used to support that file.

Upon closing a file write, the Handler:

o writes any residual data from its file buffer for that file to the diskette.

o updates the directory and allocation map for the associated diskette.

o releases all internal resources being utilized to support that file

GET CHARACTERS and GET RECORD

Characters are read from the diskette and passed to CIO as a raw data stream. None of the ATASCII control characters have any special significance. A status of $88 is returned if an attempt is made to read past the last byte of a file.

PUT CHARACTERS and PUT RECORD

Characters are obtained from CIO and written to the diskette as a raw data stream. None of the ATASCII control characters have any special significance.

GET STATUS

The indicated file is checked and one of the following status byte values is returned in ICSTA and register Y:

$01 -- File found and unlocked.
$A7 -- File locked.
$AA -- File not found.

Special CIO Functions

The DFM supports a number of SPECIAL commands, that are device specific. These are explained in the paragraphs that follow:

NOTE (COMMAND BYTE = $25)

This command returns to the caller the exact diskette location of the next byte to be read or written, in the variables shown below:

ICAX3 = LSB of the diskette sector number.
ICAX4 = MSB of the diskette sector number.
ICAX5 = relative sector displacement to byte (0-124).

POINT (COMMAND BYTE = $26)

This command allows you to specify the exact diskette location of the next byte to be read or written. In order to use this commmand, the file must have been opened with the "update" option.

ICAX3 = LSB of the diskette sector number.
ICAX4 = MSB of the diskette sector number.
ICAX5 = relative sector displacement to byte (0-124).

LOCK

This command allows you to prevent write access to any number of named files. Locked files can not be deleted, renamed, nor opened for output unless they are first unlocked. Locking a file that is already locked is a valid operation. The Handler expects a device/filename specification; then all occurrences of the filename specified will be locked, using the wild-card rules.

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $23

BUFFER ADDRESS = pointer to device/filename specification.

After a LOCK operation, the following IOCB parameter will have been altered:

STATUS = result of LOCK operation; see Appendix B for a list of possible status codes.

UNLOCK

This command allows you to remove the lock status of any number of named files. Unlocking a file that is not locked is a valid operation. The Handler expects a device/filename specification; then all occurrences of the filename specified will be unlocked, using the wild-card rules.

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $24

BUFFER ADDRESS = pointer to device/filename specification.

After an UNLOCK operation, the following IOCB parameter will have been altered:

STATUS = result of UNLOCK operation; see Appendix B for a list of possible status codes.

DELETE

This command allows you to delete any number of unlocked named files from the directory of the selected diskette and to deallocate the diskette space used by the files involved. The Handler expects a device/filename specification; then all occurences of the filename specified will be deleted, using the wild-card rules.

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $21

BUFFER ADDRESS = pointer to device/filename specification.

After a DELETE operation, the following IOCB parameter will have been altered:

STATUS = result of DELETE operation; see Appendix B for a list of possible status codes.

RENAME

This command allows you to change the filenames of any number of unlocked files on a single diskette. The Handler expects to find a device/filename specification that follows:

<device spec>:<filename spec>,<filename spec><EOL>

All occurrences of the first filename will be replaced with the second filename, using the wild-card rules. No protection is provided against forming duplicate names. Once formed, duplicate names cannot be separately renamed or deleted; however, an OPEN FILE command will always select the first file found that matches the filename specification, so that file will always be accessible. The RENAME command does not alter the content of the files involved, merely the name in the directory.

Examples of some valid RENAME name specifications are shown below:

```
D1:*.SRC,*.TXT
D:TEMP,FDATA
D2:F*,F*.OLD
```

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $20

BUFFER ADDRESS = pointer to device/filename specification.

After a RENAME operation, the following IOCB parameter will have been altered:

STATUS = result of RENAME operation; see Appendix B for a list of possible status codes.

FORMAT

Soft-sector diskettes must be formatted before they can store data. The FORMAT command allows you to physically format a diskette. The physical formatting process writes a new copy of every sector on the soft-sectored diskette, with the data portion of each sector containing all zeros. The FORMAT process creates an "empty" non system diskette. When the formatting process is complete, the FMS creates an initial Volume Table of Contents (VTOC) and an initial File Directory. The boot sector (#1) is permanently reserved as part of this process.

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $FE

BUFFER ADDRESS = pointer to device specification.

After a FORMAT operation, the following IOCB parameter will have been altered:

STATUS = result of FORMAT operation; see Appendix B for a list of possible status codes.

To create a system diskette, a copy of the boot file must then be written to sectors #2-n. This is accomplished by writing the file named DOS.SYS. This is a name that is recognized by the FMS even though it is not in the directory initially.

Theory of Operation

The resident OS initiates the disk-boot process (see Section 10). The OS reads diskette sector #1 to memory and then transfers control to the "boot continuation address" (boot address + 6). The boot-continuation program contained in sector #1 then continues to load the remainder of the File Management Subsystem to memory using additional information contained in sector #1. The File Management Subsystem loaded, will contain a Disk File Manager ,and optionally, a Disk Utilities (DOS) package.

When the boot process is complete, the Disk File Manager will allocate additional RAM for the creation of sector buffers. Sector buffers are allocated based upon information in the boot record as shown below:

Byte 9 = maximum number of open files; one buffer per (the maximum value is 8).

Byte 10 = drive select bits; one buffer per (1-4 only).

The Disk File Manager will then insert the name D and the Handler vector table address in the device table.

NOTE: There is a discrepancy between the Disk File Manager's numbering of diskette sectors (0-719) and the disk controller's numbering of diskette sectors (1-720); as a result, only sectors 1- 719 are used by the Disk File Manager.

The Disk File Manager uses the Disk Handler to perform all diskette reads and writes; the DFM's function is to support and maintain the directory/file/bitmap structures as described in the following pages:

FMS Diskette Utilization

The map below shows the diskette sector utilization for a standard 720 sector diskette.

```
+----------------+
|  BOOT record   |     Sector 1
+----------------+
|    FMS BOOT    |     Sector 2    -+
=      file      =                  |
|    DOS.SYS     |     Sector n     +-- Note 1
+----------------+                  |
|      User      |     Sector n+1  -+
=      File      =
|      Area      |     Sector 359 ($167)
+----------------+
|  VTOC(note 2)  |     Sector 360 ($168)
+----------------+
|      File      |     Sector 361 ($169)
=   Directory    =
|                |     Sector 368 ($170)
+----------------+
|      User      |
=      File      =
|      Area      |     Sector 719 ($2CF)
+----------------+
|     unused     |     Sector 720 ($2D0)
+----------------+
```

Figure 5-11   File Management Subsystem Diskette Sector Utilization Map

NOTE 1 - If the diskette is not a system diskette, then your File Area starts at sector 2 and no space is reserved for the FMS BOOT file. However, "DOS" (DOS.SYS and DUP.SYS) may still be written to a diskette that has already used sectors "2-N."

NOTE 2 -- VTOC stands for Volume Table of Contents.

FMS Boot Record Format

The FMS BOOT record (sector #1) is a special case of diskette-booted software (see Section 10). The format for the FMS BOOT record is shown below:

```
+-----------------+
¦ boot flag = 0   ¦     Byte 0
+-----------------+
¦ # sectors = 1   ¦          1
+-----------------+
¦  boot address   ¦          2
+                 +
¦     = 0700      ¦
+-----------------+
¦  init address   ¦          4
+                 +
¦                 ¦
+-----------------+
¦    JMP = $4B    ¦          6
+                 +
¦    boot read    ¦
+  continuation   +
¦     address     ¦
+-----------------+ ---+
¦ max files = 3   ¦    ¦     9  Note 1
+-----------------+    ¦
¦ drive bits = 1  ¦    ¦    10  Note 2
+-----------------+    ¦
¦ alloc dirc = 0  ¦    ¦    11  Note 3
+-----------------+    ¦
¦ boot image end  ¦    ¦
+                 +    ¦               FMS
¦   address + 1   ¦    +-------------- configuration
+-----------------+    ¦               data
¦ boot flag <> 0  ¦    ¦    14  Note 4
+-----------------+    ¦
¦  sector count   ¦    ¦    15  Note 5
+-----------------+    ¦
¦     DOS.SYS     ¦    ¦
+     starting    +    ¦
¦ sector number   ¦    ¦
+-----------------+ ---+
¦ code for second¦
¦ phase of boot   ¦
```

Figure 5-12 File Management Subsystem Boot Record Format

NOTE 1 - Byte 9 specifies the maximum number of concurrently open files to be supported. This value can range from 1 to 8.

NOTE 2 - Byte 10 specifies the specific disk drive numbers to be supported using a bit encoding scheme as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!       !4!3!2!1! where a 1 indicates a selected drive.
+-+-+-+-+-+-+-+-+
```

NOTE 3 - Byte 11 specifies the buffer allocation direction, this byte should equal 0.

NOTE 4 - Byte 14 must be nonzero for the second phase of the boot process to initiate. This flag indicates that the file DOS.SYS has been written to the diskette.

NOTE 5 - This byte is assigned as being the sector count for the DOS.SYS file. It is actually an unused byte.

Boot Process Memory Map

The diagram below shows how the boot sector (part of file DOS.SYS) and following sectors are loaded to memory as part of the boot process.

```
+----------------+      Memory address 0700
| data from boot |                      |
= sector read by =                      |
| resident OS    |                     077C
+----------------+
| data from rest |                     077D
| of DOS.SYS     |                      |
| read by the    |                      |
= program in the =                      |
| boot sector.   |                      |
|                |                      |
+----------------+                     end of boot
```

Figure 5-13   File Management Subsystem Boot Process Memory Map

Volume Table of Contents

The format for the FMS volume table of contents (VTOC, sector 360) is shown in the diagram below:

```
+----------------+
| directory type |     Byte 0  Note 1
+----------------+
| maximum   (lo) |          1  Note 2
+ sector #       +
| = 02C5    (hi) |
+----------------+
| number of (lo) |          3  Note 3
+ sectors        +
| available (hi) |
+----------------+
|                |
=                =
|                |
+----------------+
|                |          10
= volume bit map =
|                |
+----------------+
|                |
=                =
|                |
+----------------+
```

Figure 5-14 File Management Subsystem Volume Table of Contents

The volume bit map organization location follows:

```
 7              0
+-+-+-+-+-+-+-+-+
| |1 2 3 4 5 6 7|     Byte 10 of VTOC
+-+-+-+-+-+-+-+-+
|8 9 . . . . . .|          11
=               =
|               |          99
+-+-+-+-+-+-+-+-+
```

Figure 5-15 File Management Subsystem Volume Bit Map

At each map bit position, a 0 indicates the corresponding sector is in use and a 1 indicates that the sector is available.

NOTE 1 - The directory type byte must equal 0.

NOTE 2 - The maximum sector number is not used because it is incorrectly set to 709 decimal. The true maximum sector number is actually 719 for the DFM.

NOTE 3 - The number of sectors available is initially set to 709 after a diskette is freshly formatted; this number is adjusted as files are created and deleted to show the number of sectors available. The sectors that are initially reserved are 1 and 360-368.

File Directory Format

The FMS reserves eight sectors (361-368) for a file directory. Each sector containing directory information for up to eight files, thus providing for a maximum of 64 files for any volume. The format of a single 16-byte file entry is shown below:

```
+---------------+
!   flag byte   !   Byte 0
+---------------+
!  sector   (lo)!        1
+  count        +
!           (hi)!
+---------------+
! starting  (lo)!        3
+ sector        +
! number    (hi)!
+---------------+
!            (1)!        5
+               +
!            (2)!
+               +
!            (3)!
+               +
!      file  (4)!
+               +
!      name  (5)!
+               +
!    primary (6)!
+               +
!            (7)!
+               +
!            (8)!
+---------------+
!      file  (1)!        13
+               +
!      name  (2)!
+               +
!  extension (3)!
+---------------+
```

Figure 5-16   File Directory Format

Where the flag byte has the following bits assigned:

bit 7 = 1 if the file has been deleted.
bit 6 = 1 if the file is in use.
bit 5 = 1 if the file is locked.
bit 0 = 1 if OPEN output.

The flag byte can take on the following values:

$00 = entry not yet used (no file).
$40 = entry in use (normal CLOSEd file).
$41 = entry in use (OPEN output file).
$60 = entry in use (locked file).
$80 = entry available (prior file deleted).

Sector count is the number of sectors comprising the file.

FMS File Sector Format

The format of a sector in your data file is shown below:

```
 7             0
+-+-+-+-+-+-+-+-+
!     data      !  +0
=               =
!               !
+-+-+-+-+-+-+-+-+
! file #    !hi !  +125
+-+-+-+-+-+-+   +
!forward pointer!  +126
+-+-+-+-+-+-+-+-+
!S! byte count  !  +127
+-+-+-+-+-+-+-+-+
```

Figure 5-17   File Management Subsystem File Sector Format

The FMS uses the file # to verify file integrity. The file # is a redundant piece of information. The file number field contains the value of the directory position of that file. If a mismatch occurs between the file's directory position, and the file number as contained in each sector, then the DFM will generate the error $A4.

The forward pointer field contains the 10-bit value for the diskette sector number of the next sector of the file. The pointer equals zero for the last sector of a file.

The S bit indicates whether or not the sector is a "short sector" (a sector containing fewer than 125 data bytes). S is equal to 1 when the sector is short.

The byte-count field contains the number of data bytes in the sector.

Non-CIO I/O

Some portions of the I/O subsystem are accessed independently of the Central I/O Utility (CIO); this section discusses those areas.

Resident Device Handler Vectors

All of the OS ROM resident device handlers can be accessed via sets of vectors that are part of the OS ROM. These vectors increase the speed of I/O operations that utilize fixed device assignments, such as output to the Display Handler. For each resident Handler there is a set of vectors ordered as shown below:

```
+---------------+
+-    OPEN    -+      +0
+---------------+
+-   CLOSE    -+      +2
+---------------+
+-  GET BYTE  -+      +4
+---------------+
+-  PUT BYTE  -+      +6
+---------------+
+- GET STATUS -+      +8
+---------------+
+-  SPECIAL   -+      +10
+---------------+
+-    JMP     -+      +12
+-    INIT    -+
+---------------+
+-   SPARE    -+
+-    BYTE    -+
+---------------+
```

Figure 5-18   Resident Device Handler Vectors

See Section 9 for a detailed description of the data interface for each of these Handler entry points.

Each of the vectors contains the address (lo,hi) of the Handler entry point minus 1. A technique similar to the one shown below is required to access the desired routines:

```
VTBASE=$E400                           ; BASE OF VECTOR TABLE.

        .
        LDX     #xx                    ; OFFSET TO DESIRED ROUTINE.
        LDA     data
        JSR     GOVEC                  ; SEND DATA TO ROUTINE.
        .
        LDX     #yy                    ; OFFSET TO DIFFERENT ROUTINE.
        JSR     GOVEC                  ; GET DATA FROM ROUTINE.
        STA     data
        .

GOVEC   TAY                            ; SAVE REGISTER A.
        LDA     VTBASE+1,X             ; ADDRESS MSB TO STACK.
        PHA
        LDA     VTBASE,X               ; ADDRESS LSB TO STACK.
        PHA
        TYA                            ; RESTORE REGISTER A.
        RTS                            ; JUMP TO ROUTINE.
```

The JMP INIT slot in each set of vectors jumps to the Handler initialization entry (not minus 1).

The base address of the vector set for each of the resident handlers is shown below:

Screen Editor (E:)	E400.
Display Handler (S:)	E410.
Keyboard Handler (K:)	E420.
Printer Handler (P:)	E430.
Cassette Handler (C:)	E440.

The resident diskette Handler is not CIO-compatible, so its interface does not use a vector set.

Resident Diskette Handler

The resident Diskette Handler (not to be confused with the Disk File Manager) is responsible for all physical accesses to the diskette. The unit of data transfer for this Handler is a single diskette sector containing 128 data bytes.

Communication between you and the Diskette Handler is effected using the system's Device Control Block (DCB), that is also used for Handler/SIO communication (see Section 9). The DCB is 12 bytes long. Some bytes are user-alterable and some are for use by the Diskette Handler and/or the Serial I/O Utility (SIO). You supply the required DCB parameters and then do a JSR DSKINV [E453].

Each of the DCB bytes will now be described, and the system-equate file name for each will be given.

SERIAL BUS ID -- DDEVIC [0300]

The Diskette Handler sets up this byte to contain the Serial Bus ID for the drive to be accessed. It is not user-alterable.

DEVICE NUMBER -- DUNIT [0301]

You set up this byte to contain the disk drive number to be accessed (1 - 4).

COMMAND BYTE -- DCOMND [0302]

You set up this byte to contain the disk device command to be performed.

STATUS BYTE -- DSTATS [0303]

This byte contains the status of the command upon return to the caller. See Appendix C for a list of the possible status codes.

BUFFER ADDRESS -- DBUFLO [0304] and DBUFHI [0305]

This 2-byte pointer contains the address of the source or destination of the diskette sector data. You need not supply an address for the disk status command. The Disk Handler will obtain the status and insert the address of the status buffer into this field.

DISK TIMEOUT VALUE -- DTIMLO [0306]

The Handler supplies this timeout value (in whole seconds) for use by SIO.

BYTE COUNT -- DBYTLO [0308] and DBYTHI [0309]

This 2-byte counter indicates the number of bytes transferred to or from the disk as a result of the most recent command, and is set up by the Handler.

SECTOR NUMBER -- DAUX1 [030A] and DAUX2 [030B]

This 2-byte number specifies the diskette sector number (1 - 720) to read or write. DAUX1 contains the least significant byte, and

DAUX2 contains the most significant byte.

Diskette Handler Commands

There are five commands supported by the Diskette Handler:

GET SECTOR (PUT SECTOR --*** not supported by current handler ***)
PUT SECTOR WITH VERIFY
STATUS REQUEST
FORMAT DISK

GET SECTOR (Command byte = $52)

The Handler reads the specified sector to your buffer and returns the operation status. You set the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $52.

DEVICE NUMBER = disk drive number (1-4).

BUFFER ADDRESS = pointer to your 128-byte buffer.

SECTOR NUMBER = sector number to read.

Upon return from the sector, several of the other DCB parameters will have been altered. The STATUS BYTE will be the only parameter of interest to you, however.

PUT SECTOR (Command byte = $50)

*** Not supported by current Handler ***
(But can be accessed through SIO directly.)

The Handler writes the specified sector from your buffer and returns the operation status. You set the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $50.

DEVICE NUMBER = disk drive number (1-4).

BUFFER ADDRESS = pointer to your 128 byte buffer.

SECTOR NUMBER = sector number to write.

Upon return from the operation, several of the other DCB parameters will have been altered. The STATUS BYTE will be the only one of interest you, however.

PUT SECTOR WITH VERIFY (Command Byte = $57)

The Handler writes the specified sector from your buffer and returns the operation status. This command differs from PUT SECTOR in that the diskette controller reads the sector data after writing to verify the write operation. Aside from the COMMAND BYTE value, the calling sequence is identical to PUT SECTOR.

STATUS REQUEST (Command byte = $53)

The Handler obtains a 4-byte status from the diskette controller and puts it in system location DVSTAT [02EA]. The operation status format is shown below:

```
 7              0
+-+-+-+-+-+-+-+-+
! command stat.  !       DVSTAT + 0
+-+-+-+-+-+-+-+-+
! hardware stat. !              + 1
+-+-+-+-+-+-+-+-+
!    timeout     !              + 2
+-+-+-+-+-+-+-+-+
!    (unused)    !              + 3
+-+-+-+-+-+-+-+-+
```

Figure 5-19. DVSTAT 40-Byte Operation Status Format

The command status contains the following status bits:

Bit 0 = 1 indicates an invalid command frame was received.
Bit 1 = 1 indicates an invalid data frame was received.
Bit 2 = 1 indicates that a PUT operation was unsuccessful.
Bit 3 = 1 indicates that the diskette is write protected.
Bit 4 = 1 indicates active/standby.

The hardware status byte contains the status register of the INS1771-1 Floppy Diskette Controller chip used in the diskette controller. See the documentation for that chip to obtain information relating to the meaning of each bit in the byte.

The timeout byte contains a controller-provided maximum timeout value (in seconds) to be used by the Handler.

You set the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $53.

DEVICE NUMBER = disk drive number (1-4).

Upon return from the operation, several of the other DCB parameters will have been altered. The STATUS BYTE will be the only one of interest to you, however.

FORMAT DISK (Command Byte = $21)

The Handler commands the diskette controller to format the entire diskette and then to verify it. All bad sector numbers (up to a maximum of 63) are returned and put in the supplied buffer, followed by two bytes of all 1's ($FFFF). You set up the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $21.

DEVICE NUMBER = disk drive number (1-4).

BUFFER ADDRESS = pointer to your 128-byte buffer.

Upon return, you might be interested in the following DCB parameters:

STATUS BYTE = status of operation.

BYTE COUNT = number of bytes of bad sector information in your buffer, not including the $FFFF terminator. If there are no bad sectors, the count will equal zero.

Serial Bus I/O

Input/Output to devices other than the keyboard, the screen, and the ATARI Computer controller port devices, must utilize the Serial I/O bus. This bus contains data, control, and clock lines to be used to allow the computer to communicate with external devices on this "daisychained" bus. Every device on the bus has a unique identifier and will respond only when directly addressed.

The resident system provides a Serial I/O Utility (SIO), that provides a standardized high-level program interface to the bus. SIO is utilized by the resident Diskette, Printer, and Cassette handlers, and is intended to be used by nonresident handlers (see Section 9), or by applications, as well. For a detailed description of the program/SIO interface and for a detailed bus specification refer to Section 9.

6 INTERRUPT PROCESSING

Section 6 describes system actions for the various interrupt causing events, defines the many RAM vectors and provides recommended procedures for dealing with interrupts.

The 6502 microcomputer processes three general interrupt types: chip-reset, nonmaskable interrupts (NMI) and maskable interrupts (IRQ). The IRQ interrupt type can be enabled and disabled using the 6502 CLI and SEI instructions. The NMI type cannot be disabled at the processor level; but the NMI interrupts other than [SYSTEM.RESET] key can be disabled at the ANTIC chip.

The system events that can cause interrupts are listed below:

```
chip-reset - power-up

NMI - Display list interrupt (unused by OS)
      vertical-blank (50/60 Hz)
      [SYSTEM.RESET] key

IRQ - Serial bus output ready
      Serial bus output complete
      Serial bus input ready
      Serial bus proceed line (unused by system)
      Serial bus interrupt line (unused by system)
      POKEY timers 1, 2 and 4
      Keyboard key
      [BREAK] key
      6502 BRK instruction (unused by OS)
```

Figure 6-1 List of System-Interrupt Events

The chip-reset interrupt is vectored via location FFFC to E477, where a JMP vector to the power-up routine is located. All NMI interrupts are vectored via location FFFA to the NMI interrupt service routine at E7B4, and all IRQ interrupts are vectored via location FFFE to the IRQ interrupt service routine at E6F3; at that point the cause of the interrupt must be determined by a series of tests. For some of the events there are built in monitor actions and for other events the corresponding interrupts are disabled or ignored. The system provides RAM vectors so that you can intercept interrupts when necessary.

CHIP-RESET

The OS generates chip-reset in response to a power-up condition. The system is completely initialized (see Section 7).

NONMASKABLE INTERRUPTS

When an NMI interrupt occurs, control is transferred through the ROM vector directly to the system NMI interrupt service routine. A cause for the interrupt is determined by examining hardware register NMIST [D40F]. The NMI makes a jump through the global RAM vector VDSLST [0200] if a display list interrupt is pending. The OS does not use display list interrupts, so VDSLST is initialized to point to an RTI instruction, and you must not change it before VDSLST generates a display interrupt.

If the interrupt is not a display-list interrupt, then a test is made to see if it is a [SYSTEM.RESET] key interrupt. If so, then a jump is made to the system reset initialization routine (see Section 7 for details of system reset initialization).

If the interrupt is neither a display list interrupt nor a [SYSTEM.RESET] key interrupt; then it is assumed to be a vertical-blank (VBLANK) interrupt, and the following actions occur:

Registers A, X and Y are pushed to the stack.

The interrupt request is cleared (NMIRES [D40F]).

A jump is made through the "immediate" vertical-blank global RAM vector VVBLKI [0222] that normally points to the Stage 1 VBLANK processor.

The following actions occur assuming that you have not changed VVBLKI.

The stage 1 VBLANK processor is executed.

The OS tests to see if a critical code section has been interrupted. If so; then all registers are restored, and an RTI instruction returns from the interrupt to the critical section. A critical section is determined by examining the CRITIC flag [0042], and the processor I bit. If either are set, then the interrupted section is assumed to be critical.

If the interrupt was not from a critical section, then the stage 2 VBLANK processor is executed.

The OS then jumps through the "deferred" vertical-blank global RAM vector VBLKD [0224], that normally points to the VBLANK exit routine.

The following actions occur assuming that you have not changed VVBLKD.

o The 6502 A, X and Y registers are restored.

o An RTI instruction is executed.

NOTE: You can alter the deferred and immediate VBLANK RAM vectors, but still enable normal system processes; or restore original vectors without having to save them. The instruction at E45F is a JMP to the stage 1 VBLANK processor; the address at [E460,2] is the value normally found in VVBLKI. The instruction at E462 is a JMP to the VBLANK exit routine; the address at [E463,2] is the value normally found in VVBLKD. These ROM vectors to stage 1 VBLANK processor and to the VBLANK exit routine will accomplish your goal.

NOTE: Every VBLANK interrupt jumps through vector VVBLKI. Only VBLANK interrupts from noncritical code sections jump through vector VVBLKD.

Stage 1 VBLANK Process

The following stage 1 VBLANK processing is performed at every VBLANK interrupt:

The stage 1 VBLANK process increments the 3-byte frame counter RTCLOK [0012-0014]; RTCLOK+0 is the MSB and RTCLOK+2 is the LSB. This counter wraps to zero when it overflows (every 77 hours or so), and continues counting.

The Attract mode variables are processed (see Appendix L, B10-12).

The stage 1 VBLANK process decrements the System Timer 1 CDTMV1 [0218,2] if it is nonzero; if the timer goes from nonzero to zero then an indirect JSR is performed via CDTMA1 [0226,2].

Stage 2 VBLANK Process

The stage 2 VBLANK processing performs the following for those VBLANK interrupts that do not interrupt critical sections:

The stage 2 VBLANK process clears the 6502 processor I bit. This enables the IRQ interrupts.

The stage 2 VBLANK process updates various hardware registers with data from the OS data base, as shown below:

Data Base Item	Hardware Register	Reason for Update
SDLSTH [0231]	DLISTH [D403]	Display list start
SDLSTL [0230]	DLISTL [D402]	
SDMCTL [022F]	DMACTL [D400]	
CHBAS [02F4]	CHBASE [D409]	
CHACT [02F3]	CHACTL [D401]	
GPRIOR [026F]	PRIOR [D01B]	
COLOR0 [02C4]	COLPF0 [D016]	Attract mode.
COLOR1 [02C5]	COLPF1 [D017]	
COLOR2 [02C6]	COLPF2 [D018]	
COLOR3 [02C7]	COLPF3 [D019]	
COLOR4 [02C8]	COLBK [D01A]	
PCOLR0 [02C0]	COLPM0 [D012]	
PCOLR1 [02C1]	COLPM1 [D013]	
PCOLR2 [02C2]	COLPM2 [D014]	
PCOLR3 [02C3]	COLPM3 [D015]	
Constant = 8	CONSOL [D01F]	Console speaker off.

The stage 2 VBLANK process decrements the System Timer 2 CDTMV2 [021A,2] if it is nonzero; if the timer goes from nonzero to zero, then an indirect JSR is performed through CDTMA2 [0228,2].

The stage 2 VBLAMK process decrements System Timers 3, 4 and 5 if they are nonzero; the corresponding flags are set to zero for each timer that changes from nonzero to zero.

Timer	Timer Value	Timer Flag
3	CDTMV3 [021C,2]	CDTMF3 [022A,1]
4	CDTMV4 [021E,2]	CDTMF4 [022C,1]
5	CDTMV5 [0220,2]	CDTMF5 [022E,1]

A character is read from the POKEY keyboard register and stored in CH [02FC], if auto repeat is active.

The stage 2 VBLANK process decrements the keyboard debounce counter if it is not equal to zero, and if no key is pressed.

The stage 2 VBLANK process processes the keyboard auto repeat (see Appendix L, E8).

The stage 2 VBLANK process reads game controller data from the hardware to the RAM data base, as shown below:

Hardware Register	Data Base Item	Function
PORTA [D300]	STICK0 [0278]	Joysticks and
	STICK1 [0279]	
	PTRIG0 [027C]	Paddle Controllers
	PTRIG1 [027D]	
	PTRIG2 [027E]	
	PTRIG3 [027F]	
PORTB [D301]	STICK2 [027A]	
	STICK3 [027B]	
	PTRIG4 [0280]	
	PTRIG5 [0281]	
	PTRIG6 [0282]	
	PTRIG7 [0283]	
POT 0 [D200]	PADDL0 [0270]	Paddle Controllers
POT 1 [D201]	PADDL1 [0271]	
POT 2 [D202]	PADDL2 [0272]	
POT 3 [D203]	PADDL3 [0273]	
POT 4 [D204]	PADDL4 [0274]	
POT 5 [D205]	PADDL5 [0275]	
POT 6 [D206]	PADDL6 [0276]	
POT 7 [D207]	PADDL7 [0277]	
TRIG0 [D001]	STRIG0 [0284]	Joystick triggers.
TRIG1 [D002]	STRIG1 [0285]	
TRIG2 [D003]	STRIG2 [0286]	
TRIG3 [D004]	STRIG3 [0287]	

MASKABLE INTERRUPTS

An IRQ interrupt causes control to be transferred through the immediate IRQ global RAM vector VIMIRQ [0216]. Ordinarily this vector points to the system IRQ Handler. The Handler performs these following actions:

The IRQ Handler determines a cause for the interrupt by examining the IRQST [D20E] register and the PIA status registers PACTL [D302] and PBCTL [D303]. The interrupt status bit is cleared when it is found. One interrupt event is cleared and processed for each interrupt-service entry. If multiple IRQs are pending, then a separate interrupt will be generated for each pending IRQ, until all are serviced.

The system IRQ interrupt service routine deals with each of the possible IRQ causing events, in the following ways:

- The 6502 A register is pushed to the stack.
- If the interrupt is due to serial I/O bus output ready, then clear the interrupt and jump through global RAM vector VSEROR [020C].
- If the interrupt is due to serial I/O bus input ready, then clear the interrupt and jump through global RAM vector VSERIN [020A].
- If the interrupt is due to serial I/O bus output complete, then clear the interrupt and jump through global RAM vector VSEROC [020E].
- If the interrupt is due to POKEY timer #1, then clear the interrupt and jump through global RAM vector VTIMR1 [0210].
- If the interrupt is due to POKEY timer #2, then clear the interrupt and jump through global RAM vector VTIMR2 [0212].
- If the interrupt is due to POKEY timer #4, then clear the interrupt. The service routine contains a bug, and falls into the following test.
- If pressing a keyboard key caused the interrupt (other than [BREAK], [START], [OPTION], or [SELECT]); then clear the interrupt and jump through global RAM vector VKEYBD [0208].
- If pressing the [BREAK] key caused the interrupt; then clear the interrupt. Set the BREAK flag BRKKEY [0011] to zero, proceed to clear the following:

  Start/stop flag SSFLAG [02FF]
  Cursor inhibit flag CRSINH [02F0]
  Attract mode flag ATRACT [004D]

Return from the interrupt after restoring the 6502 A register from the stack.

- o If the interrupt is due to the serial I/O bus proceed line; then clear the interrupt, and jump through global RAM vector VPRCED [0202].

- o If the interrupt is due to the serial I/O bus interrupt line, then clear the interrupt and jump through global RAM vector VINTER [0204].

- o If the interrupt is due to a 6502 BRK instruction, then jump through global RAM vector VBREAK [0206].

- o If none of the above, restore the 6502 A register and return from the interrupt (RTI).

INTERRUPT INITIALIZATION

The interrupt subsystem completely reinitializes itself whenever the system is powered up or the [SYSTEM.RESET] key is pressed. The OS clears the hardware registers, and sets the interrupt global RAM vectors to the following configurations:

Vector	Type	Function
VDSLST [0200]	NMI	RTI -- ignore interrupt.
VVBLKI [0222]	"	System stage 1 VBLANK.
CDTMA1 [0226]	"	SIO timeout timer.
CDTMA2 [0228]	"	No system function.
VVBLKD [0224]	"	System return from interrupt.
VIMIRQ [0216]	IRQ	System IRQ processor.
VSEROR [020C]	"	SIO.
VSERIN [020A]	"	SIO.
VSEROC [020E]	"	SIO.
VTIMR1 [0210]	"	PLA, RTI -- ignore interrupt.
VTIMR2 [0212]	"	PLA, RTI -- ignore interrupt.
VTIMR4 [0214]	"	*** doesn't matter ***
VKEYBD [0208]	"	System keyboard interrupt handler.
VPRCED [0202]	"	PLA, RTI -- ignore interrupt.
VINTER [0204]	"	PLA, RTI -- ignore interrupt.
VBREAK [0206]	BRK	PLA, RTI -- ignore interrupt.

Figure 6-2 Interrupt RAM Vector Initialization

System initialization sets the interrupt enable status as follows:

NMI	VBLANK enabled, display list disabled.
IRQ	[BREAK] key and data key interrupts enabled, all others disabled.

SYSTEM TIMERS

The OS contains five general purpose software timers, plus an OS-supported frame counter. The timers are 2 bytes in length (lo,hi) and the frame counter RTCLOK [0012] is three bytes in length (hi,mid,lo). The timers count downward from any nonzero value to zero. Upon reaching zero, they either clear an associated flag, or JSR through a RAM vector. The frame counter counts upward, wrapping to zero when it overflows.

The following table shows the timers and the frame counter characteristics:

	Timer Name	Flag/Vector	Use
*	CDTMV1 [0218]	CDTMA1 [0226]	2-byte vector -- SIO timeout.
	CDTMV2 [021A]	CDTMA2 [0228]	2-byte vector
	CDTMV3 [021C]	CDTMF3 [022A]	1-byte flag
	CDTMV4 [021E]	CDTMF4 [022C]	1-byte flag
	CDTMV5 [0220]	CDTMF5 [022E]	1-byte flag
*	RTCLOK [0012]		3-byte frame counter.

* These two timers are maintained as part of every VBLANK interrupt (stage 1 process). The other timers are subject to the critical section test (stage-2 process), that can defer their updating to a later VBLANK interrupt.

USAGE NOTES

This subsection describes the techniques you need to know in order to utilize interrupts in conjunction with the operating system.

POKEY Interrupt Mask

ANTIC (display-list and vertical-blank) and PIA (interrupt and proceed lines) interrupts can be masked directly (see the Hardware Manual). However, eight bits of a single byte IRQEN [D20E] mask the POKEY interrupts ([BREAK] key, data key, serial input ready, serial output ready, serial output done and timers 1,2 and 4).

IRQEN is a write-only register. Thus, we must maintain a current value of that register in RAM in order to update individual mask bits selectively, while not changing other bits. The name of the variable used is POKMSK [0010], and it is used as shown in the examples below:

```
; EXAMPLE OF INTERRUPT ENABLE

        SEI              ; TO AVOID CONFLICT WITH IRQ ...
        LDA     POKMSK   ; ... PROCESSOR WHICH ALTERS VAR.
        ORA     #$xx     ; ENABLE BIT(S).
        STA     POKMSK
        STA     IRQEN    ; TO HARDWARE REG TOO.
        CLI

; EXAMPLE OF INTERRUPT DISABLE

        SEI              ; TO AVOID CONFLICT WITH IRQ ...
        LDA     POKMSK   ; ... PROCESSOR WHICH ALTERS VAR.
        AND     #$FF-xx  ; DISABLE BIT(S).
        STA     POKMSK
        STA     IRQEN    ; TO HARDWARE REGISTER TOO.
        CLI
```

Figure 6-3   POKEY Interrupt Mask Example

Note that the OS IRQ service routine uses and alters POKMSK, so alterations to the variable must be done with interrupts inhibited. If done at the interrupt level there is no problem, as the I bit is already set; if done at a background level then the SEI and CLI instructions should be used as shown in the examples.

Setting Interrupt and Timer Vectors

Because vertical-blank interrupts are generally kept enabled so that the frame counter RTCLOK is maintained accurately, there is a problem with setting the VBLANK vectors (VVBLKI and VVBLKD) or the timer values (CDTMV1 through CDTMV5) directly. A VBLANK interrupt could occur when only one byte of the two-byte value had been updated, leading to undesired consequences. For this reason, the SETVBV [E45F] routine is provided to perform the desired update in safe manner. The calling sequence is shown below:

```
A = update item indicator
    1 - 5 for timers 1 - 5.
    6 for immediate VBLANK vector VVBLKI.
    7 for deferred VBLANK vector VVBLKD.
X = MSB of value to store.
Y = LSB of value to store.

JSR    SETVBV
```

The A, X and Y registers can be altered.
The display list interrupt will always be disabled on return, even if enabled upon entry.

It is possible to fully process a vertical-blank interrupt during a call to this routine.

When working with the System Timers, the vectors for timers 1 and 2 and the flags for timers 3, 4 and 5 should be set while the associated timer is equal to zero, then the timer should be set to its (nonzero) value.

Stack Content at Interrupt Vector Points

The following table shows the stack content at every one of the RAM interrupt vector points:

RAM STACK CONTENT

INTERRUPT VECTOR	DESCRIPTION	OS RETURN CONTROL
VDSLST [0200]	Display list	return, P
VVBLKI [0222] *	VBLANK immediate	return, P, A, X, Y
CDTMA1 [0226]	System Timer 1	return, P, A, X, Y, return
CDTMA2 [0228]	System Timer 2	return, P, A, X, Y, return
VVBLKD [0224] *	VBLANK defer.	return, P, A, X, Y
VIMIRQ [0216] *	IRQ immediate	return, P, A
VSEROR [020C] *	Serial out ready	return, P, A
VSERIN [020A] *	Serial in ready	return, P, A
VSEROC [020E] *	Serial out compare	return, P, A
VTIMR1 [0210]	POKEY timer 1	return, P, A
VTIMR2 [0212]	POKEY timer 2	return, P, A
VTIMR4 [0214]	POKEY timer 4	return, P, A
VKEYBD [0208] *	Keyboard data	return, P, A
VPRSED [0202]	Serial proceed	return, P, A
VINTER [0204]	Serial interrupt	return, P, A
VBREAK [0206]	BRK instruction	return, P, A

Figure 6-4 Interrupt and Timer Vector RAM Stack Content Table

* The OS initializes these entries at power-up. Improperly changing these vectors will alter system performance.

Miscellaneous Considerations

The following paragraphs list a set of miscellaneous considerations for the writer of an interrupt service routine.

Restrictions on Clearing of "I" Bit

Display list, immediate vertical-blank and System Timer #1 routines should not clear the 6502 I bit. If the NMI leading to one of these routines occurred while an IRQ was being processed, then clearing the I bit will cause the IRQ to re-interrupt with an unknown result.

The OS VBLANK processor carefully checks this condition after the stage 1 process and before the stage 2 process.

Interrupt Process Time Restrictions

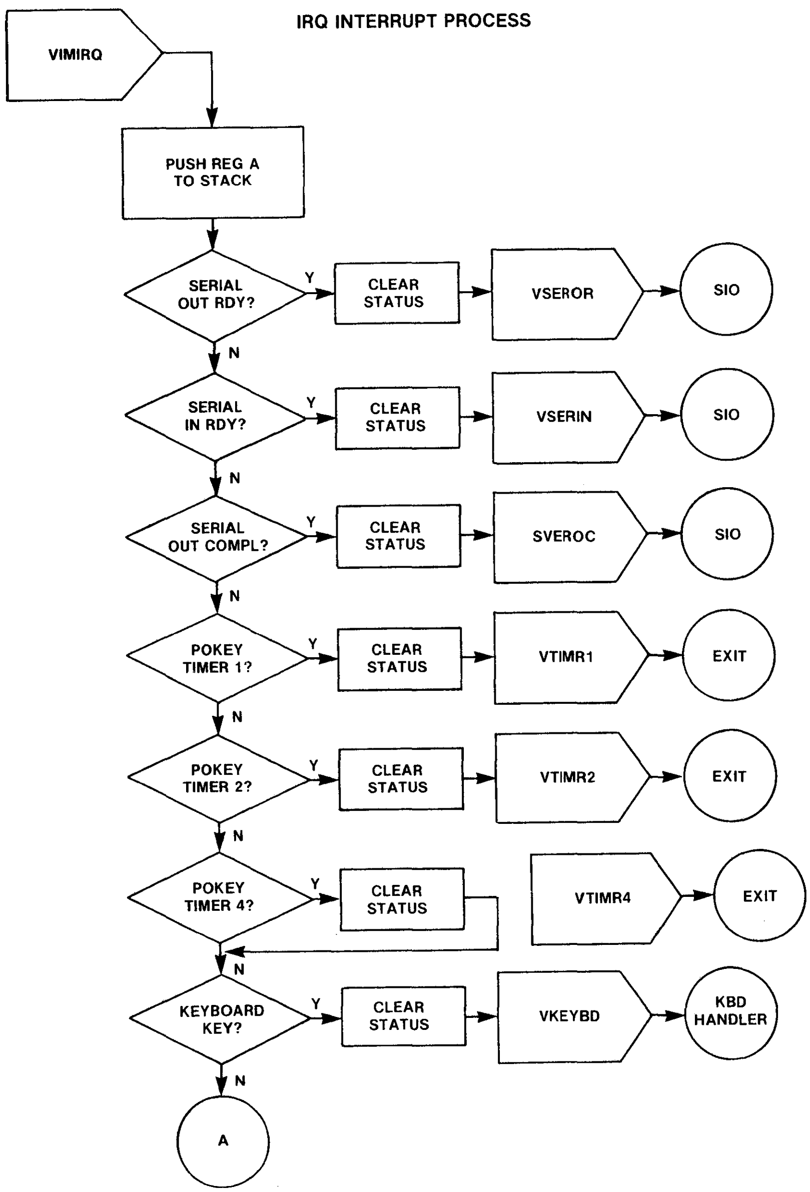
You should not write an interrupt routine that exceeds 400 msec. when added to the stage 1 VBLANK, if the serial I/O is being used. The SIO sets the CRITIC flag while serial bus I/O is in progress.

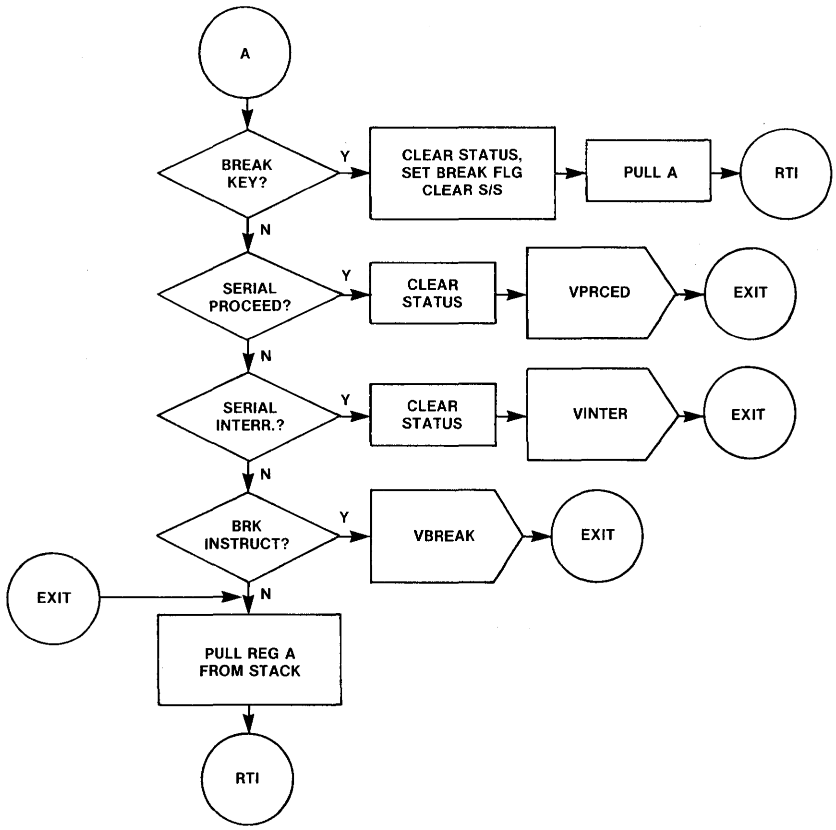
Interrupt Delay Due to "WAIT FOR SYNC"

Whenever a key is read from the keyboard, the Keyboard Handler sets WSYNC [D40A] repeatedly while generating the audible click on the console speaker. A problem occurs when interrupts are generated during the wait-for-sync period; the processing of such interrupts will be delayed by one horizontal scan line. This condition cannot be prevented. You can work around the condition by examining the line count VCOUNT [D40B] and delaying interrupt processing by one line when no WSYNC delay has occurred.

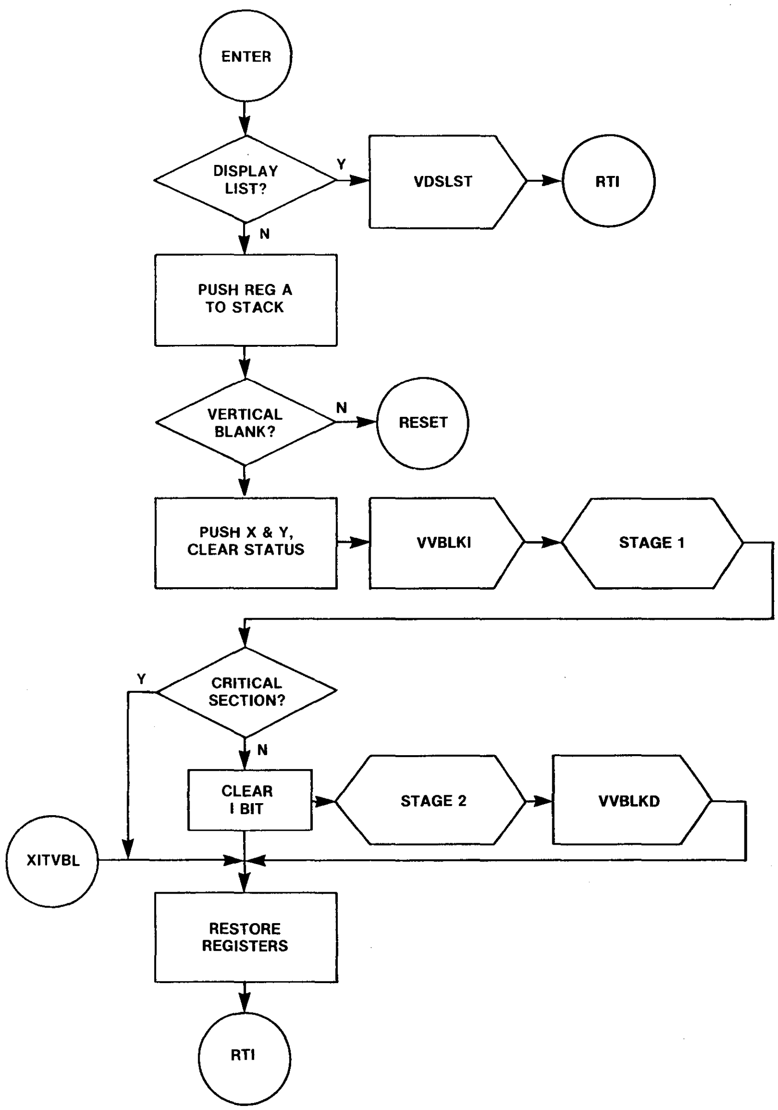
FLOWCHARTS

The following pages contain process flowcharts showing the main events that occur in the NMI and IRQ interrupt processes.




A
BREAK KEY?
Y
N
CLEAR STATUS, SET BREAK FLG CLEAR S/S
PULL A
RTI
SERIAL PROCEED?
Y
N
CLEAR STATUS
VPRCED
EXIT
SERIAL INTERR.?
Y
N
CLEAR STATUS
VINTER
EXIT
BRK INSTRUCT?
Y
N
VBREAK
EXIT
EXIT
PULL REG A FROM STACK
RTI

NMI INTERRUPT PROCESS



7 SYSTEM INITIALIZATION

Section 7 discusses the details of the power-up and system reset processes. The power-up process will be explained first, and then the system reset process will be explained in terms of its differences from the power-up process.

Both power-up (also called coldstart) and pressing [SYSTEM.RESET] (warmstart) will cause system initialization: In addition, there are vectors for these processes at E474 (system reset) and E477 (power-up) so that they can be user-initiated.

The power-up initialization process is a superset of the system reset initialization process. Power-up initializes both the OS and user RAM regions, whereas system reset initializes only the OS RAM region. In both cases, the OS calls the outer level software initialization entry points allow the application to initialize its own variables.

Pressing the [SYSTEM.RESET] key produces an NMI interrupt. It does not perform a 6502 chip-reset. If the processor is locked up, the [SYSTEM.RESET] key cannot be sufficient to unlock it, and the system must have power cycled to clear the problem.

POWER-UP INITIALIZATION (COLDSTART) PROCEDURE

The OS performs the following functions in the order shown, as part of the power-up initialization process:

1. The following 6502 processor states are set:

o IRQ interrupts are disabled using the SEI instruction.

o The decimal flag is cleared using the CLD instruction.

o The stack pointer is set to $FF.

2. The OS sets the warmstart flag WARMST [0008] to 0 (false).

3. The OS tests to see if a diagnostic cartridge is in the A slot:

- o Cartridge address BFFC = 00?
- o The memory at BFFC is not RAM?
- o Bit 7 of the byte at BFFD = 1?

If all of the above tests are true, then control is passed to the diagnostic cartridge via the vector at BFFE. No return is expected.

4. The OS determines the lowest memory address containing non-RAM, by testing the first byte of every 4K "block" to see if the content can be complemented. If it can be complemented, then the original value is restored and testing continues. If it can't be complemented; then the content is assumed to be the first non-RAM address in the system. The MSB of the address is stored temporarily in TRAMSZ [0006].

5. Zero is stored to all of the hardware register addresses shown below (most of that aren't decoded by the hardware):

    D000 through D0FF
    D200 through D2FF
    D300 through D3FF
    D400 through D4FF

6. The OS clears RAM from location 0008, to the address determined in step 4, above.

7. The default value for the "noncartridge" control vector DOSVEC [000A] is set to point to the blackboard routine. At the end of initialization, control is passed through this vector if a cartridge does not take control.

8. The coldstart flag COLDST [0244] is set to -1 (local use).

9. The screen margins are set: left margin = 2, right margin = 39, for a 38 character physical line. The maximum line size of 40 characters can be obtained by setting the margins to 0 and 39. The OS insets the left margin because the two leftmost columns of the video picture on many television sets are not entirely visible on the screen.

10. The interrupt RAM vectors VDSLST [0200] through VVBLKD [0224] are initialized. See Section 6 for the initialization values.

11. Portions of the OS RAM are set to their required nonzero values as shown below:

- o The [BREAK] key flag BRKKEY [0011] = -1 (false).

- o The top of memory pointer MEMTOP [02E5] = the lowest non-RAM address (from step 4); MEMTOP will be altered later when the Screen Editor is opened in step 15.

- o The bottom of memory pointer MEMLO [02E7] = 0700; MEMLO can be changed later if there is either a diskette- or cassette-boot operation.

- o The following resident routines are called for initialization:

  Screen Editor
  Display Handler
  Keyboard Handler
  Printer Handler
  Cassette Handler
  Central I/O Monitor (CIO)
  Serial I/O Monitor (SIO)
  Interrupt processor

- o The [START] key is checked, and if pressed, the cassette-boot request flag CKEY [004A] is set.

12. 6502 IRQ interrupts are enabled using the CLI instruction.

13. The device table HATABS [031A] is initialized to point to the resident handlers. See Section 9 for information relating to the Device Handler table.

14. The cartridge slot addresses for cartridges B and A are examined to determine if cartridges are inserted, if RAM does not extend into the cartridge address space.

    If the content of location 9FFC is zero, then a JSR is executed through the vector at 9FFE, thus initializing cartridge "B". The cartridge is expected to return.

    If the content of location BFFC is zero, then a JSR is executed through the vector at BFFE, thus initializing cartridge "A". The cartridge is expected to return.

15. IOCB #0 is set up for an OPEN of the Screen Editor (E) and the OPEN is performed. The Screen Editor will use the highest portion of RAM for the screen and will adjust MEMTOP accordingly. If this operation should fail, the entire initialization process is repeated.

16. A delay is effected to assure that a VBLANK interrupt has occurred. This is done so that the screen will be established before continuing.

17. If the cassette-boot request flag is set (see step 11 above), then a cassette-boot operation is attempted. See Section 10

for details of the cassette-boot operation.

18. If any of the three conditions stated below exists, an attempt is made to boot from the disk.

    There are no cartridges in the slots.

    Cartridge B is inserted and bit 0 of 9FFD is 1.

    Cartridge A is inserted and bit 0 of BFFD is 1.

    See Section 10 for details of the diskette-boot operation.

19. The coldstart flag COLDST is reset to indicate that the coldstart process went to completion.

20. The initialization process is now complete, and the controlling application is now determined via the remaining steps.

    If there is an A cartridge inserted and bit-2 of BFFD is 1, then a JMP is executed through the vector at BFFA.

    Or, if there is a B cartridge inserted and bit-2 of 9FFD is 1, then a JMP is executed through the vector at 9FFA.

    Or, a jump is executed through the vector DOSVEC that can point to the blackboard routine (default case), cassette booted software or diskette booted software. DOSVEC can be altered by the booted software as explained in Section 10.

SYSTEM RESET INITIALIZATION (WARMSTART) PROCEDURE

The functions listed below are performed, in the order shown, as part of the system reset initialization process:

A. Same as power-up step 1.

B. The warmstart flag WARMST [0008] is set to -1 (true).

C. Same as power-up steps 3 through 5.

D. OS RAM is zeroed from locations 0200-03FF and 0010-007F.

E. Same as power-up steps 9 through 16.

F. If a cassette-boot was successfully completed during the power-up initialization, then a JSR is executed through the vector CASINI [0002]. See Section 10 for details of the cassette-boot process.

G. Same as power-up step 18, except instead of booting the diskette software, a JSR is executed through the vector DOSINI [000C] if the diskette-boot was successfully completed during the Power-up initialization. See Section 10 for details of the diskette-boot process.

H. Same as power-up steps 19 and 20.

Note that the initialization procedures and main entries for all software entities are executed at every system reset as well as at power up (see steps 14, 17, 18, 20, F and G). If the user-supplied initialization/startup code must behave differently in response to system reset than it does to power-up, then the warmstart flag WARMST [0008] should be interrogated; WARMST = 0 means power-up entry, else system reset entry.

8 FLOATING POINT ARITHMETIC PACKAGE

This section describes the BCD floating point (FP) package that is resident in the OS ROM in both the models 400 and 800.

The floating point package maintains numbers internally as 6-byte quantities: a 5-byte (10 BCD digit) mantissa with a 1-byte exponent. BCD internal representation was chosen so that decimal division would not lead to the rounding errors typically found in binary representation implementations.

The package provides the following operations:

- ASCII to FP conversion.
- FP to ASCII conversion.
- Integer to FP conversion.
- FP to integer conversion.
- FP add, subtract, multiply,and divide.
- FP logarithm, exponentiation, and polynomial evaluation.
- FP zero, load, store, and move.

A floating point operation is performed by calling one of the provided routines (each at a fixed address in ROM) after having set one or more floating point pseudo registers in RAM. The result of the desired operation will also involve floating point pseudo registers. The primary pseudo registers are described below and their addresses given within the square brackets:

```
FR0 [00D4] =    6-byte internal form of FP number.
FR1 [00E0] =    6-byte internal form of FP number.
FLPTR [00FC] =  2-byte pointer (lo,hi) to a FP.
                number.
INBUFF [00F3] = 2-byte pointer (lo,hi) to an ASCII text
                buffer.
CIX [00F2] =    1-byte index, used as offset to buffer
                pointed to by INBUFF.
LBUFF [0580] =  result buffer for the FASC routine.
```

FUNCTIONS/CALLING SEQUENCES

Descriptions of these floating point routines assume that a pseudo register is not altered by a given routine. The numbers in square brackets [xxxx] are the ROM addresses of the routines.

ASCII to Floating Point Conversion (AFP)

Function: This routine takes an ASCII string as input and produces a floating point number in internal form.

Calling sequence:

```
INBUFF = pointer to buffer containing the ASCII
         representation of the number.
CIX = the buffer offset to the first byte of the ASCII
      number.

JSR      AFP [D800]
BCS      first byte of ASCII number is invalid

FR0 = floating point number.
CIX = the buffer offset to the first byte after the ASCII
      number.
```

Algorithm: The routine takes bytes from the buffer until it encounters a byte that cannot be part of the number. The bytes scanned to that point are then converted to a floating point number. If the first byte encountered is invalid, the carry bit is set as a flag.

Floating Point to ASCII Conversion (FASC)

Function: This routine converts a floating point number from internal form to its ASCII representation.

Calling sequence:

FR0 = floating point number.

JSR FASC [D8E6]

INBUFF = pointer to the first byte of the ASCII number. The last byte of the ASCII representation has the most significant bit (sign bit) set; no EOL follows.

Algorithm: The routine converts the number from its internal floating point representation to a printable form (ATASCII). The pointer INBUFF will point to part of LBUFF, where the result is stored.

Integer to Floating Point Conversion (IFP)

Function: This routine converts a 2-byte unsigned integer (0 to 65535) to floating point internal representation.

Calling sequence:

FR0 = integer (FR0+0 = LSB, FR0+1 = MSB).

JSR IFP [D9AA]

FR0 = floating point representation of integer.

Floating Point to Integer Conversion (FPI)

Function: This routine converts a positive floating point number from its internal representation to the nearest 2-byte integer.

Calling sequence:

FR0 = floating point number.

JSR FPI [D9D2]
BCS FP number is negative or >= 65535.5

FR0 = 2-byte integer (FR0+0 = LSB, FR0+1 = MSB).

Algorithm: The routine performs true rounding, not truncation, during the conversion process.

Floating Point Addition (FADD)

Function: This routine adds two floating point numbers and checks the result for out-of-range.

Calling sequence:

```
     FR0 = floating point number.
     FR1 = floating point number.

     JSR       FADD [DA66]
     BCS       out-of-range result.

     FR0 = result of FR0 + FR1.
     FR1 is altered.
```

Floating Point Subtraction (FSUB)

Function: This routine subtracts two floating point numbers and checks the result for out-of-range.

Calling sequence:

```
     FR0 = floating point minuend.
     FR1 = floating point subtrahend.

     JSR       FSUB [DA60]
     BCS       out-of-range result.

     FR0 = result of FR0 - FR1.
     FR1 is altered.
```

Floating Point Multiplication (FMUL)

Function: This routine multiplies two floating point numbers and checks the result for out-of-range.

Calling sequence:

```
     FR0 = floating point multiplier.
     FR1 = floating point multiplicand.

     JSR       FMUL [DADB]
     BCS       out-of-range result.

     FR0 = result of FR0 * FR1.
     FR1 is altered.
```

Floating Point Division (FDIV)

Function: This routine divides two floating point numbers and checks for division by zero and for result out-of-range.

Calling sequence:

```
     FR0 = floating point dividend.
     FR1 = floating point divisor.

     JSR      FDIV [DB28]
     BCS      out-of-range result or divisor is zero.

     FR0 = result of FR0 / FR1.
     FR1 is altered.
```

Floating Point Logarithms (LOG and LOG10)

Function: These routines take the natural or base 10 logarithms of a floating point number.

Calling sequence:

```
     FR0 = floating point number.

     JSR      LOG [DECD] for natural logarithm
or
     JSR      LOG10 [DED1] for base 10 logarithm
     BCS      negative number or overflow.

     FR0 = floating point logarithm.
     FR1 is altered.
```

Algorithm: Both logarithms are first computed as base 10 logarithms using a 10 term polynomial approximation; the natural logarithm is computed by dividing the base 10 result by the constant LOG10(e).

The logarithm of a number Z is computed as follows:

```
     F * (10 ** Y) = Z where 1 <= F < 10 (normalization).
     L = LOG10(F) by 10 term polynomial approximation.
     LOG10(Z) = Y + L.  LOG(Z) = LOG10(Z) / LOG10(e).
```

NOTE: This routine does not return an error if the number input is zero; the LOG10 result in this case is approximately -129.5, which is not useful.

Floating Point Exponentiation (EXP and EXP10)

Function: This routine exponentiates.

Calling sequence:

```
     FRO = floating point exponent (Z).

     JSR       EXP [DDC0] for e ** Z
or
     JSR       EXP10 [DDCC] for 10 ** Z
     BCS       overflow.

     FRO = floating point result.
     FR1 is altered.
```

Algorithm: Both exponentials are computed internally as base 10, with the base e exponential using the identity:
e ** X = 10 ** (X * LOG10(e)).

The base 10 exponential is evaluated in two parts using the identity:

10 ** X = 10 ** (I + F) = (10 ** I) * (10 ** F) -- where I is the integer portion of X and F is the fraction.

The term 10 ** F is evaluated using a polynomial approximation, and 10 ** I is a straightforward modification to the floating point exponent.

Floating Point Polynomial Evaluation (PLYEVL)

Function: This routine performs an n degree polynomial evaluation.

Calling sequence:

```
     X,Y = pointer (X = LSB) to list of FP coefficients (A(i))
           ordered from high order to low order (six bytes per
           coefficient).
     A = number of coefficients in list.
     FRO = floating point independent variable (Z).

     JSR       PLYEVL [DD40]
     BCS       overflow or other error.

     FRO = result of A(n)*Z**n + A(n-1)*Z**n-1 ... + A(1)*Z +
           A(0).
     FR1 is altered.
```

Algorithm: The polynomial P(Z) = SUM(i=0 to n) (A(i)*Z**i) is computed using the standard method shown below:

```
     P(Z) = (...(A(n)*Z + A(n-1))*Z + ... + A(1))*Z + A(0)
```

Clear FRO (ZFRO)

Function: This routine sets the contents of pseudo register FRO to all zeros.

Calling sequence:

```
JSR       ZFRO [DA44]

FRO = zero.
```

Clear Page Zero Floating Point Number (ZF1)

Function: This routine sets the contents of a zero-page floating point number to all zeroes.

Calling sequence:

```
X = Zero-page address of FP number to clear.

JSR       ZF1 [DA46]

Zero-page FP number(X) = zero.
```

Load Floating Point Number to FRO (FLDOR and FLDOP)

Function: These routines load pseudo register FRO with the floating point number specified by the calling sequence.

Calling sequences:

```
X,Y = pointer (X = LSB) to FP number.

JSR       FLDOR [DD89]
```

or

```
FLPTR = pointer to FP number.

JSR       FLDOP [DD8D]


FRO = floating point number (in either case).
FLPTR = pointer to FP number (in either case).
```

Load Floating Point Number to FR1 (FLD1R and FLD1P)

Function: These routines load pseudo register FR1 with the floating point number specified by the calling sequence.

Calling sequences:

As in prior description, except the result goes to FR1 instead of FR0. FLD1R [DD98] and FLD1P [DD9C].

Store Floating Point Number From FR0 (FSTOR and FSTOP)

Function: These routines store the contents of pseudo register FR0 to the address specified by the calling sequence:

Calling sequence:

As in prior descriptions, except the floating point number is stored from FR0 rather than loaded to FR0. FSTOR [DDA7] and FSTOP [DDAB].

Move Floating Point Number From FR0 to FR1 (FMOVE)

Function: This routine moves the floating point number in FR0 to pseudo register FR1.

Calling sequence:

```
JSR       FMOVE [DDB6]

FR1 = FR0  (FR0 remains unchanged).
```

RESOURCE UTILIZATION

The floating point package uses the following RAM locations in the course of performing the functions described in this section:

00D4 through 00FF
057E through 05FF

All of these locations are available for program coding if your program does not call the floating point package.

IMPLEMENTATION DETAILS

Floating point numbers are maintained internally as 6-byte quantities, with 5 bytes (10 BCD digits) of mantissa and 1 byte of exponent. The mantissa is always normalized such that the most significant byte is nonzero (note "byte" and not "BCD digit").

The most significant bit of the exponent byte provides the sign for the mantissa; 0 for positive and 1 for negative. The remaining 7 bits of the exponent byte provide the exponent in excess 64 notation. The resulting number represents powers of 100 decimal (not powers of 10). This storage format allows the mantissa to hold 10 BCD digits when the value of the exponent is an even power of 10, and 9 BCD digits when the value of the exponent is an odd power of 10.

The implied decimal point is always to the immediate right of the first byte. An exponent less than 64 indicates a number less than 1. An exponent equal to or greater than 64 represents a number equal to or greater than 1.

Zero is represented by a zero mantissa and a zero exponent. To test for a result from any of the standard routines; test either the exponent or the first mantissa byte for zero.

The absolute value of floating point numbers must be greater than 10**-98, and less than 10**+98, or be equal to zero. There is perfect symmetry between positive and negative numbers with the exception that negative zero is never generated.

The precision of all computations is maintained at 9 or 10 decimal digits, but accuracy is somewhat less for those functions involving polynomial approximations (logarithm and exponentiation). Also, the problems inherent in all floating point systems are present here; for example: subtracting two very nearly equal numbers, adding numbers of disparate magnitude, or successions of any operation, will all result in a loss of significant digits. An analysis of the data range and the order of evaluation of expressions may be required for some types of applications.

The examples below compare floating point numbers with their internal representations, as an aid to understanding storage format. All numbers prior to this point have been expressed in decimal notation, but these examples will use hexadecimal notation. Note that 64 decimal (the excess number of the exponent) is 40 when expressed in hexadecimal:

```
Number: +0.02 = 2 * 10**-2 = 2 * 100**-1
Stored: 3F 02 00 00 00 00  (FP exponent = 40 - 1)

Number: -0.02 = -2 * 10**-2 = -2 * 100**-1
Stored: BF 02 00 00 00 00  (FP exponent = 80 + 40 - 1)
```

```
Number: +37.0 = 3.7 * 10**1 = 37 * 100**0
Stored: 40 37 00 00 00 00  (FP exponent = 40 + 0)

Number: -4.60312486 * 10**11 = -46.03...  * 100**5
Stored: C5 46 03 01 24 86  (FP exponent = 80 + 40 + 5)

Number: 0.0
Stored: 00 00 00 00 00 00  (special case)
```

9 ADDING NEW DEVICE HANDLERS/PERIPHERALS

This section describes the interface requirements for a nonresident Device Handler that is to be accessed via the Central I/O utility (CIO). The Serial bus I/O utility (SIO) interface is defined for those handlers that utilize the Serial I/O bus.

The I/O subsystem is organized with three levels of software between you and your hardware: The CIO, the individual device handlers, and the SIO.

The CIO performs the following functions:

Logical device name to Device Handler mapping (on OPEN).

I/O Control Block (IOCB) maintenance.

Logical record handling.

User buffer handling.

The device handlers are below CIO. They perform the following functions:

Device initialization on power-up and system reset.

Device-dependent support of OPEN and CLOSE commands.

Byte-at-a-time data input and output.

Device-dependent special operations.

Device-dependent command support.

Device data buffer management.

The SIO is at the bottom level (for Serial I/O bus peripheral handlers). It performs the following functions:

Control of all Serial bus I/O, conforming to the bus protocol.

Bus operation retries on errors.

Return of unified error statuses on error conditions.

A separate control structure is used for communication at each interface, as follows:

User/CIO	I/O Control Block (IOCB)
CIO/Handler	Zero-page IOCB (ZIOCB)
Handler/SIO	Device Control Block (DCB)

```
               +----------+
               |   user   |
               |  program |---------------------------+
               +----------+                           |
+----------+        |                                 |
|  IOCB's  |********|                                 |
+----------+        |                                 |
               +----------+                           |
               |   CIO    |        +----------+       |
               |  utility |        |   DCB    |*******|
               +----------+        +----------+  *    |
                    |                            *    |
                    |                            *    |
+----------+   +----------+        +----------+  *    |
|  ZIOCB   |   |  Device  |        |Disk File |  *    |
+----------+   |  Table   |    +---| Manager  |-----+ |
     *         +----------+    |   +----------+     | |
     *              |          |                    | |
+------------+------+-------+--+--....---+          | |
|            |              |            |          | |
     +----------+ +----------+  +----------+  +----------+ +----------+
     | Printer  | | Cassette |  | ...      |  | Keyboard | |   Disk   |
     | Handler  | | Handler  |  | Handler  |  | Handler  | | Handler  |
     +----------+ +----------+  +----------+  +----------+ +----------+
|            |              |                         |
+------------+------+-------+-----....----------------+
                    |
+----------+        |
|   DCB    |********|
+----------+        |
               +----------+
               |   SIO    |
               | Utility  |
               +----------+
```

Where: ---- shows a control path.
       **** shows the data structure required for a path.

Note the following:

1. The Keyboard/Display/Screen Editor handlers don't use SIO.
2. The Diskette Handler cannot be called directly from CIO.
3. The DCB is shown twice in the diagram.

Figure 9-1   I/O Subsystem Flow Diagram

DEVICE TABLE

The device table is a RAM-resident table that contains the single-character device name (e.g. K, D, C, etc). and the handler address for each of the handlers known to CIO. The table is initialized to contain entries for the following resident handlers: Keyboard (K), Display (S), Screen Editor (E), Cassette (C), and Printer (P) at power-up and system reset. To install a new handler, some procedure must insert a device table entry after the table is initialized.

The table format is shown below:

```
                +----------------+  -+
HATABS [031A]   !  device name   !   !
                +----------------+   !
                ! handler vector !   +- one entry
                +                +   !
                ! table address  !   !
                +----------------+  -+
                !      more      !
                =                =
                !    entries     !
                +----------------+
                !  zero fill to  !
                =                =
                !  end of table  !
                +----------------+
```

Figure 9-2   Device Table Format

This 38-byte table will hold a maximum of 12 entries, with the last 2 bytes being zero. CIO scans the table from the end to the beginning (high to low address); so the entry nearest the end of the table will take precedence in case of multiple occurrences of a device name..

The device name for each entry is a single ATASCII character, and the handler address points to the handler's vector table, that will be described in the following section.

CIO/HANDLER INTERFACE

This section describes the interface between the Central I/O utility and the individual device handlers that are represented in the Device Table (as described in the preceding section).

Calling Mechanism

Each handler has a vector table as shown below:

```
+----------------+
+ OPEN vector    +      (low address)
+----------------+
+ CLOSE vector   +
+----------------+
+ GETBYTE vector +
+----------------+
+ PUTBYTE vector +
+----------------+
+ GETSTAT vector +
+----------------+
+ SPECIAL vector +
+----------------+
+ JMP init code  +
+                +      (high address)
+----------------+
```

Figure 9-3 Handler Vector Table

The device table entry for the handler points to the first byte of the vector table.

The first six entries in the table are vectors (lo,hi) that contain the address - 1 of the handler routine that handles the indicated function. The seventh entry is a 6502 JMP instruction to the handler initialization routine. CIO uses only the addresses contained in this table for handler entry. Each user/CIO command translates to one or more calls to one of the handler entries defined in the vector table.

The vector table provides the handler addresses for certain fixed functions to be performed to CIO. In addition, operation parameters also must be passed for most functions. Parameter passing is accomplished using the 6502 A, X, and Y registers and an IOCB in page 0 named ZIOCB [0020]. In general, register A is used to pass data, register X contains the index to the originating IOCB, and register Y is used to pass status information to CIO. The zero-page IOCB, is a copy of the originating IOCB; but in the course of processing some commands, CIO can alter the buffer address and buffer length parameters in ZIOCB, but not in the originating IOCB (see Section 5 for information relating to the originating IOCB).

See Appendix B for the standard status byte values to be returned to CIO in register Y.

The following sections describe the CIO/handler interface for each of the vectors in the handler vector table.

Handler Initialization

NOTE: This entry doesn't appear to have any function for nonresident handlers due to a bug in the current OS -- the device table is cleared in response to system reset as well as power-up. This prevents this entry point from ever being called. The rest of this section discusses the intended use of this entry point. Conformation would be in order to allow compatibility with possible corrected versions of the OS in the future.

The entry was to have been called on all occurrences of power-up and system reset; the handler is to perform initialization of its hardware and RAM data using a routine that assures proper processing of all CIO commands that follow.

Functions Supported

This section describes the functions associated with the first six vectors from the handler vector table. This section also presents a brief, device-independent description of the CIO/handler interface and recommended actions for each function vector.

OPEN

This entry is called in response to an OPEN command to CIO. The handler is expected to validate the OPEN parameters and perform any required device initialization associated with a device OPEN.

At handler entry, the following parameters can be of interest:

X = index to originating IOCB.
Y = $92 (status = function not implemented by handler).

ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICBALZ/ICBAHZ [0024/0025] = address of device/filename specification.
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler attempts to perform the indicated OPEN and indicates the status of the operation by the value of the Y register. The responsibility for checking for multiple OPENs to the same device or file, where it is illegal, lies with the handler.

CLOSE

This vector table entry is called in response to a CLOSE command to CIO. The handler is expected to release any held resources that relate specifically to that device/filename, and for output files to:

1) send any data remaining in handler buffers to the device,

2) mark the end of file

3) update any associated directories, allocation maps, etc.

At handler entry, the following parameters can be of interest:

X = index to originating IOCB.
Y = $92 (status = function not implemented by handler).

ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler attempts to perform the indicated CLOSE and indicates the status of the operation by the value of the Y register.

CIO releases the associated IOCB after the handler returns, regardless of the operation status value.

GETBYTE

This vector table entry is called in response to a GET CHARACTERS or GET RECORD command to CIO. The handler is expected to return a single byte in the A register, or return an error status in the Y register.

At handler entry, the following parameters can be of interest:

X = index to originating IOCB.
Y = $92 (status = function not implemented by handler).

ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler will obtain a data byte directly from the device or from a handler-maintained buffer and return to CIO with the byte in the A register and the operation status in the Y register.

Handlers that do not have short timeouts associated with the reading of data (such as the Keyboard and Cassette Handlers), must monitor the [BREAK] key flag BRKKEY [0011] and return with a status of $80 when a [BREAK] condition occurs. See Appendix L, E5; and Section 12 for a discussion of [BREAK] key monitoring.

CIO checks for reads from device/files that have not been opened or have been opened for output only; the handler will not be called in those cases.

PUTBYTE

This entry is called in response to a PUT CHARACTERS or PUT RECORD command to CIO. The handler is expected to accept a single byte in the A register or return an error status in the Y register.

At handler entry, the following parameters can be of interest:

```
X = index to originating IOCB.
Y = $92 (status = function not implemented by handler).
A = data byte.

ICDNOZ [0021] = device number (1-4, for multiple device
                handlers).
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.
```

The handler sends the data byte directly to the device, or to a handler-maintained buffer, and returns to CIO with the operation status in the Y register. If a handler-maintained buffer fills, the handler will send the buffered data to the device before returning to CIO.

CIO checks for WRITEs to device/files that have not been opened, or have been opened for input only. The handler will not be called in those cases.

Now that the normal operation of PUTBYTE has been defined, a special case must be added. Any handler that will operate within the environment of the ATARI 8K BASIC language interpreter has a different set of rules. Because BASIC can call the handler PUTBYTE entry directly. without going through CIO, the zero-page IOCB (ZIOCB) can or may not have a relation to the PUTBYTE call. Therefore, the handler must use the outer level IOCB to obtain any information that would normally be obtained from ZIOCB. Note also that the OPEN protection normally provided by CIO is bypassed (i.e. PUTBYTE to a non-OPEN device/file and PUTBYTE to a read-only OPEN).

GETSTAT

This entry is called in response to a GET STATUS command to CIO. The handler is expected to return four bytes of status to memory or return an error status in the Y register.

At handler entry, the following parameters can be of interest:

X = index to originating IOCB. Y = $92 (status = function not implemented by handler).

ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICBALZ/ICBAHZ [0024/0025] = address of device/filename specification.
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler gets device status information from the device controller and puts the status bytes in DVSTAT [02EA] through DVSTAT+3, and finally returns to CIO with the operation status in register Y.

The IOCB need not be opened nor closed in order for you to request CIO to perform a GET STATUS operation; the handler must check where there are restrictions. See Section 5 for a discussion of the CIO actions involved with a GET STATUS operation using both open and closed IOCB's, and note the impact of this operation on the use of the buffer address parameter.

SPECIAL

This handler entry is used to support all functions not handled by the other entry points, such as diskette file RENAME, display DRAW, etc. Specifically, if the IOCB command byte value is greater than $0D, then CIO will use the SPECIAL entry point. The handler must interrogate the command byte to determine if the requested operation is supported.

At handler entry, the following parameters can be of interest:

X = index to originating IOCB.
Y = $92 (status = function not implemented by handler).

ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICCOMZ [0022] = command byte.
ICBALZ/ICBALH [0024/0025] = buffer address.
ICBLLZ/ICBLHZ [0028/0029] = buffer length.
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler will perform the indicated operation, if possible, and return to CIO with the operation status in register Y.

The IOCB need not be opened nor closed in order for you to request CIO to perform a SPECIAL operation; the handler must check where there are restrictions. See Section 5 for a discussion of the CIO actions involved with a SPECIAL operation using both open and closed IOCB's, and note the impact of this on the use of the buffer address parameter.

Error Handling

Error handling has been simplified somewhat by having CIO handle outer level errors and having SIO handle Serial bus errors, leaving the handler to process the remaining errors. These errors include:

out-of-range parameters.
[BREAK] key abort.
Invalid command.
Read after end of file.

The current handlers respond to errors using the following guidelines:

They keep the recovery simple (and therefore predictable and repeatable).

They Do not interact directly with you for recovery instructions.

They lose as little data as possible.

They make all attempts to maintain the integrity of file oriented device storage -- this involves the initial design of the structural elements as well as error recovery techniques.

Resource Allocation

Nonresident handlers needing code and/or data space in RAM should use the techniques listed below, to assure nonconflict with other parts of the OS, including other nonresident handlers.

Zero-Page RAM

Zero-page RAM has no spare bytes, and even if there were, there is no allocation scheme to support multiple program assignment of the spares. Therefore, the nonresident handler must save and restore the bytes of zero-page RAM it is going to use. The bytes to use must be chosen carefully, according to the following criteria:

The bytes cannot be accessed by an interrupt routine.

The bytes cannot be accessed by any noninterrupt code between the time the handler modifies the bytes and then restores the original values.

A simple save/restore technique would utilize the stack in a manner similar to that shown below:

```
        LDA     COLCRS          ; (for example)
        PHA                     ; SAVE ON STACK.
        LDA     COLCRS+1
        PHA

        LDA     HPOINT          ; HANDLER'S POINTER.
        STA     COLCRS
        LDA     HPOINT+1
        STA     COLCRS+1

        XXX     (COLCRS),Y      ; DO YOUR POINTER THING.

        PLA                     ; RESTORE OLD DATA.
        STA     COLCRS+1
        PLA
        STA     COLCRS
```

Note that the Display Handler or Screen Editor should not be called before restoring the original value of COLCRS, because COLCRS is a variable used by those routines.

Nonzero-Page RAM

There is no allocation scheme to support the assignment of fixed regions of nonzero-page RAM to any specific process, so the handler has three choices:

1. Make a dynamic allocation at initialization time by altering MEMLO [02E7].

2. Include the variables with the handler for RAM-resident handlers. This still involves altering MEMLO at the time the handler is booted.

3. If the handler replaces one of the resident handlers (by removing the resident handler's entry in the device table), then the new handler can use any RAM that the

formerly resident handler would have used.

Stack Space

In most cases, there are no restrictions on the use of the stack by a handler. However, if the handler plans to push more than a couple dozen bytes to the stack; then it should do a stack overflow test, and always leave stack space for interrupt processing.

HANDLER/SIO INTERFACE

This section describes the interface between serial bus device handlers and the serial bus I/O utility (SIO). SIO completely handles all bus transactions following the device-independent bus protocol. SIO is responsible for the following functions:

Bus data format and timing from computer end.

Error detection, retries and statuses.

Bus timeout.

Transfer of data between the bus and the caller's buffer.

Calling Mechanism

SIO has a single entry point SIOV [E459] for all operations. The device control block (DCB) [0300] contains all parameters passed to SIO. The DCB contains the following bytes:

DEVICE BUS ID -- DDEVIC [0300]

The bus ID of the device is set by the handler prior to calling SIO (see Appendix I).

DEVICE UNIT # -- DUNIT [0301]

This byte indicates that of n units of a given device type to access, and is set by the handler prior to calling SIO. This value usually comes from ICDNOZ. SIO accesses the bus device whose address is equal to the value of DDEVIC plus DUNIT minus 1 (the lowest unit number is normally equal to 1).

DEVICE COMMAND -- DCOMND [0302]

The handler sets this byte prior to calling SIO. It will be sent to the bus device as part of the command frame. See Appendix I for device command byte values.

DEVICE STATUS -- DSTATS [0303]

This byte is bidirectional. The handler will use DSTATS to indicate to SIO what to do after the command frame is sent and acknowledged. SIO will use it to indicate to the handler the status of the requested operation.

Prior to an SIO call:

```
 7             0
+-+-+-+-+-+-+-+-+
|W|R|  unused   |
+-+-+-+-+-+-+-+-+
```

Where: W,R = 0,0 indicates no data transfer is associated with the operation.
0,1 indicates a data frame is expected from the device.
1,0 indicates a data frame is to be sent to the device.
1,1 is invalid.

After an SIO call:

```
 7             0
+-+-+-+-+-+-+-+-+
|  status code  |
+-+-+-+-+-+-+-+-+
```

See Appendix C for the possible SIO operation status codes.

HANDLER BUFFER ADDRESS -- DBUFLO/DBUFHI [0304/0305]

The handler sets this 2-byte pointer. It indicates the source or destination buffer for device data or status information.

DEVICE TIMEOUT -- DTIMLO [0306]

The handler sets this byte. It specifies the device timeout time in units of 64/60 of a second. For example, a count of 6 specifies a timeout of 6.4 seconds.

BUFFER LENGTH/BYTE COUNT -- DBYTLO/DBYTHI [0308/0309]

The handler sets this 2-byte count for the current operation, and indicates the number of data bytes to be transferred into or out of the buffer. This parameter is not required if the STATUS byte W and R bits are both zero. These values indicate that no data transfer is to take place.

> WARNING: There is a bug in SIO that causes incorrect actions when the last byte of a buffer is in a memory address ending in $FF, such as 13FF, 42FF, etc.

AUXILIARY INFORMATION -- DAUX1/DAUX2 [030A/030B]

The handler sets these 2-bytes. The SIO includes them in the bus command frame; they have device-specific meanings.

Functions Supported

SIO does not examine the COMMAND byte it sends to the device, because all bus transactions are expected to conform to a universal protocol. The protocol includes three forms, stated below (as seen from the computer):

Send command frame.

Send command frame and send data frame.

Send command frame and receive data frame.

The values of the W and R bits in the status byte select the command form.

Error Handling

SIO handles most of the serial bus errors for the handler, as indicated below:

Bus timeout -- SIO provides a uniform command frame and data frame ACK byte timeout of 1/60 of a second - 0 / + 1/60. The handler specifies the maximum COMPLETE byte timeout value in DTIMLO.

Bus errors -- SIO detects and reports UART overrun and framing errors. The sensing of these errors in any received byte will cause the entire associated frame to be considered bad.

Data frame checksum error -- SIO validates the checksum on all received data frames and generates a checksum for all transmitted frames.

Invalid response from device -- In addition to the error conditions stated above, SIO checks that the ACK and COMPLETE responses are proper (ACK = $41 and COMPLETE = $43). ACK stands for acknowledge.

Bus operation retries -- SIO will attempt one complete command retry if the first attempt is not error free, where a complete command try consists of up to 14 attempts to send (and acknowledge) a command frame, followed by a single attempt to receive the COMPLETE code and possibly a data frame.

> NOTE: There is a bug in the retry logic for data writes, such that if the command frame is acknowledged by the controller, but the data frame is not acknowledged, then SIO will retry indefinitely.

Unified error status codes -- SIO provides device-independent error codes (see Appendix C).

SERIAL I/O BUS CHARACTERISTICS AND PROTOCOL

This section describes:

- o The electrical characteristics of the ATARI 400 and ATARI 800 Home Computers serial bus
- o The use of the bus to send bytes of data,
- o The organization of the bytes as "frames" (records),
- o The overall command sequences that utilize frames and response bytes to provide computer/peripheral communication.

Hardware/Electrical Characteristics

The ATARI 400 and the ATARI 800 Home Computers communicate with peripheral devices over a 19,200 baud asynchronous serial port. The serial port consists of a serial DATA OUT (transmission) line, a serial DATA IN (receiver) line and other miscellaneous control lines.

Data is transmitted and received as 8 bits of serial data (LSB sent first) preceded by a logic zero start bit and succeeded by a logic one stop bit. The serial DATA OUT is transmitted as positive logic (+4v = one/true/high, 0v = zero/false/low). The serial DATA OUT line always assumes its new state when the serial CLOCK OUT line goes high; CLOCK OUT then goes low in the center of the DATA OUT bit time.

An end view of the Serial bus connector at the computer or peripheral is shown below (the cable connectors would of course be a mirror image):

```
           2   4   6   8  10  12
           o   o   o   o   o   o

         o   o   o   o   o   o   o
         1   3   5   7   9  11  13
```

```
where:  1 = computer CLOCK IN.
        2 = computer CLOCK OUT.
        3 = computer DATA IN.
        4 = GND.
        5 = computer DATA OUT.
        6 = GND.
        7 = COMMAND-.
        8 = MOTOR CONTROL.
        9 = PROCEED-.
       10 = +5v/READY.
       11 = computer AUDIO IN.
       12 = +12v.
       13 = INTERRUPT-.
```

Figure 9-4   Serial Bus Connector Pin Descriptions

CLOCK IN is not used by the present OS and peripherals. This line can be used in future synchronous communications schemes.

CLOCK OUT is the serial bus clock. CLOCK OUT goes high at the start of each DATA OUT bit and returns to low in the middle of each bit.

DATA IN is the serial bus data line to the computer.

Pin 4 GND is the signal/shield ground line.

DATA OUT is the serial bus data line from the computer.

Pin 6 GND is the signal/shield ground line.

COMMAND- is normally high and goes low when a command frame is being sent from the computer.

MOTOR CONTROL is the cassette motor control line (high=on, low= off).

PROCEED- is not used by the present OS and peripherals; this line is pulled high.

+5v/READY indicates that the computer is turned on and ready. This line can also be used as a +5 volt supply of 50ma current rating for ATARI peripherals only.

AUDIO IN accepts an audio signal from the cassette.

+12V is a +12 volt supply of unknown current rating for ATARI peripherals only.

INTERRUPT- is not used by the present OS and peripherals; this line is pulled high.

There are no pin reassignments made in the Serial bus cable, so pin 3, the computer's DATA IN line, is the peripheral's data output line; and similarly for pin 5.

Serial Port Electrical Specifications

Peripheral input:

VIH = 2.0v min.
VIL = 0.4v max.

IIH = 20ua. max. @ VIH = 2.0v
IIL = 5ua. max. @ VIL = .4v

Peripheral output (open collector bipolar):

VOL = 0.4v max. @ 1.6 ma.
VOH = 4.5v min. with external 100Kohm pull-up.

Vcc/READY input:

VIH = 2.0v min. @ IIH = 1ma. max.
VIL = 0.4v max.
Input goes to logic zero when open.

Bus Commands

The bus protocol specifies that all commands must originate from the computer, and that peripherals will present data on the bus only when commanded to. Every bus operation will go to completion before another bus operation is initiated (no overlap). An error detected at any point in the command sequence will abort the entire sequence.

A bus operation consists of the following elements:

Command frame from the computer.

Acknowledgement (ACK) from the peripheral.

Optional data frame to or from the computer.

Operation complete (COMPLETE) from the peripheral.

Command Frame

The serial bus protocol provides for three types of commands: 1) data send, 2) data receive and 3) immediate (no data -- command only). There is a common element in all three types, a command frame consisting of five bytes of information sent from the computer while the COMMAND- line is held low. The format of the command frame is
shown below:

```
+----------------+
|  device ID     |
+----------------+
|    command     |
+----------------+
| auxiliary #1   |
+----------------+
| auxiliary #2   |
+----------------+
|    checksum    |
+----------------+
```

Figure 9-5   Serial Bus Command Frame Format

The device ID specifies that of the serial bus devices is being addressed (see Appendix I for a list of device IDs).

The command byte contains a device-dependent command (see Appendix I for a list of device commands).

The auxiliary bytes contain more device-dependent information.

The checksum byte contains the arithmetic sum of the first four bytes (with the carry added back after every addition).

Command Frame Acknowledge

The peripheral being addressed would normally respond to a command frame by sending an ACK byte ($41) to the computer; if there is a problem with the command frame, the peripheral should not respond.

Data Frame

Following the command frame (and ACK) can be an optional data frame that is formatted as shown below:

```
+----------------+
|                |
|                |
|      data      |
=                =
|      bytes     |
|                |
+----------------+
|    checksum    |
+----------------+
```

This data frame can originate at the computer or at the device controller, depending upon the command. Current device controllers expect fixed-length data frames as does the computer, where the data frame length is a fixed function of the device type and command.

The checksum value in the data frame is the arithmetic sum of all of the frame data preceding the checksum, with the carry from each addition being added back (the same as for the command frame).

In the case of the computer sending a data frame to a peripheral, the peripheral is expected to send an ACK if the data frame is acceptable, and send a NAK ($4E), or do nothing if the data frame is unacceptable. See the first flowchart in Section 9.

Operation Complete

A peripheral is also expected to send an operation-COMPLETE byte ($43) at the time the commanded operation is complete. The location of this byte in the command sequence for each command type is shown in the timing diagrams in Section 9. If the operation cannot go to normal, error-free completion, the peripheral should respond with an ERROR byte ($45) instead of COMPLETE.

Bus Timing

This section provides timing diagrams for the three types of command sequences: data send, data receive, and immediate.

DATA SEND sequence:

```
              ---+        +------------------------------------
COMMAND-         !        !
                 +--------+


                  +------+          +---//---+
DATA OUT          ! cmnd !          !  data  !
              ----+frame +------//--+ frame  +-----------------


                             +-+                 +-+      +-+
DATA IN                      ! !                 ! !      ! !
              ---------------+ +-----------------+ +--//--+ +-
                             ACK                 ACK      CMPL

                 !!      !!  ! !    !         !  ! !      !

                 t0     t1 t2    t3            t4     t5
```

DATA RECEIVE sequence:

```
              ---+        +------------------------------------
COMMAND-         !        !
                 +--------+


                  +------+
DATA OUT          ! cmnd !
              ----+frame +-------------------------------------


                             +-+      +-+ +----//-----+
DATA IN                      ! !      ! ! !   data   !
              ---------------+ +--//--+ +-+  frame   +---------
                             ACK      CMPL

                 !!      !!  ! !      !

                 t0     t1 t2    t5
```

IMMEDIATE sequence:

```
             ---+        +------------------------------------
COMMAND-        !        !
                +--------+


                 +------+
DATA OUT         ! cmnd !
             ----+frame +-------------------------------------


                           +-+                  +-+
DATA IN                    ! !                  ! !
             --------------+ +---------//-------+ +----------
                           ACK                  CMPL

                !!      !! ! !                  !

                t0      t1 t2          t5
```

Figure 9-6   Serial Bus Timing Diagram

The computer generates a delay (t0) between the lowering of COMMAND- and the transmission of the first byte of the command frame.

computer t0 (min) = 750 microsec.
computer t0 (max) = 1600 microsec.

peripheral t0 (min) = ??
peripheral t0 (max) = ??

The computer generates a delay (t1) between the transmission of the last bit of the command frame and the raising of the COMMAND- line.

computer t1 (min) = 650 microsec.
computer t1 (max) = 950 microsec.

peripheral t1 (min) = ??
peripheral t1 (max) = ??

The peripheral generates a delay (t2) between the raising of COMMAND- and the transmission of the ACK byte by the peripheral.

computer t2 (min) = 0 microsec.
computer t2 (max) = 16 msec.

peripheral t2 (min) = ??
peripheral t2 (max) = ??

The computer generates a delay (t3) between the receipt of the last bit of the ACK byte and the transmission of the first bit of the data frame by the computer.

computer t3 (min) = 1000 microsec.
computer t3 (max) = 1800 microsec.

peripheral t3 (min) = ??
peripheral t3 (max) = ??

The peripheral generates a delay (t4) between the transmission of the last bit of the data frame and the receipt of the first bit of the ACK byte by the computer.

computer t4 (min) = 850 microsec.
computer t4 (max) = 16 msec.

peripheral t4 (min) = ??
peripheral t4 (max) = ??

The Peripheral generates a delay (t5) between the the receipt of the last bit of the ACK byte and the first bit of the COMPLETE byte by the computer.

computer t5 (min) = 250 microsec.
computer t5 (max) = 255 sec. (handler-dependent)

peripheral t5 (min) = ??
peripheral t5 (max) = N/A

HANDLER ENVIRONMENT

Nonresident handlers can be installed in at least three different manners:

1. As booted software from diskette or cassette.

2. Resident in a cartridge (A or B).

3. Downloaded from a serial bus device.

This section will discuss the basic mechanisms for handler installation for these environments. In order to fully utilize the information in this section, you must have read and understood the following sections:

Program environments . . . . Section 3
RAM region . . . . Section 4
Memory dynamics. . . . . Section 4
System initialization. . . . . Section 7
Adding new device handlers/peripherals . Section 9
Program environment and initialization . Section 10

Bootable Handler

The diskette- or cassette-booted software will insert the handler's vector table pointer and name to the device table whenever the booted software's initialization entry point is entered (on power-up and system reset). Remember that both power-up and system reset clear the device table of all but the resident handler entries.

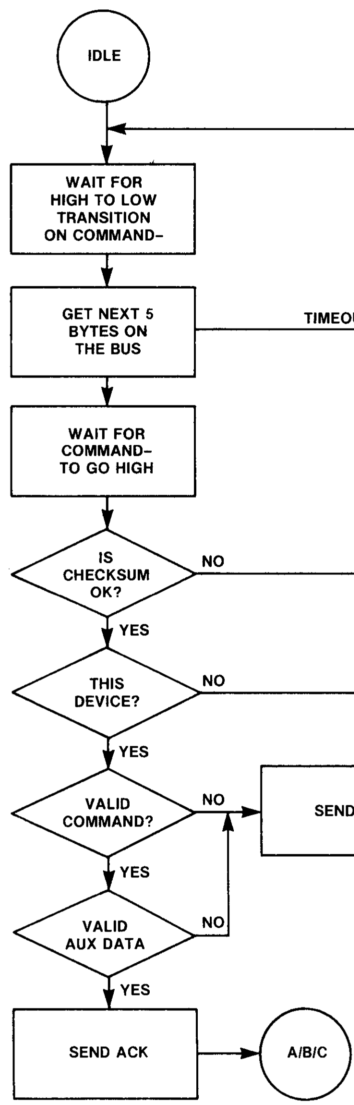
Cartridge Resident Handler

The cartridge software will insert the handler's vector table pointer and name to the device table whenever the cartridge's initialization entry point is entered (on power-up and system reset). Remember that both power-up and system reset clear the device table of all but the resident handler entries; therefore the device table must be reestablished by the handler-initialization procedure upon every entry.

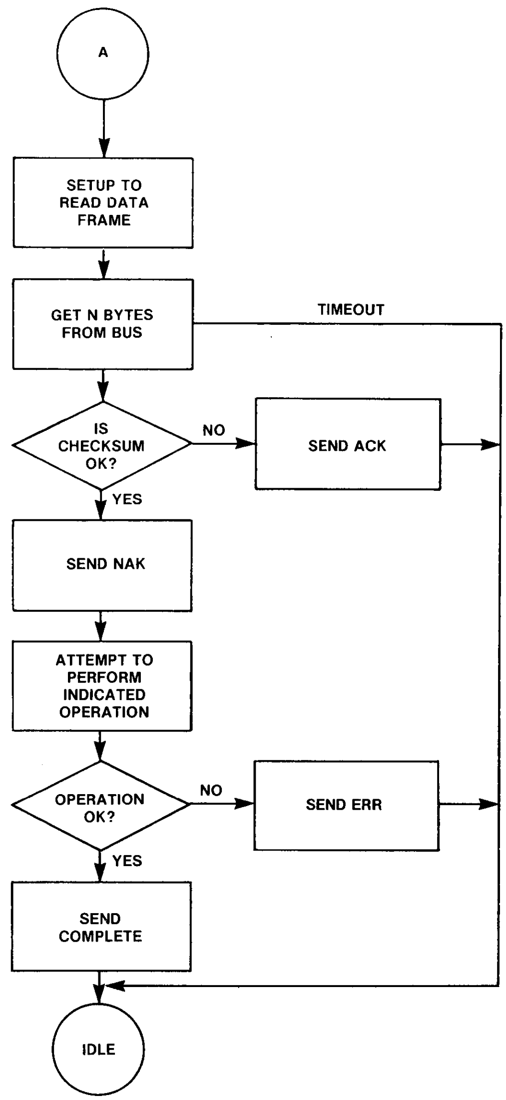
FLOWCHARTS

The following pages contain process flowcharts showing the SIO and peripheral actions for the Serial bus command forms.

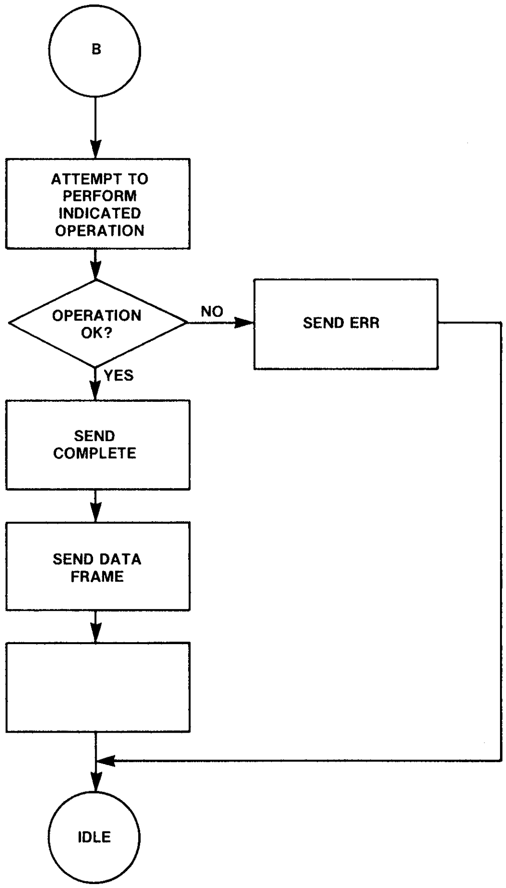
PERIPHERAL'S COMMAND FRAME PROCESSING



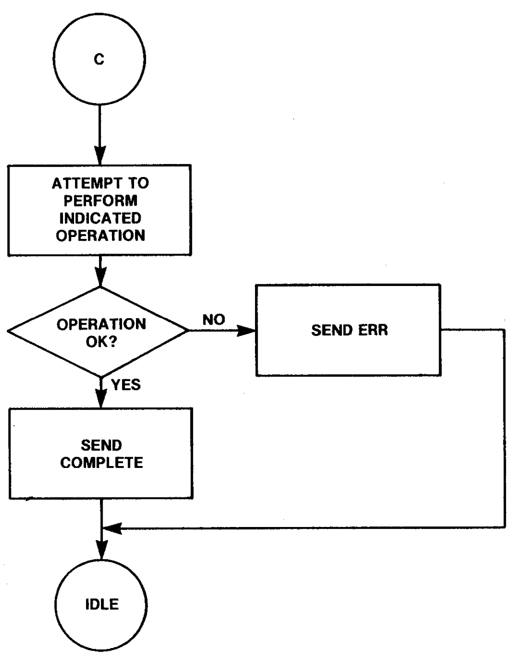
DATA FRAME TO PERIPHERAL



DATA FRAME TO COMPUTER



IMMEDIATE



10 PROGRAM ENVIRONMENT AND INITIALIZATION

This section discusses possible alternative software environments using OS Configurations. Environments other than those discussed here are also possible. A thorough understanding of the power-up and system reset processes (see Section 7) will be necessary to evaluate all alternative environments.

CARTRIDGE

Most games (and some language processors) are supported via the cartridge environment. The cartridge resident software is in control of the system, sometimes using the OS and sometimes not. A cartridge can specify whether the diskette is to be booted at power-up time, whether the cartridge is to provide the controlling software, or whether the cartridge is a special diagnostic cartridge. These options are specified by bits in the cartridge header, as shown below:

```
+----------------+
!   cartridge    !   BFFA (9FFA for cartridge B)
+-              -+
! start address  !
+----------------+
!       00       !
+----------------+
!  option byte   !
+----------------+
!   cartridge    !
+-              -+
!  init address  !   BFFF (9FFF for cartridge B)
+----------------+
```

Figure 10-1   Cartridge Header Format

The byte of "00" is used to allow the OS to determine when a cartridge is inserted; locations BFFC and 9FFC will not read zero when there is neither RAM at those locations nor a cartridge inserted. RAM is differentiated from a cartridge by its ability to be altered.

The option byte has the following option bits:

```
bit 0 = 0, then do not boot the diskette.
        1, then boot the diskette.

Bit 2 = 0, then init but do not start the cartridge.
        1, then init and start the cartridge.

bit 7 = 0, then cartridge is not a diagnostic cartridge.
        1, then cartridge is a diagnostic cartridge and control
           will be given to the cartridge before any of the OS
           is initialized (JMP (BFFE)).
```

The cartridge init address specifies the location to which the OS will JSR during all power-up and system reset operations. As a minimum, this vector should point to an RTS instruction.

The cartridge start address specifies the location to which the OS will JMP during all power-up and system reset operations, if bit 1 of the option byte is = 1. The application should examine the variable WARMST [0008] if system reset action is to be different than power-up (WARMST will be zero on power-up and nonzero thereafter).

Cartridge Without Booted Support Package

A cartridge that does not specify the diskette-boot option and does not support the cassette-boot possibility can use lower memory (from 0480 to the address in MEMTOP [02E5]) in any way it sees fit.

Cartridge With Booted Support Package

A cartridge that does specify the diskette-boot option or does support the cassette-boot possibility must use some care in its use of lower memory. The following regions are defined:

```
0480-06FF  is always available to the cartridge.
MEMLO/MEMTOP region is always available to the cartridge.
```

DISKETTE-BOOTED SOFTWARE

Software can be booted from the disk drive at power-up time in response to one of the following conditions:

Neither Cartridge A nor B is inserted.

Cartridge A is inserted and has bit 0 of its option byte [BFFD] = 1.

Cartridge B is inserted and has bit 0 of its option byte [9FFD] = 1.

If any of these conditions are met, the OS will attempt to read the boot record from sector #1 of disk drive 1 and then transfer control to the software that was read in. The exact sequence of operations will be explained later in this section.

Diskette-Boot File Format

The key region of a diskette-boot file is the first six bytes, which are formatted as shown below:

```
+----------------+
|     flags      |  first byte
+----------------+
|  # of sectors  |
+----------------+
| memory address |
+-              -+
| to start load  |
+----------------+
|      init      |
+-              -+
|    address     |  sixth byte
+----------------+
|      boot      |
|  continuation  |
|      code      |
```

Figure 10-2   Diskette-Boot File Format

The first byte is stored in DFLAGS [0240], but is otherwise unused. It should equal zero.

The second byte contains the number of 128-byte diskette sectors to be read as part of the boot process (including the record containing this information). This number can range from 1 to 255, with 0 meaning 256.

The third and fourth bytes contain the address (lo,hi) at which to start loading the first byte of the file.

The fifth and sixth bytes contain the address (lo,hi) to which the booter will transfer control after the boot process is complete and whenever the [SYSTEM.RESET] key is pressed.

The Diskette File Management System (FMS) has extra bytes assigned to its boot record, but this is a special case of the generalized diskette-boot and is discussed in Section 5.

Diskette-Boot Process

If no cartridge is installed, then the diskette will follow these steps to boot up:

1. Read the first diskette record to the cassette buffer [0400].

2. Extract information from the first six bytes:

   Save the flags byte to DFLAGS [0240,1]. Save the # of sectors to boot to DBSECT [0241,1]. Save the load address to BOOTAD [0242,2]. Save the initialization address in DOSINI [000C,2].

3. Move the record just read to the load address specified.

4. Read the remaining records directly to the load area.

5. JSR to the load address+6 where a multistage boot process can continue. The carry bit indicates the success of this operation (carry set = error, carry reset = success).

   NOTE: During step 5, after the initial boot process is complete, the booter will transfer control to the seventh byte of the first record. The software should continue the boot process at this point, if it is a multistage boot. The value of MEMLO [02E7] should point to the first free RAM location beyond the software just booted. It should be established by the booted software as shown below:

```
        LDA     #END+1          ; SET UP LSB.
        STA     MEMLO
        STA     APPMHI
        LDA     #END+1/256      ; SET UP MSB.
        STA     MEMLO+1
        STA     APPMHI+1
```

   If the booted software is to take control of the system at the end of the boot operation, the vector DOSVEC [000A] must be set up by the application at this time; DOSVEC points to the restart entry for the booted application. If the booted software is not to take control, then DOSVEC should remain unchanged.

```
        LDA     #RESTRT         ; RESTART LSB.
        STA     DOSVEC
        LDA     #RESTRT/256
        STA     DOSVEC+1
```

6. JSR indirectly through DOSINI for initialization of the application; the application will initialize and return.

   NOTE: The OS enters the initialization point on every system reset and power-up. Internal initialization can take place during system reset and power-up as well. Initialization can also be deferred until Step 7 for controlling applications.

7. JMP indirectly through DOSVEC to transfer control to the application.

   NOTE: Pressing the [SYSTEM.RESET] key after the application is fully booted will cause steps 6 and 7 to be repeated.

Sample Diskette-Bootable Program Listing

This skeletal program can be booted from the diskette. It retains control when it is entered.

```
; THIS IS THE START OF THE PROGRAM FILE.

PST=    $0700                   ; (OR SOME OTHER LOCATION).
*=      PST                     ; (.ORG).

; THIS IS THE diskette-boot CONTROL INFORMATION.

        .BYTE   0               ;
        .BYTE   PND-PST+127/128 ; NUMBER OF RECORDS.
        .WORD   PST             ; MEMORY ADDRESS TO START LOAD.
        .WORD   PINIT           ; PROGRAM INIT.
```

```
; THIS IS THE START OF THE BOOT CONTINUATION.

        LDA     #PND                ; ESTABLISH LOW MEMORY LIMITS.
        STA     MEMLO
        STA     APPMHI
        LDA     #PND/256
        STA     MEMLO+1
        STA     APPMHI+1

        LDA     #RESTRT             ; ESTABLISH RESTART VECTOR.
        STA     DOSVEC
        LDA     #RESTRT/256
        STA     DOSVEC+1

        CLC                         ; SET FLAG FOR SUCCESSFUL BOOT.
        RTS


; APPLICATION INITIALIZATION ENTRY POINT.

PINIT   RTS                         ; NOTHING TO DO HERE FOR ...
                                    ; ... CONTROLLING APPLICATION.

; THE MAIN BODY OF THE PROGRAM FOLLOWS.

RESTRT=*

; THE MAIN BODY OF THE PROGRAM ENDS HERE.

PND=    *                           ; 'PND' = NEXT FREE LOCATION.
        .END
```

Figure 10-3   Diskette-Bootable Program Listing Example

Program to Create Diskette-Boot Files

This section provides a program that can be used to make bootable files on diskettes. The program given is not the only one possible, and no claims are made as to its elegance.

Shown below is a listing of the program to create diskette-boot files.

```
; THIS PROGRAM WRITES A SINGLE "FILE" TO THE DISKETTE AND IS
; USED IN CONJUNCTION WITH A PROCEDURE TO MAKE DISKETTE-
; BOOTABLE FILES.  THE FOLLOWING TWO SYMBOLS MUST BE EQUATED
; USING THE MEMORY LIMITS OF THE PROGRAM TO BE COPIED:
;
;      'PST' = PROGRAM START ADDRESS (SEE SAMPLE PROGRAM).
;      'PND' = PROGRAM END ADDRESS (SEE SAMPLE PROGRAM).


SECSIZ=128                      ; DISKETTE SECTOR SIZE.
PST=   $0700
PND=   $1324
FLEN=  PND-PST+SECSIZ-1/SECSIZ ; # OF SECTORS IN FILE.

*=     $B000                    ; THIS PROGRAM'S ORIGIN.




BOOTB  BRK                      ; *** LOAD APPLICATION ***

; SET UP DEVICE CONTROL BLOCK FOR DISKETTE HANDLER CALL

       LDA    #FLEN             ; # OF SECTORS TO WRITE.
       STA    COUNT

       LDA    #1                ; DISK DRIVE #1.
       STA    DUNIT

       LDA    #'W               ; SET UP FOR WRITE WITH CHECK.
       STA    DCOMND

       LDA    #PST              ; POINT TO START OF APPLIC. PROG.
       STA    DBUFLO
       LDA    #PST/256
       STA    DBUFHI

       LDA    #01               ; SET UP STARTING SECTOR # = 0001.
       STA    DAUX1
       LDA    #00
       STA    DAUX2
```

```
; NOW WRITE THE FILE ONE SECTOR AT A TIME.

BOTO10 JSR     DSKINV            ; WRITE ONE SECTOR.
       BMI     DERR              ; ERROR.

       LDA     DBUFLO            ; INCREMENT MEMORY ADDRESS.
       CLC
       ADC     #SECSIZ
       STA     DBUFLO
       LDA     DBUFHI
       ADC     #0
       STA     DBUFHI

       INC     DAUX1             ; INCREMENT SECTOR #.
       BNE     BOTO20
       INC     DAUX2

BOTO20 DEC     COUNT             ; MORE SECTORS TO WRITE?
       BNE     BOTO10            ; YES.

       BRK                       ; STOP WHEN DONE.

DERR   BRK                       ; STOP ON ERROR.

COUNT  *=*+1                     ; SECTOR COUNT.

; THIS IS THE CARTRIDGE HEADER

*=     $BFF9                     ; "A" CARTRIDGE.

INIT   RTS
       .WORD   BOOTB
       .BYTE   0,4
       .WORD   INIT

       .END
```

CASSETTE-BOOTED SOFTWARE

You can boot software from the cassette as well as from the diskette, at power-up. The following requirements must be met in order to boot from the cassette:

- o You must be pressing the [START] key as power is applied to the system.

- o A cassette tape with a proper boot format file must be installed in the cassette drive, and the PLAY button must be pressed.

o When you are given the audio prompt by the cassette handler you must press the [RETURN] key.

If all of these conditions are met, the OS will read the boot file from the cassette and then transfer control to the software that was read in. The exact sequence of operations will be explained later in this section.

Cassette-Boot File Format

The key region of a cassette-boot file is the first six bytes, that are formatted as shown below:

```
+----------------+
!                !
+----------------+
! # of Records   !
+----------------+
! Memory Address !
+--            --+
! To Start Load  !
+----------------+
!      Init      !
+--            --+
!    address     !
+----------------+
```

The first byte is not used by the cassette-boot process.

The second byte contains the number of 128-byte cassette records to be read as part of the boot process (including the record containing this information). This number can range from 1 to 255, with 0 meaning 256.

The third and fourth bytes contain the address (lo,hi) to which the booter will transfer control after the boot process is complete and whenever the [SYSTEM.RESET] key is pressed.

Cassette-Boot Process

The cassette-boot process is described step-by-step for a configuration in that no cartridge is installed and no diskettes are attached. For the general case see Section 7.

1. Read the first cassette record to the cassette buffer.

2. Extract information from the first six bytes:

Save the # of records to boot. Save the load address. Save the initialization address in CASINI [0002]

3. Move the record just read to the load address specified.

4. Read the remaining records directly to the load area.

5. JSR to the load address+6 where a multistage boot process can continue; the carry bit will indicate the success of this operation (carry set=error, carry reset=success).

6. JSR indirectly through CASINI for initialization of the application; the application will initialize and return.

7. JMP indirectly through DOSVEC to transfer control to the application.

Pressing the [SYSTEM.RESET] key after the application is fully booted will cause steps 6 and 7 to be repeated.

NOTE: After the initial boot process is complete, the booter will transfer control to the seventh byte of the first record; at this point the software should continue the boot process (if it is a multistage boot) and then stop the cassette drive, which due to a system bug will still be running, using the following instruction sequence:

```
        LDA     #$3C
        STA     PACTL [D302]
```

The application should then set a value in MEMLO [0237] that points to the first free RAM location beyond the software just booted, as shown below:

```
        LDA     #END+1
        STA     MEMLO
        STA     APPMHI
        LDA     #END+1/256
        STA     MEMLO+1
        STA     APPMHI+1
```

If the booted software is to take control of the system at the end of the boot operation, the vector DOSVEC [000A] must be set up by the application at this time; DOSVEC points to the restart entry for the booted application. If the booted software is not to take control, then DOSVEC should remain unchanged.

```
    LDA     #RESTRT             ; RESTART LSB
    STA     DOSVEC
    LDA     #RESTRT/256
    STA     DOSVEC+1
```

NOTE: The initialization point is entered on every system reset and power-up; internal initialization can take place here.

For controlling applications initialization can also be deferred until step 7.

Sample Cassette-Bootable Program Listing

Shown below is a skeletal program that can be booted from the cassette and that retains control when it is entered.

```
; THIS IS THE START OF THE PROGRAM FILE.

PST=    $0700                   ; (OR SOME OTHER LOCATION).
*=      PST                     ; (.ORG).


; THIS IS THE cassette-boot CONTROL INFORMATION.

        .BYTE   0               ; (DOESN'T MATTER).
        .BYTE   PND-PST+127/128 ; NUMBER OF RECORDS.
        .WORD   PST             ; MEMORY ADDRESS TO START LOAD.
        .WORD   PINIT           ; PROGRAM INIT.

; THIS IS THE START OF THE BOOT CONTINUATION.

        LDA     #$3C            ; STOP THE CASSETTE.
        STA     PACTL

        LDA     #PND            ; ESTABLISH LOW MEMORY LIMITS.
        STA     MEMLO
        STA     APPMHI
        LDA     #PND/256
        STA     MEMLO+1
        STA     APPMHI+1

        LDA     #RESTRT         ; ESTABLISH RESTART VECTOR.
        STA     DOSVEC
        LDA     #RESTRT/256
        STA     DOSVEC+1

        CLC                     ; SET FLAG FOR SUCCESSFUL BOOT.
        RTS

; APPLICATION INITIALIZATION ENTRY POINT.

PINIT   RTS                     ; NOTHING TO DO HERE FOR ...
                                ; ... CONTROLLING APPLICATION.

; THE MAIN BODY OF THE PROGRAM FOLLOWS.

RESTRT=*

; THE MAIN BODY OF THE PROGRAM ENDS HERE.
```

```
PND=    *                          ; 'PND' = NEXT FREE LOCATION.
        .END
```

Figure 10-4   Sample Cassette-Bootable Program

Program to Create Cassette-Boot Files

This section provides a program listing that can be used to make bootable files on cassette tapes. The program given is not the only one possible, and no claims are made as to its elegance.

Shown below is a listing of the program to create a cassette-boot file:

```
;   THIS PROGRAM WRITES A SINGLE FILE TO THE CASSETTE AND IS
;   USED IN CONJUNCTION WITH A PROCEDURE TO MAKE CASSETTE-
;   BOOTABLE FILES. THE FOLLOWING TWO SYMBOLS MUST BE EQUATED
;   USING THE MEMORY LIMITS OF THE PROGRAM TO BE COPIED:

;       'PST' = PROGRAM START ADDRESS (SEE SAMPLE PROGRAM).
;       'PND' = PROGRAM END ADDRESS (SEE SAMPLE PROGRAM).

PST=    $0700
PND=    $1324
FLEN=   PND-PST+127/128*128     ; ROUND UP TO MULTIPLE OF 128.

*=      $B000                   ; THIS PROGRAM'S ORIGIN.

BOOTB   LDX     #$10            ; USE IOCB #1.

; FIRST OPEN THE CASSETTE FILE FOR WRITING.

        LDA     #OPEN           ; SET UP FOR DEVICE "OPEN."
        STA     ICCOM,X

        LDA     #OPNOT          ; DIRECTION IS "OUTPUT."
        STA     ICAX1,X
        LDA     #$80            ; SELECT SHORT IRG.
        STA     ICAX2,X

        LDA     #CFILE          ; SET UP POINTER TO DEVICE NAME.
        STA     ICBAL,X
        LDA     #CFILE/256
        STA     ICBAH,X

        JSR     CIOV            ; ATTEMPT TO OPEN FILE.
        BMI     CERR            ; ERROR.

; NOW WRITE THE ENTIRE FILE AS ONE OPERATION.
```

```
        LDA     #PUTCHR         ; SET UP FOR "PUT CHARACTERS."
        STA     ICCOM,X

        LDA     #PST            ; POINT TO START OF APPLIC. PROG.
        STA     ICBAL,X
        LDA     #PST/256
        STA     ICBAH,X

        LDA     #FLEN           ; SET UP # OF BYTES TO WRITE.
        STA     ICBLL,X
        LDA     #FLEN/256
        STA     ICBLH,X

        JSR     CIOV            ; WRITE ENTIRE FILE.
        BMI     CERR            ; ERROR.

; NOW CLOSE THE FILE AFTER SUCCESSFUL WRITE.

        LDA     #CLOSE          ; SET UP FOR "CLOSE."
        STA     ICCOM,X

        JSR     CIOV            ; CLOSE THE FILE.
        BMI     CERR            ; ERROR.

        BRK                     ; STOP WHEN DONE.

CERR    BRK                     ; STOP ON ERROR.

CFILE   .BYTE   "C:",CR         ; FILE NAME.

; THIS IS THE CARTRIDGE HEADER

*=      $BFF9                   ; "A" CARTRIDGE.

INIT    RTS
        .WORD   BOOTB
        .BYTE   0,4

        .WORD   INIT
        .END
```

11 ADVANCED TECHNIQUES AND APPLICATION NOTES

This section presents information to use the capabilities of the OS and some of the hardware capabilites that aren't directly available through the OS ,and in fact, can be in direct conflict with parts of the OS.

SOUND GENERATION

The OS uses the POKEY sound generation capabilities only in the I/O subsystem, for cassette FSK tone generation, and for the "noisy bus" option in SIO.

Capabilities

The hardware provides four independently programmable audio channels that are mixed and sent to the television set as part of the composite video signal. The POKEY registers shown below are all concerned with sound control (as described in the ATARI Home Computer Hardware Manual).

AUDCTL [D208]	Audio control.
AUDC1 [D201] and AUDF1 [D200]	Channel 1 control.
AUDC2 [D203] and AUDF2 [D202]	Channel 2 control.
AUDC3 [D205] and AUDF3 [D204]	Channel 3 control.
AUDC4 [D207] and AUDF4 [D206]	Channel 4 control.

Conflicts With OS

There are two potential conflicts with the OS involving sound generation:

- o The OS can generate its own sounds and then turn off all sounds as part of I/O operations to the cassette and the serial bus peripherals.

- o The OS does not turn off sounds when you press [SYSTEM.RESET] or [BREAK]. If the sounds are to be turned off under those conditions, the controlling program must provide that capability.

SCREEN GRAPHICS

Hardware Capabilities

The hardware capabilities for screen presentations are quite versatile; the OS uses a very small amount of the capability provided. The means of extension, however, are non-trivial; and making changes to a screen format while still utilizing the resident Display Handler will be difficult. See the ATARI Home Computer Hardware Manual for information regarding screen presentations.

OS Capabilities

The resident Display Handler arbitrarily supports 8 of the 11 possible full screen modes (11 of 14 modes if the GTIA chip is used in place of the CTIA). The resident Display Handler allows for an optional "split-screen" text window of fixed size. The hardware allows for many more options than the Display Handler supports, as will be seen by reading the ATARI Home Computer Hardware Manual.

Cursor Control

You can control the Display Handler text and graphics cursors directly (see Section 5 and Appendix L, B1-4).

Color Control

You can alter the color register assignments that the Display Handler makes upon all OPEN commands (see Appendix L B7-8 and elsewhere). Note that every system reset or Display Handler OPEN will reset the values back to the system default.

Alternate Character Sets

Two character sets are available in screen text modes 1 and 2. The value stored in the data base variable CHBAS [02F4] selects the character set of interest to you. The default value of $E0 provides capital (uppercase) letters, numbers and the punctuation characters corresponding to display codes $20 through $5F in Appendix E). The alternate value of $E2 provides lowercase letters and the special character graphics set (corresponding to display codes $60 through $7F and $00 through $1F in Appendix E).

User-defined character sets can also be obtained for text modes 0, 1, and 2 by providing the character matrix definitions in RAM and setting CHBAS to point to those definitions. CHBAS always contains the most significant bits of the memory address of the start of the character definitions, as shown below:

```
           7             0
          +-+-+-+-+-+-+-+-+
CHBAS     !    MSB     !x x!   Text mode 0
          +-+-+-+-+-+-+-+-+

          +-+-+-+-+-+-+-+-+
          !    MSB       !x!   Text modes 1 and 2
          +-+-+-+-+-+-+-+-+
```

Figure 11-1   User-Defined Character Set Bit Memory Addresses

(X indicates an ignored address bit assumed to be 0.)

Each character is defined by an 8 x 8 bit matrix; the character '@' is defined as shown below:

```
         7            0
  Byte +-+-+-+-+-+-+-+-+
       !0!0!0!0!0!0!0!0!    0
       +-+-+-+-+-+-+-+-+
       !0!0!1!1!1!1!0!0!    1
       +-+-+-+-+-+-+-+-+
       !0!1!1!0!1!1!0!0!    2
       +-+-+-+-+-+-+-+-+
       !0!1!1!0!1!1!0!0!    3
       +-+-+-+-+-+-+-+-+
       !0!1!1!0!1!1!0!0!    4
       +-+-+-+-+-+-+-+-+
       !0!1!1!0!0!0!0!0!    5
       +-+-+-+-+-+-+-+-+
       !0!0!1!1!1!1!0!0!    6
       +-+-+-+-+-+-+-+-+
       !0!0!0!0!0!0!0!0!    7
       +-+-+-+-+-+-+-+-+
```

Figure 11-2   User Defined 8 x 8 Character Matrix Bit Table

The storage for the character set involves eight consecutive bytes for each character with characters ordered consecutively by their internal code value (see the discussion in Appendix L relating to B55).

```
                     +---------------+
   Character base    ! Character for !
                     !  code $00     ! 8 bytes
                     +---------------+
                     ! Character for !
                     !  code $01     ! increasing addresses
                     +---------------+
                     !               !
                     =               =
                     !               !
                     +---------------+
                     ! Character for !
                     !  code $7E     !
                     +---------------+
                     ! Character for !
                     !  code $7F     !
                     +---------------+
```

Figure 11-3   Character Base Diagram

PLAYER/MISSILE GRAPHICS

The OS makes no use of the player/missile generation capability of the hardware. It can be used independently of the OS with no conflict.

Hardware Capabilities

The hardware allows a number of independently moveable screen objects of limited width to be positioned and moved about the screen without affecting the "playfield" (bit-mapped graphics or character) data. Priority control allows the various objects to have a display precedence in case of conflict (overlap).

Conflicts With OS

You must assure that the player/missile data is address-aligned as required by PMBASE [D407]. You also must find a suitable free area that the OS guarantees to be free under all environments.

READING GAME CONTROLLERS

The OS reads the game controllers (shown below) as part of the stage 2 VBLANK process (see Appendix L J1-9):

- Joysticks/triggers 1-4.
- Paddle controllers/triggers 1-8.
- Driving controllers/triggers 1-4.
- Light pen/trigger

In addition to these controllers, other information can be sensed or sent using the PIA chip to that the console connectors are interfaced.

Keyboard Controller Sensing

Data can be read from an ATARI keyboard controller connected to the first port. This program alters registers on a chip called a PIA. To set these back to the default values to do further I/O, hit [SYSTEM.RESET] or POKE PACTL,60. If this program is to be loaded from diskette, use LOAD, not RUN and wait for the busy light on the disk drive to go out. Do not execute the program before this light goes out, otherwise the diskette continues to spin.

```
1 GRAPHICS 0
5 PRINT :PRINT "    KEYBOARD CONTROLLER DEMO"
10 DIM ROW(3),I$(13),BUTTON$(1)
30 GOSUB 6000
40 FOR CNT=1 TO 4
60 POSITION 2,CNT*2+5:PRINT "CONTROLLER # ";CNT;":";
```

```
70 NEXT CNT
80 FOR CNT=1 TO 4:GOSUB 7000:POSITION 19,CNT+CNT+5:PRINT BUTTON$;
   :NEXT CNT
120 GOTO 80
6000 REM ** SET UP FOR CONTROLLERS **
6010 PORTA=54016:PORTB=54017:PACTL=54018:PBCTL=54019
6020 POKE PACTL,48:POKE PORTA,255:POKE PACTL,52:POKE PORTA,221
6025 POKE PBCTL,48:POKEPORTB,255:POKE PBCTL,52:POKE PORTB,221
6030 ROW(0)=238:ROW(1)=221:ROW(2)=187:ROW(3)=119
6040 I$=" 123456789*0#"
6050 RETURN
7000 REM ** RETURN BUTTON$ WITH CHARACTER FOR BUTTON WHICH HAS
     BEEN PRESSED ON CONTROLLER CNT (1-4). **
7001 REM ** NOTE: A 1 WILL BE RETURNED IF NO CONTROLLER IS
     CONNECTED. **
7002 REM ** A SPACE WILL BE RETURNED IF THE CONTROLLER IS
     CONNECTED BUT NO KEY HAS BEEN PRESSED. **
7003 PORT=PORTA:IF CNT>2 THEN PORT=PORTB
7005 P=1
7008 PA0=CNT+CNT-2
7010 FOR J=0 TO 3
7020 POKE PORT,ROW(J)
7030 FOR I=1 TO 10:NEXT I
7050 IF PADDLE(PA0+1)>10 THEN P=J+J+J+2:GOTO 7090
7060 IF PADDLE(PA0)>10 THEN P=J+J+J+3:GOTO 7090
7070 IF STRIG(CNT-1)=0 THEN P=J+J+J+4:GOTO 7090
7080 NEXT J
7090 BUTTON$=I$(P,P)
7095 RETURN
```

Figure 11-4 Reading Data From an ATARI Keyboard Controller

The table below shows the variable/register values used for reading a keyboard controller from each of the four controller ports.

	Port 1	Port 2	Port 3	Port 4
PORT A direction bits	OF	FO	-	-
PORT B direction bits	-	-	OF	FO
Port A row sel ect	FE, FD, FB, F7	EF, DF BF, 7F	-	-
Port B row se- lect	-	-	FE, FD, FB, F7	EF, DF, BF, 7F
Column 1 Sense	PADDL1	PADDL3	PADDL5	PADDL7
Column 2 Sense	PADDLO	PADDL2	PADDL4	PADDL6
Column 3 Sense	STRIGO	STRIG1	STRIG2	STRIG3

Figure 11-5 ATARI Keyboard Controller Variable/Register Value Table

Front Panel Connectors as I/O Ports

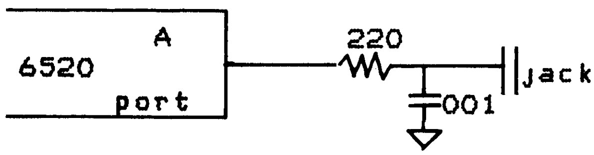
The three pages that follow show how some of the pins in the front panel (game controller) connectors can be used as general I/O pins.

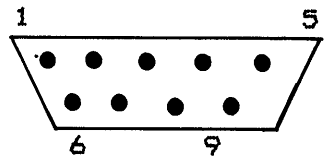
Hardware Information

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

For more information refer to 6520 chip manual.

Port A Circuit (typical):



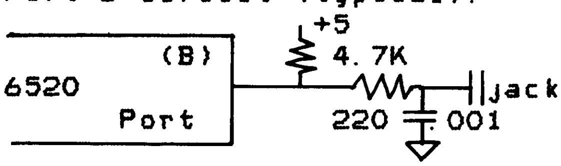


Male connector, FRONT view
Pin 8 = Ground
Pin 7 = Vcc 8+5v *)

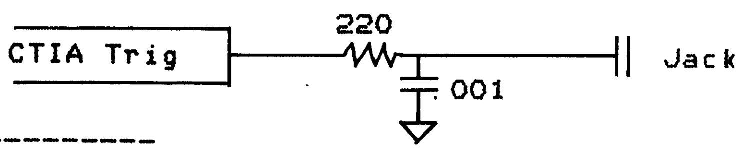
Note: 50mA maximum total external drain on power supply allowed

---

Port B Circuit (typical):



"Trigger" Port Circuit (typical):



Software Information

6520 PIA: (This also pertains to all of the following: **)
Port A control (address D302)

7	6	5	4	3	2	1	0	
0	0	1	1	1	X	0	0	Write this into this register

Port A Data/Data direction addres ing control
0 = Data Direction is at D300
1 = data is at D300

Port A data direction (address D300)

7	6	5	4	3	2	1	0	
X	X	X	X	X	X	X	X	Write this into this register.

Data direction control for Port A
1 = Out
0 = In

Port A data (address D300)

7	6	5	4	3	2	1	0
4	3	2	1	4	3	2	1

Read or Write this register

$\underbrace{4\ 3\ 2\ 1}_{\text{Jack 2}}\ \underbrace{4\ 3\ 2\ 1}_{\text{Jack 1}}$

Pin Numbers

Port B Control (address D303)

0	0	1	1	1	X	0	0

6520 PIA:

Port B Control (address D303)

7	6	5	4	3	2	1	0
0	0	1	1	1	X	0	0

write this into this register

Bit 2 — Port B Data/Data direction addressing control
0 = D301 contains data direction
1 = $D301 contains

Port B data direction (address D301)

7	6	5	4	3	2	1
X	X	X	X	X	X	X

write this into this register

data direction control for Port B
1 = Out
0 = In

Port B data (address D301)

7	6	5	4	3	2	1	0
4	3	2	1	4	3	2	1

$\underbrace{4\ 3\ 2\ 1}_{\text{Jack 4}}\ \underbrace{4\ 3\ 2\ 1}_{\text{J ack 3}}$

Pin Numbers

Four "Trigger" ports: D010, D011, D012, D013

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	X

Read this port

Bit 0 — Trigger Value
D010 = Port 1 Pin 6
D013 = Port 4 pin 6

Other Miscellaneous Software Information

1). The OS sets up all PIA ports as inputs during initialization.
2). The OS usually reads the above once per television frame (during
vertical-blank) into RAM as follows:

Data Base Name	Address	Data	Pins S
STICK0	0278	7 6 5 4 3 2 1 0	Jack 1, pins 4,3,2, if 10053,7
		0 0 0 0 X X X X	
STICK1	0729		Jack 2, Pins 4,3,2,1
STICK2	027A		Jack 3, Pins 4,3,2,1
STICK3	027B		Jack 4, Pins 4,3,2,1
STRIG0	0284		Jack 1, Pin 6
		7 6 5 4 3 2 1 0	
STRIG1	0285	0 0 0 0 0 0 0	Jack 2, Pin 6
STRIG2	0286		Jack 3, Pin 6
STRIG3	0287		Jack 4, Pin 6
PADDL1	0270	7 6 5 4 3 2 1 0	
		X X X X X X X X	Jack 1, Pin 5
PADDL3	0272		Jack 2, Pin 5
PADDL5	0274		Jack 3, Pin 5
PADDL7	0276		Jack 4, Pin 5
PADDL0	0271		Jack 1, Pin 9
PADDL2	0273		Jack 2, Pin 9
PADDL4	0275		Jack 3, Pin 9
PADDL 6	0277		Jack 4, Pin 9

Figure 11-6 Using Front Panel Connectors As I/O Ports: Pin Function Tables

* Pins 5 and 9 are read through the paddle controller circuitry
a nominal value of 7 indicates that the pin is high (or floating)
and a nominal value of 228 indicates that the pin is pulled low.

Appendix A -- CIO COMMAND BYTE VALUES

The following hex values are known to be legitimate CIO commands.

Most handlers:

03 -- OPEN
05 -- GET RECORD
07 -- GET CHARACTERS
09 -- PUT RECORD
0B -- PUT CHARACTERS
0C -- CLOSE
0D -- GET STATUS

Display Handler only:

11 -- FILL
12 -- DRAW

Diskette File Manager only:

20 -- RENAME
21 -- DELETE
22 -- FORMAT
23 -- LOCK
24 -- UNLOCK
25 -- POINT
26 -- NOTE

Appendix B -- CIO STATUS BYTE VALUES

Shown below are the known CIO STATUS BYTE values.

```
01 (001) -- OPERATION COMPLETE (NO ERRORS)

80 (128) -- [BREAK] KEY ABORT
81 (129) -- IOCB ALREADY IN USE (OPEN)
82 (130) -- NON-EXISTENT DEVICE
83 (131) -- OPENED FOR WRITE ONLY
84 (132) -- INVALID COMMAND
85 (133) -- DEVICE OR FILE NOT OPEN
86 (134) -- INVALID IOCB NUMBER (Y reg only)
87 (135) -- OPENED FOR READ ONLY
88 (136) -- END OF FILE
89 (137) -- TRUNCATED RECORD
8A (138) -- DEVICE TIMEOUT (DOESN'T RESPOND)
8B (139) -- DEVICE NAK
8C (140) -- SERIAL BUS INPUT FRAMING ERROR
8D (141) -- CURSOR out-of-range
8E (142) -- SERIAL BUS DATA FRAME OVERRUN ERROR
8F (143) -- SERIAL BUS DATA FRAME CHECKSUM ERROR
90 (144) -- DEVICE DONE ERROR
91 (145) -- BAD SCREEN MODE
92 (146) -- FUNCTION NOT SUPPORTED BY HANDLER
93 (147) -- INSUFFICIENT MEMORY FOR SCREEN MODE

A0 (160) -- DISK DRIVE # ERROR
A1 (161) -- TOO MANY OPEN DISK FILES
A2 (162) -- DISK FULL
A3 (163) -- FATAL DISK I/O ERROR
A4 (164) -- INTERNAL FILE # MISMATCH
A5 (165) -- FILE NAME ERROR
A6 (166) -- POINT DATA LENGTH ERROR
A7 (167) -- FILE LOCKED
A8 (168) -- COMMAND INVALID FOR DISK
A9 (169) -- DIRECTORY FULL (64 FILES)
AA (170) -- FILE NOT FOUND
AB (171) -- POINT INVALID
```

Appendix C -- SIO STATUS BYTE VALUES

Shown below are the known SIO STATUS BYTE hexadecimal values.

01 (001) -- OPERATION COMPLETE (NO ERRORS)

8A (138) -- DEVICE TIMEOUT (DOESN'T RESPOND)
8B (139) -- DEVICE NAK
8C (140) -- SERIAL BUS INPUT FRAMING ERROR
8E (142) -- SERIAL BUS DATA FRAME OVERRUN ERROR
8F (143) -- SERIAL BUS DATA FRAME CHECKSUM ERROR
90 (144) -- DEVICE DONE ERROR

	ØX	2X	4X	6X	8X	AX	CX	EX
ØØ		Space	@					
Ø1		!	A	a				
Ø2		"	B	b				
Ø3		#	C	c				
Ø4		$	D	d				
Ø5		%	E	e				
Ø6		&	F	f				
Ø7		'	G	g				
Ø8		(	H	h				
Ø9		)	I	i				
ØA		*	J	j				
ØB		+	K	k				
ØC		,	L	l				
ØD		-	M	m				
ØE		.	N	n				
ØF		/	O	o				
1Ø		Ø	P	p				
11		1	Q	q				
12		2	R	r				
13		3	S	s				
14		4	T	t				
15		5	U	u				
16		6	V	v				
17		7	W	w				
18		8	X	x				
19		9	Y	y				
1A		:	Z	z				
1B	ESC	;	[		EOL			
1C		<	\	|	DEL LINE			
1D		=	]	CLEAR	INS LINE			BELL
1E		>	^	BACKSP	CLR TAB			DEL CHAR
1F		?	_	TAB	SET TAB			INS CHAR

	ØX	2X	4X	6X	8X	AX	CX	EX
ØØ		Space	@					
Ø1		!	A	a				
Ø2		"	B	b				
Ø3		#	C	c				
Ø4		$	D	d				
Ø5		%	E	e				
Ø6		&	F	f				
Ø7		'	G	g				
Ø8		(	H	h				
Ø9		)	I	i				
ØA		*	J	j				
ØB		+	K	k				
ØC		,	L	l				
ØD		-	M	m				
ØE		.	N	n				
ØF		/	O	o				
1Ø		Ø	P	p				
11		1	Q	q				
12		2	R	r				
13		3	S	s				
14		4	T	t				
15		5	U	u				
16		6	V	v				
17		7	W	w				
18		8	X	x				
19		9	Y	y				
1A		:	Z	z				
1B		;	[					
1C		<	\	|				
1D		=	]					
1E		>	^					
1F		?	_					

CODES 8Ø-FF SHOW AS THE INVERSE VIDEO OF CODES ØØ-7F

Appendix F -- KEYBOARD CODES (ATASCII)

CTRL			SHIFT & LOWER			SHIFT			LOWER		
00	,	20	20	<space>	21	40	@	35	60	^.	22
01	A	3F	21	!	1F	41	A	3F	61	a	3F
02	B	15	22	"	1E	42	B	15	62	b	15
03	C	12	23	#	1A	43	C	12	63	c	12
04	D	3A	24	$	18	44	D	3A	64	d	3A
05	E	2A	25	%	1D	45	E	2A	65	e	2A
06	F	38	26	&	1B	46	F	38	66	f	38
07	G	3D	27	'	33	47	G	3D	67	g	3D
08	H	39	28	(	30	48	H	39	68	h	39
09	I	0D	29	)	32	49	I	0D	69	i	0D
0A	J	01	2A	*	07	4A	J	01	6A	j	01
0B	K	05	2B	+	06	4B	K	05	6B	k	05
0C	L	00	2C	,	20	4C	L	00	6C	l	00
0D	M	25	2D	-	0E	4D	M	25	6D	m	25
0E	N	23	2E	.	22	4E	N	23	6E	n	23
0F	O	08	2F	/	26	4F	O	08	6F	o	08
10	P	0A	30	0	32	50	P	0A	70	p	0A
11	Q	2F	31	1	1F	51	Q	2F	71	q	2F
12	R	28	32	2	1E	52	R	28	72	r	28
13	S	3E	33	3	1A	53	S	3E	73	s	3E
14	T	2D	34	4	18	54	T	2D	74	t	2D
15	U	0B	35	5	1D	55	U	0B	75	u	0B
16	V	10	36	6	1B	56	V	10	76	v	10
17	W	2E	37	7	33	57	W	2E	77	w	2E
18	X	16	38	8	35	58	X	16	78	x	16
19	Y	2B	39	9	30	59	Y	2B	79	y	2B
1A	Z	17	3A	:	02	5A	Z	17	7A	z	17
1B	<esc>	1C	3B	;	0D	5B	[	20	7B	;	02
1C	^<up>	0E	3C	<	36	5C	\	06	7C	¦	0F
1D	^<down>	0F	3D	=	0F	5D	]	22	7D	<clear>	36
1E	^<left>	06	3E	>	37	5E	^	07	7E	<back>	34
1F	^<right>	07	3F	?	26	5F	_	0E	7F	<tab>	2C

80-9A	/¦\	00-1A
9B	<return> and ^3	0C, 1A
9C	s<del>	34
9D	s<insert>	37
9E	^<tab>	2C
9F	s<tab>	2C
A0-FC	/¦\	20-7C
FD	^2	1E
FE	^<del>	34
FF	^<insert>	37

<clear> ::= s< or ^<
<return> ::= <return> or s<return> or ^<return>
<esc> ::= <esc> or s<esc> or ^<esc>
<space> ::= <space> or s<space> or ^<space>

Where: s as a prefix indicates [SHIFT].
^ as a prefix indicates [CTRL].
/¦\ as a prefix indicates ATARI key inverse active.

Appendix G -- PRINTER CODES (ATASCII)

Character set for "normal" mode printing:

20	<space>	40	@	60	`
21	!	41	A	61	a
22	"	42	B	62	b
23	#	43	C	63	c
24	$	44	D	64	d
25	%	45	E	65	e
26	&	46	F	66	f
27	'	47	G	67	g
28	(	48	H	68	h
29	)	49	I	69	i
2A	*	4A	J	6A	j
2B	+	4B	K	6B	k
2C	,	4C	L	6C	l
2D	-	4D	M	6D	m
2E	.	4E	N	6E	n
2F	/	4F	O	6F	o
30	0	50	P	70	p
31	1	51	Q	71	q
32	2	52	R	72	r
33	3	53	S	73	s
34	4	54	T	74	t
35	5	55	U	75	u
36	6	56	V	76	v
37	7	57	W	77	w
38	8	58	X	78	x
39	9	59	Y	79	y
3A	:	5A	Z	7A	z
3B	;	5B	[	7B	{
3C	<	5C	\	7C	¦
3D	=	5D	]	7D	}
3E	>	5E	^	7E	~
3F	?	5F	_	7F	<space>

Note: The following codes print differently than defined by the ATASCII definition.

00 through 1F print blank.
60 prints ` instead of "diamond".
7B prints { instead of "spade".
7D prints } instead of "clear".
7E prints ~ instead of "backspace".
7F prints blank instead of "tab".

Character set for "sideways" mode printing:

		40	@	60	@
		41	A	61	A
		42	B	62	B
		43	C	63	C
		44	D	64	D
		45	E	65	E
		46	F	66	F
		47	G	67	G
		48	H	68	H
		49	I	69	I
		4A	J	6A	J
		4B	K	6B	K
		4C	L	6C	L
		4D	M	6D	M
		4E	N	6E	N
		4F	O	6F	O
30	0	50	P	70	P
31	1	51	Q	71	Q
32	2	52	R	72	R
33	3	53	S	73	S
34	4	54	T	74	T
35	5	55	U	75	U
36	6	56	V	76	V
37	7	57	W	77	W
38	8	58	X	78	X
39	9	59	Y	79	Y
3A	:	5A	Z	7A	Z
3B	;	5B	[	7B	[
3C	<	5C	\	7C	\
3D	=	5D	]	7D	]
3E	>	5E	<up>	7E	<up>
3F	?	5F	<left>	7F	<left>

Note: the following codes print differently than defined by the ATASCII definition.

00 through 2F print blank.
5E prints "up arrow" instead of .
5F prints "left arrow" instead of _.
60 through 7F repeats 40 through 5F instead of proper set.

Mode #	Horiz. Posit.	Vert. W/O Sp	Vert. W Sp	Colors	Data Value	Color Reg.	Memory Reqd. (split)	(full)
0	40	24	--	2	backgd.	BAK	992	992
					00-FF	PF 2		
					"	PF 1*		
1	20	24	20	5	backgd.	BAK	674	672
					00-3F	PF 0		
					40-7F	PF 1		
					80-BF	PF 2		
					C0-FF	PF 3		
2	20	12	10	5	backgd.	BAK	424	420
					00-3F	PF 0		
					40-7F	PF 1		
					80-BF	PF 2		
					C0-FF	PF 3		
3	40	24	20	4	0	BAK	434	432
					1	PF 0		
					2	PF 1		
					3	PF 2		
4	80	48	40	2	0	BAK	694	696
					1	PF 0		
5	80	48	40	4	0	BAK	1174	1176
					1	PF 0		
					2	PF 1		
					3	PF 2		
6	160	96	80	2	0	BAK	2174	2184
					1	PF 0		
7	160	96	80	4	0	BAK	4190	4200
					1	PF 0		
					2	PF 1		
					3	PF 2		
8	320	192	160	2	0	PF 2	8112	8138
					1	PF 1*		
9	80	192	--	1	Note 2			8138
10	80	192	--	9	0	PM 0		8138
					1	PM 1		
					2	PM 2		
					3	PM 3		
					4	PF 0		

					5	PF 1	
					6	PF 2	
					7	PF 3	
					8	BAK	
					9	BAK	
					A	BAK	
					B	BAK	
					C	PF 0	
					D	PF 1	
					E	PF 2	
					F	PF 3	
11	80	192	--	16	Note 3		8138

Notes:

*   Uses color of PF 2, lum of PF 1.
2   Uses color of BAK, lum of data value ($0-F).
3   Uses color of data value ($0-F), lum of BAK.

PF x ::= Playfield color register x.
PM x ::= Player/Missile Graphics color register x.
BAK ::= Background color register (also known as PF 4).

The default values for the color registers are shown below:

BAK = $00
PF0 = $28
PF1 = $CA
PF2 = $94
PF3 = $46

The form of a color register byte is shown below:

```
             7 6 5 4 3 2 1 0
            +-+-+-+-+-+-+-+-+
            | color | lum |0|
            +-+-+-+-+-+-+-+-+

Where: color (hex values)     lum

       0 = gray               0 = minimum luminance
       1 = light orange       1 =         |
       2 = orange             2 =         |
       3 = red orange         3 =   (increasing
       4 = pink               4 =    luminance)
       5 = purple             5 =         |
       6 = purple-blue        6 =         |
       7 = blue               7 = maximum luminance
       8 = blue
       9 = light blue
       A = turquoise
       B = green-blue
       C = green
       D = yellow-green
       E = orange-green
       F = light orange
```

Appendix I -- SERIAL BUS ID AND COMMAND SUMMARY

Serial bus device IDs

Floppy diskettes	D1-D4	$31-34
Printer	P1	$40
RS-232-C	R1-R4	$50-53

Serial bus control codes

ACK	- $41 ('A')
NAK	- $4E ('N')
COMPLETE	- $43 ('C')
ERR	- $45 ('E')

Serial bus command codes

READ	- $52 ('R')	Disk
WRITE	- $57 ('W')	Printer/Disk
STATUS	- $53 ('S')	Printer/Disk
PUT(no check)	- $50 ('P')	Disk
FORMAT	- $21 ('!')	Disk
READ ADDRESS	- $54 ('T')	
READ SPIN	- $51 ('Q')	Disk
MOTOR ON	- $55 ('U')	Disk
VERIFY SECTOR	- $56 ('V')	Disk

Appendix J -- ROM VECTORS

The fixed address OS ROM JMP vectors are shown below; at each address is a JMP instruction to the indicated routine.

Name	Addr	Reference	Function
DISKIV	E450	*	Diskette Handler initialization
DSKINV	E453	5.4.2	Diskette Handler entry.
CIOV	E456	5.2	CIO utility entry.
SIOV	E459	9.3	SIO utility entry.
SETVBV	E45C	6.7.2	Set System Timers routine.
SYSVBV	E45F	6.3	Stage 1 VBLANK entry.
XITVBV	E462	6.3	Exit VBLANK entry.
SIOINV	E465	*	SIO utility initialization.
SENDEV	E468	*	Send enable routine.
INTINV	E46B	*	Interrupt Handler initialization.
CIOINV	E46E	*	CIO utility initialization.
BLKBDV	E471	3.1.1	Blackboard mode entry.
WARMSV	E474	7.	Warmstart ([SYSTEM.RESET]) entry.
COLDSV	E477	7.	Coldstart (power-up) entry.
RBLOKV	E47A	*	Cassette-read block entry.
CSOPIV	E47D	*	Cassette-OPEN input entry.

* These vectors are for OS internal use only.

The fixed address Floating Point Package ROM routine entry point addresses are shown below; complete descriptions of the corresponding routines are provided in Section 8.

AFP	D800	ASCII to FP convert.
FASC	D8E6	FP to ASCII convert.
IFP	D9AA	Integer to FP convert.
FPI	D9D2	FP to integer convert.
FADD	DA66	FP add.
FSUB	DA60	FP subtract.
FMUL	DADB	FP multiply
FDIV	DB28	FP divide.
LOG	DECD	FP base e logarithm.
LOG10	DED1	FP base 10 logarithm.
EXP	DDC0	FP base e exponentiation.
EXP10	DDCC	FP base 10 exponentiation.
PLYEVL	DD40	FP polynomial evaluation.
ZFR0	DA44	Clear FR0.
ZF1	DA46	Clear FP number.
FLD0R	DD89	Load FP number.
FLD0P	DD8D	Load FP number.
FLD1R	DD98	Load FP number.
FLD1P	DD9C	Load FP number.
FST0R	DDA7	Store FP number.
FST0P	DDAB	Store FP number.
FMOVE	DDB6	Move FP number.

The base addresses of the Handler vectors for the resident handlers are shown below:

Screen Editor (E)	E400
Display Handler (S)	E410
Keyboard Handler (K)	E420
Printer Handler (P)	E430
Cassette Handler (C)	E440

See Section 5 for the format of the entry for each Handler.

The 6502 Computer interrupt vector values are shown below:

Function	Address	Value
NMI	FFFA	E7B4
RESET	FFAC	E477
IRQ	FFFE	E6FE

Appendix K -- DEVICE CHARACTERISTICS

This appendix describes the physical characteristics of the devices that interface to the ATARI 400 and ATARI 800 Home Computers. Where applicable, data capacity, data transfer rate, storage format, SIO interface, and cabling will be detailed.

KEYBOARD

The keyboard input rate is limited by the OS keyboard reading procedure to be 60 characters per second. The code for each key is shown in Table 5-4. The keyboard hardware has no buffering and is rate-limited by the debounce algorithm used.

DISPLAY

The television screen display generator has many capabilities that are not used by the Display Handler (as described in Section 5 and shown in Appendix H). There are additional display modes, object generators, hardware display scrolling, and many other features that are described in the ATARI Home Computer Hardware Manual.

Since all display data is stored in RAM, the display data update rate is limited primarily by the software routines that generate and format the data and access the RAM. The generation of the display from the RAM is accomplished by the ANTIC and CTIA or GTIA chips using Direct Memory Access (DMA) to access the RAM data.

The internal storage formats for display data for the various modes are detailed in the ATARI Home Computer Hardware Manual.

ATARI 410 PROGRAM RECORDER

The ATARI 410 Program Recorder has the following characteristics:

DATA CAPACITY:

100 characters per C-60 tape (unformatted).

DATA TRANSFER RATES:

* 600 Baud (60 characters per second)

*Note: The OS has the ability to adjust to different tape speeds (447 - 895 Baud).

STORAGE FORMAT:

Tapes are recorded in 1/4 track stereo format at 1 7/8 inches per second. The tape can be recorded in both directions, where tracks 1 and 2 are side A left and right; and tracks 3 and 4 are side B right and left (industry standard). On each side, the left channel (1 or 4) is used for audio and the right channel (2 and 3) is used for digital information.

The audio channel is recorded the normal way. The digital channel is recorded using the POKEY two-tone mode producing FSK data at up to 600 baud. The MARK frequency is 5327 Hz and the SPACE frequency is 3995 Hz. The transmission of data is asynchronous byte serial as seen from the computer; POKEY reads or writes a bit serial FSK sequence for each byte, in the following order:

```
1 start bit (SPACE)
data bit 0 -+
data bit 1  !
.           +-  0 = SPACE, 1 = MARK.
data bit 6  !
data bit 7 -+
1 stop bit (MARK)
```

The only control the computer has over tape motion is motor start/stop; and this only if the PLAY button is pressed by the user. In order for recording to take place, the user must press both the REC and PLAY buttons on the cassette. The computer has no way to sense the position of these buttons, nor even if an ATARI 410 Program Recorder is cabled to the computer, so the user must be careful when using this device.

SIO INTERFACE

The cassette device utilizes portions of the serial bus hardware, but does not follow any of the protocol as defined in Section 9.

ATARI 820[TM] 40-COLUMN IMPACT PRINTER

The ATARI 820 Printer has the following characteristics:

DATA CAPACITY:

40 characters per line (normal printing)
29 characters per line (sideways printing)

DATA TRANSFER RATES:

Bus rate: xx characters per second.
Print time (burst): xx characters per second.
Print time (average): xx characters per second.

STORAGE FORMAT:

3 7/8 inch wide paper.
5X7 dot matrix, impact printing.

Normal format --
40 characters per line.
6 lines per inch (vertical).
12 characters per inch (horizontal).

Sideways format --
29 characters per line.
6 lines per inch (vertical).
9 characters per inch (horizontal).

SIO INTERFACE

The controller serial bus ID is $40.

The controller supports the following SIO commands (see Section 5 for more information regarding the Handler and Section 9 for a general discussion of bus commands):

GET STATUS

The computer sends a command frame of the format shown below:

Device ID = $40.
Command byte = $53.
auxiliary 1 = doesn't matter.
auxiliary 2 = doesn't matter.
Checksum = checksum of bytes above.

The printer controller responds with a data frame of the format shown earlier in this appendix as part of the GET STATUS discussion.

PRINT LINE

The computer sends a command frame of the format shown below:

Device ID = $40.
Command byte = $57.

auxiliary 1 = doesn't matter.
auxiliary 2 = $4E for normal print or $53 for sideways.
Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:

Leftmost character of line (column 1).
Next character of line (column 2).
.
.
Rightmost character of line (column 40 or 29).
Checksum byte.

Note that the data frame size is variable, either 41 or 30 bytes in length, depending upon the print mode specified in the command frame.

ATARI 810 DISK DRIVE

The ATARI 810[TM] Disk Drive has the following characteristics:

DATA CAPACITY:

720 sectors of 128 bytes each (Disk Handler format).
709 sectors of 125 data bytes each (Disk File Manager format).

DATA TRANSFER RATES:

Bus rate: 1920 characters per second.
Seek time: 5.25 msec. per track + 10 to 210 msec.
Rotational latency: 104 msec maximum (288 rpm).

STORAGE FORMAT:

5 1/4 inch diskette, soft sectored by the controller.
40 tracks per diskette.
18 sectors per track.
128 bytes per sector.
Controlled by National INS1771-1 formatter/controller chip.
Sector sequence per track is: 18, 1, 3, 5, 7, 9, 11, 13, 15, 17, 2, 4, 6, 8, 10, 12, 14, 16

SIO INTERFACE

The controller serial bus IDs range from $31 (for 'D1') to $34 (for 'D4').

The controller supports the following SIO commands (see earlier in this Appendix for information about the Diskette Handler and Section 9 for a general discussion of bus commands):

GET STATUS

The computer sends a command frame of the format shown below:

Device ID = $31-34.
Command byte = $53.
auxiliary 1 = doesn't matter.
auxiliary 2 = doesn't matter.
Checksum = checksum of bytes above.

The diskette controller responds with a data frame of the format shown earlier in this Appendix as part of the STATUS REQUEST discussion.

PUT SECTOR (WITH VERIFY)

The computer sends a command frame of the format shown below:

Device ID = $31-34
Command byte = $57.
auxiliary 1 = low byte of sector number.
auxiliary 2 = high byte of sector number (1-720).
Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:

128 data bytes.
Checksum byte.

The diskette controller writes the frame data to the specified sector, then reads the sector and compares the content with the frame data. The COMPLETE byte value indicates the status of the operation.

PUT SECTOR (NO VERIFY)

The computer sends a command frame of the format shown below:

Device ID = $31-34
Command byte = $50.
auxiliary 1 = low byte of sector number.
auxiliary 2 = high byte of sector number (1-720).
Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:

128 data bytes.
Checksum byte.

The diskette controller writes the frame data to the specified sector, then sends a COMPLETE byte value that indicates the status of the operation.

GET SECTOR

The computer sends a command frame of the format shown below:

Device ID = $31-34
Command byte = $52.
auxiliary 1 = low byte of sector number.
auxiliary 2 = high byte of sector number (1-720).
Checksum = checksum of bytes above.

The diskette controller sends a data frame of the format shown below:

128 data bytes.
Checksum byte.

FORMAT DISKETTE

The computer sends a command frame of the format shown below:

Device ID = $31-34
Command byte = $21.
auxiliary 1 = doesn't matter.
auxiliary 2 = doesn't matter.
Checksum = checksum of bytes above.

The diskette controller completely formats the diskette (generates 40 tracks of 18 soft sectors per track with the data portion of each sector equal to all zeros) and then reads each sector to verify its integrity. A data frame of 128 bytes plus checksum is returned in that the sector numbers of all bad sectors (up to a maximum of 63 sectors) are contained, followed by two consecutive bytes of $FF. If there are no bad sectors on the diskette the first 2 bytes of the data

Appendix L -- OS DATA BASE VARIABLE FUNCTIONAL DESCRIPTIONS

CENTRAL DATA BASE DESCRIPTION

This appendix provides detailed information for those variables in the OS data base that can be altered by the user. Remaining variables are provided narrative descriptions. Information on the variables is presented in a multiple access scheme: Lookup tables are referenced to a common set of narratives, that is itself ordered by function.

Variable descriptions are referenced by a label called a variable identifier (VID) number. The label comprises a single letter followed by a number. A different letter is assigned for each major functional area being described, and the numbers are assigned sequentially within each functional area. Those variables that are not considered to be of interest to any user are flagged with an asterisk (*) after their names. The data base lookup tables provided are:

1. Functional grouping -- index to the function narrative and descriptions of variables, giving VID and variable name.

2. Alphabetic list of names -- giving VID of description.

3. Address ordered list -- giving VID of description.

Item 1, the functional grouping index, starts on the next page; the other two lookup tables are at the end of Appendix L.

FUNCTIONAL INDEX TO DATA BASE VARIABLE DESCRIPTIONS

A. Memory configuration
   A1 MEMLO
   A2 MEMTOP
   A3 APPMHI
   A4 RAMTOP
   A5 RAMSIZ

B. Text/graphics screen

   Cursor control
      B1 CRSINH
      B2 ROWCRS, COLCRS
      B3 OLDROW, OLDCOL
      B4 TXTROW, TXTCOL

   Screen margins
      B5 LMARGN
      B6 RMARGN

   Color control
      B7 PCOLR0 - PCOLR3
      B8 COLOR0 - COLOR4

   Text scrolling
      B9 SCRFLG*

   Attract mode
      B10 ATRACT
      B11 COLRSH*
      B12 DRKMSK*

   Tabbing
      B13 TABMAP

   Logical text lines
      B14 LOGMAP*
      B15 LOGCOL*

   Split screen

B16 BOTSCR*

FILL/DRAW function

B17 FILDAT
B18 FILFLG*
B19 NEW SROW*, NEWCOL*
B20 HOLD4*

B21 ROWINC*, COLINC*
B22 DELTAR*, DELTAC*
B23 COUNTR*
B24 ROWAC*, COLAC*
B25 ENDPT*

Displaying control characters

Escape (display following control char)

B26 ESCFLG*

Display control characters mode

B27 DSPFLG

Bit mapped graphics

B28 DMASK*
B29 SHFAMT*

Internal working variables

B30 HOLD1*
B31 HOLD2*
B32 HOLD3*
B33 TMPCHR*
B34 DSTAT*
B35 DINDEX
B36 SAVMSC
B37 OLDCHR*
B38 OLDADR*
B39 ADRESS*
B40 MLTTMP/OPNTMP/TOADR*
B41 SAVADR/FRMADR*
B42 BUFCNT*
B43 BUFSTR*
B44 SWPFLG*
B45 INSDAT*
B46 TMPROW*, TMPCOL*
B47 TMPLBT*
B48 SUBTMP*
B49 TINDEX*
B50 BITMSK*
B51 LINBUF*
B52 TXTMSC
B53 TXTOLD*

Internal character code conversion
- B54 ATACHR
- B55 CHAR*

C. Disk Handler
- C1 BUFADR*
- C2 DSKTIM*

D. Cassette (part in SIO part in Handler)

Baud rate determination
- D1 CBAUDL*, CBAUDH*
- D2 TIMFLG*
- D3 TIMER1*, TIMER2*
- D4 ADDCOR*
- D5 TEMP1*
- D6 TEMP3*
- D7 SAVIO*

Cassette mode
- D8 CASFLG*

Cassette buffer
- D9 CASBUF*
- D10 BLIM*
- D11 BPTR*

Internal working variables
- D12 FEOF*
- D13 FTYPE*
- D14 WMODE*
- D15 FREQ*

E. Keyboard

Key reading and debouncing
- E1 CH1*
- E2 KEYDEL*
- E3 CH

Special functions

Start/stop
E4 SSFLAG

[BREAK]
E5 BRKKEY

[SHIFT]/[CONTROL] lock
E6 SHFLOK
E7 HOLDCH*

Autorepeat
E8 SRTIMR*

Inverse video
E9 INVFLG

Console switches ([SELECT],[START],and [OPTION])

F. Printer

printer-buffer

F1 PRNBUF*
F2 PBUFSZ*
F3 PBPNT*

Internal working variables
F4 PTEMP*
F5 PTIMOT*

G. Central I/O routine (CIO)

User call parameters

G1 IOCB
G2 ICHID
G3 ICDNO
G4 ICCOM
G5 ICSTA
G6 ICBAL, ICBAH
G7 ICPTL, ICPTH
G8 ICBLL, ICBLH
G9 ICAX1, ICAX2
G10 ICSPR

Device status
G11 DVSTAT

device table
G12 HATABS

CIO/Handler interface Parameters

G13 ZIOCB (IOCBAS)
G14 ICHIDZ
G15 ICDNOZ
G16 ICCOMZ
G17 ICSTAZ
G18 ICBALZ, ICBALH
G19 ICPTLZ, ICPTHZ
G20 ICBLLZ, ICBLHZ
G21 ICAX1Z, ICAX2Z
G22 ICSPRZ (ICIDNO, CIOCHR)

Internal working variables
G23 ICCOMT*
G24 ICIDNO*
G25 CIOCHR*

H. Serial I/O routine (SIO)

User call parameters

H1 DCB control block
H2 DDEVIC
H3 DUNIT
H4 DCOMND
H5 DSTATS
H6 DBUFLO, DBUFHI
H7 DTIMLO
H8 DBYTLO, DBYTHI
H9 DAUX1, DAUX2

Bus sound control
H10 SOUNDR

Serial bus control

Retry logic
H11 CRETRY*
H12 DRETRY*

Checksum
H13 CHKSUM*
H14 CHKSNT*
H15 NOCKSM*

Data buffering

General buffer control

H16 BUFRLO*, BUFRHI*
H17 BFENLO*, BFENHI*

Command frame output buffer
H18 CDEVIC*
H19 CCOMND*
H20 CAUX1*, CAUX2*

Receive/transmit data buffering
H21 BUFRFL*
H22 RECVDN*
H23 TEMP*
H24 XMTDON*

SIO timeout
H25 TIMFLG*
H26 CDTMV1*
H27 CDTMA1*

Internal working variables
H28 STACKP*
H29 TSTAT*
H30 ERRFLG*
H31 STATUS*
H32 SSKCTL*

J. ATARI controllers

Joysticks
J1 STICK0 - STICK3
J2 STRIG0 - STRIG3

Paddles
J3 PADDL0 - PADDL7
J4 PTRIG0 - PTRIG7

Paddle controllers
J8 STICK0 - STICK3
J9 STRIG0 - STRIG3

K. Disk file manager
K1 FMSZPG*
K2 ZBUFP*
K3 ZDRVA*
K4 ZSBA*
K5 ERRNO*

L. Disk utilities (DOS)
   L1 DSKUTL*

M. Floating point package
   M1 FR0
   M2 FRE*
   M3 FR1
   M4 FR2*
   M5 FRX*
   M6 EEXP*
   M7 NSIGN*
   M8 ESIGN*
   M9 FCHRFLG*
   M10 DIGRT*
   M11 CIX
   M12 INBUFF
   M13 ZTEMP1*
   M14 ZTEMP4*
   M15 ZTEMP3*
   M16 FLPTR
   M17 FPTR2*
   M18 LBPR1*
   M19 LBPR2*
   M20 LBUFF
   M21 PLYARG*
   M22 FPSCR/FSCR*
   M23 FPSCR1/FSCR1*
   M24 DEGFLG/RADFLG*

N. Power-Up and System Reset
   RAM sizing
      N1 RAMLO*, TRAMSZ*
      N2 TSTDAT*

   Diskette/cassette-boot
      N3 DOSINI
      N4 CKEY*
      N5 CASSBT*
      N6 CASINI
      N7 BOOT?*
      N8 DFLAGS*
      N9 DBSECT*
      N10 BOOTAD*

   Environmental control
      N11 COLDST
      N12 DOSVEC

[S RESET]
    N13 WARMST

P. Interrupts
   P1 CRITIC
   P2 POKMSK

   System Timers

      Real-time clock
         P3 RTCLOK

      System Timer 1
         P4 CDTMV1
         P5 CDTMA1

      System Timer 2
         P6 CDTMV2
         P7 CDTMA2

      System Timers 3-5
         P8 CDTMV3, CDTMV4, CDTMV5
         P9 CDTMF3, CDTMF4, CDTMF5

   RAM-interrupt vectors

      NMI-interrupt vectors
         P10 VDSLST
         P11 VVBLKI
         P12 VVBLKD

      IRQ-interrupt vectors
         P13 VIMIRQ
         P14 VPRCED
         P15 VINTER
         P16 VBREAK
         P17 VKEYBD
         P18 VSERIN
         P19 VSEROR
         P20 VSEROC
         P21 VTIMR1, VTIMR2, VTIMR4

   Hardware register updates
      P22 SDMCTL
      P23 SDLSTL, SDLSTH
      P24 GPRIOR
      P25 CHACT
      P26 CHBAS
      P27 PCOLRx, COLORx

Internal working variable
P28 INTEMP*

R. User areas
R1 (unlabeled)
R2 USAREA

This appendix contains descriptions of many of the data base variables; descriptions are included for all of the user-accessible variables and for some of the "internal" variables as well. Those variables that are not considered to be normally of interest to any user are flagged with an asterisk (*) after their names; the other variables can be of interest to one or more of the following classes of users:

- o End user.
- o Game developer.
- o Applications programmer.
- o System utility writer.
- o Language processor developer.
- o Device Handler Writer.

Each variable is specified by its system equate file name followed by its address (in hex) and the number of bytes reserved in the data base (in decimal), in the following form:

<name> [<address>,<size>]

For example:

MEMLO [02E7,2]

Note that most word (2 byte) variables are ordered with the least significant byte at the lower address.

A. MEMORY CONFIGURATION

See Section 4 for a general discussion of memory dynamics and section 7 for details of system initialization.

A1 MEMLO [02E7,2] -- User-free memory low address

MEMLO contains the address of the first location in the free memory region. The value is established by the OS during power-up and system reset initialization and is never altered by the OS thereafter.

A2 MEMTOP [02E5,2] -- User-free memory high address

MEMTOP contains the address of the first non-useable memory location above the free memory region. The value is established by the OS during power-up and system reset initialization; and then is re-established whenever the display is opened, based upon the requirements of the selected graphics mode.

A3 APPMHI [000E,2] -- User-free memory screen lower limit

APPMHI is a user-controlled variable that contains the address within the free memory region below which the Display Handler cannot go in setting up a display screen. This variable is initialized to zero by the OS at power-up.

A4 RAMTOP* [006A,1] -- Display Handler top of RAM address (MSB)

RAMTOP permanently retains the RAM top address that was contained in TRAMSZ (as described in N1) for the Display Handler's use. The value is set up as part of Handler initialization.

A5 RAMSIZ [02E4,1] -- Top of RAM address (MSB only)

RAMSIZ permanently retains the RAM top address that was contained in TRAMSZ (as described in N1).

B. TEXT/GRAPHICS SCREEN

See Section 5 for a discussion of the text and graphics screens and their Handlers.

Cursor Control

For the text screen and split-screen text window there is a visible cursor on the screen which shows the position of the next input or output operation. The cursor is represented by inversing the video of the character upon which it resides; but the cursor can be made invisible, at the user's option. The graphics screen always has an invisible cursor.

The cursor position is sensed by examining data base variables and can be moved by altering those same variables; in addition, when using the Screen Editor, there are cursor movement control codes that can be sent as data (as explained in Section 5).

B1 CRSINH [02F0,1] -- Cursor display inhibit flag

When CRSINH is zero, all outputs to the text screen will be followed by a visible cursor (inversed character); and when CRSINH is nonzero, no visible cursor will be generated.

CRSINH is set to zero by power-up, the [SYSTEM.RESET] or [BREAK] keys or an OPEN command to the Display Handler or Screen Editor.

Note that altering CRSINH does not cause the visible cursor to change states until the next output to the screen; if an immediate change to the cursor state is desired, without altering the screen data, follow the CRSINH change with the output of CURSOR UP, CURSOR DOWN, or some other innocuous sequence.

B2 ROWCRS [0054,1] and COLCRS [0055,2] -- Current cursor position

ROWCRS and COLCRS define the cursor location (row and column, respectively) for the next data element to be read from or written to the main screen segment. When in split-screen mode, the variables TXTROW and TXTCOL define the cursor for the text window at the bottom of the screen as explained in B4 below.

The row and column numbering start with the value zero, and increase in increments of one to the number of rows or columns minus 1; with the upper left corner of the screen being the origin (0,0).

ROWCRS is a single-byte variable with a maximum allowable value of 191 (screen modes 8-11); COLCRS is a 2-byte variable with a maximum allowable value of 319 (screen mode 8).

B3 OLDROW [005A,1] and OLDCOL [005B,2] -- Prior cursor position

OLDROW and OLDCOL are updated from ROWCRS and COLCRS before every operation. The variables are used only for the DRAW and FILL operations.

B4 TXTROW [0290,1] and TXTCOL [0291,2] -- Split-screen text cursor position

TXTROW and TXTCOL define the cursor location (row and column, respectively) for the next data element to be read from or written to the split-screen text window.

The row and column numbering start with the value zero, and increase in increments of one to 3 and 39, respectively; with the upper left corner of the split-screen text window being the origin (0,0).

Screen Margins

The text screen and split-screen text window have user-alterable left and right margins that define the normal domain of the text cursor.

B5 LMARGN [0052,1] -- Text column left margin

LMARGN contains the column number (0-39) of the text screen left margin; the text cursor will remain on or to the right of the left margin as a result of all operations, unless the cursor column variable is directly updated by the user (see B2 and B4 above). The default value for LMARGN is 2 and is established upon power-up or system reset.

B6 RMARGN [0053,1] -- Text column right margin

RMARGN contains the column number (0-39) of the text screen right margin; the text cursor will remain on or to the left of the right margin as a result of all operations, unless the cursor column variable is directly updated by the user (see B2 and B4 above). The default value for RMARGN is 39 and is established upon power-up or system reset.

Color Control

As part of the stage 2 VBLANK process (see Section 6), the values of nine data base variables are stored in corresponding hardware color control registers. The color registers are divided into two groups: the player/missile colors and the playfield colors. The playfield color registers are utilized by the different screen modes as shown in Appendix H. The player/missile color registers are not used by the standard OS.

B7 PCOLR0 - PCOLR3 [02C0,4] -- Player/missile graphics colors

Each color variable is stored in the corresponding hardware register as shown below:

```
PCOLR0 [02C0]     COLPM0 [D012]
PCOLR1 [02C1]     COLPM1 [D013]
PCOLR2 [02C2]     COLPM2 [D014]
PCOLR3 [02C3]     COLPM3 [D015]
```

Each color variable has the format shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
| color | lum |x|
+-+-+-+-+-+-+-+-+
```

See Appendix H for information regarding the color and luminance field values.

B8 COLOR0 - COLOR4 [02C5,5] -- Playfield colors

Each color variable is stored in the corresponding hardware register as shown below:

```
COLOR0 [02C4]     COLPF0 [D016]
COLOR1 [02C5]     COLPF1 [D017]
COLOR2 [02C6]     COLPF2 [D018]
COLOR3 [02C7]     COLPF3 [D019]
COLOR4 [02C8]     COLBK  [D01A]
```

Each color variable has the format shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
| color | lum |x|
+-+-+-+-+-+-+-+-+
```

See Appendix H for information regarding the color and luminance field values.

Text Scrolling

The text screen or split-screen text window "scrolls" upward whenever one of the two conditions shown below occurs:

- o A text line at the bottom row of the screen extends past the right margin.
- o A text line at the bottom row of the screen is terminated by an EOL.

Scrolling has the effect of removing the entire logical line that starts at the top of the screen and then moving all subsequent lines upward to fill in the void. The cursor will also move upward if the logical line deleted exceeds one physical line.

B9 SCRFLG* [02BB,1] -- Scroll flag

SCRFLG is a working variable that counts the number of physical lines minus 1 that were deleted from the top of the screen; since a logical line ranges in size from 1 to 3, SCRFLG ranges from 0 to 2.

Attract Mode

Attract mode is a mechanism that protects the television screen from having patterns "burned into" the phosphors due to a fixed display being left on the screen for extended periods of time. When the computer is left unattended for more than 9 minutes, the color intensities are limited to 50 percent of maximum and the hues are continually varied every 8.3 seconds. Pressing any keyboard data key will be sufficient to remove the attract mode for 9 more minutes.

As part of the stage 2 VBLANK process, the color registers from the data base are sent to the corresponding hardware color registers; before they are sent, they undergo the following transformation:

    hardware register = database variable XOR COLRSH AND DRKMSK

Normally COLRSH = $00 and DRKMSK = $FE, thus making the above calculation a null operation; however, once attract mode becomes active, COLRSH = the content of RTCLOK+1 and DRKMSK = $F6, that has the effect of modifying all of the colors and keeping their luminance always below the 50 percent level.

Since RTCLOK+1 is incremented every 256/60 of a second and since the least significant bit of COLRSH is of no consequence, a color/lum change will be effected every 8.3 seconds (512/60).

B10 ATRACT [004D,1] -- Attract mode timer and flag

ATRACT is the timer (and flag) that controls the initiation and termination of attract mode. Whenever a keyboard key is pressed, the keyboard IRQ service routine sets ATRACT to zero, thus terminating attract mode; the [BREAK] key logic behaves accordingly. As part of the stage 1 VBLANK process, ATRACT is incremented every 4 seconds; if the value exceeds 127 (after 9 minutes without keyboard activity), the value of ATRACT will be set to $FE and will retain that value until attract mode is terminated.

Since the attract mode is prevented and terminated by the OS based only upon keyboard activity, some users can want to reset ATRACT based upon Atari-controller event detection, user-controlled Serial I/O bus activity or any other signs of life.

B11 COLRSH* [004F,1] -- Color shift mask

COLRSH has the value $00 when attract mode is inactive, thus effecting no change to the screen colors; when attract mode is active, COLRSH contains the current value of the timer variable middle digit (RTCLOK+1).

B12 DRKMSK* [004E,1] -- Dark (luminance) mask

DRKMSK has the value $FE when attract mode is inactive, which does not alter the luminance; and has the value $F6 when attract mode is active, which forces the most significant bit of the luminance field to zero, thus guaranteeing that the luminance will never exceed 50 percent.

Tabbing

See Section 5 for a discussion of the use of tabs in conjunction with the Screen Editor.

B13 TABMAP [02A3,15] -- Tab stop setting map

The tab settings are retained in a 15-byte (120 bit) map, where a bit value of 1 indicates a tab setting; the diagram below shows the mapping of the individual bits to tab positions.

```
  7   6   5   4   3   2   1   0
+---+---+---+---+---+---+---+---+
! 0 ! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 !   TABMAP+0
+---+---+---+---+---+---+---+---+
! 8 ! 9 ! 10! 11! 12! 13! 14! 15!         +1
+---+---+---+---+---+---+---+---+
!                               !
=                               =
!                               !
+---+---+---+---+---+---+---+---+
!112!113!114!115!116!117!118!119!         +14
+---+---+---+---+---+---+---+---+
```

Whenever the Display Handler or Screen Editor is opened, this map is initialized to contain the value of $01 in every byte, thus providing the default tab stops at 7, 15, 23, etc.

Logical Text Lines

The text screen is invisibly divided into logical lines of text, each comprising from one to three physical lines of text. The screen is initialized to 24 logical lines of one physical line each; but data entry and/or data insertion can increase the size of a logical line to two or three physical lines.

B14 LOGMAP* [02B2,4] -- Logical line starting row map

The beginning physical line number for each logical line on the screen is retained in a four byte (32 bit) map, where a bit value of one indicates the start of a logical line; the diagram below shows the mapping of the individual bits to physical line (row) numbers.

```
 7  6  5  4  3  2  1  0
+--+--+--+--+--+--+--+--+
! 0! 1! 2! 3! 4! 5! 6! 7!      LOGMAP+0
+--+--+--+--+--+--+--+--+
! 8! 9!10!11!12!13!14!15!            +1
+--+--+--+--+--+--+--+--+
!16!17!18!19!20!21!22!23!            +2
+--+--+--+--+--+--+--+--+
!  !  !  !  !  !  !  !  !            +3
+--+--+--+--+--+--+--+--+
```

The map bits are all set to 1 whenever the text screen is opened or cleared. From that point, the map is updated as logical lines are entered, edited and deleted from the screen.

B15 LOGCOL* [0063,1] -- Cursor/logical line column number

LOGCOL contains the logical-line column number for the current cursor position; note that a logical line can comprise up to three physical lines. This variable is for the internal use of the Display Handler.

Split Screen

The Display Handler and Screen Editor together support the operation of a split-screen mode (see Section 5) in which the main portion of the screen is in one of the graphics modes and is controlled by the Display Handler, and there are 4 physical lines in the text window at the bottom of the screen which is controlled by the Screen Editor.

B16 BOTSCR* [02BF,1] -- Text screen lines count

BOTSCR contains the number of lines of text for the current screen: 24 for mode 0 or 4 for a split-screen mode. The Handler also uses this variable as an indication of the split-screen status; tests are made for the specific values 4 and 24.

DRAW/FILL Function

The DRAW function line drawing algorithm is shown below translated to the PASCAL language from assembly language.

```
NEWROW := ROWCRS; NEWCOL := COLCRS;

DELTAR := ABS (NEWROW-OLDROW);
ROWINC := SIGN (NEWROW-OLDROW); { +1 or -1 }

DELTAC := ABS (NEWCOL-OLDCOL);
COLINC := SIGN (NEWCOL-OLDCOL); { +1 or -1 }

ROWAC := 0; COLAC := 0;
ROWCRS := OLDROW; COLCRS := OLDCOL;

COUNTR := MAX (DELTAC,DELTAR);
ENDPT := COUNTR;
IF COUNTR = DELTAC
   THEN ROWAC := ENDPT DIV 2
   ELSE COLAC := ENDPT DIV 2;

WHILE COUNTR > 0 DO
   BEGIN
```

```
        ROWAC := ROWAC + DELTAR;
        IF ROWAC >= ENDPT
           THEN
              BEGIN
                 ROWAC := ROWAC - ENDPT;
                 ROWCRS := ROWCRS + ROWINC
              END;

        COLAC := COLAC + DELTAC;
        IF COLAC >= ENDPT
           THEN
              BEGIN
                 COLAC := COLAC - ENDPT;
                 COLCRS := COLCRS + COLINC
              END;

        PLOT_POINT; { point defined by ROWCRS and COLCRS }

        IF FILFLG <> 0 THEN FILL_LINE;

        COUNTR := COUNTR - 1

     END;
```

The FILL function algorithm (FILL_LINE above) is described briefly in Section 5.

B17 FILDAT [02FD,1] -- Fill data

FILLDAT contains the fill region data value as part of the calling sequence for a FILL command as described in Section 5.

B18 FILFLG* [02B7,1] -- Fill flag

FILFLG indicates to the shared code within the Display Handler whether the current operation is FILL (FILFLG <> 0) or DRAW (FILFLG = 0).

B19 NEWROW* [0060,1] and NEWCOL* [0061,2] -- Destination point

NEWROW and NEWCOL are initialized to the values in ROWCRS and COLCRS, which represent the destination endpoint of the DRAW/FILL command. This is done so that ROWCRS and COLCRS can be altered during the performance of the command.

B20 HOLD4* [02BC,1] -- Temporary storage

HOLD4 is used to save and restore the value in ATACHR during the FILL process; ATACHR is temporarily set to the value in FILDAT to accomplish the filling portion of the command.

B21 ROWINC* [0079,1] and COLINC* [007A,1] -- Row/column increment/decrement

ROWINC and COLINC are the row and column increment values; they are each set to +1 or -1 to control the basic direction of line drawing. ROWINC and COLINC represent the signs of NEWROW - ROWCRS and NEWCOL - COLCRS, respectively.

B22 DELTAR* [0076,1] and DELTAC* [0077,2] -- Delta row and delta column

DELTAR and DELTAC contain the absolute values of NEWROW - ROWCRS and NEWCOL - COLCRS, respectively; together with ROWINC and COLINC, they define the slope of the line to be drawn.

B23 COUNTR* [007E,2] -- Draw iteration count

COUNTR initially contains the larger of DELTAR and DELTAC, that is the number of iterations required to generate the desired line. COUNTR is then decremented after every point on the line is plotted, until it reaches a value of zero.

B24 ROWAC* [0070,2] and COLAC* [0072,2] -- Accumulators

ROWAC and COLAC are working accumulators that control the row-and column-point plotting and increment (or decrement) function.

B25 ENDPT* [0074,2] -- Line length

ENDPT contains the larger of DELTAR and DELTAC, and is used in conjunction with ROWAC/COLAC and DELTAR/DELTAC to control the plotting of line points.

Displaying Control Characters

Often it is useful to have ATASCII control codes (such as CLEAR, CURSOR UP, etc). displayed in their graphic forms instead of having them perform their control function. This display capability is provided in two forms when outputting to the Screen Editor: 1) a data content form in which a special character (ESC) precedes each control character to be displayed and 2) a mode control form.

Escape (Display Following Control Character)

Whenever an ESC character is detected by the Screen Editor, the next character following this code is displayed as data, even if it would normally be treated as a control code; the EOL code is the sole exception. It is always treated as a control code. The sequence ESC ESC will cause the second ESC character to be displayed.

B26 ESCFLG* [02A2,1] -- Escape flag

ESCFLG is used by the Screen Editor to control the escape sequence function; the flag is set (to $80) by the detection of an ESC character ($1B) in the data stream and is reset (to 0) following the output of the next character.

Display Control Characters Mode

When it is desired to display ATASCII control codes other than EOL in their graphics form, but not have an ESC character associated with each control code, a display mode can be established by setting a flag in the data base. This capability is used by language processors when displaying high-level language statements, that can contain control codes as data elements.

B27 DSPFLG [02FE,1] -- Display control characters flag

When DSPFLG is nonzero, ATASCII control codes other than EOL are treated as data and displayed on the screen when output to the Screen Editor. When DSPFLG is zero, ATASCII control codes are processed normally.

DSPFLG is set to zero by Power-up and [SYSTEM.RESET].

Bit-Mapped Graphics

A number of temporary variables are used by the Display Handler when handling data elements (pixels) going to or from the screen; of interest here are those variables that are used to control the packing and unpacking of graphics data, where a memory byte typically contains more than one data element (for example, screen mode 8 contains 8 pixels per memory byte).

B28 DMASK* [02A0,1] -- Pixel location mask

DMASK is a mask that contains zeros for all bits that do not correspond to the specific pixel to be operated upon, and 1's for all bits that do correspond. DMASK can contain the values shown below in binary notation:

```
11111111  --  screen modes 1 and 2; one pixel per byte.

11110000  --  screen modes 9-11; two pixels per byte.
00001111

11000000  --  screen modes 3, 5 and 7; four pixels per byte.
00110000
00001100
00000011

10000000  --  screen modes 4, 6 and 8; eight pixels per byte.
01000000

00000010
00000001
```

B29 SHFAMT* [006F,1] -- Pixel justification

SHFAMT indicates the amount to shift the right-justified pixel data on output, or the amount to shift the input data to right justify it on input. The value is always the same as for DMASK prior to the justification process.

Internal Working Variables

B30 HOLD1* [0051,1] -- Temporary storage

B31 HOLD2* [029F,1] -- Temporary storage

B32 HOLD3* [029D,1] -- Temporary storage

B33 TMPCHR* [0050,1] -- Temporary storage

B34 DSTAT* [004C,1] -- Display status

B35 DINDEX [0057,1] -- Display mode

DINDEX contains the current screen mode obtained from the low order four bits of the most recent OPEN AUX1 byte.

B36 SAVMSC [0058,2] -- Screen Memory Address

SAVMSC contains the lowest address of the screen data region; the data at that address is displayed at the upper left corner of the screen.

B37 OLDCHR* [005D,1] -- Cursor character save/restore

OLDCHR retains the value of the character under the visible text cursor; this variable is used to restore the original character value when the cursor is moved.

B38 OLDADR* [005E,2] -- Cursor memory address

OLDADR retains the memory address of the current visible text cursor location; this variable is used in conjunction with OLDCHR (B37) to restore the original character value when the cursor is moved.

B39 ADRESS* [0064,2] -- Temporary storage

B40 MLTTMP/OPNTMP/TOADR* [0066,2] -- Temporary storage

B41 SAVADR/FRMADR* [0068,2] -- Temporary storage

B42 BUFCNT* [006B,1] -- Screen Editor current logical line size

B43 BUFSTR* [006C,2] -- Temporary storage

B44 SWPFLG* [007B,1] -- Split-screen cursor control

In split-screen mode, the graphics cursor data and the text window cursor data are frequently swapped as shown below in order to get the variables associated with the region being accessed into the ROWCRS-OLDADR variables.

```
ROWCRS B2 ------- TXTROW B4
COLCRS B2 ------- TXTCOL B4
DINDEX B35 ------ TINDEX B49
SAVMSC B36 ------ TXTMSC B52
OLDROW B3 ------- TXTOLD B53
OLDCOL B3 -------   "     "
OLDCHR B37 ------   "     "
OLDADR B38 ------   "     "
```

SWPFLG is used to keep track of what data set is currently in the ROWCRS-OLDADR region; SWPFLG is equal to $FF when split-screen text window cursor data is in the main region, otherwise SWPFLG is equal to 0.

B45 INSDAT* [007D,1] -- Temporary storage

B46 TMPROW* [02B8,1] and TMPCOL* [02B9,2] -- Temporary storage

B47 TMPLBT* [02A1,1] -- Temporary storage

B48 SUBTMP* [029E,1] -- Temporary storage

B49 TINDEX* [0293,1] -- Split screen text window screen mode

TINDEX is the split-screen text window equivalent of DINDEX and is always equal to zero when SWPFLG is equal to zero (see B44).

B50 BITMSK* [006E,1] -- Temporary storage

B51 LINBUF* [0247,40] -- Physical line buffer

LINBUF is used to temporarily buffer one physical line of text when the Screen Editor is moving screen data.

B52 TXTMSC [0294,2] -- Split screen memory address

TXTMSC is the split-screen text window version of SAVMSC (B36).

See B44 for more information.

B53 TXTOLD* [0296,6] -- Split screen cursor data

See B44 for more information.

Internal Character Code Conversion

Two variables are used to retain the current character being processed (for both reading and writing); ATACHR contains the value passed to or from CIO, and CHAR contains the internal code corresponding to the value in ATACHR. Because the hardware does not interpret ATASCII characters directly, the transformations shown below are applied to all text data read and written:

ATASCII CODE	INTERNAL CODE
00-1F	40-5F
20-3F	00-1F
40-5F	20-3F
60-7F	60-7F
80-9F	C0-DF

A0-BF	80-9F
C0-DF	A0-BF
E0-FF	E0-FF

See P26 for more information.

B54 ATACHR [02FB,1] -- Last ATASCII character or plot point

ATACHR contains the ATASCII value for the most recent character read or written, or the value of the graphics point. This variable can also be considered to be a parameter of the FILL/DRAW commands, as the value in ATACHR will determine the line color when a DRAW or FILL is performed.

B55 CHAR* [02FA,1] -- Internal character code

CHAR contains the internal code value for the most recent character read or written.

C. DISKETTE HANDLER

See Section 5 for a discussion of the resident Diskette Handler.

C1 BUFADR* [0015,2] -- Data buffer pointer

BUFADR acts as temporary page zero pointer to the current diskette buffer.

C2 DSKTIM* [0246,1] -- Disk format operation timeout time

DSKTIM contains the timeout value for SIO calling sequence variable DTIMLO (see Section 9). DSKTIM is set to 160 (which represents a 171-second timeout) at initialization time, and is updated after each diskette status request operation. It contains the value returned in the third byte of the status frame (see Section 5). Note that all diskette operations other than format have a fixed (7) second timeout, established by the Diskette Handler.

D. CASSETTE

See Section 5 for a general description of the Cassette Handler. The cassette uses the Serial I/O bus hardware, but does not conform with the Serial I/O bus protocol as defined in Section 9. Hence, the Serial

I/O utility (SIO) has cassette specific code within it. Some variables in this subsection are utilized by SIO and some by the Cassette Handler.

Baud Rate Determination

The input baud rate is assumed to be a nominal 600 baud, but will be adjusted, if necessary, by the SIO routine to account for drive-motor variations, stretched tape, etc. The beginning of every cassette record contains a pattern of alternating 1's and zeros that is used solely for speed correction; by measuring the time to read a fixed number of bits, the true-receive baud rate is determined and the hardware adjusted accordingly. Input baud rates ranging from 318 to 1407 baud can theoretically be handled using this technique.

The input baud rate is adjusted by setting the POKEY counter that controls the bit sampling period.

D1 CBAUDL* [02EE,1] and CBAUDH* [02EF,1] -- Cassette baud rate

Initialized to 05CC hex, which represents a nominal 600 baud. After baud rate calculation, these variables will contain POKEY counter values for the corrected baud rate.

D2 TIMFLG* [0317,1] -- Baud rate determination timeout flag

TIMFLG is used by SIO to timeout an unsuccessful baud rate determination. The flag is initially set to 1, and if it attains a value of zero (after 2 seconds) before the first byte of the cassette record has been read, the operation will be aborted. See also H24.

D3 TIMER1* [030C,2] and TIMER2* [0310,2] -- Baud rate timers

These timers contain reference times for the beginning and end of the fixed bit pattern receive period. The first byte of each timer contains the then current vertical line counter value read from ANTIC, and the second byte of each timer contains the then current value of the least significant byte of the OS real time clock (RTCLOK+2).

The difference between the timers is converted to raster pair counts and is then used to perform a table lookup with interpolation to determine the new values for CBAUDL and CBAUDH.

D4 ADDCOR* [030E,1] -- Interpolation adjustment variable

ADDCOR is a temporary variable used for the interpolation calculation of the above computation.

D5 TEMP1* [0312,2] -- Temporary storage

D6 TEMP3* [0315,1] -- Temporary storage

D7 SAVIO* [0316,1] -- Serial in data detect

SAVIO is used to retain the state of SKSTAT [D20F] bit 4 (serial data in); it is used to detect (and is updated after) every bit arrival.

Cassette Mode

D8 CASFLG* [030F,1] -- Cassette I/O flag

CASFLG is used internally by SIO to control the program flow through shared code. A value of zero indicates that the current operation is a standard Serial I/O bus operation, and a nonzero value indicates a cassette operation.

Cassette Buffer

D9 CASBUF* [03FD,131] -- Cassette record buffer

CASBUF is the buffer used by the Cassette Handler for the packing and unpacking of cassette-record data, and by the initialization cassette-boot logic. The format for the standard cassette record in the buffer is shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0 1 0 1 0 1 0 1!      CASBUF+0
+-+-+-+-+-+-+-+-+
!0 1 0 1 0 1 0 1!            +1
+-+-+-+-+-+-+-+-+
! control byte  !            +2
+-+-+-+-+-+-+-+-+
!      128      !            +3
=     data      =
!     bytes     !            +130
+-+-+-+-+-+-+-+-+
```

See Section 5 for an explanation of the standard cassette-record format.

D10 BLIM* [028A,1] -- Cassette record data size

BLIM contains the count of the number of data bytes in the current cassette record being read. BLIM will have a value ranging from 1 to 128, depending upon the record control byte as explained in Section 5.

D11 BPTR* [003D,1] -- Cassette-record data index

BPTR contains an index into the data portion of the cassette record being read or written. The value will range from 0 to the then current value of BLIM. When BPTR equals BLIM then the buffer (CASBUF) is full if writing or empty if reading.

Internal Working Variables

D12 FEOF* [003F,1] -- Cassette end-of-file flag

FEOF is used by the Cassette Handler to flag the detection of an end of file condition (control byte = $FE). FEOF equal to zero indicates that an EOF has not yet been detected, and a nonzero value indicates that an EOF has been detected. The flag is reset at every OPEN.

D13 FTYPE* [003E,1] -- Interrecord gap type

FTYPE is a copy of ICAX2Z from the OPEN command and indicates the type of interrecord gap selected; a positive value indicates normal record gaps, and a negative value indicates continuous mode gaps.

D14 WMODE* [0289,1] -- Cassette read/write mode flag

WMODE is used by the Cassette Handler to indicate whether the current operation is a read or write operation; a value of zero indicates read, and a value of $80 indicates write.

D15 FREQ* [0040,1] -- Beep count

FREQ is used to retain and count the number of beeps requested of the BEEP routine by the Cassette Handler during the OPEN command process.

E. KEYBOARD

See Section 5 for a general description of the Keyboard Handler.

Key Reading and Debouncing

The console key code register is read in response to an IRQ interrupt that is generated whenever a key stroke is detected by the hardware. The key code is compared with the prior key code accepted (CH1); if the codes are not identical, then the new code is accepted and stored in the key code FIFO (CH) and in the prior key code variable (CH1). If the codes are identical, then the new code is accepted only if a suitable key debounce delay has transpired since the prior value was accepted.

If the key code read and accepted is the code for [CTRL] 1, then the display start/stop flag (SSFLAG) is complemented and the value is not stored in the key code FIFO (CH).

In addition to the reading of the key data, SRTIMR is set to $30 for all interrupts received (see E8), and ATRACT is set to 0 whenever a new code is accepted (see B10).

The Keyboard Handler obtains all key data from CH; whenever a code is extracted from that 1-byte FIFO, the Handler stores a value of $FF to the FIFO to indicate that the code has been read. See Section 5 for further discussion of the Keyboard Handler's processing of the key codes.

E1 CH1* [02F2,1] -- Prior keyboard character code.

CH1 contains the key code value of the key most recently read and accepted.

E2 KEYDEL* [02F1,1] -- Debounce delay timer.

KEYDEL is set to a value of 3 whenever a key code is accepted, and is decremented every 60th of a second by the stage 2 VBLANK process (until it reaches zero).

E3 CH [02FC,1] -- Keyboard character code FIFO.

CH is a 1-byte FIFO that contains either the value of the most recently read and accepted key code or the value $FF (which indicates that the FIFO is empty). The FIFO is normally read by the Keyboard Handler, but can be read by a user program.

Key data can also be stored into CH by the Autorepeat logic as explained in the discussion relating to E8.

Special Functions

Start/Stop

Display Handler and Screen Editor output to the text or graphics mode screen can be stopped and started (without losing any of the output data) through the use of the [CTRL] 1 key combination. Each key depression toggles a flag that is monitored by the above mentioned Handlers. When the flag is nonzero, the handlers wait for it to go to zero before continuing any output.

E4 SSFLAG [02FF,1] -- Start/stop flag

The flag is normally zero, indicating that screen output is not to be stopped. The flag is complemented by every occurrence of the [CTRL] 1 key combination by the keyboard IRQ service routine.

The flag is set to zero upon power-up, [SYSTEM.RESET] or [BREAK] key processing.

[BREAK] Key

E5 BRKKEY [0011,1] -- [BREAK] key flag

BRKKEY is used to indicate that the [BREAK] key has been pressed. The value is normally nonzero and is set to zero whenever the [BREAK] key is pressed. The code that detects and processes the [BREAK] condition (flag = 0) should set the flag nonzero again.

BRKKEY is monitored by the following OS routines: Keyboard Handler, Display Handler, Screen Editor, Cassette Handler, xx? The detection of a [BREAK] condition during an I/O operation will cause the operation to be aborted and a status of $80 to be returned to the user.

The flag is set to nonzero upon Power-up, [SYSTEM.RESET] or upon aborting a pending I/O operation.

[SHIFT]/[CONTROL] Lock

The keyboard control has three different modes for code generation that apply to the alphabetic keys A through Z: 1) normal, 2) caps lock, and 3) control lock.

In normal mode, all unmodified alphabetic character keys generate the lowercase letter ATASCII code ($61-7A).

In caps lock mode, all unmodified alphabetic character keys generate the uppercase letter ATASCII code ($41-5A).

In control lock mode, all unmodified alphabetic character keys generate the control letter ATASCII code ($01-1A).

In all three modes, any alphabetic character key that is modified (by being pressed in conjunction with the [SHIFT] or [CTRL] key) will generate the desired modified code.

E6 SHFLOK [02BE,1] -- Shift/control lock control flag

SHFLOK normally has one of three values:

    $00 = normal mode (no locks in effect).
    $40 = caps lock.
    $80 = control lock.

SHFLOK is set to $40 upon Power-up and [SYSTEM.RESET] and is modified thereafter by the OS only when the [CAPS.LOWER] key is pressed (either by itself or in conjunction with the [SHIFT] or [CTRL] key).

E7 HOLDCH* [007C,1] -- Character holding variable

HOLDCH is used to retain the current character value prior to the [SHIFT]/[CONTROL] logic process.

Autorepeat

The Autorepeat feature responds to the continuous depression of a keyboard key by replicating the key code 10 times per second, after an initial 1/2 second delay. The timer variable SRTIMR is used to control both the initial delay and the repeat rate.

Whenever SRTIMR is equal to zero and a key is being held down, the value of the key code is stored in the key code FIFO (CH). This logic is part of the stage 2 VBLANK process.

E8 SRTIMR* [022B,1] -- Autorepeat timer

SRTIMR is controlled by two independent processes: 1) the keyboard IRQ service routine, which establishes the initial delay value and 2) the stage 2 VBLANK routine that establishes the repeat rate, decrements the timer and implements the auto repeat logic.

Inverse Video Control

The Keyboard Handler allows the direct generation of more than half of the 256 ATASCII codes; but codes $80-9A and codes $A0- FC can be generated only with the "inverse video mode" active. The ATARI key acts as an on/off toggle for this mode, and all characters (except for screen editing control characters) will be subject to inversion when the mode is active.

E9 INVFLG [02B6,1] -- Inverse video flag

INVFLG is normally zero, indicating that normal video ATASCII codes (bit 7 = 0) are to be generated from keystrokes; whenever INVFLG is nonzero, inverse video ATASCII codes (bit 7 = 1) will be generated. The special control codes are exempt from this bit manipulation.

INVFLG is set to zero by power-up and system reset.

The Keyboard Handler inverts bit 7 of INVFLG whenever the ATARI key is pressed; the lower order bits are not altered and are assumed to be zero.

The Keyboard Handler's "exclusive or's" (XOR's) the ATASCII key data with the value in INVFLG at all times; the normal values of $00 and $80 thus lead to control of the inverse video bit (bit 7).

Console Keys: [SELECT],[START], and [OPTION]

The console keys are sensed directly from the hardware register CONSOL [D01F]; see the ATARI Home Computer Hardware Manual for details.

F. PRINTER

See Section 5 for a general description of the Printer Handler.

Printer-Buffer

F1 PRNBUF* [03C0,40] -- Printer-record buffer

PRNBUF is the buffer used by the Printer Handler for packing printer data to be sent to the device controller. The buffer is 40 bytes long and contains nothing but printer data.

F2 PBUFSZ* [001E,1] -- Printer-record size

PBUFSZ contains the size of the Printer-record for the current mode selected; the modes and respective sizes (in decimal bytes) are shown below:

Normal	40	
Double width	20	(not currently supported by the device)
Sideways	29	
Status request	4	

F3 PBPNT* [001D,1] -- Printer-buffer index

PBPNT contains the current index to the Printer-buffer. PBPNT ranges in value from zero to the value of PBUFSZ.

Internal Working Variables

F4 PTEMP* [001F,1] -- Printer Handler temporary data save

PTEMP is used by the Printer Handler to temporarily save the value of a character to be output to the printer.

F5 PTIMOT* [001C,1] -- Printer timeout value

PTIMOT contains the timeout value for SIO calling sequence variable DTIMLO (see Section 9); PTIMOT is set to 30 (which represents a 32 second timeout) at intialization time, and is updated after each printer status request operation to contain the value returned in the third byte of the status frame (see Section 5).

G. CENTRAL I/O ROUTINE (CIO)

See Section 5 for a description of the Central I/O Utility.

User Call Parameters

CIO call parameters are passed primarily through an I/O Control Block (IOCB); although additional device status information can be returned in DVSTAT, and Handler information is obtained from the device table (HATABS).

I/O Control Block

IOCB is the name applied collectively to the 16 bytes associated with each of the 8 provided control structures; see Section 5.

G1 IOCB [0340,16] -- I/O Control Block

The label IOCB is the location of the first byte of the first IOCB in the data base. For VIDs G2 through G10, the addresses given are for IOCB #0 only, the addresses for all of the IOCB's are shown below:

0340-034F	IOCB #0
0350-035F	IOCB #1
0360-036F	IOCB #2
0370-037F	IOCB #3
0380-038F	IOCB #4
0390-039F	IOCB #5
03A0-03AF	IOCB #6
03B0-03BF	IOCB #7

G2 ICHID [0340,1] -- Handler ID

See Section 5. Initialized to $FF at power-up and system reset.

G3 ICDNO [0341,1] -- Device number

See Section 5.

G4 ICCOM [0342,1] -- Command byte

See Section 5.

G5 ICSTA [0343,1] -- Status

See Section 5.

G6 ICBAL,ICBAH [0344,2] -- Buffer address

See Section 5.

G7 ICPTL,ICPTH [0346,2] -- PUT BYTE vector

See Section 5. Initialized to point to CIO's "IOCB not OPEN" routine at power-up and system reset.

G8 ICBLL,ICBLH [0348,2] -- Buffer length / byte count

See Section 5.

G9 ICAX1,ICAX2 [034A,2] -- Auxiliary information

See Section 5.

G10 ICSPR [034C,4] -- Spare bytes for Handler use

There is no fixed assignment of these four bytes; the Handler associated with an IOCB can or may not use these bytes.

Device Status

G11 DVSTAT [02EA,4] -- Device status

See Section 5 for a discussion of the GET STATUS command.

Device Table

G12 HATABS [031A,38] -- Device table

See Section 9 for a description of the device table.

CIO/Handler Interface Parameters

Communication between CIO and a Handler is accomplished using the 6502 machine registers, and a data structure called the Zero-page IOCB (ZIOCB). The ZIOCB is essentially a copy of the particular IOCB being used for the current operation.

Zero-Page IOCB

G13 ZIOCB (IOCBAS) [0020,16] -- Zero-page IOCB

The Zero-page IOCB is an exact copy (except as noted in the discussions that follow) of the IOCB specified by the 6502 X register upon entry to CIO; CIO copies the outer level IOCB to the Zero-page IOCB, performs the indicated function, moves the (possibly altered) Zero-page IOCB back to the outer level IOCB, and then returns to the caller.

Although both the outer level IOCB and the Zero-page IOCB are defined to be 16 bytes in size, only the first 12 bytes are moved by CIO.

G14 ICHIDZ [0020,1] -- Handler index number

See Section 5. Set to $FF on CLOSE.

G15 ICDNOZ [0021,1] -- Device drive number

See Section 5.

G16 ICCOMZ [0022,1] -- Command byte

See Section 5.

G17 ICSTAZ [0023,1] -- Status byte

See Section 5.

G18 ICBALZ,ICBALH [0024,2] -- Buffer address

See Section 5. This pointer variable is modified by CIO in the course of processing some commands; however, the original value is restored before returning to the caller.

G19 ICPTLZ,ICPTHZ

See Section 5. Set to point to CIO's "IOCB not OPEN" routine on CLOSE.

G20 ICBLLZ,ICBLHZ [0028,2] -- Buffer length / byte count

See Section 5. This double-byte variable, which starts out representing the buffer length, is modified by CIO in the course of processing some commands; then, before returning to the caller, the transaction byte count is stored therein.

G21 ICAX1Z,ICAX2Z [002A,2] -- Auxiliary information

See Section 5.

G22 ICSPRZ (ICIDNO,CIOCHR) [002C,4] -- CIO working variables

ICSPRZ and ICSPRZ+1 are used by CIO in obtaining the appropriate Handler entry point from the handler's vector table (see Section 9).

ICSPRZ+2 is also labeled ICIDNO and retains the value of the 6502 X register from CIO entry. The X register is loaded from ICIDNO as CIO returns to the caller.

ICSPRZ+3 is also labeled CIOCHR and retains the value of the 6502 A register from CIO entry, except for data reading type commands, in which case the most recent data byte read is stored in CIOCHR. The 6502 A register is loaded from CIOCHR as CIO returns to the caller.

Internal Working Variables

G23 ICCOMT* [0017,1] -- Command table index

ICCOMT is used as an index to CIO's internal command table, which maps command byte values to Handler entry offsets (see Section 9 for more information). ICCOMT contains the value from ICCOMZ except when ICCOMZ is greater than $0E, in which case ICCOMT is set to $0E.

G24 ICIDNO* [002E,1] -- CIO call X register save/restore

See G22.

G25 CIOCHR* [002F,1] -- CIO call A register save/restore

See G22.

H. SERIAL I/O ROUTINE (SIO)

See Section 9 for discussions relating to SIO.

User Call Parameters

SIO call parameters are passed primarily through a Device Control Block; although an additional "noisy bus" option exists that is selectable through a separate variable.

Device Control Block

H1 DCB [0300,12] -- Device Control Block

DCB is the name applied collectively to the 12 bytes at locations 0300-030B. These bytes provide the parameter passing mechanism for SIO and are described individually below.

H2 DDEVIC [0300,1] -- Device bus ID

See Section 9.

H3 DUNIT [0301,1] -- Device unit number

See Section 9.

H4 DCOMND [0302,1] -- Device command

See Section 9.

H5 DSTATS [0303,1] -- Device status

See Section 9.

H6 DBUFLO,DBUFHI [0304,2] -- Handler buffer address

See Section 9.

H7 DTIMLO [0306,1] -- Device timeout

See Section 9.

H8 DBYTLO,DBYTHI [0308,2] -- Buffer length / byte count

See Section 9.

H9 DAUX1, DAUX2 [030A, 2] -- Auxiliary information

See Section 9.

Bus Sound Control

H10 SOUNDR [0041, 1] -- Quiet/noisy I/O flag

SOUNDR is a flag used to indicate to SIO whether noise is to be generated on the television audio circuit when Serial I/O bus activity is in progress. SOUNDR equal to zero indicates that sound is to be inhibited, and nonzero indicates that sound is to be enabled. SIO sets SOUNDR to 3 at power-up and system reset.

Serial Bus Control

Retry Logic

SIO will attempt one complete command retry if the first attempt is not error free, where a complete command try consists of up to 14 attempts to send (and acknowledge) a command frame, followed by a single attempt to receive COMPLETE and possibly a data frame.

H11 CRETRY* [0036, 1] -- Command frame retry counter

CRETRY controls the inner loop of the retry logic, that associated with sending and receiving an acknowledgement of the command frame. CRETRY is set to 13 by SIO at the beginning of every command initiation, thus allowing for an initial attempt and up to 13 additional retries.

H12 DRETRY* [0037, 1] -- Device retry counter

DRETRY controls the outer loop of the retry logic, that associated with initiating a command retry after a failure subsequent to the command frame acknowledgement. DRETRY is set to 1 by SIO at entry, thus allowing for an initial attempt and 1 additional retry.

Checksum

The Serial I/O bus protocol specifies that all command and data frames must contain a checksum validation byte; this byte is the arithmetic sum (with end-around carry) of all of the other bytes in the frame.

H13 CHKSUM* [0031,1] -- Checksum value

CHKSUM contains the frame checksum as computed by SIO for all frame transfers.

H14 CHKSNT* [003B,1] -- Checksum sent flag

CHKSNT indicates to the serial bus transmit interrupt service routine whether the frame checksum byte has been sent yet. CHKSNT equal to zero indicates that the checksum byte has not yet been sent; after the checksum is sent, CHKSNT is then set nonzero.

H15 NOCKSM* [003C,1] -- No checksum follows data flag

NOCKSM is a flag used to communicate between the SIO top level code and the Serial bus receive interrupt service routine that the next input will not be followed by a checksum byte. A value of zero specifies that a checksum byte will follow, nonzero specifies that a checksum byte will not follow.

Data Buffering

General Buffer Control

H16 BUFRLO* [0032,1] and BUFRHI* [0033,1] -- Next byte address

BUFRLO and BUFRHI comprise a pointer to the next buffer location to be read from or written to. For a data frame transfer, the pointer is initially set to the value contained in the SIO call parameters DBUFLO and DBUFHI, and is then incremented by the interrupt service routines as a part of normal bus data transfer. For a command frame transfer, the pointer is set to point to the SIO-maintained command frame output buffer.

H17 BFENLO* [0034,1] and BFENHI* [0035,1] -- Buffer end address

BFENLO/BFENHI form a pointer to the byte following the last frame data byte (not including the checksum) to be sent or received.

BFENLO/BFENHI is the arithmetic sum of BUFRLO/BUFRHI plus the frame size plus -1.

Command Frame Output Buffer

See Section 9 for the command frame format and description.

H18 CDEVIC* [023A,1] -- Command frame device ID

CDEVIC is set to the value obtained by adding SIO call parameter DDEVIC to DUNIT and subtracting 1.

H19 CCOMND* [023B,1] -- Command frame command.

CCOMND is set to the value obtained from SIO call parameter DCOMND.

H20 CAUX1* [023C,1] and CAUX2* [023D,1] -- Auxiliary information

CAUX1 and CAUX2 are set to the values obtained from SIO call parameters DAUX1 and DAUX2, respectively.

Receive/Transmit Data Buffering

H21 BUFRFL* [0038,1] -- Buffer full flag

BUFRFL is a flag used by the serial bus receive interrupt service routine to indicate when the main portion of a bus frame has been received -- all but the checksum byte. BUFRFL equal to zero indicates that the main portion has not been completely received, a nonzero value indicates that the main portion has been received.

H22 RECVDN* [0039,1] -- Receive frame done flag

RECVDN is a flag used by SIO to communicate between the Serial bus receive interrupt service routine and the main SIO code. The flag is initially set to zero by SIO, and later set nonzero by the interrupt service routine after the last byte of a bus frame has been received.

H23 TEMP* [023E,1] -- SIO 1-byte I/O data

TEMP is used to receive 1-byte responses from serial bus controllers, such as ACK, NAK, COMPLETE or ERROR.

H24 XMTDON* [003A,1] -- Transmit frame done flag

XMTDON is a flag used by SIO to communicate between the Serial bus transmit interrupt service routine and the main SIO code. The flag is initially set to zero by SIO, and later set nonzero by the interrupt service routine after the last byte of a bus frame has been transmitted.

SIO Timeout

SIO uses System Timer 1 to provide the timeout capability for various operations initiated internally. See Section 6 for a discussion of the capabilities of the System Timers. TIMFLG is the flag used to communicate between SIO and the timer initiated code pointed to by CDTMA1.

H25 TIMFLG* [0317,1] -- SIO operation timeout flag

TIMFLG is used to indicate a timeout situation for a bus operation . The flag is initially set to 1, and if it attains a value of zero (after the timeout period) before the current operation is complete, the operation will be aborted. See also D2.

H26 CDTMV1* [0218,2] -- System Timer 1 value

This 2-byte count takes on various values depending upon the operation being timed. See also P4.

H27 CDTMA1* [0226,2] -- System Timer 1 address

This vector always points to the JTIMER routine, whose only function is to set TIMFLG to zero. This vector is initialized by SIO before every use, so that System Timer 1 can be used by any process that does not use SIO within a timing function. See also P5.

Internal Working Variables

H28 STACKP* [0318,1] -- Stack pointer save/restore

STACKP contains the value of the 6502 SP register at entry to SIO; this is retained to facilitate a direct error exit from an SIO subroutine.

H29 TSTAT* [0319,1] -- Temporary status

TSTAT is used to return the operation status from the WAIT routine and will contain one of the SIO status byte values as shown in Appendix B.

H30 ERRFLG* [023F,1] -- I/O error flag

ERRFLG is used for communication between the WAIT routine and the outer level SIO code. ERRFLG is normally zero, but is set to $FF when a device responds with an invalid response byte.

H31 STATUS* [0030,1] -- SIO operation status

STATUS is a zero-page variable that is used within SIO to contain the operation status that will be stored to the calling sequence parameter variable DSTATS when SIO returns to the caller.

H32 SSKCTL* [0232,1] -- SKCTL copy

SSKCTL is utilized by SIO to keep track of the content of the SKCTL [D20F] register, which is a write-only register.

J. ATARI CONTROLLERS

The ATARI controllers are read as part of the Stage 2 VBLANK process. The encoded data is partially decoded and processed as shown in the subsections that follow.

Joysticks

Up to four joystick controllers can be attached to the computer console, each with a 9-position joystick plus a trigger button.

J1 STICK0 - STICK3 [0278,4] -- Joystick position sense

The 4 joystick position sense variables contain a bit-encoded position sense as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0|R|L|D|U|
+-+-+-+-+-+-+-+-+
```

```
where: R = 0 indicates joystick RIGHT sensor true.
       L = 0 indicates joystick LEFT sensor true.
       D = 0 indicates joystick DOWN sensor true.
       U = 0 indicates joystick UP sensor true.
```

Nine unique combinations are possible, indicating the possible joystick positions shown below:

CENTER	$0F
UP	$0E
UP/RIGHT	$06
RIGHT	$07
DOWN/RIGHT	$05
DOWN	$0D
DOWN/LEFT	$09
LEFT	$0B
UP/LEFT	$0A

J2 STRIG0 - STRIG3 [0284,4] -- Joystick trigger sense

The four joystick trigger sense variables each contain a single bit indicating the position of the joystick trigger as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0|T|
+-+-+-+-+-+-+-+-+
```

where: T = 0 indicates trigger pressed.

Paddles

Up to eight paddle controllers can be connected to the computer, each with a potentiometer and a trigger sense.

J3 PADDL0 - PADDL7 [0270,8] -- Paddle position sense

There is a single-byte variable associated with each paddle position sense; the values range from 228 for full counterclockwise rotation to 1 for full clockwise rotation.

The paddle values are often converted by the user, as shown below, to give a result of 0 for full counterclockwise rotation and 227 for full clockwise rotation:

```
VALUE := 228 - PADDLX;
```

J4 PTRIG0 - PTRIG7 [027C,8] -- Paddle trigger sense

The 8-paddle trigger sense variables each contain a single bit indicating the position of the paddle trigger as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
!0 0 0 0 0 0 0!T!
+-+-+-+-+-+-+-+-+
```

where: T = 0 indicates trigger pressed.

Light Pen

The OS reads the position of a single light pen and stores the horizontal and vertical position codes in two variables; these codes are not the same as the actual screen coordinates. The pen position codes for different portions of the screen are shown below:

Left edge -- 67.
Codes increase in increments of one to a value of 227, then go to 0 and continue to increase monotonically (one count per color clock).
Right edge -- 7.

Upper edge -- 16.
Codes increase in increments of one (one count per two raster lines). Lower edge -- 111.

The light pen hardware will read and latch the pen position 60 times per second, independent of the pen button position, which is separately sensed.

In order for the light pen to operate it must be positioned over a portion of the screen which has sufficient luminance to activate the photosensor in the pen; a blank (dark) screen will generally not provide enough luminance to utilize the light pen.

J5 LPENH [0234,1] -- Light pen horizontal position code

LPENH contains the horizontal position code for the light pen; the algorithm below (written in Pascal) shows the conversion from position code to screen coordinate (screen mode 7):

```
IF LPENH < 33          { check for rollover point }
   THEN                { adjust values to right of rollover }
```

```
        XPOS := LPENH + 227
     ELSE                { no adjustment to left of rollover point }
        XPOS := LPENH;
  XPOS := XPOS - 67;    { adjust for left edge offset }
  IF XPOS < 0 THEN XPOS := 0;
  IF XPOS > 159 THEN XPOS := 159;
```

J6 LPENV [0235, 1] -- Light pen vertical position code

LPENV contains the vertical position code for the light pen; the algorithm below (written in Pascal) shows the conversion from position code to screen coordinate (screen mode 7):

```
  YPOS := LPENV - 16;   { adjust for upper edge offset }
  IF YPOS < 0 THEN YPOS := 0;
  IF YPOS > 95 THEN YPOS := 95;
```

J7 STICK0 - STICK3 [0278, 4] -- Light pen button sense

The light pen button sense is encoded in one of STICK0 - STICK3 (depending upon the actual controller port used) as shown below:

```
      7             0
     +-+-+-+-+-+-+-+-+
     !       !0!0!0!T!
     +-+-+-+-+-+-+-+-+
```

where: T = 0 indicates the light pen button is pressed.

Driving Controllers

The driving controller has no position stops and thus allows unlimited rotation in either direction; the output of the controller is a 2-bit Gray code which can be used to determine the direction of rotation. The controller is sensed using the same internal hardware as the joystick, thus the same data base variables are used for both.

J8 STICK0 - STICK3 [0278,4] -- Driving controller sense

The 4 driving controller sense variables contain an encoded rotation (position) sense value, as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0 1 1|val|
+-+-+-+-+-+-+-+-+
```

where a clockwise rotation of the controller produces the following continuous sequence of four values (shown in hexadecimal):

OF,OD,OC,OE,OF,OD,.......

and a counterclockwise rotation of the controller produces the following continuous sequence of four values:

OF,OE,OC,OD,OF,OE,.......

J9 STRIG0 - STRIG3 [0284,4] -- Driving trigger sense

The four driving trigger sense variables each contain a single bit indicating the position of the driving trigger as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0|T|
+-+-+-+-+-+-+-+-+
```

where: T = 0 indicates trigger pressed.

K. DISK FILE MANAGER

See Section 5 for information relating to the Disk File Manager.

K1 FMSZPG* [0043,7] -- FMS reserved space

FMSZPG is the reserved space in the database for the variables shown below; the names associated with K2 through K5 are not in the system equate file.

K2 ZBUFP* [0043,2] -- Buffer pointer

K3 ZDRVA* [0045,2] -- Drive pointer

K4 ZSBA* [0047,2] -- Sector buffer pointer

K5 ERRNO* [0049,1] -- Error number

L. DISK UTILITY POINTER

L1 DSKUTL* [001A,2] -- Page-zero pointer variable

M. FLOATING POINT PACKAGE

See Section 8 for a description of the Floating Point Package.

M1 FR0 [00D4,6] -- FP register 0

M2 FRE* [00DA,6] -- FP register (internal)

M3 FR1 [00E0,6] -- FP register 1

M4 FR2* [00E6,6] -- FP register 2 (internal)

M5 FRX* [00EC,1] -- Spare (unused)

M6 EEXP* [00ED,1] -- Exponent value (internal)

M7 NSIGN* [00EE,1] -- Sign of mantissa (internal)

M8 ESIGN* [00EF,1] -- Sign of exponent (internal)

M9 FCHRFLG* [00F0,1] -- First character flag (internal)

M10 DIGRT* [00F1,1] -- Digits to right of decimal point

M11 CIX [00F2,1] -- Character index

M12 INBUFF [00F3,2] -- Input text buffer pointer

M13 ZTEMP1* [00F5,2] -- Temporary storage

M14 ZTEMP4* [00F7,2] -- Temporary storage

M15 ZTEMP3* [00F9,2] -- Temporary storage

M16 FLPTR [00FC,2] -- Pointer to FP number

M17 FPTR2* [00FE,2] -- FP package use

M18 LBPR1* [057E,1] -- LBUFF preamble

M19 LBPR2* [057F,1] -- LBUFF preamble

M20 LBUFF [0580,96] -- Text buffer

M21 PLYARG* [05E0,6] -- FP register (internal)

M22 FPSCR/FSCR* [05E6,6] -- FP register (internal)

M23 FPSCR1/SCR1* [05EC,6] -- FP register (internal)

M24 DEGFLG/RADFLG [00FB,1] -- Degrees/radians flag

DEGFLG = 0 indicates radians, 6 indicates degrees.

N. Power-Up and SYSTEM RESET

See Section 7 for details of the power-up and system reset operations.

RAM Sizing

During power-up and system reset the first non-RAM address above 1000 hex is located and its address retained using a nondestructive test. The first byte of every 4K memory "block" is tested to see if it is alterable; if so, the original value is restored and the next block is tested, and if not, that address is considered to be the end of RAM.

N1 RAMLO*/TRAMSZ* [0004,3] -- RAM data/test pointer (temporary)

RAMLO+1 contains the LSB of the address to be tested (always = 0) and TRAMSZ (same as RAMLO+2) contains the MSB of the address to be tested. RAMLO+0 contains the complemented value of the data originally contained in the memory location being tested.

Later in the initialization process these variables are used for totally unrelated functions; but first the value in TRAMSZ is moved to the variables RAMSIZ and MEMTOP+1.

N2 TSTDAT* [0007,1] -- Test data byte save

TSTDAT contains the original value of the memory location being tested.

Diskette/Cassette-Boot

As a part of the Power-up sequence, software can be booted from an attached disk drive or cassette player as explained in Section 10.

N3 DOSINI [000C,2] -- Diskette-boot initialization vector.

DOSINI contains the disk booted software initialization address from the beginning of the boot file (see Section 10) whenever a diskette-boot is successfully completed.

N4 CKEY* [004A,1] -- Cassette-boot request flag

CKEY is an internal flag used to indicate that the console [START] key was pressed during Power-up, thus indicating that a cassette-boot is desired. CKEY equals zero when no cassette-boot is requested, and is nonzero when a cassette-boot is requested. The flag is cleared to zero after a cassette-boot.

N5 CASSBT* [004B,1] -- Cassette-booting flag

CASSBT is used during the cassette-boot process to indicate to shared code that the cassette is being booted and not the diskette. CASSBT equal to zero indicates a diskette-boot, and nonzero indicates a cassette-boot.

N6 CASINI [0002,2] -- Cassette-boot initialization vector

CASINI contains the cassette-booted software initialization address from the beginning of the boot file (see Section 10) whenever a cassette-boot is successfully completed.

N7 BOOT?* [0009,1] -- Successful diskette/cassette-boot flag.

BOOT? indicates to the initialization processor which, if any, of the boot operations went to successful completion. The flag values are set by the OS and the format for the variable is shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|           |C|D|
+-+-+-+-+-+-+-+-+
```

where: C = 1 indicates that the cassette-boot was completed.
D = 1 indicates that the diskette-boot was completed.

N8 DFLAGS* [0240,1] -- Diskette flags

DFLAGS contains the value of the first byte of the boot file, after a diskette-boot. See Section 10.

N9 DBSECT* [0241,1] -- Diskette-boot sector count

DBSECT is initially set to the value of the second byte of the boot file, during a diskette-boot, and is then used to control the number of additional diskette sectors read, if any.

N10 BOOTAD* [0242,2] -- Diskette-boot memory address

BOOTAD is initially set to the value of the third and fourth bytes of the boot file, during a diskette-boot, and is not modified thereafter.

Environment Control

If, at the end of a power-up or system reset, control is not given to one of the cartridges (as explained in Sections 7 and 10), then program control passes to the address contained in the data base variable DOSVEC.

N11 COLDST* [0244,1] -- Coldstart complete flag

COLDST is used by the initialization routine to detect the case of a system reset occurring before the completion of the power-up process. COLDST is set to $FF at the beginning of the power-up sequence and is set to 0 at the completion; if a system reset occurs while the value is nonzero, the power-up sequence will be reinitiated (rather than initiating a system reset sequence).

N12 DOSVEC [000A,2] -- Noncartridge control vector

At the beginning of power-up the OS sets DOSVEC to point to the "blackboard" routine; DOSVEC can then be altered as a consequence of a diskette-boot or cassette-boot (as explained in Section 10) to establish a new control program. Control will be passed through DOSVEC on all power-up and system reset conditions in which a cartridge does not take control first.

System Reset

N13 WARMST [0008,1] -- Warmstart flag

WARMST equals $FF during a system reset (warmstart) initialization and equals 0 during a power-up initialization (coldstart).

P. INTERRUPTS

See Section 6 for a discussion of interrupt processing.

P1 CRITIC [0042,1] -- Critical code section flag

CRITIC is used to signal to the VBLANK interrupt processor that a critical code section is executing without IRQ interrupts being inhibited; the VBLANK interrupt processor will stop interrupt processing after stage 1 and before stage 2, just as if the 6502 processor I bit were set, when CRITIC is set.

CRITIC equal to zero indicates that the currently executing code section is noncritical, while any nonzero value indicates that the currently executing code section is critical.

P2 POKMSK [0010,1] -- POKEY interrupt mask

POKMSK is a software maintained interrupt mask that is used in conjunction with the enabling and disabling of the various POKEY interrupts. This mask is required because the POKEY interrupt enable register IRQEN [D20E] is a write-only register, and at any point in time the system can have several users independently enabling and disabling POKEY interrupts. POKMSK is updated by the users to always contain the current content of IRQEN.

System Timers

The System Timers are discussed in detail in Section 6.

Realtime Clock

The realtime clock (or frame counter, as it is sometimes called) is incremented as part of the stage 1 VBLANK process as explained in Section 6.

P3 RTCLOK [0012,3] -- Realtime frame counter

RTCLOK+0 is the most significant byte, RTCLOK+1 the next most significant byte, and RTCLOK+2 the least significant byte. See the discussions at D3 and preceding B10 for OS use of RTCLOK.

System Timer 1

System Timer 1 is maintained as part of the stage 1 VBLANK process, and thus has the highest priority of any of the user timers.

P4 CDTMV1 [0218,2] -- System Timer 1 value

CDTMV1 contains zero when the timer is inactive, otherwise it contains the number of VBLANKs remaining until timeout. Also see H26.

P5 CDTMA1 [0226,2] -- System Timer 1 jump address

CDTMA1 contains the address to which to JSR should the timer timeout. See also H27 and Section 6.

System Timer 2

System Timer 2 is maintained as part of the stage 2 VBLANK process, and has the second highest priority of the user timers. The OS does not have any direct use for System Timer 2.

P6 CDTMV2 [021A,2] -- System Timer 2 value

CDTMV2 contains zero when the timer is inactive, otherwise it contains the number of VBLANKs remaining until timeout.

P7 CDTMA2 [0228,2] -- System Timer 2 jump address

CDTMA2 contains the address to which to JSR should the timer timeout. See Section 6.

System Timers 3, 4 and 5

System Timers 3, 4 and 5 are maintained as part of the stage 2 VBLANK process, and have the lowest priority of the user timers. The OS does not have any direct use for these timers.

P8 CDTMV3 [021C,2], CDTMV4 [021E,2] and CDTMV5 [0220,2]

These variables contain zero when the corresponding timers are inactive, otherwise they contain the number of VBLANKs remaining until timeout.

P9 CDTMF3 [022A,1], CDTMF4 [022C,1] and CDTMF5 [022E,2]

Each of these 1-byte variables will be set to zero should its corresponding timer timeout. The OS never modifies these bytes except to set them to zero upon timeout (and initialization).

RAM Interrupt Vectors

There are RAM vectors for many of the interrupt conditions within the system. See Section 6 for a discussion of the placing of values to these vectors.

NMI Interrupt Vectors

P10 VDSLST [0200,2] -- Display-list interrupt vector

This vector is not used by the OS. See Section 6.

P11 VVBLKI [0222,2] -- Immediate VBLANK vector

This vector is initialized to point to the OS stage 1 VBLANK

P12 VVBLKD [0224,2] -- Deferred VBLANK vector

This vector is initialized to point to the OS VBLANK exit routine. See Section 6.

IRQ Interrupt Vectors

P13 VIMIRQ [0216,2] -- General IRQ vector

This vector is initialized to point to the OS IRQ interrupt processor. See Section 6.

P14 VPRCED [0202,2] -- Serial I/O bus proceed signal

The serial bus line that produces this interrupt is not used in the current system. See Section 6.

P15 VINTER [0204,2] -- Serial I/O bus interrupt signal

The serial bus line that produces this interrupt is not used in the current system. See Section 6.

P16 V[BREAK] [0206,2] -- BRK instruction vector

This vector is initialized to point to a PLA, RTI sequence as the OS proper does not utilize the BRK instruction. See Section 6.

P17 VKEYBD [0208,2] -- Keyboard interrupt vector

This vector is initialized to point to the Keyboard Handler's interrupt service routine. See Section 6 and the discussion preceding E1.

P18 VSERIN [020A,2] -- Serial I/O bus receive data ready

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P19 VSEROR [020C,2] -- Serial I/O bus transmit ready

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P20 VSEROC [020E,2] -- Serial I/O bus transmit complete

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P21 VTIMR1 [0210,2], VTIMR2 [0212,2] and VTIMR4 [0214,2] -- POKEY timer vectors

The POKEY timer interrupts are not used by the OS See Section 6.

Hardware Register Updates

As part of the stage 2 VBLANK process, certain hardware registers are updated from OS data base variables as explained in Section 6.

P22 SDMCTL* [022F,1] -- DMA control

SDMCTL is set to a value of $02 at the beginning of a Display Handler OPEN command, and then later set to a value of $22. The value of SDMCTL is stored to DMACTL [D400] as part of the stage 2 VBLANK process.

P23 SDLSTL* [0230,1] and SDLSTH* [0231,1] -- Display list address

The Display Handler formats a new display list with every OPEN command and puts the display list address in SDLSTL and SDLSTH. The value of these bytes are stored to DLISTL [D402] and DLISTH [D403] as part of the stage 2 VBLANK process.

0360-036F	IOCB #2
0370-037F	IOCB #3
0380-038F	IOCB #4
0390-039F	IOCB #5
03A0-03AF	IOCB #6
03B0-03BF	IOCB #7

NOTE: There is a potential timing problem associated with the updating of the hardware registers from the data base variables. Since the stage 2 VBLANK process is performed with interrupts enabled, it is possible for an IRQ interrupt to occur before the updating of DLISTH and DLISTL. If the processing of that interrupt (plus other nested interrupts) exceeds the vertical-blank delay (1 msec), then the display list pointer register will not have been updated when display list processing commences for the new frame, and a screen glitch will result.

P24 GPRIOR* [026F,1] -- Priority control

The Display Handler alters bits 6 and 7 of GPRIOR as part of establishing the GTIA mode. The value of GPRIOR is stored to PRIOR [D01B] as part of the stage 2 VBLANK process.

P25 CHACT* [02F3,1] -- Character control

The Display Handler sets CHACT to $02 on every OPEN command. The value of CHACT is stored to CHACTL [D401] as part of the stage 2 VBLANK process.

P26 CHBAS [02F4,1] -- Character address base

The Display Handler sets CHBAS to $E0 on every OPEN command. The value of CHBAS is stored to CHBASE [D409] as part of the stage 2 VBLANK process. This variable controls the character subset for screen modes 1 and 2; a value of $E0 provides the capital letters and number set whereas a value of $E2 provides the lowercase letters and special graphics set. See B55 for more information.

P27 PCOLRx [02C0,4] and COLORx [02C4,5] -- Color registers

See B7 and B8.

Internal Working Variables

P28 INTEMP* [022D,1] -- Temporary storage

INTEMP is used by the SETVBL (SETVBV) routine.

R. USER AREAS

The areas shown below are available to the user in a non-nested environment. See Section 4 for further information.

R1 [0080,128]

R2 [0480,640]

ALPHABETICAL LIST OF DATA BASE VARIABLES

NAME	VID	ADDRESS SIZE
ADDCOR	D4	030E, 1
ADRESS	B39	0064, 2
APPMHI	A3	000E, 2
ATACHR	B54	02FB, 1
ATRACT	B10	004D, 1
BFENHI	H17	0035, 1
BFENLO	H17	0034, 1
BITMSK	B50	006E, 1
BLIM	D10	028A, 1
BOOT?	N7	0009, 1
BOOTAD	N10	0242, 2
BOTSCR	B16	02BF, 1
BPTR	D11	003D, 1
BRKKEY	E5	0011, 1
BUFADR	C1	0015, 2
BUFCNT	B42	006B, 1
BUFRFL	H21	0038, 1
BUFRHI	H16	0033, 1
BUFRLO	H16	0032, 1
BUFSTR	B43	006C, 2
CASBUF	D9	03FD, 131
CASFLG	D8	030F, 1
CASINI	N6	0002, 2
CASSBT	N5	004B, 1
CAUX1	H20	023C, 1
CAUX2	H20	023D, 1
CBAUDH	D1	02EF, 1
CBAUDL	D1	02EE, 1
CCOMND	H19	023B, 1
CDEVIC	H18	023A, 1
CDTMA1	P5, H27	0226, 2
CDTMA2	P7	0228, 2
CDTMF3	P9	022A, 1
CDTMF4	P9	022C, 1
CDTMF5	P9	022E, 1
CDTMV1	P4, H26	0226, 2
CDTMV2	P6	021A, 2
CDTMV3	P8	021C, 2

CDTMV4	P8	021E,2
CDTMV5	P8	0220,2
CH	E3	02FC,1
CHKSNT	H14	003B,1
CH1	E1	02F2,1
CHACT	P25	02F3,1
CHAR	B55	02FA,1
CHBAS	P26	02F4,1
CHKSNT	H14	003B,1
CHKSUM	H13	0031,1
CIOCHR	G25	002F,1
CIX	M11	00F2,1
CKEY	N4	004A,1
COLAC	B24	0072,2
COLCRS	B2	0055,2
COLDST	N11	0244,1
COLINC	B21	007A,1
COLOR0	B8,P27	02C4,1
COLOR1	B8,P27	02C5,1
COLOR2	B8,P27	02C6,1
COLOR3	B8,P27	02C7,1
COLOR4	B8,P27	02C8,1
COLRSH	B11	004F,1
COUNTR	B23	007E,2
CRETRY	H11	0036,1
CRITIC	P1	0042,1
CRSINH	B1	02F0,1
CSTAT	S2	0288,1

DAUX1	H9	030A,1
DAUX2	H9	030B,2
DBSECT	N9	0241,1
DBUFHI	H6	0304,1
DBUFLO	H6	0305,1
DBYTHI	H8	0308,1
DBYTLO	H8	0309,1
DCB	H1	0300,12
DCOMND	H4	0302,1
DDEVIC	H2	0300,1
DEGFLG	M24	00FB,1
DELTAC	B22	0077,2
DELTAR	B22	0076,1
DFLAGS	N8	0240,1
DIGRT	M10	00F1,1
DINDEX	B35	0057,1
DMASK	B28	02A0,1
DOSINI	N3	000C,2
DOSVEC	N12	000A,2
DRETRY	H12	0037,1
DRKMSK	B12	004E,1
DSKTIM	C2	0246,1
DSKUTL	L1	001A,2
DSPFLG	B27	02FE,1
DSTAT	B34	004C,1

DSTATS	H5	0303,1
DTIMLO	H7	0306,1
DUNIT	H3	0301,1
DUNUSE	S3	0307,1
DVSTAT	G11	02EA,4
EEXP	M6	00ED,1
ENDPT	B25	0074,2
ERRFLG	H30	023F,1
(ERRNO	K5)	0049,1
ESCFLG	B26	02A2,1
ESIGN	M8	00EF,1
FCHRFL	M9	00F0,1
FEOF	D12	003F,1
FILDAT	B17	02FD,1
FILFLG	B18	02B7,1
FLPTR	M16	00FC,2
FMSZPG	K1	0043,7
FPSCR	M22	05E6,6
FPSCR1	M23	05EC,6
FPTR2	M17	00FE,2
FR0	M1	00D4,6
FR1	M3	00E0,6
FR2	M4	00E6,6
FRE	M2	00DA,6
FREQ	D15	0040,1
FRMADR	B41	0068,2
FRX	M5	00EC,1
FSCR	M22	05E6,6
FSCR1	M23	05EC,6
FTYPE	D13	003E,1
GPRIOR	P24	026F,1
HATABS	G12	031A,38
HOLD1	B30	0051,1
HOLD2	B31	029F,1
HOLD3	B32	029D,1
HOLD4	B20	02BC,1
HOLDCH	E7	007C,1
ICAX1	G9	034A,1
ICAX1Z	G21	002A,1
ICAX2	G9	034B,1
ICAX2Z	G21	002B,1
ICBAH	G6	0345,1
ICBAHZ	G18	0025,1
ICBAL	G6	0344,1
ICBALZ	G18	0024,1
ICBLH	G8	0349,1
ICBLHZ	G20	0029,1
ICBLL	G8	0348,1
ICBLLZ	G20	0028,1

ICCOM	G4	0342, 1
ICCOMT	G23	0017, 1
ICCOMZ	G16	0022, 1
ICDNO	G3	0341, 1
ICDNOZ	G15	0021, 1
ICHID	G2	0340, 1
ICHIDZ	G14	0020, 1
ICIDNO	G24, G2	2002E, 1
ICPTH	G7	0347, 1
ICPTHZ	G19	0027, 1
ICPTL	G7	0346, 1
ICPTLZ	G19	0026, 1
ICSPR	G10	034C, 4
ICSPRZ	G22	002C, 4
ICSTA	G5	0343, 1
ICSTAZ	G17	0023, 1
INBUFF	M12	00F3, 2
INSDAT	B45	007D, 1
INTEMP	P28	022D, 1
INVFLG	E9	02B6, 1
IOCB	G1	0340, 16
IOCBAS	G13	0020, 16
KEYDEL	E2	02F1, 1
LBFEND	M20	0580, 96
LBPR1	M18	057E, 1
LBPR2	M19	057F, 1
LBUFF	M20	0580, 96
LINBUF	B51	0247, 40
LMARGN	B5	0052, 1
LOGCOL	B15	0063, 1
LOGMAP	B14	02B2, 4
MEMLO	A1	02E7, 2
MEMTOP	A2	02E5, 2
MLTTMP	B40	0066, 2
NEWCOL	B19	0061, 2
NEWROW	B19	0060, 1
NOCKSM	H15	003C, 1
NSIGN	M7	00EE, 1
OLDADR	B38	005E, 2
OLDCHR	B37	005D, 1
OLDCOL	B3	005B, 2
OLDROW	B3	005A, 1
OPNTMP	B40	0066, 2
PADDL0	J3	0270, 1
PADDL1	J3	0271, 1
PADDL2	J3	0272, 1
PADDL3	J3	0273, 1
PADDL4	J3	0274, 1

PADDL5	J3	0275, 1
PADDL6	J3	0276, 1
PADDL7	J3	0277, 1
PBPNT	F3	001D, 1
PBUFSZ	F2	001E, 1
PCOLR0	B7, P27	02C0, 1
PCOLR1	B7, P27	02C1, 1
PCOLR2	B7, P27	02C2, 1
PCOLR3	B7, P27	02C3, 1
PLYARG	M21	05E0, 6
POKMSK	P2	0010, 1
PRNBUF	F1	03C0, 40
PTEMP	F4	001F, 1
PTIMOT	F5	001C, 1
PTRIG0	J4	027C, 1
PTRIG1	J4	027D, 1
PTRIG2	J4	027E, 1
PTRIG3	J4	027F, 1
PTRIG4	J4	0280, 1
PTRIG5	J4	0281, 1
PTRIG6	J4	0282, 1
PTRIG7	J4	0283, 1
RADFLG	M24	00FB, 1
RAMLO	N1	0004, 3
RAMSIZ	A5	02E4, 1
RAMTOP	A4	006A, 1
RECVDN	H22	0039, 1
RMARGN	B6	0053, 1
ROWAC	B24	0070, 2
ROWCRS	B2	0054, 1
ROWINC	B21	0079, 1
RTCLOK	P3	0012, 3
SAVADR	B41	0068, 2
SAVIO	D7	0316, 1
SAVMSC	B36	0058, 2
SCRFLG	B9	02BB, 1
SDLSTH	P23	0231, 1
SDLSTL	P23	0230, 1
SDMCTL	P22	022F, 1
SHFAMT	B29	006F, 1
SHFLOK	E6	02BE, 1
SOUNDR	H10	0041, 1
SRTIMR	E8	022B, 1
SSFLAG	E4	02FF, 1
SSKCTL	H32	0232, 1
STACKP	H28	0318, 1
STATUS	H31	0030, 1
STICK0	J1, J7, J8	0278, 1
STICK1	J1, J7, J8	0279, 1
STICK2	J1, J7, J8	027A, 1
STICK3	J1, J7, J8	027B, 1
STRIG0	J2, J7, J9	0284, 1

STRIG1	J2, J7, J9	0285, 1
STRIG2	J2, J7, J9	0286, 1
STRIG3	J2, J7, J9	0284, 4
SUBTMP	B48	029E, 1
SWPFLG	B44	007B, 1
TABMAP	B13	02A3, 15
TEMP	H23	023E, 1
TEMP1	D5	0312, 2
TEMP3	D6	0315, 1
TIMER1	D3	030C, 2
TIMER2	D3	0310, 2
TIMFLG	D2, H25	0317, 1
TINDEX	B49	0293, 1
TMPCHR	B33	0050, 1
TMPCOL	B46	02B9, 2
TMPLBT	B47	02A1, 1
TMPROW	B46	02B8, 1
TOADR	B40	0066, 2
TRAMSZ	N1	0004, 3
TSTAT	H29	0319, 1
TSTDAT	N2	0007, 1
TXTCOL	B4	0291, 2
TXTMSC	B52	0294, 2
TXTOLD	B53	0296, 6
TXTROW	B4	0290, 1
USAREA	R1	0080, 128
VBREAK	P16	0206, 2
VDSLST	P10	0200, 2
VIMIRQ	P13	0216, 2
VINTER	P15	0204, 2
VKEYBD	P17	0208, 2
VPRCED	P14	0202, 2
VSERIN	P18	020A, 2
VSEROC	P20	020E, 2
VSEROR	P19	020C, 2
VTIMR1	P21	0210, 2
VTIMR2	P21	0212, 2
VTIMR4	P21	0214, 2
VVBLKD	P12	0224, 2
VVBLKI	P11	0222, 2
WARMST	N13	0008, 1
WMODE	D14	0289, 1
XMTDON	H24	003A, 1
(ZBUFF	K2)	0043, 2
(ZDRVA	K3)	0045, 2
ZIOCB	G13	0020, 16
(ZSBA	K4)	0047, 2
ZTEMP1	M13	00F5, 2

ZTEMP3	M15	00F9,2
ZTEMP4	M14	00F7,2

MEMORY ADDRESS ORDERED LIST OF DATABASE VARIABLES

ADDRESS	VID	NAME
0000-0001	S7	LNZBS
0002-0003	N6	CASINI
0004-0006	N1	RAMLO, TRAMSZ
0007	N2	TSTDAT
0008	N13	WARMST
0009	N7	BOOT?
000A-000B	N12	DOSVEC
000C-000D	N3	DOSINI
000E-000F	A3	APPMHI
0010	P2	POKMSK
0011	E5	BRKKEY
0012-0014	P3	RTCLOK
0015-0016	C1	BUFADR
0017	G23	ICCOMT
001A-001B	L1	DSKUTL
001C	F5	PTIMOT
001D	F3	PBPNT
001E	F2	PBUFSZ
001F	F4	PTEMP
0020	G13, G14	ICHIDZ
0021	G15	ICDNOZ
0022	G16	ICCOMZ
0023	G17	ICOBAS
0024-0025	G18	ICBALZ, ICBAHZ
0026-0027	G19	ICPTLZ, ICPTHZ
0028-0029	G20	ICBLLZ, ICBLHZ
002A-002B	G21	ICAX1Z, ICAX2Z
002C-002F	G22, G24, G25	ICSPRZ
0030	H31	STATUS
0031	H13	CHKSUM
0032-0033	H16	BUFRLO, BUFFRHI
0034-0035	H17	BFENLO, BFENHI
0036	H11	CRETRY
0037	H12	DRETRY
0038	H21	BUFRFL
0039	H22	RECVDN
003A	H24	XMTDON
003B	H14	CHKSNT
003C	H15	NOCKSM
003D	D11	BPTR
003E	D13	FTYPE
003F	D12	FEOF
0040	D15	FREQ
0041	H10	SOUNDR
0042	P1	CRITIC
0043-0049	K1, K2, K3, K4, K5	ZBUFF, ZBUFP, ZDRVA, ZSBA
004A	N4	CKEY
004B	N5	CASSBT
004C	B34	DSTAT

004D	B10	ATRACT
004E	B12	DRKMSK
004F	B11	COLRSH
0050	B33	TMPCHR
0051	B30	HOLD1
0052	B5	LMARGN
0053	B6	RMARGN
0054-0056	B2	ROWCRS, COLCRS
0057	B35	DINDEX
0058-0059	B36	SAVMSC
005A-005C	B3	OLDROW, OLDCOL
005D	B37	OLDCHR
005E-005F	B38	OLDADR
0060-0062	B19	NEWROW, NEWCOL
0063	B15	LOGCOL
0064-0065	B39	ADRESS
0066-0067	B40	MLTTMP, OPNTMP, TOADR
0068-0069	B41	SAVADR/FRMADR
006A	A4	RAMTOP
006B	B42	BUFCNT
006C-006D	B43	BUFSTR
006E	B50	BITMSK
006F	B29	SHFAMT
0070-0073	B24	ROWAC, COLAC
0074-0075	B25	ENDPT
0076-0078	B22	DELTAR, DELTAC
0079-007A	B21	ROWINC, COLINC
007B	B44	SWPFLG
007C	E7	HOLDCH
007D	B45	INSDAT
007E-007F	B23	COUNTR
0080-00FF	SEE FLOATING POINT VARIABLE LIST AT END.	
0100-01FF	6502 STACK	
0200-0201	P10	VDSLST
0202-0203	P14	VPRCED
0204-0205	P15	VINTER
0206-0207	P16	VBREAK
0208-0209	P17	VKEYBD
020A-020B	P18	VSERIN
020C-020D	P19	VSEROR
020E-020F	P20	VSEROC
0210-0215	P21	VITMR1, VITMR2, VITMR4
0216-0217	P13	VIMIRQ
0218-0219	P4, H26	CDTMV1
021A-021B	P6	CDTMV2
021C-0221	P8	CDTMV3, CDTMV4, CDTMV5
0222-0223	P11	VVBLKI
0224-0225	P12	VVBLKD
0226-0227	P5, H27	CDTMA1
0228-0229	P7	CDTMA2
022A	P9	CDTMF3

022B	E8	SRTIMR
022C	P9	CDTMF4
022D	P28	INTEMP
022E	P9	CDTMF5
022F	P22	SDMCTL
0230-0231	P23	SDLSTL, SDLSTH
0232	H32	SSKCTL
023A	H18	CDEVIC
023B	H19	CCOMND
023C-023D	H20	CAUX1, CAUX2
023E	H23	TEMP
023F	H30	ERRFLG
0240	N8	DFLAGS
0241	N9	DBSECT
0242-0243	N10	BOOTAD
0244	N11	COLDST
0246	C2	DSKTIM
0247-026E	B51	LINBUF
026F	P24	GPRIOR
0270-0277	J3	PADDLO -- PADDL7
0278-027B	J1, J7, J8	STICKO -- STICK3
027C-0283	J4	PTRIGO -- PTRIG7
0284-0287	J2, J7, J9	STRIGO -- STRIG3
0289	D14	WMODE
028A	D10	BLIM
028B-028F	S10	unused
0290-0292	B4	TXTROW, TXTCOL
0293	B49	TINDEX
0294-0295	B52	TXTMSC
0296-029B	B53	TXTOLD
029D	B32	HOLD3
029E	B48	SUBTMP
029F	B31	HOLD2
02A0	B28	DMASK
02A1	B47	TMPLBT
02A2	B26	ESCFLG
02A3-02B1	B13	TABMAP
02B2-02B5	B14	LOGMAP
02B6	E9	INVFLG
02B7	B18	FILFLG
02B8-02BA	B46	TMPROW, TMPCOL
02BB	B9	SCRFLG
02BC	B20	HOLD4
02BE	E6	SHFLOK
02BF	B16	BOTSCR
02C0-02C3	B7, P27	PCOLRO -- PCOLR3
02C4-02C8	B8, P27	PCOLRO -- PCOLR4
02E4	A5	RAMSIZ
02E5-02E6	A2	MEMTOP
02E7-02E8	A1	MEMLO
02EA-02ED	G11	DVSTAT
02EE-02EF	D1	CHBAUDL, CHBAUDH
02F0	B1	CRSINH
02F1	E2	KEYDEL

02F2	E1	CH1
02F3	P25	CHACT
02F4	P26	CHBAS
02FA	B55	CHAR
02FB	B54	ATACHR
02FC	E3	CH
02FD	B17	FILDAT
02FE	B27	DSPFLG
02FF	E4	SSFLAG
0300	H1, H2	DCB/DDEVIC
0301	H3	DUNIT
0302	H4	DCOMND
0303	H5	DSTATS
0304-0305	H6	DBUFLO, DBUFHI
0306	H7	DTIMLO
0308-0309	H8	DBYTLO, DBYTHI
030A-030B	H9	DAUX1, DAUX2
030C-030D	D3	TIMER1
030E	D4	ADDCOR
030F	D8	CASFLG
0310-0311	D3	TIMER2
0312-0313	D5	TEMP1
0315	D6	TEMP3
0316	D7	SAVIO
0317	D2, H25	TIMFLG
0318	H28	STACKP
0319	H29	TSTAT
031A-033F	G12	HATABS
0340	G1, G2	IOCB, ICHID
0341	G3	ICDNO
0342	G4	ICCOM
0343	G5	ICSTA
0344-0345	G6	ICBAL, ICBAH
0346-0347	G7	ICPTL, ICPTH
0348-0349	G8	ICBLL, ICBLH
034A-034B	G9	ICAX1, ICAX2
034C-034F	G10	ICSPR
0350-035F	G2-G10	(IOCB #1)
0360-036F	G2-G10	(IOCB #2)
0370-037F	G2-G10	(IOCB #3)
0380-038F	G2-G10	(IOCB #4)
0390-039F	G2-G10	(IOCB #5)
03A0-03AF	G2-G10	(IOCB #6)
03B0-03BF	G2-G10	(IOCB #7)
03C0-03E7	F1	PRNBUF
03FD-047F	D9	CASBUF
0480-06FF	R2	User Area

FLOATING POINT PACKAGE VARIABLES

00D4-00D9	M1	FR0
00DA-00DF	M2	FRE
00E0-00E5	M3	FR1
00E6-00EB	M4	FR2
00EC	M5	FRX
00ED	M6	EEXP
00EE	M7	NSIGN
00EF	M8	ESIGN
00F0	M9	FCHRFLG
00F1	M10	DIGRT
00F2	M11	CIX
00F3-00F4	M12	INBUFF
00F5-00F6	M13	ZTEMP1
00F7-00F8	M14	ZTEMP4
00F9-00FA	M15	ZTEMP3
00FB	M24	RADFLG/DEGFLG
00FC-00FD	M16	FLPTR
00FE-00FF	M17	FPTR2
057E	M18	LBPR1
057F	M19	LBPR2
0580-05FF	M20	LBFEND, LBUFF
05E0-05E5	M21	PLYARG
05E6-05EB	M22	FPSCR/FSCR
05EC-05F1	M23	FPSCR1/SCR1

INDEX

The subject index contains three forms of references:

Section number, such as '3.'
Appendix, such as 'App B'
Variable ID from Appendix L, such as 'B7'.

ATARI® 400/800™

ATARI® HOME COMPUTER SYSTEM

OPERATING SYSTEM SOURCE LISTING

ATARI®

A Warner Communications Company

```
 LIST X
; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
; SYSTEM LISTING, MODIFIED TO ASSEMBLE ON THE MICROTEC CROSS
; ASSEMBLER.
; THIS VERSION IS THE ONE WHICH WAS BURNED INTO ROM.
; THERE IS A RESIDUAL PIECE OF CODE WHICH IS FOR LNBUG.  THIS
; IS AT LOCATION $9000 WHICH IS NOT IN ROM.
;
; THIS IS THE REVISION B EPROM VERSION
```

```
                         .PAGE
                  ;
                  ;
                  ;      COLLEEN OPERATING SYSTEM EQUATE FILE
                  ;
                  ;      NTSC/PAL ASSEMBLY FLAG
                  ;
0000              PALFLG  =      0            ;0 = NTSC   1 = PAL
                  ;
                  ;
                  ;      MODULE ORIGIN TABLE
                  ;
E000              CHRORG  =      $E000        ;CHARACTER SET
E400              VECTBL  =      $E400        ;VECTOR TABLE
E480              VCTABL  =      $E480        ;RAM VECTOR INITIAL VALUE TABLE
E4A6              CIOORG  =      $E4A6        ;CENTRAL I/O HANDLER
E6D5              INTORG  =      $E6D5        ;INTERRUPT HANDLER
E944              SIOORG  =      $E944        ;SERIAL I/O DRIVER
EDEA              DSKORG  =      $EDEA        ;DISK HANDLER
EE78              PRNORG  =      $EE78        ;PRINTER HANDLER
EF41              CASORG  =      $EF41        ;CASSETTE HANDLER
F0E3              MONORG  =      $F0E3        ;MONITOR/POWER UP MODULE
F3E4              KBDORG  =      $F3E4        ;KEYBOARD/DISPLAY HANDLER
                  ;
                  ;
                  ;
                  ;
                  ;     VECTOR TABLE
                  ;
                  ;HANDLER ENTRY POINTS ARE CALLED OUT IN THE FOLLOWING VECTOR
                  ;TABLE. THESE ARE THE ADDRESSES MINUS ONE.
                  ;
                  ;
                  ;EXAMPLE FOR EDITOR
                  ;
                  ;    E400       OPEN
                  ;       2       CLOSE
                  ;       4       GET
                  ;       6       PUT
                  ;       8       STATUS
                  ;       A       SPECIAL
                  ;       C       JUMP TO POWER ON INITIALIZATION ROUTINE
                  ;     . F       NOT USED
                  ;
                  ;
E400              EDITRV  =      $E400        ;EDITOR
E410              SCRENV  =      $E410        ;TELEVISION SCREEN
E420              KEYBDV  =      $E420        ;KEYBOARD
E430              PRINTV  =      $E430        ;PRINTER
E440              CASETV  =      $E440        ;CASSETTE
                  ;
                  ;      JUMP VECTOR TABLE
                  ;
                  ;THE FOLLOWING IS A TABLE OF JUMP INSTRUCTIONS
```

```
                    ;TO VARIOUS ENTRY POINTS IN THE OPERATING SYSTEM.
                    ;
E450                DISKIV  =     $E450        ;DISK INITIALIZATION
E453                DSKINV  =     $E453        ;DISK INTERFACE
E456                CIOV    =     $E456        ;CENTRAL INPUT OUTPUT ROUTINE
E459                SIOV    =     $E459        ;SERIAL INPUT OUTPUT ROUTINE
E45C                SETVBV  =     $E45C        ;SET SYSTEM TIMERS ROUTINE
E45F                SYSVBV  =     $E45F        ;SYSTEM VERTICAL BLANK CALCULATIONS
E462                XITVBV  =     $E462        ;EXIT VERTICAL BLANK CALCULATIONS
E465                SIOINV  =     $E465        ;SERIAL INPUT OUTPUT INITIALIZATION
E468                SENDEV  =     $E468        ;SEND ENABLE ROUTINE
E46B                INTINV  =     $E46B        ;INTERRUPT HANDLER INITIALIZATION
E46E                CIOINV  =     $E46E        ;CENTRAL INPUT OUTPUT INITIALIZATION
E471                BLKBDV  =     $E471        ;BLACKBOARD MODE
E474                WARMSV  =     $E474        ;WARM START ENTRY POINT
E477                COLDSV  =     $E477        ;COLD START ENTRY POINT
E47A                RBLOKV  =     $E47A        ;CASSETTE READ BLOCK ENTRY POINT VECTOR
E47D                CSOPIV  =     $E47D        ;CASSETTE OPEN FOR INPUT VECTOR
                    ;VCTABL = $E480
                    ;
                    ;
                    ; OPERATING SYSTEM EQUATES
                    ;
                    ; COMMAND CODES FOR IOCB
0003                OPEN    =     3            ;OPEN FOR INPUT/OUTPUT
0005                GETREC  =     5            ;GET RECORD (TEXT)
0007                GETCHR  =     7            ;GET CHARACTER(S)
0009                PUTREC  =     9            ;PUT RECORD (TEXT)
000B                PUTCHR  =     $B           ;PUT CHARACTER(S)
000C                CLOSE   =     $C           ;CLOSE DEVICE
000D                STATIS  =     $D           ;STATUS REQUEST
000E                SPECIL  =     $E           ;BEGINNING OF SPECIAL ENTRY COMMANDS
                    ;
                    ; SPECIAL ENTRY COMMANDS
0011                DRAWLN  =     $11          ;DRAW LINE
0012                FILLIN  =     $12          ;DRAW LINE WITH RIGHT FILL
0020                RENAME  =     $20          ;RENAME DISK FILE
0021                DELETE  =     $21          ;DELETE DISK FILE
0022                FORMAT  =     $22          ;FORMAT
0023                LOCKFL  =     $23          ;LOCK FILE TO READ ONLY
0024                UNLOCK  =     $24          ;UNLOCK LOCKED FILE
0025                POINT   =     $25          ;POINT SECTOR
0026                NOTE    =     $26          ;NOTE SECTOR
00FF                IOCFRE  =     $FF          ;IOCB "FREE"
                    ;
                    ; AUX1 EQUATES
                    ; () INDICATES WHICH DEVICES USE BIT
0001                APPEND  =     $1           ;OPEN FOR WRITE APPEND (D), OR SCREEN READ (
0002                DIRECT  =     $2           ;OPEN FOR DIRECTORY ACCESS (D)
0004                OPNIN   =     $4           ;OPEN FOR INPUT (ALL DEVICES)
0008                OPNOT   =     $8           ;OPEN FOR OUTPUT (ALL DEVICES)
000C                OPNINO  =     OPNIN+OPNOT  ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
0010                MXDMOD  =     $10          ;OPEN FOR MIXED MODE (E,S)
0020                INSCLR  =     $20          ;OPEN WITHOUT CLEARING SCREEN (E,S)
```

```
                        ;
                        ; DEVICE NAMES
     0045               SCREDT  =       'E          ;SCREEN EDITOR (R/W)
     004B               KBD     =       'K          ;KEYBOARD (R ONLY)
     0053               DISPLY  =       'S          ;SCREEN DISPLAY (R/W)
     0050               PRINTR  =       'P          ;PRINTER (W ONLY)
     0043               CASSET  =       'C          ;CASSETTE
     004D               MODEM   =       'M          ;MODEM
     0044               DISK    =       'D          ;DISK (R/W)
                        ;
                        ; SYSTEM EOL (CARRIAGE RETURN)
     009B               CR      =       $9B
                        ;
                        ;
                        ;       OPERATING SYSTEM STATUS CODES
                        ;
     0001               SUCCES  =       $01         ;SUCCESSFUL OPERATION
                        ;
     0080               BRKABT  =       $80         ;BREAK KEY ABORT
     0081               PRVOPN  =       $81         ;IOCB ALREADY OPEN
     0082               NONDEV  =       $82         ;NON-EXISTANT DEVICE
     0083               WRONLY  =       $83         ;IOCB OPENED FOR WRITE ONLY
     0084               NVALID  =       $84         ;INVALID COMMAND
     0085               NOTOPN  =       $85         ;DEVICE OR FILE NOT OPEN
     0086               BADIOC  =       $86         ;INVALID IOCB NUMBER
     0087               RDONLY  =       $87         ;IOCB OPENED FOR READ ONLY
     0088               EOFERR  =       $88         ;END OF FILE
     0089               TRNRCD  =       $89         ;TRUNCATED RECORD
     008A               TIMOUT  =       $8A         ;PERIPHERAL DEVICE TIME OUT
     008B               DNACK   =       $8B         ;DEVICE DOES NOT ACKNOWLEDGE COMMAND
     008C               FRMERR  =       $8C         ;SERIAL BUS FRAMING ERROR
     008D               CRSROR  =       $8D         ;CURSOR OVERRANGE
     008E               OVRRUN  =       $8E         ;SERIAL BUS DATA OVERRUN
     008F               CHKERR  =       $8F         ;SERIAL BUS CHECKSUM ERROR
                        ;
     0090               DERROR  =       $90         ;PERIPHERAL DEVICE ERROR (OPERATION NOT COMP
     0091               BADMOD  =       $91         ;BAD SCREEN MODE NUMBER
     0092               FNCNOT  =       $92         ;FUNCTION NOT IMPLEMENTED IN HANDLER
     0093               SCRMEM  =       $93         ;INSUFICIENT MEMORY FOR SCREEN MODE
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       PAGE ZERO RAM ASSIGNMENTS
                        ;
                                *=$0000
     0000               LINZBS: .RES    2           ;LINBUG RAM (WILL BE REPLACED BY MONITOR RAM
                        ;
                        ; THESE LOCATIONS ARE NOT CLEARED
     0002               CASINI: .RES    2           ;CASSETTE INIT LOCATION
     0004               RAMLO:  .RES    2           ;RAM POINTER FOR MEMORY TEST
     0006               TRAMSZ: .RES    1           ;TEMPORARY REGISTER FOR RAM SIZE
```

```
0007            TSTDAT: .RES    1           ;RAM TEST DATA REGISTER
                ;
                ; CLEARED ON COLDSTART ONLY
0008            WARMST: .RES    1           ;WARM START FLAG
0009            BOOT?:  .RES    1           ;SUCCESSFUL BOOT FLAG
000A            DOSVEC: .RES    2           ;DISK SOFTWARE START VECTOR
000C            DOSINI: .RES    2           ;DISK SOFTWARE INIT ADDRESS
000E            APPMHI: .RES    2           ;APPLICATIONS MEMORY HI LIMIT
                ;
                ; CLEARED ON COLD OR WARM START
0010            INTZBS  =*                  ;INTERRUPT HANDLER
0010            POKMSK: .RES    1           ;SYSTEM MASK FOR POKEY IRQ ENABLE
0011            BRKKEY: .RES    1           ;BREAK KEY FLAG
0012            RTCLOK: .RES    3           ;REAL TIME CLOCK (IN 16 MSEC UNITS)
                ;
0015            BUFADR: .RES    2           ;INDIRECT BUFFER ADDRESS REGISTER
                ;
0017            ICCOMT: .RES    1           ;COMMAND FOR VECTOR
                ;
0018            DSKFMS: .RES    2           ;DISK FILE MANAGER POINTER
001A            DSKUTL: .RES    2           ;DISK UTILITIES POINTER
                ;
001C            PTIMOT: .RES    1           ;PRINTER TIME OUT REGISTER
001D            PBPNT:  .RES    1           ;PRINT BUFFER POINTER
001E            PBUFSZ: .RES    1           ;PRINT BUFFER SIZE
001F            PTEMP:  .RES    1           ;TEMPORARY REGISTER
                ;
0020            ZIOCB   =*                  ;ZERO PAGE I/O CONTROL BLOCK
0010            IOCBSZ  =       16          ;NUMBER OF BYTES PER IOCB
0080            MAXIOC  =       8*IOCBSZ    ;LENGTH OF THE IOCB AREA
0020            IOCBAS  =*
0020            ICHIDZ: .RES    1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
0021            ICDNOZ: .RES    1           ;DEVICE NUMBER (DRIVE NUMBER)
0022            ICCOMZ: .RES    1           ;COMMAND CODE
0023            ICSTAZ: .RES    1           ;STATUS OF LAST IOCB ACTION
0024            ICBALZ: .RES    1           ;BUFFER ADDRESS LOW BYTE
0025            ICBAHZ: .RES    1
0026            ICPTLZ: .RES    1           ;PUT BYTE ROUTINE ADDRESS - 1
0027            ICPTHZ: .RES    1
0028            ICBLLZ: .RES    1           ;BUFFER LENGTH LOW BYTE
0029            ICBLHZ: .RES    1
002A            ICAX1Z: .RES    1           ;AUXILIARY INFORMATION FIRST BYTE
002B            ICAX2Z: .RES    1
002C            ICSPRZ: .RES    4           ;TWO SPARE BYTES (CIO LOCAL USE)
002E            ICIDNO  =       ICSPRZ+2    ;IOCB NUMBER X 16
002F            CIOCHR  =       ICSPRZ+3    ;CHARACTER BYTE FOR CURRENT OPERATION
                ;
0030            STATUS: .RES    1           ;INTERNAL STATUS STORAGE
0031            CHKSUM: .RES    1           ;CHECKSUM (SINGLE BYTE SUM WITH CARRY)
0032            BUFRLO: .RES    1           ;POINTER TO DATA BUFFER (LO BYTE)
0033            BUFRHI: .RES    1           ;POINTER TO DATA BUFFER (HI BYTE)
0034            BFENLO: .RES    1           ;NEXT BYTE PAST END OF THE DATA BUFFER (LO B
0035            BFENHI: .RES    1           ;NEXT BYTE PAST END OF THE DATA BUFFER (HI B
0036            CRETRY: .RES    1           ;NUMBER OF COMMAND FRAME RETRIES
```

```
0037        DRETRY: .RES    1           ;NUMBER OF DEVICE RETRIES
0038        BUFRFL: .RES    1           ;DATA BUFFER FULL FLAG
0039        RECVDN: .RES    1           ;RECEIVE DONE FLAG
003A        XMTDON: .RES    1           ;TRANSMISSION DONE FLAG
003B        CHKSNT: .RES    1           ;CHECKSUM SENT FLAG
003C        NOCKSM: .RES    1           ;NO CHECKSUM FOLLOWS DATA FLAG
            ;
            ;
003D        BPTR:   .RES    1
003E        FTYPE:  .RES    1
003F        FEOF:   .RES    1
0040        FREQ:   .RES    1
0041        SOUNDR: .RES    1           ;NOISY I/O FLAG. (ZERO IS QUIET)
0042        CRITIC: .RES    1           ;DEFINES CRITICAL SECTION (CRITICAL IF NON-Z
            ;
0043        FMSZPG: .RES    7           ;DISK FILE MANAGER SYSTEM ZERO PAGE
            ;
            ;
004A        CKEY:   .RES    1           ;FLAG SET WHEN GAME START PRESSED
004B        CASSBT: .RES    1           ;CASSETTE BOOT FLAG
004C        DSTAT:  .RES    1           ;DISPLAY STATUS
            ;
004D        ATRACT: .RES    1           ;ATRACT FLAG
004E        DRKMSK: .RES    1           ;DARK ATRACT MASK
004F        COLRSH: .RES    1           ;ATRACT COLOR SHIFTER (EOR'ED WITH PLAYFIELD
            ;
0002        LEDGE   =       2           ;LMARGN'S VALUE AT COLD START
0027        REDGE   =       39          ;RMARGN'S VALUE AT COLD START
0050        TMPCHR: .RES    1
0051        HOLD1:  .RES    1
0052        LMARGN: .RES    1           ;LEFT MARGIN (SET TO 1 AT POWER ON)
0053        RMARGN: .RES    1           ;RIGHT MARGIN (SET TO 38 AT POWER ON)
0054        ROWCRS: .RES    1           ;CURSOR COUNTERS
0055        COLCRS: .RES    2
0057        DINDEX: .RES    1
0058        SAVMSC: .RES    2
005A        OLDROW: .RES    1
005B        OLDCOL: .RES    2
005D        OLDCHR: .RES    1           ;DATA UNDER CURSOR
005E        OLDADR: .RES    2
0060        NEWROW: .RES    1           ;POINT DRAW GOES TO
0061        NEWCOL: .RES    2
0063        LOGCOL: .RES    1           ;POINTS AT COLUMN IN LOGICAL LINE
0064        ADRESS: .RES    2
0066        MLTTMP: .RES    2
0066        OPNTMP  =       MLTTMP      ;FIRST BYTE IS USED IN OPEN AS TEMP
0068        SAVADR: .RES    2
006A        RAMTOP: .RES    1           ;RAM SIZE DEFINED BY POWER ON LOGIC
006B        BUFCNT: .RES    1           ;BUFFER COUNT
006C        BUFSTR: .RES    2           ;EDITOR GETCH POINTER
006E        BITMSK: .RES    1           ;BIT MASK
006F        SHFAMT: .RES    1
0070        ROWAC:  .RES    2
0072        COLAC:  .RES    2
```

```
0074            ENDPT:   .RES   2
0076            DELTAR:  .RES   1
0077            DELTAC:  .RES   2
0079            ROWINC:  .RES   1
007A            COLINC:  .RES   1
007B            SWPFLG:  .RES   1        ;NON-0 IF TXT AND REGULAR RAM IS SWAPPED
007C            HOLDCH:  .RES   1        ;CH IS MOVED HERE IN KGETCH BEFORE CNTL & SH
007D            INSDAT:  .RES   1
007E            COUNTR:  .RES   2
                ;
                ;
                ;
                ;
                ;        80 - FF ARE RESERVED FOR USER APPLICATIONS
                ;
                ;
                ;
                ;        NOTE : SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS
                ;
                ;
                ;
                ;
                ;        PAGE 1  -  STACK
                ;
                ;
                ;
                ;
                ;        PAGE TWO RAM ASSIGNMENTS
                ;
                         *=$0200
0200            INTABS   =*              ;INTERRUPT RAM
0200            VDSLST:  .RES   2        ;DISPLAY LIST NMI VECTOR
0202            VPRCED:  .RES   2        ;PROCEED LINE IRQ VECTOR
0204            VINTER:  .RES   2        ;INTERRUPT LINE IRQ VECTOR
0206            VBREAK:  .RES   2        ;SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR
0208            VKEYBD:  .RES   2        ;POKEY KEYBOARD IRQ VECTOR
020A            VSERIN:  .RES   2        ;POKEY SERIAL INPUT READY IRQ
020C            VSEROR:  .RES   2        ;POKEY SERIAL OUTPUT READY IRQ
020E            VSEROC:  .RES   2        ;POKEY SERIAL OUTPUT COMPLETE IRQ
0210            VTIMR1:  .RES   2        ;POKEY TIMER 1 IRQ
0212            VTIMR2:  .RES   2        ;POKEY TIMER 2 IRQ
0214            VTIMR4:  .RES   2        ;POKEY TIMER 4 IRQ
0216            VIMIRQ:  .RES   2        ;IMMEDIATE IRQ VECTOR
0218            CDTMV1:  .RES   2        ;COUNT DOWN TIMER 1
021A            CDTMV2:  .RES   2        ;COUNT DOWN TIMER2
021C            CDTMV3:  .RES   2        ;COUNT DOWN TIMER 3
021E            CDTMV4:  .RES   2        ;COUNT DOWN TIMER 4
0220            CDTMV5:  .RES   2        ;COUNT DOWN TIMER 5
0222            VVBLKI:  .RES   2        ;IMMEDIATE VERTICAL BLANK NMI VECTOR
0224            VVBLKD:  .RES   2        ;DEFERRED VERTICAL BLANK NMI VECTOR
0226            CDTMA1:  .RES   2        ;COUNT DOWN TIMER 1 JSR ADDRESS
0228            CDTMA2:  .RES   2        ;COUNT DOWN TIMER 2 JSR ADDRESS
022A            CDTMF3:  .RES   1        ;COUNT DOWN TIMER 3 FLAG
022B            SRTIMR:  .RES   1        ;SOFTWARE REPEAT TIMER
```

```
022C          CDTMF4: .RES    1           ;COUNT DOWN TIMER 4 FLAG
022D          INTEMP: .RES    1           ;IAN'S TEMP (RENAMED FROM T1 BY POPULAR DEMA
022E          CDTMF5: .RES    1           ;COUNT DOWN TIMER FLAG 5
022F          SDMCTL: .RES    1           ;SAVE DMACTL REGISTER
0230          SDLSTL: .RES    1           ;SAVE DISPLAY LIST LOW BYTE
0231          SDLSTH: .RES    1           ;SAVE DISPLAY LIST HI BYTE
0232          SSKCTL: .RES    1           ;SKCTL REGISTER RAM
0233                  .RES    1           ;
              ;
0234          LPENH:  .RES    1           ;LIGHT PEN HORIZONTAL VALUE
0235          LPENV:  .RES    1           ;LIGHT PEN VERTICAL VALUE
0236          BRKKY:  .RES    2           ;BREAK KEY VECTOR
              ;
0238                  .RES    2           ;SPARE
              ;
023A          CDEVIC: .RES    1           ;COMMAND FRAME BUFFER - DEVICE
023B          CCOMND: .RES    1           ;COMMAND
023C          CAUX1:  .RES    1           ;COMMAND AUX BYTE 1
023D          CAUX2:  .RES    1           ;COMMAND AUX BYTE 2
              ;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
023E          TEMP:   .RES    1           ;TEMPORARY RAM CELL
              ;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
023F          ERRFLG: .RES    1           ;ERROR FLAG - ANY DEVICE ERROR EXCEPT TIME O
              ;
0240          DFLAGS: .RES    1           ;DISK FLAGS FROM SECTOR ONE
0241          DBSECT: .RES    1           ;NUMBER OF DISK BOOT SECTORS
0242          BOOTAD: .RES    2           ;ADDRESS WHERE DISK BOOT LOADER WILL BE PUT
0244          COLDST: .RES    1           ;COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)
              ;
0245                  .RES    1           ;SPARE
              ;
0246          DSKTIM: .RES    1           ;DISK TIME OUT REGISTER
              ;
0247          LINBUF: .RES    40          ;CHAR LINE BUFFER
              ;
026F          GPRIOR: .RES    1           ;GLOBAL PRIORITY CELL
              ;
0270          PADDL0: .RES    1           ;POTENTIOMETER 0 RAM CELL
0271          PADDL1: .RES    1
0272          PADDL2: .RES    1
0273          PADDL3: .RES    1
0274          PADDL4: .RES    1
0275          PADDL5: .RES    1
0276          PADDL6: .RES    1
0277          PADDL7: .RES    1
0278          STICK0: .RES    1           ;JOYSTICK 0 RAM CELL
0279          STICK1: .RES    1
027A          STICK2: .RES    1
027B          STICK3: .RES    1
027C          PTRIG0: .RES    1           ;PADDLE TRIGGER 0
027D          PTRIG1: .RES    1
027E          PTRIG2: .RES    1
027F          PTRIG3: .RES    1
0280          PTRIG4: .RES    1
```

```
0281          PTRIG5: .RES    1
0282          PTRIG6: .RES    1
0283          PTRIG7: .RES    1
0284          STRIG0: .RES    1          ;JOYSTICK TRIGGER 0
0285          STRIG1: .RES    1
0286          STRIG2: .RES    1
0287          STRIG3: .RES    1
              ;
0288          CSTAT:  .RES    1
0289          WMODE:  .RES    1
028A          BLIM:   .RES    1
028B          IMASK:  .RES    1
028C          JVECK:  .RES    2
              ;
028E                  .RES    2          ;SPARE
              ;
              ;
              ;
              ;
0290          TXTROW: .RES    1          ;TEXT ROWCRS
0291          TXTCOL: .RES    2          ;TEXT COLCRS
0293          TINDEX: .RES    1          ;TEXT INDEX
0294          TXTMSC: .RES    2          ;FOOLS CONVRT INTO NEW MSC
0296          TXTOLD: .RES    6          ;OLDROW & OLDCOL FOR TEXT (AND THEN SOME)
029C          TMPX1:  .RES    1
029D          HOLD3:  .RES    1
029E          SUBTMP: .RES    1
029F          HOLD2:  .RES    1
02A0          DMASK:  .RES    1
02A1          TMPLBT: .RES    1
02A2          ESCFLG: .RES    1          ;ESCAPE FLAG
02A3          TABMAP: .RES    15
02B2          LOGMAP: .RES    4          ;LOGICAL LINE START BIT MAP
02B6          INVFLG: .RES    1          ;INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
02B7          FILFLG: .RES    1          ;RIGHT FILL FLAG FOR DRAW
02B8          TMPROW: .RES    1
02B9          TMPCOL: .RES    2
02BB          SCRFLG: .RES    1          ;SET IF SCROLL OCCURS
02BC          HOLD4:  .RES    1          ;TEMP CELL USED IN DRAW ONLY
02BD          HOLD5:  .RES    1          ;DITTO
02BE          SHFLOK: .RES    1
02BF          BOTSCR: .RES    1          ;BOTTOM OF SCREEN : 24 NORM 4 SPLIT
              ;
              ;
02C0          PCOLR0: .RES    1          ;P0 COLOR
02C1          PCOLR1: .RES    1          ;P1 COLOR
02C2          PCOLR2: .RES    1          ;P2 COLOR
02C3          PCOLR3: .RES    1          ;P3 COLOR
02C4          COLOR0: .RES    1          ;COLOR 0
02C5          COLOR1: .RES    1
02C6          COLOR2: .RES    1
02C7          COLOR3: .RES    1
02C8          COLOR4: .RES    1
              ;
```

```
                   ;
02C9                       .RES    23          ;SPARE
                   ;
                   ;
                   ;
02E0               GLBABS  =*                  ;GLOBAL VARIABLES
                   ;
02E0                       .RES    4           ;SPARE
                   ;
02E4               RAMSIZ: .RES    1           ;RAM SIZE (HI BYTE ONLY)
02E5               MEMTOP: .RES    2           ;TOP OF AVAILABLE USER MEMORY
02E7               MEMLO:  .RES    2           ;BOTTOM OF AVAILABLE USER MEMORY
02E9                       .RES    1           ;SPARE
02EA               DVSTAT: .RES    4           ;STATUS BUFFER
02EE               CBAUDL: .RES    1           ;CASSETTE BAUD RATE LOW BYTE
02EF               CBAUDH: .RES    1
                   ;
02F0               CRSINH: .RES    1           ;CURSOR INHIBIT (00 = CURSOR ON)
02F1               KEYDEL: .RES    1           ;KEY DELAY
02F2               CH1:    .RES    1
                   ;
02F3               CHACT:  .RES    1           ;CHACTL REGISTER RAM
02F4               CHBAS:  .RES    1           ;CHBAS REGISTER RAM
                   ;
02F5                       .RES    5           ;SPARE BYTES
                   ;
02FA               CHAR:   .RES    1
02FB               ATACHR: .RES    1           ;ATASCII CHARACTER
02FC               CH:     .RES    1           ;GLOBAL VARIABLE FOR KEYBOARD
02FD               FILDAT: .RES    1           ;RIGHT FILL DATA (DRAW)
02FE               DSPFLG: .RES    1           ;DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
02FF               SSFLAG: .RES    1           ;START/STOP FLAG FOR PAGING (CNTL 1). CLEARE
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;       PAGE THREE RAM ASSIGNMENTS
                   ;
0300               DCB     =*                  ;DEVICE CONTROL BLOCK
0300               DDEVIC: .RES    1           ;PERIPHERAL UNIT 1 BUS I.D. NUMBER
0301               DUNIT:  .RES    1           ;UNIT NUMBER
0302               DCOMND: .RES    1           ;BUS COMMAND
0303               DSTATS: .RES    1           ;COMMAND TYPE/STATUS RETURN
0304               DBUFLO: .RES    1           ;DATA BUFFER POINTER LOW BYTE
0305               DBUFHI: .RES    1
0306               DTIMLO: .RES    1           ;DEVICE TIME OUT IN 1 SECOND UNITS
0307               DUNUSE: .RES    1           ;UNUSED BYTE
0308               DBYTLO: .RES    1           ;NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
0309               DBYTHI: .RES    1
030A               DAUX1:  .RES    1           ;COMMAND AUXILIARY BYTE 1
030B               DAUX2:  .RES    1
```

```
                             ;
      030C                   TIMER1: .RES    2           ;INITIAL TIMER VALUE
      030E                   ADDCOR: .RES    1           ;ADDITION CORRECTION
      030F                   CASFLG: .RES    1           ;CASSETTE MODE WHEN SET
      0310                   TIMER2: .RES    2           ;FINAL TIMER VALUE. THESE TWO TIMER VALUES
                             ; ARE USED TO COMPUTE INTERVAL FOR BAUD RATE
      0312                   TEMP1:  .RES    2           ;TEMPORARY STORAGE REGISTER
      0314                   TEMP2:  .RES    1           ;TEMPORARY STORAGE REGISTER
      0315                   TEMP3:  .RES    1           ;TEMPORARY STORAGE REGISTER
      0316                   SAVIO:  .RES    1           ;SAVE SERIAL IN DATA PORT
      0317                   TIMFLG: .RES    1           ;TIME OUT FLAG FOR BAUD RATE CORRECTION
      0318                   STACKP: .RES    1           ;SIO STACK POINTER SAVE CELL
      0319                   TSTAT:  .RES    1           ;TEMPORARY STATUS HOLDER
                             ;
                             ;
                             ;
      031A                   HATABS: .RES    38          ;HANDLER ADDRESS TABLE
      0021                   MAXDEV  =       *-HATABS-5  ;MAXIMUM HANDLER ADDRESS INDEX
                             ;
                             ;     NOTE : THE ENTIRE IOCB DEFINITIONS HAVE BEEN MODIFIED
                             ;
                             IOCB:   .ORG    *           ;I/O CONTROL BLOCKS
      0340                   ICHID:  .RES    1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
      0341                   ICDNO:  .RES    1           ;DEVICE NUMBER (DRIVE NUMBER)
      0342                   ICCOM:  .RES    1           ;COMMAND CODE
      0343                   ICSTA:  .RES    1           ;STATUS OF LAST IOCB ACTION
      0344                   ICBAL:  .RES    1           ;BUFFER ADDRESS LOW BYTE
      0345                   ICBAH:  .RES    1
      0346                   ICPTL:  .RES    1           ;PUT BYTE ROUTINE ADDRESS - 1
      0347                   ICPTH:  .RES    1
      0348                   ICBLL:  .RES    1           ;BUFFER LENGTH LOW BYTE
      0349                   ICBLH:  .RES    1
      034A                   ICAX1:  .RES    1           ;AUXILIARY INFORMATION FIRST BYTE
      034B                   ICAX2:  .RES    1
      034C                   ICSPR:  .RES    4           ;FOUR SPARE BYTES
      0350                           .RES    MAXIOC-IOCBSZ
                             ;
      03C0                   PRNBUF: .RES    40          ;PRINTER BUFFER
                             ;
      03E8                           .RES    21          ;SPARE BYTES
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;       PAGE FOUR RAM ASSIGNMENTS
                             ;
      03FD                   CASBUF: .RES    131         ;CASSETTE BUFFER
                             ;
                             ; USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE
      0480                   USAREA: .RES    128         ;SPARE
                             ;
```

```
                    ;
                    ;
                    ;
                    ;
                    ;
                    ;
                    ;       PAGE FIVE RAM ASSIGNMENTS
                    ;
                    ;       PAGE FIVE IS RESERVED AS A USER WORK SPACE
                    ;
                    ;       NOTE: SEE FLOATING POINT SUBROUTINE AREA FOR PAGE FIVE CELLS
                    ;
                    ;
                    ;       PAGE SIX RAM ASSIGNMENTS
                    ;
                    ; PAGE SIX IS RESERVED AS A USER'S USER WORK SPACE
                    ;
                    ;
                    ;
                    ;
                    ;       FLOATING POINT SUBROUTINES
                    ;
0006                FPREC   =       6           ;FLOATING PT PRECISION (# OF BYTES)
                    ; IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR
D800                AFP     =       $D800       ;ASCII->FLOATING POINT (FP)
                    ;                               INBUFF+CIX -> FRO, CIX, CARRY
D8E6                FASC    =       $D8E6       ;FP -> ASCII FRO-> LBUFF (INBUFF)
D9AA                IFP     =       $D9AA       ;INTEGER -> FP
                    ;                               0-$FFFF (LSB,MSB) IN FRO,FRO+1->FRO
D9D2                FPI     =       $D9D2       ;FP -> INTEGER FRO -> FRO,FRO+1, CARRY
DA60                FSUB    =       $DA60       ;FRO <- FRO - FR1 ,CARRY
DA66                FADD    =       $DA66       ;FRO <- FRO + FR1 ,CARRY
DADB                FMUL    =       $DADB       ;FRO <- FRO * FR1 ,CARRY
DB28                FDIV    =       $DB28       ;FRO <- FRO / FR1 ,CARRY
DD89                FLDOR   =       $DD89       ;FLOATING LOAD REGO   FRO  <- (X,Y)
DD8D                FLDOP   =       $DD8D       ;   "       "     "   FRO  <- (FLPTR)
DD98                FLD1R   =       $DD98       ;   "       "   REG1  FR1  <- (X,Y)
DD9C                FLD1P   =       $DD9C       ;   "       "     "   FR1  <- (FLPTR)
DDA7                FSTOR   =       $DDA7       ;FLOATING STORE REGO (X,Y) <- FRO
DDAB                FSTOP   =       $DDAB       ;    "       "     " (FLPTR)  <- FRO
DDB6                FMOVE   =       $DDB6       ;FR1 <- FRO
DD40                PLYEVL  =       $DD40       ;FRO <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR
                    ;                           INPUT: (X,Y) = A(N),A(N-1)...A(0)  -> PLYARG
                    ;                                  ACC   = # OF COEFFICIENTS = DEGREE+1
                    ;                                  FRO   = Z
DDC0                EXP     =       $DDC0       ;FRO <- E**FRO = EXP10(FRO * LOG10(E)) CARRY
DDCC                EXP10   =       $DDCC       ;FRO <- 10**FRO CARRY
DECD                LOG     =       $DECD       ;FRO <- LN(FRO) = LOG10(FRO)/LOG10(E) CARRY
DED1                LOG10   =       $DED1       ;FRO <- LOG10 (FRO) CARRY
                    ; THE FOLLOWING ARE IN BASIC CARTRIDGE:
BD81                SIN     =       $BD81       ;FRO <- SIN(FRO) DEGFLG=0 =>RADS, 6=>DEG. CA
BD73                COS     =       $BD73       ;FRO <- COS(FRO) CARRY
BE43                ATAN    =       $BE43       ;FRO <- ATAN(FRO) CARRY
BEB1                SQR     =       $BEB1       ;FRO <- SQUAREROOT(FRO) CARRY
```

```
                ; FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF F.P. ROUTINES ARE CA
                        *=$D4
00D4            FRO:    .RES    FPREC           ;FP REGO
00DA            FRE:    .RES    FPREC
00E0            FR1:    .RES    FPREC           ;FP REG1
00E6            FR2:    .RES    FPREC
00EC            FRX:    .RES    1               ;FP SPARE
00ED            EEXP:   .RES    1               ;VALUE OF E
00EE            NSIGN:  .RES    1               ;SIGN OF #
00EF            ESIGN:  .RES    1               ;SIGN OF EXPONENT
00F0            FCHRFLG:.RES    1               ;1ST CHAR FLAG
00F1            DIGRT:  .RES    1               ;# OF DIGITS RIGHT OF DECIMAL
00F2            CIX:    .RES    1               ;CURRENT INPUT INDEX
00F3            INBUFF: .RES    2               ;POINTS TO USER'S LINE INPUT BUFFER
00F5            ZTEMP1: .RES    2
00F7            ZTEMP4: .RES    2
00F9            ZTEMP3: .RES    2
00FB            DEGFLG
00FB            RADFLG: .RES    1               ;0=RADIANS, 6=DEGREES
0000            RADON   =       0               ;INDICATES RADIANS
0006            DEGON   =       6               ;INDICATES DEGREES
00FC            FLPTR:  .RES    2               ;POINTS TO USER'S FLOATING PT NUMBER
00FE            FPTR2:  .RES    2
                ; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
                ; (NEEDED ONLY IF F.P. ROUTINES CALLED)
                        *=$57E
057E            LBPR1:  .RES    1               ;LBUFF PREFIX 1
057F            LBPR2:  .RES    1               ;LBUFF PREFIX 2
0580            LBUFF:  .RES    128             ;LINE BUFFER
05E0            PLYARG  =       LBUFF+$60       ;POLYNOMIAL ARGUMENTS
05E6            FPSCR   =       PLYARG+FPREC
05EC            FPSCR1  =       FPSCR+FPREC
05E6            FSCR    =       FPSCR
05EC            FSCR1   =       FPSCR1
05FF            LBFEND  =       *-1             ;END OF LBUFF
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ;       COLLEEN MNEMONICS
                ;
D200            POKEY   =       $D200           ;VBLANK ACTION:         DESCRIPTION:
D200            POT0    =       POKEY+0         ;POT0-->PADDL0         0-227 IN RAM CELL
D201            POT1    =       POKEY+1         ;POT1-->PADDL1         0-227 IN RAM CELL
D202            POT2    =       POKEY+2         ;POT2-->PADDL2         0-227 IN RAM CELL
D203            POT3    =       POKEY+3         ;POT3-->PADDL3         0-227 IN RAM CELL
D204            POT4    =       POKEY+4         ;POT4-->PADDL4         0-227 IN RAM CELL
D205            POT5    =       POKEY+5         ;POT5-->PADDL5         0-227 IN RAM CELL
D206            POT6    =       POKEY+6         ;POT6-->PADDL6         0-227 IN RAM CELL
```

```
D207      POT7    =  POKEY+7    ;POT7-->PADDL7        0-227 IN RAM CELL
D208      ALLPOT  =  POKEY+8    ;???
D209      KBCODE  =  POKEY+9
D20A      RANDOM  =  POKEY+10
D20B      POTGO   =  POKEY+11   ;STROBED
D20D      SERIN   =  POKEY+13
D20E      IRQST   =  POKEY+14
D20F      SKSTAT  =  POKEY+15
D200      AUDF1   =  POKEY+0
D201      AUDC1   =  POKEY+1
D202      AUDF2   =  POKEY+2
D203      AUDC2   =  POKEY+3
D204      AUDF3   =  POKEY+4
D205      AUDC3   =  POKEY+5
D206      AUDF4   =  POKEY+6
D207      AUDC4   =  POKEY+7
D208      AUDCTL  =  POKEY+8    ;NONE               AUDCTL<--[SIO]
D209      STIMER  =  POKEY+9
D20A      SKRES   =  POKEY+10   ;NONE                SKRES<--[SIO]
D20D      SEROUT  =  POKEY+13   ;NONE               SEROUT<--[SIO]
D20E      IRQEN   =  POKEY+14   ;POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
D20F      SKCTL   =  POKEY+15   ;SSKCTL-->SKCTL       SSKCTL<--[SIO]
          ;
D000      CTIA    =  $D000      ;VBLANK ACTION:        DESCRIPTION:
D000      HPOSP0  =  CTIA+0
D001      HPOSP1  =  CTIA+1
D002      HPOSP2  =  CTIA+2
D003      HPOSP3  =  CTIA+3
D004      HPOSM0  =  CTIA+4
D005      HPOSM1  =  CTIA+5
D006      HPOSM2  =  CTIA+6
D007      HPOSM3  =  CTIA+7
D008      SIZEP0  =  CTIA+8
D009      SIZEP1  =  CTIA+9
D00A      SIZEP2  =  CTIA+10
D00B      SIZEP3  =  CTIA+11
D00C      SIZEM   =  CTIA+12
D00D      GRAFP0  =  CTIA+13
D00E      GRAFP1  =  CTIA+14
D00F      GRAFP2  =  CTIA+15
D010      GRAFP3  =  CTIA+16
D011      GRAFM   =  CTIA+17
D012      COLPM0  =  CTIA+18    ;PCOLR0-->COLPM0       WITH ATTRACT MODE
D013      COLPM1  =  CTIA+19    ;PCOLR1-->COLPM1       WITH ATTRACT MODE
D014      COLPM2  =  CTIA+20    ;PCOLR2-->COLPM2       WITH ATTRACT MODE
D015      COLPM3  =  CTIA+21    ;PCOLR3-->COLPM3       WITH ATTRACT MODE
D016      COLPF0  =  CTIA+22    ;COLOR0-->COLPF0       WITH ATTRACT MODE
D017      COLPF1  =  CTIA+23    ;COLOR1-->COLPF1       WITH ATTRACT MODE
D018      COLPF2  =  CTIA+24    ;COLOR2-->COLPF2       WITH ATTRACT MODE
D019      COLPF3  =  CTIA+25    ;COLOR3-->COLPF3       WITH ATTRACT MODE
D01A      COLBK   =  CTIA+26    ;COLOR4-->COLBK        WITH ATTRACT MODE
D01B      PRIOR   =  CTIA+27    ;(ON OPEN S: OR E:)    GPRIOR-->PRIOR
D01C      VDELAY  =  CTIA+28
D01D      GRACTL  =  CTIA+29
```

```
     712  D01E                 HITCLR   =      CTIA+30
     713  D01F                 CONSOL   =      CTIA+31     ;$08-->CONSOL         TURN OFF SPEAKER
     714  D000                 M0PF     =      CTIA+0
     715  D001                 M1PF     =      CTIA+1
     716  D002                 M2PF     =      CTIA+2
     717  D003                 M3PF     =      CTIA+3
     718  D004                 P0PF     =      CTIA+4
     719  D005                 P1PF     =      CTIA+5
     720  D006                 P2PF     =      CTIA+6
     721  D007                 P3PF     =      CTIA+7
     722  D008                 M0PL     =      CTIA+8
     723  D009                 M1PL     =      CTIA+9
     724  D00A                 M2PL     =      CTIA+10
     725  D00B                 M3PL     =      CTIA+11
     726  D00C                 P0PL     =      CTIA+12
     727  D00D                 P1PL     =      CTIA+13
     728  D00E                 P2PL     =      CTIA+14
     729  D00F                 P3PL     =      CTIA+15
     730  D010                 TRIG0    =      CTIA+16     ;TRIG0-->STRIG0
     731  D011                 TRIG1    =      CTIA+17     ;TRIG1-->STRIG1
     732  D012                 TRIG2    =      CTIA+18     ;TRIG2-->STRIG2
     733  D013                 TRIG3    =      CTIA+19     ;TRIG3-->STRIG3
     734                       ;
     735  D400                 ANTIC    =      $D400       ;VBLANK ACTION        DESCRIPTION
     736  D400                 DMACTL   =      ANTIC+0     ;DMACTL<--SDMCTL      ON OPEN S: OR E:
     737  D401                 CHACTL   =      ANTIC+1     ;CHACTL<--CHACT       ON OPEN S: OR E:
     738  D402                 DLISTL   =      ANTIC+2     ;DLISTL<--SDLSTL      ON OPEN S: OR E:
     739  D403                 DLISTH   =      ANTIC+3     ;DLISTH<--SDLSTH      ON OPEN S: OR E:
     740  D404                 HSCROL   =      ANTIC+4
     741  D405                 VSCROL   =      ANTIC+5
     742  D407                 PMBASE   =      ANTIC+7
     743  D409                 CHBASE   =      ANTIC+9     ;CHBASE<--CHBAS       ON OPEN S: OR E:
     744  D40A                 WSYNC    =      ANTIC+10
     745  D40B                 VCOUNT   =      ANTIC+11
     746  D40C                 PENH     =      ANTIC+12
     747  D40D                 PENV     =      ANTIC+13
     748  D40E                 NMIEN    =      ANTIC+14    ;NMIEN<--40 POWER     ON AND [SETVBV]
     749  D40F                 NMIRES   =      ANTIC+15    ;STROBED
     750  D40F                 NMIST    =      ANTIC+15
     751  D300                 PIA      =      $D300       ;VBLANK ACTION        DESCRIPTION
     752  D300                 PORTA    =      PIA+0       ;PORTA-->STICK0,1     X-Y CONTROLLERS
     753  D301                 PORTB    =      PIA+1       ;PORTB-->STICK2,3     X-Y CONTROLLERS
     754  D302                 PACTL    =      PIA+2       ;NONE                 PACTL<--3C [INIT]
     755  D303                 PBCTL    =      PIA+3       ;NONE                 PBCTL<--3C [INIT]
     756                       ;
     757                       ;
     758                       ;
     759                       ;  .PAGE
```

```
     760                            .PAGE
     761                            LIST    S
     762                            .TITLE  'CENTRAL INPUT/OUTPUT (CIO) 2-7-79'
     763                    ;               UPDATED BY AL MILLER 3-9-79
     764  0030              ASCZER  =       '0          ;ASCII ZERO
     765  003A              COLON   =       $3A         ;ASCII COLON
     766  009B              EOL     =       $9B         ;END OF RECORD
```

```
      767                               .PAGE
      768                       ;
      769                       ; CIO JUMP VECTOR FOR USERS
      770                               *=CIOV
      771  E456  4C C4 E4               JMP     CIO         ;GO TO CIO
      772                       ;
      773                       ; CIO INIT JUMP VECTOR FOR POWER UP
      774                               *=CIOINV
      775  E46E  4C A6 E4               JMP     CIOINT      ;GO TO INIT
      776                       ;
      777                       ;
      778                       ; ERROR ROUTINE ADDRESS EQUATE
      779                       ; ERRTNH =ERRTN/256         "MOVED TO LINE 788"
      780                       ; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
      781                       ;
      782                       ;
      783                               *=CIOORG
      784                       ;
      785                       ; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
      786  E4A6  A2 00          CIOINT: LDX     #0
      787  E4A8  A9 FF          CIOI1:  LDA     #IOCFRE     ;SET ALL IOCB'S TO FREE
      788  E4AA  9D 40 03               STA     ICHID,X     ;BY SETTING HANDLER ID'S=$FF
      789  E4AD  A9 C0                  LDA     #ERRTNL
      790  E4AF  9D 46 03               STA     ICPTL,X     ;POINT PUT TO ERROR ROUTINE
      791  E4B2  A9 E4                  LDA     #ERRTNH
      792  E4B4  9D 47 03               STA     ICPTH,X
      793  E4B7  8A                     TXA
      794  E4B8  18                     CLC
      795  E4B9  69 10                  ADC     #IOCBSZ     ;BUMP INDEX BY SIZE
      796  E4BB  AA                     TAX
      797  E4BC  C9 80                  CMP     #MAXIOC     ;DONE?
      798  E4BE  90 E8                  BCC     CIOI1       ;NO
      799  E4C0  60                     RTS                 ;YES, RETURN
      800                       ;
      801                       ; ERROR ROUTINE FOR ILLEGAL PUT
      802  E4C0                 ERRTN   =*-1
      803  00E4                 ERRTNH  =ERRTN/256
      804  00C0                 ERRTNL  =(-ERRTNH)*256+ERRTN
      805  E4C1  A0 85                  LDY     #NOTOPN     ;IOCB NOT OPEN
      806  E4C3  60                     RTS
```

```
                          .PAGE
                   ;
                   ; CIO LOCAL RAM (USES SPARE BYTES IN ZERO PAGE IOCB)
002C               ENTVEC  =       ICSPRZ
                   ;
                   ; CIO MAIN ROUTINE
                   ;
                   ; CIO INTERFACES BETWEEN USER AND INPUT/OUTPUT DE
E4C4  85 2F        CIO:    STA     CIOCHR      ;SAVE POSSIBLE OUTPUT CHARACTER
E4C6  86 2E                STX     ICIDNO      ;SAVE IOCB NUMBER * N
                   ;
                   ; CHECK FOR LEGAL IOCB
E4C8  8A                   TXA
E4C9  29 0F                AND     #$F         ;IS IOCB MULTIPLE OF 16?
E4CB  D0 04                BNE     CIERR1      ;NO, ERROR
E4CD  E0 80                CPX     #MAXIOC     ;IS INDEX TOO LARGE?
E4CF  90 05                BCC     IOC1        ;NO
                   ;
                   ; INVALID IOCB NUMBER -- RETURN ERROR
E4D1  A0 86        CIERR1: LDY     #BADIOC     ;ERROR CODE
E4D3  4C 1B E6             JMP     CIRTN1      ;RETURN
                   ;
                   ; MOVE USER IOCB TO ZERO PAGE
E4D6  A0 00        IOC1:   LDY     #0
E4D8  BD 40 03     IOC1A:  LDA     IOCB,X      ;USER IOCB
E4DB  99 20 00             STA     IOCBAS,Y    ;TO ZERO PAGE
E4DE  E8                   INX
E4DF  C8                   INY
E4E0  C0 0C                CPY     #12         ;12 BYTES
E4E2  90 F4                BCC     IOC1A
                   ;
                   ; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
E4E4  A0 84                LDY     #NVALID     ;ASSUME INVALID CODE
E4E6  A5 22                LDA     ICCOMZ      ;COMMAND CODE TO INDEX
E4E8  C9 03                CMP     #OPEN       ;IS COMMAND LEGAL?
E4EA  90 25                BCC     CIERR4      ;NO
E4EC  A8                   TAY
                   ;
                   ; MOVE COMMAND TO ZERO BASE FOR INDEX
E4ED  C0 0E                CPY     #SPECIL     ;IS COMMAND SPECIAL?
E4EF  90 02                BCC     IOC2        ;NO
E4F1  A0 0E                LDY     #SPECIL     ;YES, SET SPECIAL OFFSET INDEX
E4F3  84 17        IOC2:   STY     ICCOMT      ;SAVE COMMAND FOR VECTOR
E4F5  B9 C6 E6             LDA     COMTAB-3,Y  ;GET VECTOR OFFSET FROM TABLE
E4F8  F0 0F                BEQ     CIOPEN      ;GO IF OPEN COMMAND
E4FA  C9 02                CMP     #2          ;IS IT CLOSE?
E4FC  F0 35                BEQ     CICLOS      ;YES
E4FE  C9 08                CMP     #8          ;IS IT STATUS OR SPECIAL?
E500  B0 4C                BCS     CISTSP      ;YES
E502  C9 04                CMP     #4          ;IS IT READ?
E504  F0 63                BEQ     CIREAD      ;YES
E506  4C C9 E5             JMP     CIWRIT      ;ELSE, MUST BE WRITE
```

```
                                .PAGE
                        ;
                        ; OPEN COMMAND
                        ;
                        ; FIND DEVICE HANDLER IN HANDLER ADDRESS TABLE
E509  A5 20             CIOPEN: LDA     ICHIDZ          ;GET HANDLER ID
E50B  C9 FF                     CMP     #IOCFRE         ;IS THIS IOCB CLOSED?
E50D  F0 05                     BEQ     IOC6            ;YES
                        ;
                        ; ERROR -- IOCB ALREADY OPEN
E50F  A0 81             CIERR3: LDY     #PRVOPN         ;ERROR CODE
E511  4C 1B E6          CIERR4: JMP     CIRTN1          ;RETURN
                        ;
                        ; GO FIND DEVICE
E514  20 9E E6          IOC6:   JSR     DEVSRC          ;CALL DEVICE SEARCH
E517  B0 F8                     BCS     CIERR4          ;GO IF DEVICE NOT FOUND
                        ;
                        ; DEVICE FOUND, INITIALIZE IOCB FOR OPEN
                        ;
                        ; COMPUTE HANDLER ENTRY POINT
E519  20 3D E6          IOC7:   JSR     COMENT
E51C  B0 F3                     BCS     CIERR4          ;GO IF ERROR IN COMPUTE
                        ;
                        ; GO TO HANDLER FOR INITIALIZATION
E51E  20 89 E6                  JSR     GOHAND          ;USE INDIRECT JUMP
                        ;
                        ; STORE PUT BYTE ADDRESS-1 INTO IOCB
E521  A9 0B                     LDA     #PUTCHR         ;SIMULATE PUT CHARACTER
E523  85 17                     STA     ICCOMT
E525  20 3D E6                  JSR     COMENT          ;COMPUTE ENTRY POINT
E528  A5 2C                     LDA     ICSPRZ          ;MOVE COMPUTED VALUE
E52A  85 26                     STA     ICPTLZ          ;TO PUT BYTE ADDRESS
E52C  A5 2D                     LDA     ICSPRZ+1
E52E  85 27                     STA     ICPTHZ
E530  4C 1D E6                  JMP     CIRTN2          ;RETURN TO USER
```

```
                             .PAGE
                     ;
                     ;
                     ; CLOSE COMMAND
E533  A0 01          CICLOS: LDY     #SUCCES     ;ASSUME GOOD CLOSE
E535  84 23                  STY     ICSTAZ
E537  20 3D E6               JSR     COMENT      ;COMPUTE HANDLER ENTRY POINT
E53A  B0 03                  BCS     CICLO2      ;GO IF ERROR IN COMPUTE
E53C  20 89 E6               JSR     GOHAND      ;GO TO HANDLER TO CLOSE DEVICE
E53F  A9 FF          CICLO2: LDA     #IOCFRE     ;GET IOCB "FREE" VALUE
E541  85 20                  STA     ICHIDZ      ;SET HANDLER ID
E543  A9 E4                  LDA     #ERRTNH
E545  85 27                  STA     ICPTHZ      ;SET PUT BYTE TO POINT TO ERROR
E547  A9 C0                  LDA     #ERRTNL
E549  85 26                  STA     ICPTLZ
E54B  4C 1D E6               JMP     CIRTN2      ;RETURN
                     ;
                     ;
                     ; STATUS AND SPECIAL REQUESTS
                     ; DO IMPLIED OPEN IF NECESSARY AND GO TO DEVICE
E54E  A5 20          CISTSP: LDA     ICHIDZ      ;IS THERE A HANDLER ID?
E550  C9 FF                  CMP     #IOCFRE
E552  D0 05                  BNE     CIST1       ;YES
                     ;
                     ; IOCB IS FREE, DO IMPLIED OPEN
E554  20 9E E6               JSR     DEVSRC      ;FIND DEVICE IN TABLE
E557  B0 B8                  BCS     CIERR4      ;GO IF ERROR IN COMPUTE
                     ;
                     ; COMPUTE AND GO TO ENTRY POINT IN HANDLER
E559  20 3D E6       CIST1:  JSR     COMENT      ;COMPUTER HANDLER ENTRY VECTOR
E55C  20 89 E6               JSR     GOHAND      ;GO TO HANDLER
                     ;
                     ; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
E55F  A6 2E                  LDX     ICIDNO      ;IOCB INDEX
E561  BD 40 03               LDA     ICHID,X     ;GET ORIGINAL HANDLER ID
E564  85 20                  STA     ICHIDZ      ;RESTORE ZERO PAGE
E566  4C 1D E6               JMP     CIRTN2      ;RETURN
```

```
                                  .PAGE
                          ;
                          ; READ -- DO GET COMMANDS
E569  A5 22               CIREAD: LDA     ICCOMZ      ;GET COMMAND BYTE
E56B  25 2A                       AND     ICAX1Z      ;IS THIS READ LEGAL?
E56D  D0 05                       BNE     RCI1A       ;YES
                          ;
                          ; ILLEGAL READ -- IOCB OPENED FOR WRITE ONLY
E56F  A0 83                       LDY     #WRONLY     ;ERROR CODE
E571  4C 1B E6            RCI1B:  JMP     CIRTN1      ;RETURN
                          ;
                          ; COMPUTE AND CHECK ENTRY POINT
E574  20 3D E6            RCI1A:  JSR     COMENT      ;COMPUTE ENTRY POINT
E577  B0 F8                       BCS     RCI1B       ;GO IF ERROR IN COMPUTE
                          ;
                          ; GET RECORD OR CHARACTERS
E579  A5 28                       LDA     ICBLLZ
E57B  05 29                       ORA     ICBLLZ+1    ;IS BUFFER LENGTH ZERO?
E57D  D0 08                       BNE     RCI3        ;NO
E57F  20 89 E6                    JSR     GOHAND
E582  85 2F                       STA     CIOCHR
E584  4C 1D E6                    JMP     CIRTN2
                          ;
                          ; LOOP TO FILL BUFFER OR END RECORD
E587  20 89 E6            RCI3:   JSR     GOHAND      ;GO TO HANDLER TO GET BYTE
E58A  85 2F                       STA     CIOCHR      ;SAVE BYTE
E58C  30 35                       BMI     RCI4        ;END TRANSFER IF ERROR
E58E  A0 00                       LDY     #0
E590  91 24                       STA     (ICBALZ),Y  ;PUT BYTE IN USER BUFFER
E592  20 70 E6                    JSR     INCBFP      ;INCREMENT BUFFER POINTER
E595  A5 22                       LDA     ICCOMZ      ;GET COMMAND CODE
E597  29 02                       AND     #2          ;IS IT GET RECORD?
E599  D0 0C                       BNE     RCI1        ;NO
                          ;
                          ; CHECK FOR EOL ON TEXT RECORDS
E59B  A5 2F                       LDA     CIOCHR      ;GET BYTE
E59D  C9 9B                       CMP     #EOL        ;IS IT AN EOL?
E59F  D0 06                       BNE     RCI1        ;NO
E5A1  20 63 E6                    JSR     DECBFL      ;YES, DECREMENT BUFFER LENGTH
E5A4  4C C3 E5                    JMP     RCI4        ;END TRANSFER
                          ;
                          ; CHECK BUFFER FULL
E5A7  20 63 E6            RCI1:   JSR     DECBFL      ;DECREMENT BUFFER LENGTH
E5AA  D0 DB                       BNE     RCI3        ;CONTINUE IF NON ZERO
```

```
                                  .PAGE
                          ;
                          ; BUFFER FULL, RECORD NOT ENDED
                          ; DISCARD BYTES UNTIL END OF RECORD
E5AC  A5 22               RCI2:   LDA     ICCOMZ      ;GET COMMAND BYTE
E5AE  29 02                       AND     #2          ;IS IT GET CHARACTER?
E5B0  D0 11                       BNE     RCI4        ;YES, END TRANSFER
                          ;
                          ; LOOP TO WAIT FOR EOL
E5B2  20 89 E6            RCI6:   JSR     GOHAND      ;GET BYTE FROM HANDLER
E5B5  85 2F                       STA     CIOCHR      ;SAVE CHARACTER
E5B7  30 0A                       BMI     RCI4        ;GO IF ERROR
                          ;
                          ; TEXT RECORD, WAIT FOR EOL
E5B9  A5 2F                       LDA     CIOCHR      ;GET GOT BYTE
E5BB  C9 9B                       CMP     #EOL        ;IS IT EOL?
E5BD  D0 F3                       BNE     RCI6        ;NO, CONTINUE
                          ;
                          ; END OF RECORD, BUFFER FULL -- SEND TRUNCATED RECORD MESSAGE
E5BF  A9 89               RCI11:  LDA     #TRNRCD     ;ERROR CODE
E5C1  85 23                       STA     ICSTAZ      ;STORE IN IOCB
                          ;
                          ; TRANSFER DONE
E5C3  20 77 E6            RCI4:   JSR     SUBBFL      ;SET FINAL BUFFER LENGTH
E5C6  4C 1D E6                    JMP     CIRTN2      ;RETURN
```

```
                             .PAGE
                    ;
                    ; WRITE -- DO PUT COMMANDS
E5C9  A5 22         CIWRIT: LDA     ICCOMZ       ;GET COMMAND BYTE
E5CB  25 2A                 AND     ICAX1Z       ;IS THIS WRITE LEGAL?
E5CD  D0 05                 BNE     WCI1A        ;YES
                    ;
                    ; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
E5CF  A0 87                 LDY     #RDONLY      ;ERROR CODE
E5D1  4C 1B E6      WCI1B:  JMP     CIRTN1       ;RETURN
                    ;
                    ; COMPUTE AND CHECK ENTRY POINT
E5D4  20 3D E6      WCI1A:  JSR     COMENT       ;COMPUTE HANDLER ENTRY POINT
E5D7  B0 F8                 BCS     WCI1B        ;GO IF ERROR IN COMPUTE
                    ;
                    ; PUT RECORD OR CHARACTERS
E5D9  A5 28                 LDA     ICBLLZ
E5DB  05 29                 ORA     ICBLLZ+1     ;IS BUFFER LENGTH ZERO?
E5DD  D0 06                 BNE     WCI3         ;NO
E5DF  A5 2F                 LDA     CIOCHR       ;GET CHARACTER
E5E1  E6 28                 INC     ICBLLZ       ;SET BUFFER LENGTH=1
E5E3  D0 06                 BNE     WCI4         ;THEN JUST TRANSFER ONE BYTE
                    ;
                    ; LOOP TO TRANSFER BYTES FROM BUFFER TO HANDLER
E5E5  A0 00         WCI3:   LDY     #0
E5E7  B1 24                 LDA     (ICBALZ),Y   ;GET BYTE FROM BUFFER
E5E9  85 2F                 STA     CIOCHR       ;SAVE
E5EB  20 89 E6      WCI4:   JSR     GOHAND       ;GO PUT BYTE
E5EE  30 25                 BMI     WCI5         ;END IF ERROR
E5F0  20 70 E6              JSR     INCBFP       ;INCREMENT BUFFER POINTER
                    ;
                    ; CHECK FOR TEXT RECORD
E5F3  A5 22                 LDA     ICCOMZ       ;GET COMMAND BYTE
E5F5  29 02                 AND     #2           ;IS IT PUT RECORD?
E5F7  D0 0C                 BNE     WCI1         ;NO
                    ;
                    ; TEXT RECORD -- CHECK FOR EOL TRANSFER
E5F9  A5 2F                 LDA     CIOCHR       ;GET LAST CHARACTER
E5FB  C9 9B                 CMP     #EOL         ;IS IT AN EOL?
E5FD  D0 06                 BNE     WCI1         ;NO
E5FF  20 63 E6              JSR     DECBFL       ;DECREMENT BUFFER LENGTH
E602  4C 15 E6              JMP     WCI5         ;END TRANSFER
                    ;
                    ; CHECK FOR BUFFER EMPTY
E605  20 63 E6      WCI1:   JSR     DECBFL       ;DECREMENT BUFFER LENGTH
E608  D0 DB                 BNE     WCI3         ;CONTINUE IF NON ZERO
```

```
                                .PAGE
                        ;
                        ; BUFFER EMPTY, RECORD NOT FILLED
                        ; CHECK TYPE OF TRANSFER
E60A  A5 22             WCI2:   LDA     ICCOMZ      ;GET COMMAND CODE
E60C  29 02                     AND     #2          ;IS IT PUT CHARACTER?
E60E  D0 05                     BNE     WCI5        ;YES, END TRANSFER
                        ;
                        ; PUT RECORD (TEXT), BUFFER EMPTY, SEND EOL
E610  A9 9B                     LDA     #EOL
E612  20 89 E6                  JSR     GOHAND      ;GO TO HANDLER
                        ;
                        ; END PUT TRANSFER
E615  20 77 E6          WCI5:   JSR     SUBBFL      ;SET ACTUAL PUT BUFFER LENGTH
E618  4C 1D E6                  JMP     CIRTN2      ;RETURN
```

```
                                 .PAGE
                         ;
                         ; CIO RETURNS
                         ; RETURNS WITH Y=STATUS
E61B  84 23              CIRTN1: STY     ICSTAZ      ;SAVE STATUS
                         ;
                         ; RETURNS WITH STATUS STORED IN ICSTAZ
                         ; MOVE IOCB IN ZERO PAGE BACK TO USER AREA
E61D  A4 2E              CIRTN2: LDY     ICIDNO      ;GET IOCB INDEX
E61F  B9 44 03                   LDA     ICBAL,Y
E622  85 24                      STA     ICBALZ      ;RESTORE USER BUFFER POINTER
E624  B9 45 03                   LDA     ICBAH,Y
E627  85 25                      STA     ICBAHZ
E629  A2 00                      LDX     #0          ;LOOP COUNT AND INDEX
E62B  B5 20              CIRT3:  LDA     IOCBAS,X    ;ZERO PAGE
E62D  99 40 03                   STA     IOCB,Y      ;TO USER AREA
E630  E8                         INX
E631  C8                         INY
E632  E0 0C                      CPX     #12         ;12 BYTES
E634  90 F5                      BCC     CIRT3
                         ;
                         ; RESTORE A,X, & Y
E636  A5 2F                      LDA     CIOCHR      ;GET LAST CHARACTER
E638  A6 2E                      LDX     ICIDNO      ;IOCB INDEX
E63A  A4 23                      LDY     ICSTAZ      ;GET STATUS AND SET FLAGS
E63C  60                         RTS                 ;RETURN TO USER
```

```
                                      .PAGE
                          ;
                          ;
                          ; CIO SUBROUTINES
                          ;
                          ; COMENT -- CHECK AND COMPUTE HANDLER ENTRY POINT
E63D  A4 20               COMENT: LDY   ICHIDZ      ;GET HANDLER INDEX
E63F  C0 22                       CPY   #MAXDEV+1   ;IS IT A LEGAL INDEX?
E641  90 04                       BCC   COM1        ;YES
                          ;
                          ; ILLEGAL HANDLER INDEX MEANS DEVICE NOT OPEN FOR OPERATION
E643  A0 85                       LDY   #NOTOPN     ;ERROR CODE
E645  B0 1B                       BCS   COM2        ;RETURN
                          ;
                          ; USE HANDLER ADDRESS TABLE AND COMMAND TABLE TO GET VECTOR
E647  B9 1B 03            COM1:   LDA   HATABS+1,Y  ;GET LOW BYTE OF ADDRESS
E64A  85 2C                       STA   ICSPRZ      ;AND SAVE IN POINTER
E64C  B9 1C 03                    LDA   HATABS+2,Y  ;GET HI BYTE OF ADDRESS
E64F  85 2D                       STA   ICSPRZ+1
E651  A4 17                       LDY   ICCOMT      ;GET COMMAND CODE
E653  B9 C6 E6                    LDA   COMTAB-3,Y  ;GET COMMAND OFFSET
E656  A8                          TAY
E657  B1 2C                       LDA   (ICSPRZ),Y  ;GET LOW BYTE OF VECTOR FROM
E659  AA                          TAX               ;HANDLER ITSELF AND SAVE
E65A  C8                          INY
E65B  B1 2C                       LDA   (ICSPRZ),Y  ;GET HI BYTE OF VECTOR
E65D  85 2D                       STA   ICSPRZ+1
E65F  86 2C                       STX   ICSPRZ      ;SET LO BYTE
E661  18                          CLC               ;SHOW NO ERROR
E662  60                  COM2:   RTS
                          ;
                          ;
                          ; DECBFL -- DECREMENT BUFFER LENGTH DOUBLE BYTE
                          ; Z FLAG = 0 ON RETURN IF LENGTH = 0 AFTER DECREMENT
E663  C6 28               DECBFL: DEC   ICBLLZ      ;DECREMENT LOW BYTE
E665  A5 28                       LDA   ICBLLZ      ;CHECK IT
E667  C9 FF                       CMP   #$FF        ;DID IT GO BELOW?
E669  D0 02                       BNE   DECBF1      ;NO
E66B  C6 29                       DEC   ICBLLZ+1    ;DECREMENT HI BYTE
E66D  05 29               DECBF1: ORA   ICBLLZ+1    ;SET Z IF BOTH ARE ZERO
E66F  60                          RTS
                          ;
                          ;
                          ; INCBFP -- INCREMENT WORKING BUFFER POINTER
E670  E6 24               INCBFP: INC   ICBALZ      ;BUMP LOW BYTE
E672  D0 02                       BNE   INCBF1      ;GO IF NOT ZERO
E674  E6 25                       INC   ICBALZ+1    ;ELSE, BUMP HI BYTE
E676  60                  INCBF1: RTS
                          ;
                          ;
                          ; SUBBFL -- SET BUFFER LENGTH = BUFFER LENGTH - WORKING BYTE COUNT
E677  A6 2E               SUBBFL: LDX   ICIDNO      ;GET IOCB INDEX
E679  38                          SEC
E67A  BD 48 03                    LDA   ICBLL,X     ;GET LOW BYTE OF INITIAL LENGTH
```

```
E67D  E5 28               SBC    ICBLLZ       ;SUBTRACT FINAL LOW BYTE
E67F  85 28               STA    ICBLLZ       ;AND SAVE BACK
E681  BD 49 03            LDA    ICBLH,X      ;GET HI BYTE
E684  E5 29               SBC    ICBLLZ+1
E686  85 29               STA    ICBLHZ
E688  60                  RTS
                   ;
                   ;
                   ; GOHAND -- GO INDIRECT TO A DEVICE HANDLER
                   ; Y= STATUS ON RETURN, N FLAG=1 IF ERROR ON RETURN
E689  A0 92        GOHAND: LDY   #FNCNOT      ;PREPARE NO FUNCTION STATUS FOR HANDLER RTS
E68B  20 93 E6            JSR    CIJUMP       ;USE THE INDIRECT JUMP
E68E  84 23               STY    ICSTAZ       ;SAVE STATUS
E690  C0 00               CPY    #0           ;AND SET N FLAG
E692  60                  RTS
                   ;
                   ; INDIRECT JUMP TO HANDLER BY PAUL'S METHOD
E693  AA           CIJUMP: TAX                ;SAVE A
E694  A5 2D               LDA    ICSPRZ+1     ;GET JUMP ADDRESS HI BYTE
E696  48                  PHA                 ;PUT ON STACK
E697  A5 2C               LDA    ICSPRZ       ;GET JUMP ADDRESS LO BYTE
E699  48                  PHA                 ;PUT ON STACK
E69A  8A                  TXA                 ;RESTORE A
E69B  A6 2E               LDX    ICIDNO       ;GET IOCB INDEX
E69D  60                  RTS                 ;GO TO HANDLER INDIRECTLY
```

```
                                        .PAGE
                              ;
                              ; DEVSRC -- DEVICE SEARCH, FIND DEVICE IN HANDLER ADDRESS TABLE
                              ;
                              ; LOOP TO FIND DEVICE
E69E  A0 00                   DEVSRC: LDY     #0
E6A0  B1 24                           LDA     (ICBALZ),Y  ;GET DEVICE NAME FROM USER
E6A2  F0 0C                           BEQ     CIERR2
E6A4  A0 21                           LDY     #MAXDEV     ;INITIAL COMPARE INDEX
E6A6  D9 1A 03                DEVS1:  CMP     HATABS,Y    ;IS THIS THE DEVICE?
E6A9  F0 0A                           BEQ     DEVS2       ;YES
E6AB  88                              DEY
E6AC  88                              DEY                 ;ELSE, POINT TO NEXT DEVICE NAME
E6AD  88                              DEY
E6AE  10 F6                           BPL     DEVS1       ;CONTINUE FOR ALL DEVICES
                              ;
                              ; NO DEVICE FOUND, DECLARE NON-EXISTENT DEVICE ERROR
E6B0  A0 82                   CIERR2: LDY     #NONDEV     ;ERROR CODE
E6B2  38                              SEC                 ;SHOW ERROR
E6B3  B0 13                           BCS     DEVS4       ;AND RETURN
                              ;
                              ; FOUND DEVICE, SET ICHID,ICDNO, AND INIT DEVICE
E6B5  98                      DEVS2:  TYA
E6B6  85 20                           STA     ICHIDZ      ;SAVE HANDLER INDEX
E6B8  38                              SEC
E6B9  A0 01                           LDY     #1
E6BB  B1 24                           LDA     (ICBALZ),Y  ;GET DEVICE NUMBER (DRIVE NUMBER)
E6BD  E9 30                           SBC     #ASCZER     ;SUBTRACT ASCII ZERO
E6BF  C9 0A                           CMP     #$A         ;IS NUMBER IN RANGE?
E6C1  90 02                           BCC     DEVS3       ;YES
E6C3  A9 01                           LDA     #1          ;NO, DEFAULT TO ONE
E6C5  85 21                   DEVS3:  STA     ICDNOZ      ;SAVE DEVICE NUMBER
E6C7  18                              CLC                 ;SHOW NO ERROR
                              ;
                              ; RETURN
E6C8  60                      DEVS4:  RTS
```

```
    1202                                 .PAGE
    1203                         ;
    1204                         ;
    1205                         ; CIO ROM TABLES
    1206                         ;
    1207                         ; COMMAND TABLE
    1208                         ; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
    1209  E6C9  00 04 04 04      COMTAB: .BYTE   0,4,4,4,4,6,6,6,6,2,8,10
    1210  E6CD  04 06 06 06
    1211  E6D1  06 02 08 0A
    1212  022F                   LENGTH  =*-CIOINT
    1213  E6D5                   CRNTP1  =*
    1214                                 *=$14
    1215  0014  00               CIOSPR: .BYTE   INTORG-CRNTP1 ;^GCIOL IS TOO LONG
```

```

                              .TITLE  'INTERRUPT HANDLER'
                      ;LIVES ON DK1:INTHV.SRC
0006                  SRTIM2  =       6            ;SECOND REPEAT INTERVAL
                      ;
                      ; THIS IS TO MAKE DOS 2 WORK WHICH USED AN ABSOLUTE ADDRESS
                      ;
                              *=$E912
E912  4C ED E8                JMP     SETVBL
                              *=SETVBV
E45C  4C ED E8                JMP     SETVBL
E45F  4C AE E7                JMP     SYSVBL
E462  4C 05 E9                JMP     XITVBL
                              *=INTINV
E46B  4C D5 E6                JMP     IHINIT
                      ;
                              *=VCTABL+INTABS-VDSLST
                      ;
E480  90 E7                   .WORD   SYRTI        ;VDSLST
E482  8F E7                   .WORD   SYIRQB       ;VPRCED
E484  8F E7                   .WORD   SYIRQB       ;VINTER
E486  8F E7                   .WORD   SYIRQB       ;VBREAK
                      ;
E488                          .RES    8
E490  8F E7                   .WORD   SYIRQB       ;VTIMR1
E492  8F E7                   .WORD   SYIRQB       ;VTIMR2
E494  8F E7                   .WORD   SYIRQB       ;VTIMR4
E496  06 E7                   .WORD   SYIRQ        ;VIMIRQ
E498  00 00 00 00             .WORD   0,0,0,0,0    ;CDTMV1-4
E49C  00 00 00 00
E4A0  00 00
E4A2  AE E7                   .WORD   SYSVBL       ;VVBLKI
E4A4  05 E9                   .WORD   XITVBL       ;VVBLKD
                      ;
                              *=$900C
                      ;
900C  A9 E6                   LDA     #PIRQH       ;SET UP RAM VECTORS FOR LINBUG VERSION
900E  8D F9 FF                STA     $FFF9
9011  A9 F3                   LDA     #PIRQL
9013  8D F8 FF                STA     $FFF8
9016  A9 E7                   LDA     #PNMIH
9018  8D FB FF                STA     $FFFB
901B  A9 91                   LDA     #PNMIL
901D  8D FA FF                STA     $FFFA
9020  60                      RTS
```

```
                                .PAGE
                        ;
                        ; IRQ HANDLER
                        ;
                        ; JUMP THRU IMMEDIATE IRQ VECTOR, WHICH ORDINARILY POINTS TO
                        ; SYSTEM IRQ: DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
                        ;
                                *=INTORG
E6D5  A9 40             IHINIT: LDA     #$40        ;VBL ON BUF DLIST OFF***FOR NOW***
E6D7  8D 0E D4                  STA     NMIEN       ;ENABLE DISPLAY LIST, VERTICAL BLANK
E6DA  A9 38                     LDA     #$38        ;LOOK AT DATA DIRECTION REGISTERS IN PIA
E6DC  8D 02 D3                  STA     PACTL
E6DF  8D 03 D3                  STA     PBCTL
E6E2  A9 00                     LDA     #0          ;MAKE ALL INPUTS
E6E4  8D 00 D3                  STA     PORTA
E6E7  8D 01 D3                  STA     PORTB
E6EA  A9 3C                     LDA     #$3C        ;BACK TO PORTS
E6EC  8D 02 D3                  STA     PACTL
E6EF  8D 03 D3                  STA     PBCTL
E6F2  60                        RTS
E6F3  6C 16 02          PIRQ:   JMP     (VIMIRQ)
E6F6  80                CMPTAB: .BYTE   $80         ;BREAK KEY
E6F7  40                        .BYTE   $40         ;KEY STROKE
E6F8  04                        .BYTE   $04         ;TIMER 4
E6F9  02                        .BYTE   $02         ;TIMER 2
E6FA  01                        .BYTE   $01         ;TIMER 1
E6FB  08                        .BYTE   $08         ;SERIAL OUT COMPLETE
E6FC  10                        .BYTE   $10         ;SERIAL OUT READY
E6FD  20                        .BYTE   $20         ;SERIAL  IN READY
E6FE
                        ; THIS IS A TABLE OF OFFSETS INTO PAGE 2.  THEY POINT TO
E6FE  36                ADRTAB: .BYTE   BRKKY-INTABS
E6FF  08                        .BYTE   VKEYBD-INTABS
E700  14                        .BYTE   VTIMR4-INTABS
E701  12                        .BYTE   VTIMR2-INTABS
E702  10                        .BYTE   VTIMR1-INTABS
E703  0E                        .BYTE   VSEROC-INTABS
E704  0C                        .BYTE   VSEROR-INTABS
E705  0A                        .BYTE   VSERIN-INTABS
E706
E706  48                SYIRQ:  PHA                 ;SAVE ACCUMULATOR
E707  AD 0E D2                  LDA     IRQST       ; CHECK FOR SERIAL IN
E70A  29 20                     AND     #$20
E70C  D0 0D                     BNE     SYIRQ2
E70E  A9 DF                     LDA     #$DF        ; MASK ALL OTHERS
E710  8D 0E D2                  STA     IRQEN
E713  A5 10                     LDA     POKMSK
E715  8D 0E D2                  STA     IRQEN
E718  6C 0A 02                  JMP     (VSERIN)
E71B  8A                SYIRQ2: TXA                 ;PUT X INTO ACC
E71C  48                        PHA                 ;SAVE X ONTO STACK
E71D  A2 06                     LDX     #$6         ;START WITH SIX OFFSET
E71F  BD F6 E6          LOOPM:  LDA     CMPTAB,X    ;LOAD MASK
E722  E0 05                     CPX     #5          ;CHECK TO SEE IF COMPLETE IS SET
```

```
E724  D0 04                BNE    LOOPM2
E726  25 10                AND    POKMSK       ;IS THIS INTERRUPT ENABLED?
E728  F0 05                BEQ    LL
E72A  2C 0E D2    LOOPM2:  BIT    IRQST        ; IS IT THE INTERUPT?
E72D  F0 06                BEQ    JMPP
E72F  CA          LL:      DEX                 ;NO DEC X AND TRY NEXT MASK
E730  10 ED                BPL    LOOPM        ;IF NOT NEG GOTO LOOPM
E732  4C 62 E7             JMP    SYIRQ8       ;DONE BUT NO INTERUPT
E735  49 FF       JMPR:    EOR    #$FF         ;COMPLEMENT MASK
E737  8D 0E D2             STA    IRQEN        ;ENABLE ALL OTHERS
E73A  A5 10                LDA    POKMSK       ; GET POKE MASK
E73C  8D 0E D2             STA    IRQEN        ; ENABLE THOSE IN POKE MASK
E73F  BD FE E6             LDA    ADRTAB,X
E742  AA                   TAX
E743  BD 00 02             LDA    INTABS,X     ; GET ADDRESS LOW PART
E746  8D 8C 02             STA    JVECK        ; PUT IN VECTOR
E749  BD 01 02             LDA    INTABS+1,X   ; GET ADDRESS HIGH PART
E74C  8D 8D 02             STA    JVECK+1      ; PUT IN VECTOR HIGH PART
E74F  68                   PLA                 ; PULL X REGISTER FROM STACK
E750  AA                   TAX                 ; PUT IT INTO X
E751  6C 8C 02             JMP    (JVECK)      ; JUMP TO THE PROPER ROUTINE
E754  A9 00       BRKKY2:  LDA    #0           ; BREAK KEY ROUTINE
E756  85 11                STA    BRKKEY       ; SET BREAK KEY FLAG
E758  8D FF 02             STA    SSFLAG       ; START/STOP FLAG
E75B  8D F0 02             STA    CRSINH       ; CURSOR INHIBIT
E75E  85 4D                STA    ATRACT       ; TURN OFF ATRACT MODE
E760  68                   PLA
E761  40                   RTI                 ;EXIT FROM INT
E762  68          SYIRQ8:  PLA
E763  AA                   TAX
E764  2C 02 D3             BIT    PACTL        ;PROCEED ***I GUESS***
E767  10 06                BPL    SYIRQ9
E769  AD 00 D3             LDA    PORTA        CLEAR INT STATUS BIT
E76C  6C 02 02             JMP    (VPRCED)
E76F  2C 03 D3    SYIRQ9:  BIT    PBCTL        ;INTERRUPT ***I GUESS***
E772  10 06                BPL    SYIRQA
E774  AD 01 D3             LDA    PORTB        ;CLEAR INT STATUS
E777  6C 04 02             JMP    (VINTER)
E77A  68          SYIRQA:  PLA
E77B  8D 8C 02             STA    JVECK
E77E  68                   PLA
E77F  48                   PHA
E780  29 10                AND    #$10         ;B BIT OF P REGISTER
E782  F0 07                BEQ    SYRTI2
E784  AD 8C 02             LDA    JVECK
E787  48                   PHA
E788  6C 06 02             JMP    (VBREAK)
E78B  AD 8C 02    SYRTI2:  LDA    JVECK
E78E  48                   PHA
E78F  68          SYIRQB:  PLA
E790  40          SYRTI:   RTI                 ;UNIDENTIFIED INTERRUPT, JUST RETURN.
```

```
                                 .PAGE
                         ;
                         ; NMI HANDLER
                         ;
                         ; DETERMINE CAUSE AND JUMP THRU VECTOR
                         ;
E791  2C 0F D4           PNMI:   BIT     NMIST
E794  10 03                      BPL     PNMI1      ;SEE IF DISPLAY LIST
E796  6C 00 02                   JMP     (VDSLST)
E799  48                 PNMI1:  PHA
E79A  AD 0F D4                   LDA     NMIST
E79D  29 20                      AND     #$20       ;SEE IF RESET
E79F  F0 03                      BEQ     *+5
E7A1  4C 74 E4                   JMP     WARMSV     ;GO THRU WARM START JUMP
E7A4  8A                         TXA                ;SAVE REGISTERS
E7A5  48                         PHA
E7A6  98                         TYA
E7A7  48                         PHA
E7A8  8D 0F D4                   STA     NMIRES     ;RESET INTERRUPT STATUS
E7AB  6C 22 02                   JMP     (VVBLKI)   ;JUMP THRU VECTOR
```

```
                               .PAGE
                        ;
                        ; SYSTEM VBLANK ROUTINE
                        ;
                        ; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET, CLEAR
                        ; SET DLISTL, DLISTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.
                        ;
E7AE  E6 14             SYSVBL: INC     RTCLOK+2      ;INC FRAME COUNTER
E7B0  D0 08                     BNE     SYSVB1
E7B2  E6 4D                     INC     ATRACT        ;INCREMENT ATRACT (CAUSES ATRACT WHEN MINUS)
E7B4  E6 13                     INC     RTCLOK+1
E7B6  D0 02                     BNE     SYSVB1
E7B8  E6 12                     INC     RTCLOK
E7BA  A9 FE             SYSVB1: LDA     #$FE          ;{ATRACT] SET DARK MASK TO NORMAL
E7BC  A2 00                     LDX     #0            ;SET COLRSH TO NORMAL
E7BE  A4 4D                     LDY     ATRACT        ;TEST ATRACT FOR NEGATIVE
E7C0  10 06                     BPL     VBATRA        ;WHILE POSITIVE, DONT GO INTO ATRACT
E7C2  85 4D                     STA     ATRACT        ;IN ATRACT, SO STAY BY STA $FE
E7C4  A6 13                     LDX     RTCLOK+1      ;COLOR SHIFT FOLLOWS RTCLOK+1
E7C6  A9 F6                     LDA     #$F6          ;SET DARK MASK TO DARK
E7C8  85 4E             VBATRA: STA     DRKMSK
E7CA  86 4F                     STX     COLRSH
E7CC  A2 00                     LDX     #0            ;POINT TO TIMER1
E7CE  20 D0 E8                  JSR     DCTIMR        ;GO DECREMENT TIMER1
E7D1  D0 03                     BNE     SYSVB2        ;BRANCH IF STILL COUNTING
E7D3  20 CA E8                  JSR     JTIMR1        ;GO JUMP TO ROUTINE
E7D6  A5 42             SYSVB2: LDA     CRITIC
E7D8  D0 08                     BNE     XXIT          ;GO IF CRITICAL SET
E7DA  BA                        TSX                   ;SEE IF I WAS SET
E7DB  BD 04 01                  LDA     $104,X        ;GET STACKED P
E7DE  29 04                     AND     #$04          ;I BIT
E7E0  F0 03                     BEQ     SYSVB3        ;BRANCH IF OK
E7E2  4C 05 E9          XXIT:   JMP     XITVBL        ;I WAS SET, EXIT
E7E5  AD 0D D4          SYSVB3: LDA     PENV
E7E8  8D 35 02                  STA     LPENV
E7EB  AD 0C D4                  LDA     PENH
E7EE  8D 34 02                  STA     LPENH
E7F1  AD 31 02                  LDA     SDLSTH
E7F4  8D 03 D4                  STA     DLISTH
E7F7  AD 30 02                  LDA     SDLSTL
E7FA  8D 02 D4                  STA     DLISTL
E7FD  AD 2F 02                  LDA     SDMCTL
E800  8D 00 D4                  STA     DMACTL
E803  AD 6F 02                  LDA     GPRIOR        ;GLOBAL PRIOR
E806  8D 1B D0                  STA     PRIOR
E809  A2 08                     LDX     #$08          ;TURN OFF KEYBOARD SPEAKER
E80B  8E 1F D0                  STX     CONSOL
E80E  58                SCOLLP: CLI                   ;DISABLE INTERUPTS
E80F  BD C0 02                  LDA     PCOLR0,X      ;LOAD COLOR REGISTERS FROM RAM
E812  45 4F                     EOR     COLRSH        ;DO COLOR SHIFT
E814  25 4E                     AND     DRKMSK        ;AND DARK ATRACT
E816  9D 12 D0                  STA     COLPM0,X
E819  CA                        DEX
E81A  10 F2                     BPL     SCOLLP
```

```
E81C  AD F4 02              LDA     CHBAS
E81F  8D 09 D4              STA     CHBASE
E822  AD F3 02              LDA     CHACT
E825  8D 01 D4              STA     CHACTL
E828  A2 02                 LDX     #2          ;POINT TO TIMER 2
E82A  20 D0 E8              JSR     DCTIMR
E82D  D0 03                 BNE     SYSVB4      ;IF DIDNT GO ZERO
E82F  20 CD E8              JSR     JTIMR2      ;GO JUMP TO TIMER2 ROUTINE
E832  A2 02        SYSVB4:  LDX     #2          ;RESTORE X
E834  E8           SYSVBB:  INX
E835  E8                    INX
E836  BD 18 02              LDA     CDTMV1,X
E839  1D 19 02              ORA     CDTMV1+1,X
E83C  F0 06                 BEQ     SYSVBA
E83E  20 D0 E8              JSR     DCTIMR      ;DECREMENT AND SET FLAG IF NONZERO
E841  9D 26 02              STA     CDTMF3-4,X
E844  E0 08        SYSVBA:  CPX     #8          ;SEE IF DONE ALL 3
E846  D0 EC                 BNE     SYSVBB      ;LOOP
                   ; CHECK DEBOUNCE COUNTER
E848  AD 0F D2              LDA     SKSTAT
E84B  29 04                 AND     #$04        ;KEY DOWN BIT
E84D  F0 08                 BEQ     SYVB6A      ;IF KEY DOWN
                   ; KEY UP SO COUNT IT
E84F  AD F1 02              LDA     KEYDEL      ;KEY DELAY COUNTER
E852  F0 03                 BEQ     SYVB6A      ;IF COUNTED DOWN ALREADY
E854  CE F1 02              DEC     KEYDEL      ;COUNT IT
                   ; CHECK SOFTWARE REPEAT TIMER
E857  AD 2B 02     SYVB6A:  LDA     SRTIMR
E85A  F0 17                 BEQ     SYSVB7      ;DOESN'T COUNT
E85C  AD 0F D2              LDA     SKSTAT
E85F  29 04                 AND     #$04        ;CHECK KEY DOWN BIT
E861  D0 60                 BNE     SYSVB6      ;BRANCH IF NO LONGER DOWN
E863  CE 2B 02              DEC     SRTIMR      ;COUNT FRAME OF KEY DOWN
E866  D0 0B                 BNE     SYSVB7      ;BRANCH IF NOT RUN OUT
                   ; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
E868  A9 06                 LDA     #SRTIM2     ;TIMER VALUE
E86A  8D 2B 02              STA     SRTIMR      ;SET TIMER
E86D  AD 09 D2              LDA     KBCODE      ;GET THE KEY
E870  8D FC 02              STA     CH          ;PUT INTO CH
                   ; READ GAME CONTROLLERS
E873  A0 01        SYSVB7:  LDY     #1
E875  A2 03                 LDX     #3
E877  B9 00 D3     STLOOP:  LDA     PORTA,Y
E87A  4A                    LSR     A
E87B  4A                    LSR     A
E87C  4A                    LSR     A
E87D  4A                    LSR     A
E87E  9D 78 02              STA     STICK0,X    ;STORE JOYSTICK
E881  CA                    DEX
E882  B9 00 D3              LDA     PORTA,Y
E885  29 0F                 AND     #$F
E887  9D 78 02              STA     STICK0,X    ;STORE JOYSTICK
E88A  CA                    DEX
E88B  88                    DEY
```

```
E88C  10 E9               BPL    STLOOP
                 ;
E88E  A2 03               LDX    #3
E890  BD 10 D0   STRL:    LDA    TRIGO,X      ;MOVE JOYSTICK TRIGGERS
E893  9D 84 02            STA    STRIGO,X
E896  BD 00 D2            LDA    POTO,X       ;MOVE POT VALUES
E899  9D 70 02            STA    PADDLO,X
E89C  BD 04 D2            LDA    POT4,X
E89F  9D 74 02            STA    PADDL4,X
E8A2  CA                  DEX
E8A3  10 EB               BPL    STRL
E8A5  8D 0B D2            STA    POTGO        ;START POTS FOR NEXT TIME
                 ;
E8A8  A2 06               LDX    #6
E8AA  A0 03               LDY    #3
E8AC  B9 78 02   PTRLP:   LDA    STICKO,Y     ;TRANSFER BITS FROM JOYSTICKS
E8AF  4A                  LSR    A            ;TO PADDLE TRIGGERS
E8B0  4A                  LSR    A
E8B1  4A                  LSR    A
E8B2  9D 7D 02            STA    PTRIG1,X
E8B5  A9 00               LDA    #0
E8B7  2A                  ROL    A
E8B8  9D 7C 02            STA    PTRIGO,X
E8BB  CA                  DEX
E8BC  CA                  DEX
E8BD  88                  DEY
E8BE  10 EC               BPL    PTRLP
                 ;
E8C0  6C 24 02            JMP    (VVBLKD)     ;GO TO DEFERRED VBLANK ROUTINE
00E8             SV7H     =      SYSVB7/256
0073             SV7L     =      (-256)*SV7H+SYSVB7
E8C3  A9 00      SYSVB6:  LDA    #0
E8C5  8D 2B 02            STA    SRTIMR       ;ZERO TIMER
E8C8  F0 A9               BEQ    SYSVB7       ;UNCOND
E8CA  6C 26 02   JTIMR1:  JMP    (CDTMA1)
E8CD  6C 28 02   JTIMR2:  JMP    (CDTMA2)
                 ;
                 ; SUBROUTINE TO DECREMENT A COUNTDOWN TIMER
                 ; ENTRY X=OFFSET FROM TIMER 1
                 ; EXIT A,P=ZERO IF WENT ZERO, FF OTHERWISE
                 ;
E8D0  BC 18 02   DCTIMR:  LDY    CDTMV1,X     ;LO BYTE
E8D3  D0 08               BNE    DCTIM1       ;NONZERO, GO DEC IT
E8D5  BC 19 02            LDY    CDTMV1+1,X   ;SEE IF BOTH ZERO
E8D8  F0 10               BEQ    DCTXF        ;YES, EXIT NONZERO
E8DA  DE 19 02            DEC    CDTMV1+1,X   ;DEC HI BYTE
E8DD  DE 18 02   DCTIM1:  DEC    CDTMV1,X     ;DEC LO BYTE
E8E0  D0 08               BNE    DCTXF
E8E2  BC 19 02            LDY    CDTMV1+1,X
E8E5  D0 03               BNE    DCTXF
E8E7  A9 00               LDA    #0           ;WENT ZERO, RETURN ZERO
E8E9  60                  RTS
E8EA  A9 FF      DCTXF:   LDA    #$FF         ;RETURN NONZERO
E8EC  60                  RTS
```

```
                                   .PAGE
                             ;
                             ; SUBROUTINE TO SET VERTICAL BLANK VECTORS AND TIMERS
                             ; ENTRY X=HI,Y=LO BYTE TO SET
                             ;        A= 1-5 TIMERS 1-5
                             ;           6 IMM VBLANK
                             ;           7 DEF VBLANK
                             ;
E8ED  0A                     SETVBL: ASL     A              ;MUL BY 2
E8EE  8D 2D 02                       STA     INTEMP
E8F1  8A                             TXA
E8F2  A2 05                          LDX     #5
E8F4  8D 0A D4                       STA     WSYNC          ;WASTE 20 CPU CYCLES
E8F7  CA                     SETLOP: DEX                    ;TO ALOWD VBLANK TO HAPPEN
E8F8  D0 FD                          BNE     SETLOP         ;IF THIS IS LINE "7C"
E8FA  AE 2D 02                       LDX     INTEMP
E8FD  9D 17 02                       STA     CDTMV1-1,X
E900  98                             TYA
E901  9D 16 02                       STA     CDTMV1-2,X
E904  60                             RTS
                             ;
                             ; EXIT FROM VERTICAL BLANK
                             ;
E905  68                     XITVBL: PLA                    ;UNSTACK Y
E906  A8                             TAY
E907  68                             PLA                    ;UNSTACK X
E908  AA                             TAX
E909  68                             PLA                    ;UNSTACK A
E90A  40                             RTI                    ;AND GO BACK FROM WHENCE.
00E6                         PIRQH   =       PIRQ/256
00F3                         PIRQL   =       (-256)*PIRQH+PIRQ
00E7                         PNMIH   =       PNMI/256
0091                         PNMIL   =       (-256)*PNMIH+PNMI
                             ; SPARE BYTE OR MODULE TOO LONG FLAG
E90B                         CRNTP2  =*
                                     *=$14
0014  39                     INTSPR: .BYTE   SIOORG-CRNTP2 ;^GINTHV IS TOO LONG
```

```

                 .TITLE  'SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )'
        ;        COLLEEN OPERATING SYSTEM
        ;
        ;        SIO  ( SERIAL BUS INPUT/OUTPUT CONTROLLER )
        ;        WITH SOFTWARE BAUD RATE CORRECTION ON CASSETTE
        ;
        ;
        ;                AL MILLER       3-APR-79
        ;
        ;
        ; THIS MODULE HAS ONE ENTRY POINT.  IT IS CALLED BY THE DEVICE
        ; HANDLERS.  IT INTERPRETS A PREVIOUSLY ESTABLISHED DEVICE CONTROL
        ; BLOCK (STORED IN GLOBAL RAM) TO ISSUE COMMANDS
        ; TO THE SERIAL BUS TO CONTROL TRANSMITTING AND RECEIVING DATA.
        ;
        ;
        ;
        ;
```

```
                    .PAGE
            ; EQUATES
            ;
            ; DCD DEVICE BUS ID NUMBERS
0030        FLOPPY  =       $30
            ;PRINTR =       $40
            ;CASSET =       $60             ;!!!!! *****
0060        CASET   =       $60             ;!!!!! *****
            ;
            ;
            ;BUS COMMANDS
            ;
0052        READ    =       'R
0057        WRITE   =       'W
            ;STATIS = 'S
            ;FORMAT = '!
            ;
            ;
            ; COMMAND AUX BYTES
            ;
0053        SIDWAY  =       'S              ;PRINT 16 CHARACTERS SIDEWAYS
004E        NORMAL  =       'N              ;PRINT 40 CHARACTERS NORMALLY
0044        DOUBLE  =       'D              ;PRINT 20 CHARACTERS DOUBLE WIDE
0050        PLOT    =       'P              ;PLOT MODE
            ;
            ;
            ; BUS RESPONSES
            ;
0041        ACK     =       'A              ;DEVICE ACKNOWLEDGES INFORMATION
004E        NACK    =       'N              ;DEVICE DID NOT UNDERSTAND
0043        COMPLT  =       'C              ;DEVICE SUCCESSFULLY COMPLETED OPERATION
0045        ERROR   =       'E              ;DEVICE INCURRED AN ERROR IN AN ATTEMPTED OP
            ;
            ;
            ; MISCELLANEOUS EQUATES
            ;
0028        B192LO  =       $28             ;19200 BAUD RATE POKEY COUNTER VALUES (LO BY
0000        B192HI  =       $00             ;(HI BYTE)
00CC        B600LO  =       $CC             ;600 BAUD (LO BYTE)
0005        B600HI  =       $05             ;(HI BYTE)
0005        HITONE  =       $05             ;FSK HI FREQ POKEY COUNTE VALUE (5326 HZ)
0007        LOTONE  =       $07             ;FSK LO FREQ POKEY COUNTER VALUE (3995 HZ)
            ;
                    .IF     PALFLG
            WIRGLO  =       150             ;WRITE INTER RECORD GAP (IN 1/60 SEC)
            RIRGLO  =       100             ;READ INTER RECORD GAP (IN 1/60 SEC)
            WSIRG   =        13             ;SHORT WRITE INTER RECORD GAP
            RSIRG   =         8             ;SHORT READ INTER RECORD GAP
                    .ENDIF
                    .IF     PALFLG-1
00B4        WIRGLO  =       180             ;WRITE INTER RECORD GAP (IN 1/60 SEC)
0078        RIRGLO  =       120             ;READ INTER RECORD GAP (IN 1/60 SEC)
000F        WSIRG   =       15              ;SHORT WRITE INTER RECORD GAP
000A        RSIRG   =       10              ;SHORT READ INTER RECORD GAP
```

```
1658                     .ENDIF
1659  0000          WIRGHI  =       0
1660  0000          RIRGHI  =       0
1661                ;
1662  0034          NCOMLO  =       $34          ;PIA COMMAND TO LOWER NOT COMMAND LINE
1663  003C          NCOMHI  =       $3C          ;PIA COMMAND TO RAISE NOT COMMAND LINE
1664  0034          MOTRGO  =       $34          ;PIA COMMAND TO TURN ON CASSETTE MOTOR
1665  003C          MOTRST  =       $3C          ;PIA COMMAND TO TURN OFF MOTOR
1666                ;
1667  0002          TEMPHI  =       TEMP/256     ;ADDRESS OF TEMP CELL (HI BYTE)
1668  003E          TEMPLO  =       (-256)*TEMPHI+TEMP ;(LO BYTE)
1669  0002          CBUFHI  =       CDEVIC/256   ;ADDRESS OF COMMAND BUFFER (HI BYTE)
1670  003A          CBUFLO  =       (-256)*CBUFHI+CDEVIC ;(LO BYTE)
1671                ;
1672  000D          CRETRI  =       13           ;NUMBER OF COMMAND FRAME RETRIES
1673  0001          DRETRI  =       1            ;NUMBER OF DEVICE RETRIES
1674  0002          CTIMLO  =       2            ;COMMAND FRAME ACK TIME OUT (LO BYTE)
1675  0000          CTIMHI  =       0            ;COMMAND FRAME ACK TIME OUT (HI BYTE)
1676                ;
1677                ;
1678                ;JTADRH  =       JTIMER/256    ;HI BYTE OF JUMP TIMER ROUTINE ADDR      "M
1679                ;JTADRL  =       (-256)*JTADRH+JTIMER    ;"MOVED TO LINE 1428"
1680                ;
```

```
                                    .PAGE
                            ;       SIO
                            ;
                            ;
                                    *=SIOV
E459  4C 59 E9                      JMP     SIO          ;SIO ENTRY POINT
                            ;
                                    *=SIOINV
E465  4C 44 E9                      JMP     SIOINT       ;SIO INITIALIZATION ENTRY POINT
                            ;
                                    *=SENDEV
E468  4C F2 EB                      JMP     SENDEN       ;SEND ENABLE ENTRY POINT
                            ;
                                    *=VCTABL-INTABS+VSERIN
                            ;
E48A  0F EB                         .WORD   ISRSIR       ;VSERIN
E48C  90 EA                         .WORD   ISRODN       ;VSEROR
E48E  CF EA                         .WORD   ISRTD        ;VSEROC
                            ;
                            ;
                            ;
                                    *=SIOORG
                            ;
                            ; SIO INITIALIZATION SUBROUTINE
                            ;
E944  A9 3C                 SIOINT: LDA     #MOTRST
E946  8D 02 D3                      STA     PACTL        ;TURN OFF MOTOR
                            ;
E949  A9 3C                         LDA     #NCOMHI
E94B  8D 03 D3                      STA     PBCTL        ;RAISE NOT COMMAND LINE
                            ;
                            ;
E94E  A9 03                         LDA     #3
E950  8D 32 02                      STA     SSKCTL       ;GET POKEY OUT OF INITIALIZE MODE
E953  85 41                         STA     SOUNDR       ;INIT POKE ADDRESS FOR QUIET I/O
E955  8D 0F D2                      STA     SKCTL
                            ;
                            ;
E958  60                            RTS                  ;RETURN
                            ;
                            ;
                            ;
                            ;
                            ;
                            ;
E959  BA                    SIO:    TSX
E95A  8E 18 03                      STX     STACKP       ;SAVE STACK POINTER
E95D  A9 01                         LDA     #1
E95F  85 42                         STA     CRITIC
                            ;
E961  AD 00 03                      LDA     DDEVIC
E964  C9 60                         CMP     #CASET
E966  D0 03                         BNE     NOTCST       ;BRANCH IF NOT CASSETTE
E968  4C 80 EB                      JMP     CASENT       ;OTHERWISE JUMP TO CASSETTE ENTER
```

```
                           ;
                           ; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
                           ;
E96B  A9 00                NOTCST: LDA    #0
E96D  8D 0F 03                     STA    CASFLG        ;INIT CASSETTE FLAG TO NO CASSETTE
                           ;
E970  A9 01                        LDA    #DRETRI       ;SET NUMBER OF DEVICE RETRIES
E972  85 37                        STA    DRETRY
E974  A9 0D                COMMND: LDA    #CRETRI       ;SET NUMBER OF COMMAND FRAME RETRIES
E976  85 36                        STA    CRETRY
                           ;
                           ; SEND A COMMAND FRAME
                           ;
E978  A9 28                COMFRM: LDA    #B192LO       ;SET BAUD RATE TO 19200
E97A  8D 04 D2                     STA    AUDF3
E97D  A9 00                        LDA    #B192HI
E97F  8D 06 D2                     STA    AUDF4
                           ;
E982  18                           CLC                  ;SET UP COMMAND BUFFER
E983  AD 00 03                     LDA    DDEVIC
E986  6D 01 03                     ADC    DUNIT
E989  69 FF                        ADC    #$FF          ;SUBTRACT 1
E98B  8D 3A 02                     STA    CDEVIC        ;SET BUS ID NUMBER
                           ;
E98E  AD 02 03                     LDA    DCOMND
E991  8D 3B 02                     STA    CCOMND        ;SET BUS COMMAND
                           ;
E994  AD 0A 03                     LDA    DAUX1         ;STORE COMMAND FRAME AUX BYTES 1 AND 2
E997  8D 3C 02                     STA    CAUX1
E99A  AD 0B 03                     LDA    DAUX2
E99D  8D 3D 02                     STA    CAUX2         ;DONE SETTING UP COMMAND BUFFER
                           ;
E9A0  18                           CLC                  ;SET BUFFER POINTER TO COMMAND FRAME BUFFER
E9A1  A9 3A                        LDA    #CBUFLO
E9A3  85 32                        STA    BUFRLO        ;AND BUFFER END ADDRESS
E9A5  69 04                        ADC    #4
E9A7  85 34                        STA    BFENLO
E9A9  A9 02                        LDA    #CBUFHI
E9AB  85 33                        STA    BUFRHI
E9AD  85 35                        STA    BFENHI        ;DONE SETTING UP BUFFER POINTER
                           ;
E9AF  A9 34                        LDA    #NCOMLO       ;LOWER NOT COMMAND LINE
E9B1  8D 03 D3                     STA    PBCTL
                           ;
E9B4  20 8A EC                     JSR    SENDIN        ;SEND THE COMMAND FRAME TO A SMART DEVICE
                           ;
E9B7  AD 3F 02                     LDA    ERRFLG
E9BA  D0 03                        BNE    BADCOM        ;BRANCH IF AN ERROR RECEIVED
                           ;
E9BC  98                           TYA
E9BD  D0 07                        BNE    ACKREC        ;BRANCH IF ACK RECEIVED
                           ;
                           ;
E9BF  C6 36                BADCOM: DEC    CRETRY        ;A NACK OR TIME OUT OCCURED
```

```
E9C1  10 B5                      BPL     COMFRM      ;SO BRANCH IF ANY RETRIES LEFT
                         ;
E9C3  4C 06 EA                   JMP     DERR1       ;OTHERWISE, JUMP TO RETURN SECTION
                         ;
                         ;
E9C6  AD 03 03           ACKREC: LDA     DSTATS      ;ACK WAS RECEIVED
E9C9  10 0C                      BPL     WATCOM      ;BRANCH TO WAIT FOR COMPLETE ,
                         ; IF THERE IS NO DATA TO BE SENT
                         ;
                         ;
                         ;
                         ; SEND A DATA FRAME TO PERIPHERAL
                         ;
E9CB  A9 0D                      LDA     #CRETRI     ;SET NUMBER OF RETRIES
E9CD  85 36                      STA     CRETRY
                         ;
E9CF  20 6A EB                   JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                         ;
E9D2  20 8A EC                   JSR     SENDIN      ;GO SEND THE DATA FRAME TO A SMART DEVICE
                         ;
E9D5  F0 E8                      BEQ     BADCOM      ;BRANCH IF BAD
                         ;
                         ;
                         ;
                         ; WAIT FOR COMPLETE SIGNAL FROM PERIPHERAL
                         ;
E9D7  20 75 EC           WATCOM: JSR     STTMOT      ;SET DDEVICE TIME OUT VALUES IN Y,X
                         ;
E9DA  A9 00                      LDA     #$00
E9DC  8D 3F 02                   STA     ERRFLG      ;CLEAR ERROR FLAG
                         ;
E9DF  20 9B EC                   JSR     WAITER      ;SET UP TIMER AND WAIT
E9E2  F0 12                      BEQ     DERR        ;BRANCH IF TIME OUT
                         ;
                         ;
                         ; DEVICE DID NOT TIME OUT
                         ;
E9E4  2C 03 03                   BIT     DSTATS
E9E7  70 07                      BVS     MODATA      ;BRANCH IF MORE DATA FOLLOWS
                         ;
E9E9  AD 3F 02                   LDA     ERRFLG
E9EC  D0 18                      BNE     DERR1       ;BRANCH IF AN ERROR OCCURRED
E9EE  F0 1D                      BEQ     RETURN      ;OTHERWISE RETURN
                         ;
                         ;
                         ;
                         ;
                         ; RECEIVE A DATA FRAME FROM PERIPHERAL
                         ;
E9F0  20 6A EB           MODATA: JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                         ;
E9F3  20 E0 EA                   JSR     RECEIV      ;GO RECEIVE A DATA FRAME
                         ;
E9F6  AD 3F 02           DERR:   LDA     ERRFLG
```

```
E9F9  F0 05                      BEQ    NOTERR     ;BRANCH IF NO ERROR PRECEEDED DATA
                         ;
E9FB  AD 19 03                   LDA    TSTAT      ;GET TEMP STATUS
E9FE  85 30                      STA    STATUS     ;STORE IN REAL STATUS
                         ;
                         ;
EA00  A5 30              NOTERR: LDA    STATUS
EA02  C9 01                      CMP    #SUCCES
EA04  F0 07                      BEQ    RETURN     ;BRANCH IF COMPLETELY SUCCESSFUL
                         ;
EA06  C6 37              DERR1:  DEC    DRETRY
EA08  30 03                      BMI    RETURN     ;BRANCH IF OUT OF DEVICE RETRIES
                         ;
EA0A  4C 74 E9                   JMP    COMMND     ;OTHERWISE, ONE MORE TIME
                         ;
                         ;
                         ;
                         ;
EA0D  20 5F EC           RETURN: JSR    SENDDS     ;DISABLE POKEY INTERRUPTS
EA10  A9 00                      LDA    #0
EA12  85 42                      STA    CRITIC
EA14  A4 30                      LDY    STATUS     ;RETURN STATUS IN Y
EA16  8C 03 03                   STY    DSTATS     ;AND THE DCB STATUS WORD
EA19  60                         RTS    RETURN
                         ;
                         ;
                         ;
                         ;
                         ; WAIT SUBROUTINE
                         ;
                         ; WAITS FOR COMPLETE OR ACK
                         ; RETURNS Y=$FF IF SUCCESSFUL, Y=$00 IF NOT
                         ;
EA1A  A9 00              WAIT:   LDA    #$00
EA1C  8D 3F 02                   STA    ERRFLG     ;CLEAR ERROR FLAG
                         ;
EA1F  18                         CLC               ;LOAD BUFFER POINTER WITH ADDRESS
EA20  A9 3E                      LDA    #TEMPLO    ;OF TEMPORARY RAM CELL
EA22  85 32                      STA    BUFRLO
EA24  69 01                      ADC    #1
EA26  85 34                      STA    BFENLO     ;ALSO SET BUFFER END +1 ADDRESS
EA28  A9 02                      LDA    #TEMPHI
EA2A  85 33                      STA    BUFRHI
EA2C  85 35                      STA    BFENHI     ;DONE LOADING POINTER
                         ;
EA2E  A9 FF                      LDA    #$FF
EA30  85 3C                      STA    NOCKSM     ;SET NO CHECKSUM FOLLOWS DATA FLAG
                         ;
EA32  20 E0 EA                   JSR    RECEIV     ;GO RECEIVE A BYTE
                         ;
EA35  A0 FF                      LDY    #$FF       ;ASSUME SUCCESS
EA37  A5 30                      LDA    STATUS
EA39  C9 01                      CMP    #SUCCES
EA3B  D0 19                      BNE    NWOK       ;BRANCH IF IT DID NOT WORK OK
```

```
                        ;
                        ;
                        ;
                        ;
EA3D  AD 3E 02          WOK:    LDA     TEMP        ;MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
EA40  C9 41                     CMP     #ACK        ;WAS ACTUALLY AN ACK OR COMPLETE
EA42  F0 21                     BEQ     GOOD
EA44  C9 43                     CMP     #COMPLT
EA46  F0 1D                     BEQ     GOOD
                        ;
EA48  C9 45                     CMP     #ERROR
EA4A  D0 06                     BNE     NOTDER      ;BRANCH IF DEVICE DID NOT SEND BACK
                        ; A DEVICE ERROR CODE
EA4C  A9 90                     LDA     #DERROR
EA4E  85 30                     STA     STATUS      ;SET DEVICE ERROR STATUS
EA50  D0 04                     BNE     NWOK
                        ;
EA52  A9 8B             NOTDER: LDA     #DNACK      ;OTHERWISE SET NACK STATUS
EA54  85 30                     STA     STATUS
                        ;
EA56  A5 30             NWOK:   LDA     STATUS
EA58  C9 8A                     CMP     #TIMOUT
EA5A  F0 07                     BEQ     BAD         ;BRANCH IF TIME OUT
                        ;
EA5C  A9 FF                     LDA     #$FF
EA5E  8D 3F 02                  STA     ERRFLG      ;SET SOME ERROR FLAG
EA61  D0 02                     BNE     GOOD        ;RETURN WITH OUT SETTING Y = 0
                        ;
EA63  A0 00             BAD:    LDY     #0
                        ;
EA65  A5 30             GOOD:   LDA     STATUS
EA67  8D 19 03                  STA     TSTAT
EA6A  60                        RTS                 ;RETURN
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; SEND SUBROUTINE
                        ;
                        ; SENDS A BUFFER OF BYTES OUT OVER THE SERIAL BUS
                        ;
                        ;
EA6B  A9 01             SEND:   LDA     #SUCCES     ;ASSUME SUCCESS
EA6D  85 30                     STA     STATUS
                        ;
EA6F  20 F2 EB                  JSR     SENDEN      ;ENABLE SENDING
                        ;
EA72  A0 00                     LDY     #0
EA74  84 31                     STY     CHKSUM      ;CLEAR CHECK SUM
EA76  84 3B                     STY     CHKSNT      ;CHECKSUM SENT FLAG
EA78  84 3A                     STY     XMTDON      ;TRANSMISSION DONE FLAG
                        ;
                        ;
```

```
EA7A  B1 32                    LDA    (BUFRLO),Y  ;PUT FIRST BYTE FROM BUFFER
EA7C  8D 0D D2                 STA    SEROUT      ;INTO THE SERIAL OUTPUT REGISTER
                      ;
                      ;
EA7F  85 31                    STA    CHKSUM      ;PUT IT IN CHECKSUM
                      ;
EA81  A5 11           NOTDON:  LDA    BRKKEY
EA83  D0 03                    BNE    NTBRKO
EA85  4C A0 ED                 JMP    BROKE       ;JUMP IF BREAK KEY PRESSED
                      ;
EA88  A5 3A           NTBRKO:  LDA    XMTDON      ;LOOP UNTIL TRANSMISSION IS DONE
EA8A  F0 F5                    BEQ    NOTDON
                      ;
EA8C  20 5F EC                 JSR    SENDDS      ;DISABLE SENDING
                      ;
EA8F  60                       RTS                ;RETURN
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ; OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
                      ;
EA90  98              ISRODN:  TYA
EA91  48                       PHA                ;SAVE Y REG ON STACK
                      ;
EA92  E6 32                    INC    BUFRLO      ;INCREMENT BUFFER POINTER
EA94  D0 02                    BNE    NOWRPO
EA96  E6 33                    INC    BUFRHI
                      ;
EA98  A5 32           NOWRPO:  LDA    BUFRLO      ;CHECK IF PAST END OF BUFFER
EA9A  C5 34                    CMP    BFENLO
EA9C  A5 33                    LDA    BUFRHI      ;HIGH PART
EA9E  E5 35                    SBC    BFENHI
EAA0  90 1C                    BCC    NOTEND      ;BRANCH IF NOT PAST END OF BUFFER
                      ;
EAA2  A5 3B                    LDA    CHKSNT
EAA4  D0 0B                    BNE    RELONE      ;BRANCH IF CHECKSUM ALREADY SENT
                      ;
EAA6  A5 31                    LDA    CHKSUM
EAA8  8D 0D D2                 STA    SEROUT      ;SEND CHECK SUM
EAAB  A9 FF                    LDA    #$FF
EAAD  85 3B                    STA    CHKSNT      ;SET CHECKSUM SENT FLAG
EAAF  D0 09                    BNE    CHKDON
                      ;
EAB1  A5 10           RELONE:  LDA    POKMSK      ;ENABLE TRANSMIT DONE INTERRUPT
EAB3  09 08                    ORA    #$08
EAB5  85 10                    STA    POKMSK
EAB7  8D 0E D2                 STA    IRQEN
                      ;
EABA  68              CHKDON:  PLA
EABB  A8                       TAY                ;RESTORE Y REG
EABC  68                       PLA                ;RETURN FROM INTERRUPT
```

```
EABD  40                      RTI
                      ;
                      ;
EABE  A0 00           NOTEND: LDY     #0
EAC0  B1 32                   LDA     (BUFRLO),Y   ;PUT NEXT BYTE FROM BUFFER
EAC2  8D 0D D2                STA     SEROUT       ;INTO THE SERIAL OUTPUT REGISTER
                      ;
EAC5  18                      CLC                  ;ADD IT TO CHECKSUM
EAC6  65 31                   ADC     CHKSUM
EAC8  69 00                   ADC     #0
EACA  85 31                   STA     CHKSUM
                      ;
EACC  4C BA EA                JMP     CHKDON       ;GO RETURN
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
                      ;
EACF  A5 3B           ISRTD:  LDA     CHKSNT
EAD1  F0 0B                   BEQ     FOOEY        ;BRANCH IF CHECKSUM NOT YET SENT
                      ; ,
EAD3  85 3A                   STA     XMTDON       ;OTHERWISE SET TRANSMISSION DONE FLAG
                      ;
EAD5  A5 10                   LDA     POKMSK       ;DISABLE TRANSMIT DONE INTERRUPT
EAD7  29 F7                   AND     #$F7
EAD9  85 10                   STA     POKMSK
EADB  8D 0E D2                STA     IRQEN
                      ;
EADE  68              FOOEY:  PLA                  ;RETURN FROM INTERRUPT
EADF  40                      RTI
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ; RECEIVE SUBROUTINE
                      ;
EAE0  A9 00           RECEIV: LDA     #0
                      ;
EAE2  AC 0F 03                LDY     CASFLG
EAE5  D0 02                   BNE     NOCLR        ;BRANCH IF CASSETTE
                      ;
EAE7  85 31                   STA     CHKSUM       ;CLEAR CHKSUM
EAE9  85 38           NOCLR:  STA     BUFRFL       ;BUFFER FULL FLAG
EAEB  85 39                   STA     RECVDN       ;RECEIVE DONE FLAG
                      ;
                      ;
                      ;
```

```
EAED  A9 01              LDA    #SUCCES
EAEF  85 30              STA    STATUS      ;SET GOOD STATUS FOR DEFAULT CASE.
EAF1  20 1B EC           JSR    RECVEN      ;DO RECEIVE ENABLE
EAF4  A9 3C              LDA    #NCOMHI     ;COMMAND FRAME HI COMMAND
EAF6  8D 03 D3           STA    PBCTL       ;STORE IN PIA
EAF9  A5 11     CHKTIM:  LDA    BRKKEY
EAFB  D0 03              BNE    NTBRK1
EAFD  4C A0 ED           JMP    BROKE       ;JUMP IF BREAK KEY PRESSED
                ;
EB00  AD 17 03  NTBRK1:  LDA    TIMFLG      ;NO,
EB03  F0 05              BEQ    TOUT        ;IF TIMEOUT, GO SET ERROR STATUS
EB05  A5 39              LDA    RECVDN
EB07  F0 F0              BEQ    CHKTIM      ;DONE ?
EB09  60        GOBACK:  RTS
EB0A  A9 8A     TOUT:    LDA    #TIMOUT     ;YES,
EB0C  85 30              STA    STATUS      ;SET TIMEOUT STATUS
                ;
                ;
                ;
                ;
                ;
                ;
EB0E  60        RRETRN:  RTS                ;RETURN
                ;
                ;
                ;
                ;
                ;
                ;
                ;
                ; SERIAL INPUT READY INTERRUPT SERVICE ROUTINE
                ;
EB0F  98        ISRSIR:  TYA
EB10  48                 PHA                ;SAVE Y REG ON STACK
                ;
                ;
                ;
EB11  AD 0F D2           LDA    SKSTAT
EB14  8D 0A D2           STA    SKRES       ;RESET STATUS REGISTER
                ; ********   THIS MAY NOT BE THE PLACE TO DO IT   *********
                ;
EB17  30 04              BMI    NTFRAM      ;BRANCH IF NO FRAMING ERROR
                ;
EB19  A0 8C              LDY    #FRMERR
EB1B  84 30              STY    STATUS      ;SET FRAME ERRORR STATUS
                ;
EB1D  29 20     NTFRAM:  AND    #$20
EB1F  D0 04              BNE    NTOVRN      ;BRANCH IF NO OVERRUN ERROR
                ;
EB21  A0 8E              LDY    #OVRRUN
EB23  84 30              STY    STATUS      ;SET OVERRUN ERROR STATUS
                ;
EB25  A5 38     NTOVRN:  LDA    BUFRFL
EB27  F0 13              BEQ    NOTYET      ;BRANCH IF BUFFER WAS NOT YET FILLED
```

```
                              ;
      EB29  AD 0D D2                  LDA    SERIN        ;THIS INPUT BYTE IS THE CHECKSUM
      EB2C  C5 31                     CMP    CHKSUM
      EB2E  F0 04                     BEQ    SRETRN       ;BRANCH IF CHECKSUMS MATCH
                              ;
      EB30  A0 8F                     LDY    #CHKERR
      EB32  84 30                     STY    STATUS       ;SET CHECKSUM ERROR STATUS
                              ;
      EB34  A9 FF             SRETRN: LDA    #$FF         ;SET RECEIVE DONE FLAG
      EB36  85 39                     STA    RECVDN
                              ;
      EB38  68                SUSUAL: PLA
      EB39  A8                        TAY                 ;RESTORE Y REG
      EB3A  68                        PLA                 ;RETURN FROM INTERRUPT
      EB3B  40                        RTI
                              ;
                              ;
                              ;
      EB3C  AD 0D D2          NOTYET: LDA    SERIN
      EB3F  A0 00                     LDY    #0
      EB41  91 32                     STA    (BUFRLO),Y   ;STORE INPUT REGISTER INTO BUFFER
                              ;
      EB43  18                        CLC                 ;ADD IT TO CHECKSUM
      EB44  65 31                     ADC    CHKSUM
      EB46  69 00                     ADC    #0
      EB48  85 31                     STA    CHKSUM
                              ;
      EB4A  E6 32                     INC    BUFRLO       ;INCREMENT BUFFER POINTER
      EB4C  D0 02                     BNE    NTWRP1
      EB4E  E6 33                     INC    BUFRHI
                              ;
      EB50  A5 32             NTWRP1: LDA    BUFRLO
      EB52  C5 34                     CMP    BFENLO
      EB54  A5 33                     LDA    BUFRHI
      EB56  E5 35                     SBC    BFENHI
      EB58  90 DE                     BCC    SUSUAL       ;BRANCH IF NEW BUFFER ADDRESS IS IN BUFFER L
                              ;
      EB5A  A5 3C                     LDA    NOCKSM
      EB5C  F0 06                     BEQ    GOON         ;BRANCH IF A CHECKSUM WILL FOLLOW DATA
                              ;
      EB5E  A9 00                     LDA    #0
      EB60  85 3C                     STA    NOCKSM       ;CLEAR NO CHECKSUM FLAG
                              ;
      EB62  F0 D0                     BEQ    SRETRN       ;GO RETURN AND SET RECEIVE DONE FLAG
                              ;
                              ;
      EB64  A9 FF             GOON:   LDA    #$FF
      EB66  85 38                     STA    BUFRFL       ;SET BUFFER FULL FLAG
                              ;
      EB68  D0 CE                     BNE    SUSUAL       ;GO RETURN
                              ;
                              ;
                              ;
                              ;
```

```
                         ;
                         ;
                         ;
                         ;
                         ; LOAD BUFFER POINTER SUBROUTINE
                         ;
                         ; LOAD BUFFER POINTER WITH DCB BUFFER INFORMATION
                         ;
EB6A  18                 LDPNTR: CLC
EB6B  AD 04 03                   LDA     DBUFLO
EB6E  85 32                      STA     BUFRLO
EB70  6D 08 03                   ADC     DBYTLO
EB73  85 34                      STA     BFENLO      ;ALSO SET BUFFER END + 1 ADDRESS
                         ;
EB75  AD 05 03                   LDA     DBUFHI
EB78  85 33                      STA     BUFRHI
EB7A  6D 09 03                   ADC     DBYTHI
EB7D  85 35                      STA     BFENHI
                         ;
EB7F  60                         RTS                 ;RETURN
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ; CASSETTE HANDLING CODE
                         ;
EB80  AD 03 03           CASENT: LDA     DSTATS
EB83  10 2E                      BPL     CASRED      ;BRANCH IF INPUT FROM CASSETTE
                         ;
                         ; WRITE A RECORD
                         ;
EB85  A9 CC                      LDA     #B600LO     ;SET BAUD RATE TO 600
EB87  8D 04 D2                   STA     AUDF3
EB8A  A9 05                      LDA     #B600HI
EB8C  8D 06 D2                   STA     AUDF4
                         ;
EB8F  20 F2 EB                   JSR     SENDEN      ;TURN ON POKEY MARK TONE
                         ;
EB92  A0 0F                      LDY     #WSIRG      ;LOAD SHORT WRITE INTER RECORD GAP TIME
EB94  AD 0B 03                   LDA     DAUX2
EB97  30 02                      BMI     SRTIRO      ;BRANCH IF SHORT GAP IS DESIRED
                         ;
EB99  A0 B4                      LDY     #WIRGLO     ;SET WRITE IRG TIME
EB9B  A2 00              SRTIRO: LDX     #WIRGHI
EB9D  20 B9 ED                   JSR     SETVBX
                         ;
EBA0  A9 34                      LDA     #MOTRGO
EBA2  8D 02 D3                   STA     PACTL       ;TURN ON MOTOR
                         ;
EBA5  AD 17 03           TIMIT:  LDA     TIMFLG      ;LOOP UNTIL DONE
```

```
EBA8  D0 FB                       BNE     TIMIT
                          ;
EBAA  20 6A EB                    JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                          ;
EBAD  20 6B EA                    JSR     SEND        ;SEND A BUFFER
                          ;
EBB0  4C DF EB                    JMP     CRETRN      ;GO RETURN
                          ;
                          ;
                          ;
                          ; RECEIVE A RECORD
                          ;
EBB3  A9 FF               CASRED: LDA     #$FF
EBB5  8D 0F 03                    STA     CASFLG      ;SET SET CASSETTE FLAG
                          ;
EBB8  A0 0A                       LDY     #RSIRG      ;LOAD SHORT READ INTER RECORD GAP TIME
EBBA  AD 0B 03                    LDA     DAUX2
EBBD  30 02                       BMI     SRTIR1      ;BRANCH IF SHORT GAP IS DESIRED
                          ;
EBBF  A0 78                       LDY     #RIRGLO     ;SET TIME OUT FOR READ IRG
EBC1  A2 00               SRTIR1: LDX     #RIRGHI
EBC3  20 B9 ED                    JSR     SETVBX
                          ;
EBC6  A9 34                       LDA     #MOTRGO
EBC8  8D 02 D3                    STA     PACTL       ;TURN ON MOTOR
                          ;
EBCB  AD 17 03            TIMIT1: LDA     TIMFLG      ;LOOP UNTIL DONE
EBCE  D0 FB                       BNE     TIMIT1
                          ;
EBD0  20 6A EB                    JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
                          ;
EBD3  20 75 EC                    JSR     STTMOT      ;SET DEVICE TIME OUT IN Y,X
EBD6  20 B9 ED                    JSR     SETVBX
                          ;
EBD9  20 10 ED                    JSR     BEGIN       ;SET INITIAL BAUD RATE
                          ;
EBDC  20 E0 EA                    JSR     RECEIV      ;GO RECEIVE A BLOCK
                          ;
EBDF  AD 0B 03            CRETRN: LDA     DAUX2
EBE2  30 05                       BMI     SRTIR2      ;BRANCH IF DOING SHORT INTER RECORD GAPS
                          ; DON'T TURN OFF CASSETTE MOTOR
EBE4  A9 3C                       LDA     #MOTRST
EBE6  8D 02 D3                    STA     PACTL       ;TURN OFF MOTOR
                          ;
EBE9  4C 0D EA            SRTIR2: JMP     RETURN      ;GO RETURN
                          ;
                          ;
                          ;
                          ;
                          ;
EBEC  A9 00               JTIMER: LDA     #$00
00EB                      JTADRH  =       JTIMER/256  ;HI BYTE OF JUMP TIMER ROUTINE ADDR
00EC                      JTADRL  =       (-256)*JTADRH+JTIMER
EBEE  8D 17 03                    STA     TIMFLG      ;SET TIME OUT FLAG
```

```
EBF1  60                         RTS
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;;
                         ; SEND ENABLE SUBROUTINE
                         ;
EBF2  A9 07              SENDEN: LDA     #$07         ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
EBF4  2D 32 02                   AND     SSKCTL
EBF7  09 20                      ORA     #$20         ;SET TRANSMIT MODE
                         ;
EBF9  AC 00 03                   LDY     DDEVIC
EBFC  C0 60                      CPY     #CASET
EBFE  D0 0C                      BNE     NOTCAS       ;BRANCH IF NOT CASSETTE
                         ;
EC00  09 08                      ORA     #$08         ;SET THE FSK OUTPUT BIT
                         ;
EC02  A0 07                      LDY     #LOTONE      ;SET FSK TONE FREQUENCIES
EC04  8C 02 D2                   STY     AUDF2
EC07  A0 05                      LDY     #HITONE
EC09  8C 00 D2                   STY     AUDF1
                         ;
EC0C  8D 32 02           NOTCAS: STA     SSKCTL       ;STORE NEW VALUE TO SYSTEM MASK
EC0F  8D 0F D2                   STA     SKCTL        ;STORE TO ACTUAL REGISTER
                         ;
EC12  A9 C7                      LDA     #$C7         ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
EC14  25 10                      AND     POKMSK
EC16  09 10                      ORA     #$10         ;ENABLE OUTPUT DATA NEEDED INTERRUPT
                         ;
                         ;
EC18  4C 31 EC                   JMP     CONTIN       ;GO CONTINUE IN RECEIVE ENABLE SUBROUTINE
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ;
                         ; RECEIVE ENABLE SUBROUTINE
                         ;
EC1B  A9 07              RECVEN: LDA     #$07         ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
EC1D  2D 32 02                   AND     SSKCTL
EC20  09 10                      ORA     #$10         ;SET RECEIVE MODE ASYNCH.
EC22  8D 32 02                   STA     SSKCTL       ;STORE NEW VALUE TO SYSTEM MASK
EC25  8D 0F D2                   STA     SKCTL        ;STORE TO ACTUAL REGISTER
                         ;
EC28  8D 0A D2                   STA     SKRES        ;RESET SERIAL PORT/KEYBOARD STATUS REGISTER
                         ;
EC2B  A9 C7                      LDA     #$C7         ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
```

```
EC2D  25 10                AND   POKMSK
EC2F  09 20                ORA   #$20       ;ENABLE RECEIVE INTERRUPT
EC31  85 10       CONTIN:  STA   POKMSK     ;STORE NEW VALUE TO SYSTEM MASK
EC33  8D 0E D2             STA   IRQEN      ;STORE TO ACTUAL REGISTER
                  ;
                  ;
EC36  A9 28                LDA   #$28       ;CLOCK CH.3 WITH 1.79 MHZ
EC38  8D 08 D2             STA   AUDCTL     ;CLOCK CH.4 WITH CH. 3
                  ;
EC3B  A2 06                LDX   #6         ;SET PURE TONES, NO VOLUME
EC3D  A9 A8                LDA   #$A8
EC3F  A4 41                LDY   SOUNDR     ;TEST QUIET I/O FLAG
EC41  D0 02                BNE   NOISE1     ;NE IS NORMAL (NOISY)
EC43  A9 A0                LDA   #$A0
EC45  9D 01 D2    NOISE1:  STA   AUDC1,X
EC48  CA                   DEX
EC49  CA                   DEX
EC4A  10 F9                BPL   NOISE1
                  ;
EC4C  A9 A0                LDA   #$A0
EC4E  8D 05 D2             STA   AUDC3      ;TURN OFF SOUND ON CHANNEL 3
EC51  AC 00 03             LDY   DDEVIC
EC54  C0 60                CPY   #CASET
EC56  F0 06                BEQ   CAS31      ;BRANCH IF CASSETTE IS DESIRED
EC58  8D 01 D2             STA   AUDC1      ;OTHERWISE TURN OFF CHANNELS 1 AND 2
EC5B  8D 03 D2             STA   AUDC2
                  ;
                  ;
EC5E  60          CAS31:   RTS              ;RETURN
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ; DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
                  ;
EC5F  EA          SENDDS:  NOP
EC60  A9 C7       RECVDS:  LDA   #$C7       ;MASK OFF SERIAL BUS INTERRUPTS
EC62  25 10                AND   POKMSK
EC64  85 10                STA   POKMSK     ;STORE NEW VALUE TO SYSTEM MASK
EC66  8D 0E D2             STA   IRQEN      ;STORE TO ACTUAL REGISTER
                  ;
EC69  A2 06                LDX   #6
EC6B  A9 00                LDA   #0
EC6D  9D 01 D2    ZERIT:   STA   AUDC1,X
EC70  CA                   DEX
EC71  CA                   DEX
EC72  10 F9                BPL   ZERIT      ;TURN OFF AUDIO VOLUME
                  ;
```

```
EC74  60                RTS                 ;RETURN
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ; SET DDEVICE TIME OUT VALUES IN Y,X SUBROUTINE
                  ;
EC75  AD 06 03    STTMOT: LDA    DTIMLO      ;GET DEVICE TIME OUT IN 1 SECOND INCR
EC78  6A                ROR    A           ;PUT 6 HI BITS IN X, LO 2 BITS IN Y
EC79  6A                ROR    A
EC7A  A8                TAY                ;TEMP SAVE
EC7B  29 3F             AND    #$3F        ;MASK OFF 2 HI BITS
EC7D  AA                TAX                ;THIS IS HI BYTE OF TIME OUT
                  ;
EC7E  98                TYA                ;RESTORE
EC7F  6A                ROR    A
EC80  29 C0             AND    #$C0        ;MASK OFF ALL BUT 2 HI BITS
EC82  A8                TAY                ;THIS IS LO BYTE OF TIME OUT
                  ;
EC83  60                RTS
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
EC84  0F EB       INTTBL: .WORD  ISRSIR      ;SERIAL INPUT READY
EC86  90 EA             .WORD  ISRODN      ;OUTPUT DATA NEEDED
EC88  CF EA             .WORD  ISRTD       ;TRANSMISSION DONE
                  ;
00EB              SIRHI   =      ISRSIR/256  ;SERIAL INPUT READY ISR ADDRESS
000F              SIRLO   =      (-256)*SIRHI+ISRSIR
00EA              ODNHI   =      ISRODN/256  ;OUTPUT DATA NEEDED ISR ADDRESS
0090              ODNLO   =      (-256)*ODNHI+ISRODN
00EA              TDHI    =      ISRTD/256   ;TRANSMISSION DONE ISR ADDRESS
00CF              TDLO    =      (-256)*TDHI+ISRTD
                  ;
                  ;
                  ;
                  ;
                  ; SEND A DATA FRAME TO AN INTELLIGENT PERIPHERAL SUBROUTINE
                  ;
                  ;
EC8A  A2 01       SENDIN: LDX    #$01
```

```
ECBC  A0 FF       DELAY0: LDY    #$FF
EC8E  88          DELAY1: DEY
EC8F  D0 FD               BNE    DELAY1
EC91  CA                  DEX
EC92  D0 F8               BNE    DELAY0
                  ;
EC94  20 6B EA            JSR    SEND         ;GO SEND THE DATA FRAME
                  ;
EC97  A0 02               LDY    #CTIMLO      ;SET ACK TIME OUT
EC99  A2 00               LDX    #CTIMHI
EC9B  20 B9 ED    WAITER: JSR    SETVBX
                  ;
EC9E  20 1A EA            JSR    WAIT         ;WAIT FOR ACK
                  ;
ECA1  98                  TYA                 ;IF Y=0, A TIME OUT OR NACK OCCURED
                  ;
ECA2  60                  RTS                 ;RETURN
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ; COMPUTE VALUE FOR POKEY FREQ REGS FOR THE BAUD RATE AS
                  ; MEASURED BY AN INTERVAL OF THE 'VCOUNT' TIMER.
                  ;
ECA3  8D 10 03    COMPUT: STA    TIMER2
ECA6  8C 11 03            STY    TIMER2+1     ;SAVE FINAL TIMER VALUE
ECA9  20 04 ED            JSR    ADJUST       ;ADJUST VCOUNT VALUE
ECAC  8D 10 03            STA    TIMER2       ;SAVE ADJUSTED VALUE
ECAF  AD 0C 03            LDA    TIMER1
ECB2  20 04 ED            JSR    ADJUST       ;ADJUST
ECB5  8D 0C 03            STA    TIMER1       ;SAVE ADJUSTED TIMER1 VALUE
ECB8  AD 10 03            LDA    TIMER2
ECBB  38                  SEC
ECBC  ED 0C 03            SBC    TIMER1
ECBF  8D 12 03            STA    TEMP1        ;FIND VCOUNT DIFFERENCE
ECC2  AD 11 03            LDA    TIMER2+1
ECC5  38                  SEC
ECC6  ED 0D 03            SBC    TIMER1+1
ECC9  A8                  TAY                 ;FIND VBLANK COUNT DIFFERENCE
                          .IF    PALFLG
                          LDA    #-$9C
                  HITIMR: CLC
                          ADC    #$9C
                          .ENDIF
                          .IF    PALFLG-1
ECCA  A9 7D               LDA    #-$83
ECCC  18          HITIMR: CLC
```

```
    2491  ECCD  69 83                         ADC     #$83          ;ACCUMULATE MULTIPLICATION
    2492                                      .ENDIF
    2493  ECCF  88                            DEY
    2494  ECD0  10 FA                         BPL     HITIMR        ;DONE?
    2495  ECD2  18                            CLC
    2496  ECD3  6D 12 03                      ADC     TEMP1         ;TOTAL VCOUNT DIFFERENCE
    2497  ECD6  A8                    FINDX:  TAY                   ;SAVE ACCUM
    2498  ECD7  4A                            LSR     A
    2499  ECD8  4A                            LSR     A
    2500  ECD9  4A                            LSR     A
    2501  ECDA  0A                            ASL     A
    2502  ECDB  38                            SEC
    2503  ECDC  E9 16                         SBC     #22           ;ADJUST TABLE INDEX
    2504  ECDE  AA                            TAX                   ;DIVIDE INTERVAL BY 4 TO GET TABLE INDEX
    2505  ECDF  98                            TYA                   ;RESTORE ACCUM
    2506  ECE0  29 07                         AND     #7
    2507  ECE2  A8                            TAY                   ;PULL OFF 3 LO BITS OF INTERVAL
    2508  ECE3  A9 F5                         LDA     #-11
    2509  ECE5  18                    DOINTP: CLC
    2510  ECE6  69 0B                         ADC     #11           ;ACCUMULATE INTERPOLATION CONSTANT
    2511  ECE8  88                            DEY
    2512  ECE9  10 FA                         BPL     DOINTP        ;INTERPOLATION CONSTANT COMPUTATION DONE?
    2513                              ;
    2514  ECEB  A0 00                 ENINTP: LDY     #0
    2515  ECED  8C 0E 03                      STY     ADDCOR        ;CLEAR ADDITION CORRECTION FLAG
    2516  ECF0  38                            SEC
    2517  ECF1  E9 07                         SBC     #7            ;ADJUST INTERPOLATION CONSTANT
    2518  ECF3  10 03                         BPL     PLUS
    2519  ECF5  CE 0E 03                      DEC     ADDCOR
    2520  ECF8  18                    PLUS:   CLC
    2521  ECF9  7D D0 ED                      ADC     POKTAB,X      ;ADD CONSTANT TO LO BYTE TABLE VALUE
    2522  ECFC  A8                            TAY                   ;LO BYTE POKEY FREQ VALUE
    2523  ECFD  AD 0E 03                      LDA     ADDCOR
    2524  ED00  7D D1 ED                      ADC     POKTAB+1,X    ;ADD CARRY TO HI BYTE TABLE VALUE
    2525                              ; HI BYTE POKEY FREQ VALUE
    2526  ED03  60                            RTS
    2527                              ;
    2528                              ;
    2529                              ;
    2530                              ;       ROUTINE TO ADJUST VCOUNT VALUE
    2531                              ;
    2532  ED04  C9 7C                 ADJUST: CMP     #$7C
    2533  ED06  30 04                         BMI     ADJ1          ;LARGER THAN '7C' ?
    2534  ED08  38                            SEC                   ;YES,
    2535  ED09  E9 7C                         SBC     #$7C
    2536  ED0B  60                            RTS
    2537  ED0C  18                    ADJ1:   CLC
    2538                                      .IF     PALFLG
    2539                                      ADC     #$20
    2540                                      .ENDIF
    2541                                      .IF     PALFLG-1
    2542  ED0D  69 07                         ADC     #$7
    2543                                      .ENDIF
    2544  ED0F  60                            RTS
```

```
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;
                          ;       INITIAL BAUD RATE MEASUREMENT -- USED TO SET THE
                          ;               BAUD RATE AT THE START OF A RECORD.
                          ;
                          ;               IT IS ASSUMED THAT THE FIRST TWO BYTES OF EVERY
                          ; RECORD ARE 'AA' HEX.
                          ;
ED10  A5 11               BEGIN:  LDA     BRKKEY
ED12  D0 03                       BNE     NTBRK2
ED14  4C A0 ED                    JMP     BROKE        ;JUMP IF BREAK KEY PRESSED
                          ;
ED17  78                  NTBRK2: SEI
                          ;
ED18  AD 17 03                    LDA     TIMFLG
ED1B  D0 02                       BNE     OKTIM1       ;BRANCH IF NOT TIMED OUT
ED1D  F0 25                       BEQ     TOUT1        ;BRANCH IF TIME OUT
                          ;
ED1F  AD 0F D2            OKTIM1: LDA     SKSTAT
ED22  29 10                       AND     #$10         ;READ SERIAL PORT
ED24  D0 EA                       BNE     BEGIN        ;START BIT?
ED26  8D 16 03                    STA     SAVIO        ;SAVE SER. DATA IN
ED29  AE 0B D4                    LDX     VCOUNT       ;READ VERTICAL LINE COUNTER
ED2C  A4 14                       LDY     RTCLOK+2     ;READ LO BYTE OF VBLANK CLOCK
ED2E  8E 0C 03                    STX     TIMER1
ED31  8C 0D 03                    STY     TIMER1+1     ;SAVE INITIAL TIMER VALUE
                          ;
ED34  A2 01                       LDX     #1           ;SET MODE FLAG
ED36  8E 15 03                    STX     TEMP3
ED39  A0 0A                       LDY     #10          ;SET BIT COUNTER FOR 10 BITS
ED3B  A5 11               COUNT:  LDA     BRKKEY
ED3D  F0 61                       BEQ     BROKE        ;BRANCH IF BREAK KEY PRESSED
                          ;
ED3F  AD 17 03                    LDA     TIMFLG
ED42  D0 04                       BNE     OKTIMR       ;BRANCH IF NOT TIMED OUT
ED44  58                  TOUT1:  CLI
ED45  4C 0A EB                    JMP     TOUT         ;BRANCH IF TIME OUT
                          ;
ED48  AD 0F D2            OKTIMR: LDA     SKSTAT
ED4B  29 10                       AND     #$10         ;READ SERIAL PORT
ED4D  CD 16 03                    CMP     SAVIO        ;DATA IN CHANGED YET?
ED50  F0 E9                       BEQ     COUNT
ED52  8D 16 03                    STA     SAVIO        ;YES,SAVE SER. DATA IN
ED55  88                          DEY                  ;DECR. BIT COUNTER
ED56  D0 E3                       BNE     COUNT        ;DONE?
                          ;
ED58  CE 15 03                    DEC     TEMP3        ;YES,
ED5B  30 12                       BMI     GOREAD       ;DONE WITH BOTH MODES?
ED5D  AD 0B D4                    LDA     VCOUNT
```

```
ED60  A4 14              LDY  RTCLOK+2     ;READ TIMER LO & HI BYTES
ED62  20 A3 EC           JSR  COMPUT       ;NO, COMPUTE BAUD RATE
ED65  8C EE 02           STY  CBAUDL
ED68  8D EF 02           STA  CBAUDH       ;SET BAUD RATE INTO RAM CELLS
ED6B  A0 09              LDY  #9           ;SET BIT COUNTER FOR 9 BITS
ED6D  D0 CC              BNE  COUNT
                ;
ED6F  AD EE 02  GOREAD:  LDA  CBAUDL
ED72  8D 04 D2           STA  AUDF3
ED75  AD EF 02           LDA  CBAUDH
ED78  8D 06 D2           STA  AUDF4        ;SET POKEY FREQ REGS FOR BAUD RATE
ED7B  A9 00              LDA  #0
ED7D  8D 0F D2           STA  SKSTAT
ED80  AD 32 02           LDA  SSKCTL
ED83  8D 0F D2           STA  SKSTAT       ;INIT. POKEY SERIAL PORT
ED86  A9 55              LDA  #$55
ED88  91 32              STA  (BUFRLO),Y   ;STORE '$55' AS FIRST RCV. BUFFER
ED8A  C8                 INY
ED8B  91 32              STA  (BUFRLO),Y
ED8D  A9 AA              LDA  #$AA
ED8F  85 31              STA  CHKSUM       ;STORE CHECKSUM FOR 2 BYTES OF '$AA'
ED91  18                 CLC
ED92  A5 32              LDA  BUFRLO
ED94  69 02              ADC  #2
ED96  85 32              STA  BUFRLO
ED98  A5 33              LDA  BUFRHI
ED9A  69 00              ADC  #0
ED9C  85 33              STA  BUFRHI       ;INCR. BUFFER POINTER BY 1
ED9E  58                 CLI
ED9F  60                 RTS
                ;
                ;
                ;
EDA0  20 5F EC  BROKE:   JSR  SENDDS       ;BREAK KEY WAS PRESSED, SO PREPARE
EDA3  A9 3C              LDA  #MOTRST      ;TO RETURN
EDA5  8D 02 D3           STA  PACTL        ;TURN OFF MOTOR
EDA8  8D 03 D3           STA  PBCTL        ;RAISE NOT COMMAND LINE
                ;
EDAB  A9 80              LDA  #BRKABT
EDAD  85 30              STA  STATUS       ;STORE BREAK ABORT STATUS CODE
                ;
EDAF  AE 18 03           LDX  STACKP
EDB2  9A                 TXS               ;RESTORE STACK POINTER
                ;
EDB3  C6 11              DEC  BRKKEY       ;SET BREAK KEY FLAG TO NONZERO
EDB5  58                 CLI               ;ALLOW IRQ'S
                ;
EDB6  4C 0D EA           JMP  RETURN       ;GO RETURN
                ;
                ;
                ;
                ;
                ;
EDB9  A9 EC     SETVBX:  LDA  #JTADRL      ;STORE TIME OUT ROUTINE ADDRESS
```

```
EDBB  8D 26 02                        STA     CDTMA1
EDBE  A9 EB                           LDA     #JTADRH
EDC0  8D 27 02                        STA     CDTMA1+1
                            ;
EDC3  A9 01                           LDA     #1             ;SET FOR TIMER 1
                            ;
EDC5  78                              SEI                    ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
EDC6  20 5C E4                        JSR     SETVBV         ;ANY VBLANKS THAT OCCUR
EDC9  A9 01                           LDA     #1             ;SET FOR TIMER 1
EDCB  8D 17 03                        STA     TIMFLG         ;SET FLAG TO NOT TIMED OUT
EDCE  58                              CLI
EDCF  60                              RTS
                            ;
                            ;
                            ;
                            ;
                            ;
                            ;
                            ;
                            ; 'VCOUNT' INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE
                            ;                 CONVERSION TABLE
                            ;
                            ;
                            ; THE VALUES STORED IN THE TABLE ARE 'AUDF+7'.
                            ;
                            ;       THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:
                            ;
                            ;               F OUT= F IN/(2*(AUDF+M)) , WHERE F IN=1.78979 MHZ. & M=7
                            ;
                            ;        FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE
                            ;        TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY
                            ;        AN INTERVAL OF THE 'VCOUNT' TIMER.
                            ;
                            ;               AUDF+7=(11.365167)*T OUT, WHERE T OUT=# OF COUNTS
                            ;               (127 USEC.RESOLUTION) OF 'VCOUNT' FOR 1
                            ;               CHARACTER TIME (10 BIT TIMES).
                            ;
                            ;
                            ;
                            ;
                            ;               AUDF+7          BAUD RATE        VCOUNT INTERVAL
                            ;               ------          -----------      ----------------
                            ;       .WORD   $27C            ;1407                   56
                            ;       .WORD   $2D7            ;1231                   64
                            ;       .WORD   $332            ;1094                   72
                            ;       .WORD   $38D            ;985                    80
EDD0  E8 03                 POKTAB: .WORD   $3E8            ;895                    88
EDD2  43 04                         .WORD   $443            ;820                    96
EDD4  9E 04                         .WORD   $49E            ;757                    104
EDD6  F9 04                         .WORD   $4F9            ;703                    112
EDD8  54 05                         .WORD   $554            ;656                    120
EDDA  AF 05                         .WORD   $5AF            ;615                    128
EDDC  0A 06                         .WORD   $60A            ;579                    136
EDDE  65 06                         .WORD   $665            ;547                    144
```

```
EDE0  C0 06                     .WORD   $6C0            ;518            152
EDE2  1A 07                     .WORD   $71A            ;492            160
EDE4  75 07                     .WORD   $775            ;469            168
EDE6  D0 07                     .WORD   $7D0            ;447            176
                        ;       .WORD   $82B            ;428            184
                        ;       .WORD   $886            ;410            192
                        ;       .WORD   $8E1            ;394            200
                        ;       .WORD   $93C            ;379            208
                        ;       .WORD   $997            ;365            216
                        ;       .WORD   $9F2            ;352            224
                        ;       .WORD   $A4D            ;339            232
                        ;       .WORD   $AA8            ;328            240
                        ;       .WORD   $B03            ;318            248
                        ;
                        ;
                        ;
                        ;
                        ;**********************************************************************
EDE8                    CRNTP3  =*
                                *=$14
0014  02                SIOSPR: .BYTE   DSKORG-CRNTP3 ;^GSIOL IS TOO LONG
```

```
                               .TITLE  'DISK   ***** DISKP.SRC ***** 3/9/79 *****   4:00:00 P.M.'
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
0002                   STATVH  =       DVSTAT/256
00EA                   STATVL  =       (-256)*STATVH+DVSTAT ;STATUS POINTER
                       ;
                       ;
                       ;
                       ;
                       ;       CONSTANT EQUATES
                       ;
0031                   DISKID  =       $31           ;SERIAL BUS DISK I.D.
0050                   PUTSEC  =       $50           ;DISK PUT SECTOR DCB COMMAND
                       ; READ  =       $52           ;DISK GET SECTOR DCB COMMAND
                       ; WRITE =       $57           ;DISK PUT SECTOR WITH READ CHECK DCB COMMAND
0053                   STATC   =       $53           ;DISK STATUS DCB COMMAND
0021                   FOMAT   =       $21           ;DISK FORMAT DCB COMMAND !!!!! *****
0000                   NODAT   =       0             ;SIO COMMAND FOR "NO DATA" OPERATION
0040                   GETDAT  =       $40           ;SIO COMMAND FOR "DATA FROM DEVICE"
0080                   PUTDAT  =       $80           ;SIO COMMAND FOR "DATA TO DEVICE"
                       ;
                       ;
                       ;       VECTORS
                       ;
                               *=$E450
                       ;
E450  4C EA ED                 JMP     DINIT         ;DISK INIT. VECTOR
E453  4C F0 ED                 JMP     DSKIF         ;DISK INTERFACE ENTRY POINT
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;       CONSTANTS
                       ;
                               *=DSKORG
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;*****************************************************************
                       ;       DISK INTERFACE ROUTINE STARTS HERE
                       ;*****************************************************************
```

```
                                ;
                                ;
                                ;
                                ;
                                ;       DISK INTERFACE INITIALIZATION ROUTINE
                                ;
EDEA  A9 A0                     DINIT:  LDA     #160
EDEC  8D 46 02                          STA     DSKTIM       ;SET INITIAL DISK TIMEOUT TO 160 SEC
EDEF  60                                RTS
                                ;
                                ;
                                ;
                                ;       DISK INTERFACE ENTRY POINT
                                ;
EDF0  A9 31                     DSKIF:  LDA     #DISKID
EDF2  8D 00 03                          STA     DDEVIC       ;SET SERIAL BUS I.D IN DCB
EDF5  AD 46 02                          LDA     DSKTIM
EDF8  AE 02 03                          LDX     DCOMND
EDFB  E0 21                             CPX     #FOMAT       ;IS COMMAND A FORMAT COMMAND?
EDFD  F0 02                             BEQ     PUTDTO
EDFF  A9 07                             LDA     #7           ;NO, SET TIMEOUT TO 7 SECS.
EE01  8D 06 03                  PUTDTO: STA     DTIMLO       ;PUT DISK TIMEOUT IN DCB
EE04  A2 40                             LDX     #GETDAT      ;SET "GET DATA" COMMAND FOR SIO
EE06  A0 80                             LDY     #$80         ;SET BYTE COUNT TO 128
EE08  AD 02 03                          LDA     DCOMND       ;READ COMMAND IN DCB
EE0B  C9 57                             CMP     #WRITE       ;IS COMMAND A "PUT SECTOR" COMMAND?
EE0D  D0 02                             BNE     CKSTC
EE0F  A2 80                             LDX     #PUTDAT      ;YES, SET "PUT DATA" COMMAND FOR SIO
EE11  C9 53                     CKSTC:  CMP     #STATC       ;IS COMMAND A STATUS COMMAND?
EE13  D0 0C                             BNE     PUTCNT
EE15  A9 EA                             LDA     #STATVL
EE17  8D 04 03                          STA     DBUFLO
EE1A  A9 02                             LDA     #STATVH
EE1C  8D 05 03                          STA     DBUFHI       ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
EE1F  A0 04                             LDY     #4           ;YES, SET BYTE COUNT TO 4
EE21  8E 03 03                  PUTCNT: STX     DSTATS       ;PUT STATUS COMMAND FOR SIO IN DCB
EE24  8C 08 03                          STY     DBYTLO
EE27  A9 00                             LDA     #0
EE29  8D 09 03                          STA     DBYTHI       ;PUT BYTE COUNT IN DCB
EE2C  20 59 E4                          JSR     SIOV         ;CALL SERIAL I/O.
EE2F  10 01                             BPL     GOODST       ;NO ERROR
EE31  60                                RTS                  ;NO, GO BACK
EE32  AD 02 03                  GOODST: LDA     DCOMND       ;READ THE COMMAND
EE35  C9 53                             CMP     #STATC       ;WAS IT A STATUS COMMAND?
EE37  D0 0A                             BNE     PUTBC
EE39  20 6D EE                          JSR     PUTADR       ;PUT BUFFER ADDR IN TEMP REG.
EE3C  A0 02                             LDY     #2
EE3E  B1 15                             LDA     (BUFADR),Y   ;READ DISK TIMEOUT VALUE BYTE OF STATUS
EE40  8D 46 02                          STA     DSKTIM       ;PUT IT IN DISK TIMEOUT REG.
EE43  AD 02 03                  PUTBC:  LDA     DCOMND
EE46  C9 21                             CMP     #FOMAT       ;WAS COMMAND A FORMAT COMMAND?
EE48  D0 1F                             BNE     ENDDIF
EE4A  20 6D EE                  FMTD:   JSR     PUTADR       ;YES, PUT BUFFER ADDR INTO TEMP REG
EE4D  A0 FE                             LDY     #$FE         ;SET BUFFER POINTER
```

```
EE4F  C8                TWICE:  INY
EE50  C8                        INY                     ;INCR BUFFER POINTER BY 2
EE51  B1 15             RDBAD:  LDA     (BUFADR),Y      ;READ LO BYTE BAD SECTOR DATA
EE53  C9 FF                     CMP     #$FF
EE55  D0 F8                     BNE     TWICE           ;IS IT "FF" ?
EE57  C8                        INY                     ;YES,
EE58  B1 15                     LDA     (BUFADR),Y      ;READ HI BYTE BAD SECTOR DATA
EE5A  C8                        INY
EE5B  C9 FF                     CMP     #$FF
EE5D  D0 F2                     BNE     RDBAD           ;IS IT "FF" ?
EE5F  88                        DEY
EE60  88                        DEY                     ;YES,
EE61  8C 08 03                  STY     DBYTLO          ;PUT BAD SECTOR BYTE COUNT INTO DCB
EE64  A9 00                     LDA     #0
EE66  8D 09 03                  STA     DBYTHI
EE69  AC 03 03          ENDDIF: LDY     DSTATS
EE6C  60                        RTS
                        ;
                        ;
                        ;
                        ;
                        ;       S U B R O U T I N E S
                        ;
                        ;
                        ;       PUT BUFFER ADDR FROM DCB INTO TEMP REG
                        ;
EE6D  AD 04 03          PUTADR: LDA     DBUFLO
EE70  85 15                     STA     BUFADR
EE72  AD 05 03                  LDA     DBUFHI
EE75  85 16                     STA     BUFADR+1        ;PUT BUFFER ADDR IN TEMP REG
EE77  60                        RTS
                        ;*************************************************************
                        ;
                        ;
                        ;       SPARE BYTE OR MODULE TOO LONG FLAG
                        ;
EE78                    CRNTP4  =       *
                        ;
                                *=$14
0014  00                DSKSPR: .BYTE   PRNORG-CRNTP4 ;^GDISKP TOO LONG
                        ;
```

```
                                     .PAGE
                                     .TITLE  'PRINTER  ***** PRINTP.SRC ***** 3/9/79 *****  4:00:00 P
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;
                             ;       DEVICE NUMBER OR CODE EQUATES
                             ;
      0002                   OPNOUT  =       $2          ;IOCB OPEN FOR OUTPUT COMMAND
      0028                   NBUFSZ  =       40          ;PRINT NORMAL BUFFER SIZE
      0014                   DBUFSZ  =       20          ;PRINT DOUBLE BUFFFER SIZE
      001D                   SBUFSZ  =       29          ;PRINT SIDEWAYS BUFFER SIZE
      0040                   PDEVN   =       $40         ;PRINTER DEVICE NUMBER
                             ; STATC =       $53         ;DCB STATUS COMMAND CODE
      0057                   WRITEC  =       $57         ;DCB WRITE COMMAND
      0020                   SPACE   =       $20         ;ASCII SPACE CHAR.
      004E                   N       =       $4E         ;ASCII "N" CHAR.
      0044                   D       =       $44         ;ASCII "D" CHAR.
      0053                   S       =       $53         ;ASCII "S" CHAR.
                             ;
                             ;
                             ;       PRINTER HANDLER ENTRY POINTS
                             ;
                             ;
                             ;
                             ;
                                     *=$E430
                             ;
      E430  9E EE                    .WORD   PHOPEN-1    ;PRINTER HANDLER OPEN
      E432  DB EE                    .WORD   PHCLOS-1    ;PH CLOSE
      E434  9D EE                    .WORD   BADST-1     ;PH READ
      E436  A6 EE                    .WORD   PHWRIT-1    ;PH WRITE
      E438  80 EE                    .WORD   PHSTAT-1    ;PH STATUS
      E43A  9D EE                    .WORD   BADST-1     ;PH SPECIAL
      E43C  4C 78 EE                 JMP     PHINIT      ;PH INIT.
      E43F  00                       .BYTE   0           ;ROM FILLER
                             ;
                             ;
                             ;
                             ;
                             ;
                                     *=PRNORG
                             ;
                             ;
                             ;
                             ;
                             ;       PRINTER HANDLER INITIALIZATION ROUTINE
```

```
                           ;
EE78  A9 1E                PHINIT: LDA     #30
EE7A  85 1C                        STA     PTIMOT      ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
EE7C  60                           RTS
                           ;
                           ;
                           ;       PRINTER HANDLER CONSTANTS
                           ;
EE7D  EA 02                PHSTLO: .WORD   DVSTAT      ;STATUS BUFFER POINTER
EE7F  C0 03                PHCHLO: .WORD   PRNBUF      ;CHAR. BUFFER POINTER
                           ;
                           ;
                           ;
                           ;       ************************
                           ;       PRINTER HANDLER ROUTINES
                           ;       ************************
                           ;
                           ;
                           ;
                           ;
                           ;
                           ;       PRINTER HANDLER STATUS ROUTINE
                           ;
EE81  A9 04                PHSTAT: LDA     #4
EE83  85 1E                        STA     PBUFSZ      ;SET BUFFER SIZE TO 4 BYTES
EE85  AE 7D EE                     LDX     PHSTLO
EE88  AC 7E EE                     LDY     PHSTLO+1    ;SET POINTER TO STATUS BUFFER
EE8B  A9 53                        LDA     #STATC      ;SET COMMAND TO "STATUS"
EE8D  8D 02 03                     STA     DCOMND      ;SET STATUS COMMAND
EE90  8D 0A 03                     STA     DAUX1
EE93  20 E6 EE                     JSR     SETDCB      ;GO SETUP DCB
EE96  20 59 E4                     JSR     SIOV        ;SEND STATUS COMMAND
EE99  30 03                        BMI     BADST       ;GO IF ERROR
EE9B  20 14 EF                     JSR     PHPUT       ;YES,PUT STATUS INTO GLOBAL BUFFER.
EE9E  60                   BADST:  RTS
                           ;
                           ;
                           ;
                           ;
                           ;       PRINTER HANDLER OPEN ROUTINE
                           ;
EE9F  20 81 EE             PHOPEN: JSR     PHSTAT      ;DO STATUS COMMAND TO SIO
EEA2  A9 00                        LDA     #0
EEA4  85 1D                        STA     PBPNT       ;CLEAR PRINT BUFFER POINTER
EEA6  60                           RTS
                           ;
                           ;
                           ;
                           ;
                           ;       PRINTER HANDLER WRITE ROUTINE
                           ;
EEA7  85 1F                PHWRIT: STA     PTEMP       ;SAVE ACCUM
EEA9  20 1A EF                     JSR     PRMODE      ;GO DETERMINE PRINT MODE
EEAC  A6 1D                        LDX     PBPNT
```

```
EEAE  A5 1F                 LDA    PTEMP       ;GET CHAR. SENT BY CIO
EEB0  9D C0 03              STA    PRNBUF,X    ;PUT CHAR. IN PRINT BUFFER
EEB3  E8                    INX                ;INCR. BUFFER POINTER
EEB4  E4 1E                 CPX    PBUFSZ      ;BUFFER POINTER=BUFFER SIZE?
EEB6  F0 13                 BEQ    BUFFUL
EEB8  86 1D                 STX    PBPNT       ;SAVE BUFFER POINTER
EEBA  C9 9B                 CMP    #CR         ;IS CHAR. = EOL ?
EEBC  F0 03                 BEQ    BLFILL      ;IF YES, GO DO BLANK FILL.
EEBE  A0 01                 LDY    #SUCCES     ;PUT GOOD STATUS IN Y REG FOR CIO.
EEC0  60                    RTS
EEC1  A9 20        BLFILL:  LDA    #SPACE      ;PUT BLANK IN ACCUM.
EEC3  9D C0 03     FILLBF:  STA    PRNBUF,X    ;STORE IT IN PRINT BUFFER.
EEC6  E8                    INX
EEC7  E4 1E                 CPX    PBUFSZ
EEC9  D0 F8                 BNE    FILLBF      ;BUFFER BLANK FILLED?
EECB  A9 00        BUFFUL:  LDA    #0
EECD  85 1D                 STA    PBPNT       ;CLEAR PRINT BUFFER POINTER
EECF  AE 7F EE              LDX    PHCHLO
EED2  AC 80 EE              LDY    PHCHLO+1    ;SET POINTER TO PRINT BUFFER
EED5  20 E6 EE              JSR    SETDCB      ;GO SETUP DCB
EED8  20 59 E4              JSR    SIOV        ;SEND PRINT COMMAND
EEDB  60                    RTS                ;YES.
                   ;
                   ;
                   ;
                   ;
                   ;        PRINTER HANDLER CLOSE ROUTINE
                   ;
EEDC  20 1A EF     PHCLOS:  JSR    PRMODE      ;GO DETERMINE PRINT MODE
EEDF  A6 1D                 LDX    PBPNT
EEE1  D0 DE                 BNE    BLFILL
EEE3  A0 01                 LDY    #SUCCES
EEE5  60                    RTS
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;        S U B R O U T I N E S
                   ;
                   ;
                   ;
                   ;
                   ;
                   ;        SET UP DCB TO CALL SIO
                   ;
EEE6  8E 04 03     SETDCB:  STX    DBUFLO
EEE9  8C 05 03              STY    DBUFHI      ;SET BUFFER POINTER
EEEC  A9 40                 LDA    #PDEVN
EEEE  8D 00 03              STA    DDEVIC      ;SET PRINTER BUS I.D. FOR DCB
EEF1  A9 01                 LDA    #1
```

```
EEF3  8D 01 03              STA     DUNIT       ;SET UNIT NUMBER TO 1
EEF6  A9 80                 LDA     #$80        ;DEVICE WILL EXPECT DATA
EEF8  AE 02 03              LDX     DCOMND
EEFB  E0 53                 CPX     #STATC      ;STATUS COMMAND?
EEFD  D0 02                 BNE     PSIOC
EEFF  A9 40                 LDA     #$40        ;EXPECT DATA FROM DEVICE
EF01  8D 03 03      PSIOC:  STA     DSTATS      ;SET SIO MODE COMMAND.
EF04  A5 1E                 LDA     PBUFSZ
EF06  8D 08 03              STA     DBYTLO      ;SET LO BYTE COUNT
EF09  A9 00                 LDA     #0
EF0B  8D 09 03              STA     DBYTHI      ;SET HI BYTE COUNT
EF0E  A5 1C                 LDA     PTIMOT
EF10  8D 06 03              STA     DTIMLO      ;SET DEVICE TIMEOUT COUNT
EF13  60                    RTS
                    ;
                    ;
                    ;
                    ;
                    ; GET DEVICE TIMEOUT FROM STATUS & SAVE IT
                    ;
EF14  AD EC 02      PHPUT:  LDA     DVSTAT+2
EF17  85 1C                 STA     PTIMOT      ;SAVE DEVICE TIMEOUT
EF19  60                    RTS
                    ;
                    ;
                    ;
                    ;
                    ; DETERMINE PRINT MODE & SETUP PRINT BUFFER SIZE, DCB PRINT
                    ; COMMAND, & DCB AUX1 FOR PRINT MODE
                    ;
EF1A  A0 57         PRMODE: LDY     #WRITEC     ;PUT WRITE COMMAND IN Y REG
EF1C  A5 2B                 LDA     ICAX2Z      ;READ PRINT MODE
EF1E  C9 4E         CMODE:  CMP     #N
EF20  D0 04                 BNE     CDUBL       ;PRINT NORMAL ?
EF22  A2 28                 LDX     #NBUFSZ     ;YES, SET NORMAL CHAR. BUFFER SIZE
EF24  D0 0E                 BNE     SETBSZ
EF26  C9 44         CDUBL:  CMP     #D
EF28  D0 04                 BNE     CSIDE       ;PRINT DOUBLE?
EF2A  A2 14                 LDX     #DBUFSZ     ;YES, SET DOUBLE CHAR. BUFFER SIZE
EF2C  D0 06                 BNE     SETBSZ
EF2E  C9 53         CSIDE:  CMP     #S          ;PRINT SIDEWAYS ?
EF30  D0 0B                 BNE     GOERR       ;IF NOT, GO TO ERROR ROUTINE
EF32  A2 1D                 LDX     #SBUFSZ     ;YES, SET SIDEWAYS BUFFER SIZE
EF34  86 1E         SETBSZ: STX     PBUFSZ      ;STORE PRINT BUFFER SIZE
EF36  8C 02 03              STY     DCOMND      ;STORE DCB COMMAND
EF39  8D 0A 03              STA     DAUX1       ;STORE DCB AUX1 PRINT MODE
EF3C  60                    RTS
EF3D  A9 4E         GOERR:  LDA     #N          ;SET DEFAULT PRINT MODE TO NORMAL
EF3F  D0 DD                 BNE     CMODE
                    ;************************************************************
                    ;
                    ;
                    ; SPARE BYTE OR MODULE TOO LONG FLAG
                    ;
```

```
EF41              CRNTP5  =        *
                  ;
                          *=$14
                  ;
0014  00          PRNSPR: .BYTE    CASORG-CRNTP5 ;^GPRINTP TOO LONG
                  ;
```

```
                                           .PAGE
                                           .TITLE  'CASSET HANDLER 3/12 (DK1:CASCV)'
 0003                           CBUFH   =          CASBUF/256
 00FD                           CBUFL   =          (-256)*CBUFH+CASBUF
 0040                           SRSTA   =          $40             ;SIO READ STATUS
 0080                           SWSTA   =          $80             ;SIO WRITE STATUS
                                ;MOTRGO =          $34
                                ;MOTRST =          $3C
                                ;
                                ;
 00FC                           DTA     =          $FC             ;DATA RECORD TYPE BYTE
 00FA                           DT1     =          $FA             ;LAST DATA RECORD
 00FE                           EOT     =          $FE             ;END OF TAPE
 00FB                           HDR     =          $FB             ;HEADER
 0002                           TONE1   =          2               ;CHANGE TO RECORD MODE TONE
 0001                           TONE2   =          1               ;PRESS PLAY TONE
                                ;
                                ;
                                ;
                                        *=CASETV
 E440  4B EF 2A F0                      .WORD   OPENC-1,CLOSEC-1,GBYTE-1,PBYTE-1,STATU-1,SPECIAL-1
 E444  D5 EF 0F F0
 E448  27 F0 4A EF
 E44C  4C 41 EF                         JMP     INIT
 E44F  00                               .BYTE   0                  ;ROM FILLER BYTE
                                ;
                                ;
                                ;
                                ; USED IN MONITP FOR CASSETTE BOOT
                                ;
                                        *=RBLOKV
 E47A  4C E9 EF                         JMP     RBLOK
                                ;
                                        *=CSOPIV
 E47D  4C 5D EF                         JMP     OPINP
                                ;
                                ;
                                        *=CASORG
                                ;
                                ;
                                ; INIT ROUTINE
                                ;
 EF41  A9 CC                    INIT:   LDA     #$CC
 EF43  8D EE 02                         STA     CBAUDL
 EF46  A9 05                            LDA     #$05
 EF48  8D EF 02                         STA     CBAUDH             ;SET CASSET BAUD RATE TO 600
                                SPECIAL:                           ;THATS ALL FOLKS
 EF4B  60                               RTS
```

```
                               .PAGE
                        ;
                        ; OPEN FUNCTION - WITH NO TIMING ADJUST
                        ;
EF4C  A5 2B             OPENC:  LDA     ICAX2Z      ;GET AX2
EF4E  85 3E                     STA     FTYPE       ;SAVE IT FOR FUTURE REFERENCE
EF50  A5 2A                     LDA     ICAX1Z
EF52  29 0C                     AND     #$0C        ;IN AND OUT BITS
EF54  C9 04                     CMP     #$04
EF56  F0 05                     BEQ     OPINP
EF58  C9 08                     CMP     #$08        ;SEE IF OPEN FOR OUTPUT
EF5A  F0 39                     BEQ     OPOUT
EF5C  60                        RTS                 ;IF ALREADY OPEN, RETURN LEAVING STATUS=$84
EF5D  A9 00             OPINP:  LDA     #0
EF5F  8D 89 02                  STA     WMODE       ;SET READ MODE
EF62  85 3F                     STA     FEOF        ;NO EOF YET
EF64  A9 01             SFH:    LDA     #TONE2      ;TONE FOR PRESS PLAY
EF66  20 58 F0                  JSR     BEEP        ;GO BEEP
EF69  30 24                     BMI     OPNRTN      ;IF ERROR DURING BEEP
EF6B  A9 34                     LDA     #MOTRGO
EF6D  8D 02 D3                  STA     PACTL       ;TURN MOTOR ON
                                .IF     PALFLG
                                LDY     #$E0
                                LDX     #1
                                .ENDIF
                                .IF     PALFLG-1
EF70  A0 40                     LDY     #$40        ;5-31-79 9 SEC READ LEADER
EF72  A2 02                     LDX     #2
                                .ENDIF
EF74  A9 03                     LDA     #3
EF76  8D 2A 02                  STA     CDTMF3
EF79  20 5C E4                  JSR     SETVBV      ;SET UP VBLANK TIMER
EF7C  AD 2A 02          WAITTM: LDA     CDTMF3
EF7F  D0 FB                     BNE     WAITTM      ;WAIT FOR MOTOR TO COME UP TO SPEED
EF81  A9 80                     LDA     #$80        ;NEXT BYTE=NO BYTES IN BUFFER
EF83  85 3D                     STA     BPTR
EF85  8D 8A 02                  STA     BLIM
EF88  4C D3 EF                  JMP     OPOK        ;OPEN OK
                        ;
                        ; OPEN FOR OUTPUT
                        ;
EF8B  A0 80             PBRK:   LDY     #BRKABT     ;BREAK KEY ABORT STATUS
EF8D  C6 11                     DEC     BRKKEY      ;RESET BREAK KEY
EF8F  A9 00             OPNRTN: LDA     #0          ;CLEAR WRITE MODE FLAG
EF91  8D 89 02                  STA     WMODE
EF94  60                        RTS                 ;AND EXIT.
                        ;
EF95  A9 80             OPOUT:  LDA     #$80
EF97  8D 89 02                  STA     WMODE       ;SET WRITE MODE
EF9A  A9 02                     LDA     #TONE1      ;TELL USER TO TURN ON RECORD MODE
EF9C  20 58 F0                  JSR     BEEP
EF9F  30 EE                     BMI     OPNRTN      ;IF ERROR DURING BEEP
EFA1  A9 CC                     LDA     #$CC        ;SET BAUD RATE
EFA3  8D 04 D2                  STA     AUDF3       ;WHICH SEEMS TO BE NESSECARY
```

```
EFA6  A9 05                 LDA    #$05        ;FOR SOME OBSCURE REASON
EFA8  8D 06 D2              STA    AUDF4
EFAB  A9 60                 LDA    #$60
EFAD  8D 00 03              STA    DDEVIC
EFB0  20 68 E4              JSR    SENDEV      ;TELL POKEY TO WRITE MARKS
EFB3  A9 34                 LDA    #MOTRGO     ;WRITE 5 SEC BLANK TAPE
EFB5  8D 02 D3              STA    PACTL
EFB8  A9 03                 LDA    #3
                            .IF    PALFLG
                            LDX    #$3
                            LDY    #$C0
                            .ENDIF
                            .IF    PALFLG-1
EFBA  A2 04                 LDX    #4          ;5/30/79 20 SEC LEADER
EFBC  A0 80                 LDY    #$80
                            .ENDIF
EFBE  20 5C E4              JSR    SETVBV
EFC1  A9 FF                 LDA    #$FF
EFC3  8D 2A 02              STA    CDTMF3
EFC6  A5 11        WDLR:    LDA    BRKKEY
EFC8  F0 C1                 BEQ    PBRK        ;IF BREAK DURING WRITE LEADER
EFCA  AD 2A 02              LDA    CDTMF3
EFCD  D0 F7                 BNE    WDLR
EFCF  A9 00                 LDA    #0          ;INIT BUFFER POINTER
EFD1  85 3D                 STA    BPTR
EFD3  A0 01        OPOK:    LDY    #SUCCES
EFD5  60                    RTS
```

```
                                     .PAGE
                              ;
                              ; GET BYTE
                              ;
EFD6  A5 3F                   GBYTE:  LDA     FEOF           ;IF AT EOF ALREADY
EFD8  30 33                           BMI     ISEOF          ;RETURN EOF STATUS
EFDA  A6 3D                           LDX     BPTR           ;BUFFER POINTER
EFDC  EC 8A 02                        CPX     BLIM           ;IF END OF BUFFER
EFDF  F0 08                           BEQ     RBLOK          ;READ ANOTHER BLOCK
EFE1  BD 00 04                        LDA     CASBUF+3,X     ;GET NEXT BYTE
EFE4  E6 3D                           INC     BPTR           ;BUMP POINTER
EFE6  A0 01                           LDY     #SUCCES        ;OK STATUS
EFE8  60                      GBX:    RTS
EFE9  A9 52                   RBLOK:  LDA     #'R            ;READ OPCODE
EFEB  20 95 F0                        JSR     SIOSB          ;SIO ON SYS BUF
EFEE  98                              TYA
EFEF  30 F7                           BMI     GBX            ;IF SIO ERRORS, RETURN
EFF1  A9 00                           LDA     #0
EFF3  85 3D                           STA     BPTR           ;RESET POINTER
EFF5  A2 80                           LDX     #$80           ;DEFAULT # BYTES
EFF7  AD FF 03                        LDA     CASBUF+2
EFFA  C9 FE                           CMP     #EOT
EFFC  F0 0D                           BEQ     ATEOF          ;IF HEADER, GO READ AGAIN
EFFE  C9 FA                           CMP     #DT1           ;IF LAST DATA REC
F000  D0 03                           BNE     NLR
F002  AE 7F 04                        LDX     CASBUF+130     ;LAST DATA RECORD, GET # BYTES
F005  8E 8A 02                NLR:    STX     BLIM
F008  4C D6 EF                        JMP     GBYTE          ;GET NEXT BYTE
F00B  C6 3F                   ATEOF:  DEC     FEOF           ;SET FEOF
F00D  A0 88                   ISEOF:  LDY     #EOFERR        ;ENDFILE STATUS
F00F  60                              RTS
```

```
                            .PAGE
                      ;
                      ; PUT BYTE TO BUFFER
                      ;
F010  A6 3D           PBYTE:  LDX     BPTR          ;BUFFER POINTER
F012  9D 00 04                STA     CASBUF+3,X    ;STORE CHAR AWAY
F015  E6 3D                   INC     BPTR          ;BUMP POINTER
F017  A0 01                   LDY     #SUCCES       ;OK STATUS
F019  E0 7F                   CPX     #127          ;IF BUFFER FULL
F01B  F0 01                   BEQ     *+3
F01D  60                      RTS
                      ; WRITE OUT THE BUFFER
F01E  A9 FC                   LDA     #DTA          ;RECORD TYPE = DATA
F020  20 D2 F0                JSR     WSIOSB        ;DO WRITE ON SYSTEM BUFFER
F023  A9 00                   LDA     #0
F025  85 3D                   STA     BPTR          ;RESET BUFFER POINTER
F027  60                      RTS                   ;EXIT.
```

```
                                 .PAGE
                         ;
                         ; STATUS - RETURN STATUS INFO THRU DVSTAT
                         ;
F028  A0 01              STATU:  LDY     #SUCCES
F02A  60                         RTS
```

```
                                  .PAGE
                          ;
                          ; CLOSE
                          ;
      F02B  AD 89 02      CLOSEC: LDA    WMODE         ;SEE IF WRITING
      F02E  30 08                 BMI    CLWRT         ;GO CLOSE FOR WRITE
                          ; CLOSE FOR READ - FLAG CLOSED
      F030  A0 01                 LDY    #SUCCES       ;SUCCESSFULL
      F032  A9 3C         FCAX:   LDA    #MOTRST       ;STOP THE MOTOR IN CASE WAS SHORT IRG MODE
      F034  8D 02 D3              STA    PACTL
      F037  60                    RTS
      F038  A6 3D         CLWRT:  LDX    BPTR          ;BUFFER POINTER
      F03A  F0 0A                 BEQ    WTLR          ;IF NO DATA BYTES IN BUFFER, NO DT1 REC
      F03C  8E 7F 04              STX    CASBUF+130    ;WRITE TO LAST RECORD
      F03F  A9 FA                 LDA    #DT1          ;REC TYPE
      F041  20 D2 F0              JSR    WSIOSB        ;WRITE OUT USER BUFFER
      F044  30 EC                 BMI    FCAX          ;GO IF ERROR
      F046  A2 7F         WTLR:   LDX    #127          ;ZERO BUFFER
      F048  A9 00                 LDA    #0
      F04A  9D 00 04      ZTBUF:  STA    CASBUF+3,X
      F04D  CA                    DEX
      F04E  10 FA                 BPL    ZTBUF
      F050  A9 FE                 LDA    #EOT          ;WRITE EOT RECORD
      F052  20 D2 F0              JSR    WSIOSB
      F055  4C 32 F0              JMP    FCAX          ;FLAG CLOSED AND EXIT
```

```
                                .PAGE
                        ;
                        ; SUBROUTINES
                        ;
                        ; BEEP - GENERATE TONE ON KEYBOARD SPEAKER
                        ; ON ENTRY A= FREQ
                        ;
F058  85 40             BEEP:   STA     FREQ
F05A  A5 14             BEEP1:  LDA     RTCLOK+2        ;CURRENT CLOCK
F05C  18                        CLC
                                .IF     PALFLG
                                ADC     #25
                                .ENDIF
                                .IF     PALFLG-1
F05D  69 1E                     ADC     #30             ;1 SEC TONE
                                .ENDIF
F05F  AA                        TAX
F060  A9 FF             WFL:    LDA     #$FF
F062  8D 1F D0                  STA     CONSOL          ;TURN ON SPEAKER
F065  A9 00                     LDA     #0
F067  A0 F0                     LDY     #$F0
F069  88                        DEY
F06A  D0 FD                     BNE     *-1
F06C  8D 1F D0                  STA     CONSOL          ;TURN OFF SPEAKER
F06F  A0 F0                     LDY     #$F0
F071  88                        DEY
F072  D0 FD                     BNE     *-1
F074  E4 14                     CPX     RTCLOK+2        ;SEE IF 1 SEC IS UP YET
F076  D0 E8                     BNE     WFL
F078  C6 40                     DEC     FREQ            ;COUNT BEEPS
F07A  F0 0B                     BEQ     WFAK            ;IF ALL DONE GO WAIT FOR KEY
F07C  8A                        TXA
F07D  18                        CLC
                                .IF     PALFLG
                                ADC     #8
                                .ENDIF
                                .IF     PALFLG-1
F07E  69 0A                     ADC     #10
                                .ENDIF
F080  AA                        TAX
F081  E4 14                     CPX     RTCLOK+2
F083  D0 FC                     BNE     *-2
F085  F0 D3                     BEQ     BEEP1           ;UNCOND GO BEEP AGIN
F087  20 8C F0          WFAK:   JSR     WFAK1           ;USE SIMULATED "JMP (KGETCH)"
F08A  98                        TYA
F08B  60                        RTS
F08C  AD 25 E4          WFAK1:  LDA     KEYBDV+5
F08F  48                        PHA
F090  AD 24 E4                  LDA     KEYBDV+4        ;SIMULATE "JMP (KGETCH)"
F093  48                        PHA
F094  60                        RTS
                        ;
                        ; SIOSB - CALL SIO ON SYSTEM BUFFER
                        ;
```

```
F095  8D 02 03     SIOSB:  STA     DCOMND       ;SAVE COMMAND
F098  A9 00                LDA     #0
F09A  8D 09 03             STA     DBYTHI       ;SET BUFFER LENGTH
F09D  A9 83                LDA     #131
F09F  8D 08 03             STA     DBYTLO
F0A2  A9 03                LDA     #CBUFH
F0A4  8D 05 03             STA     DBUFHI       ;SET BUFFER ADDRESS
F0A7  A9 FD                LDA     #CBUFL
F0A9  8D 04 03             STA     DBUFLO
F0AC  A9 60        CSIO:   LDA     #$60         ;CASSET PSEUDO DEVICE
F0AE  8D 00 03             STA     DDEVIC
F0B1  A9 00                LDA     #0
F0B3  8D 01 03             STA     DUNIT
F0B6  A9 23                LDA     #35          ;DEVICE TIMEOUT (5/30/79)
F0B8  8D 06 03             STA     DTIMLO
F0BB  AD 02 03             LDA     DCOMND       ;GET COMMAND BACK
F0BE  A0 40                LDY     #SRSTA       ;SIO READ STATUS COMMAND
F0C0  C9 52                CMP     #'R
F0C2  F0 02                BEQ     *+4
F0C4  A0 80                LDY     #SWSTA       ;SIO WRITE STATUS COMMAND
F0C6  8C 03 03             STY     DSTATS       ;SET STATUS FOR SIO
F0C9  A5 3E                LDA     FTYPE
F0CB  8D 0B 03             STA     DAUX2        ;INDICATE IF SHORT IRG MODE
F0CE  20 59 E4             JSR     SIOV         ;GO CALL SIO
F0D1  60                   RTS
                   ;
                   ; WSIOSB - WRITE SIO SYSTEM BUFFER
                   ;
F0D2  8D FF 03     WSIOSB: STA     CASBUF+2     ;STORE TYPE BYTE
F0D5  A9 55                LDA     #$55
F0D7  8D FD 03             STA     CASBUF+0
F0DA  8D FE 03             STA     CASBUF+1
F0DD  A9 57                LDA     #'W ;WRITE
F0DF  20 95 F0             JSR     SIOSB        ;CALL SIO ON SYSTEM BUFFER
F0E2  60                   RTS     AND          ;RETURN
F0E3               CRNTP6  =*
                           *=$14
0014  00           CASSPR: .BYTE   MONORG-CRNTP6 ;^GCASCV IS TOO LONG
```

```
                                   .TITLE  'MONITOR  ***** MONITP.SRC ***** 3/9/79 *****  4:00:00 P
                              ;
                              ;
                              ;
                              ;        CONSTANT EQUATES
                              ;
0009                          PUTTXT  =       $9           ;"PUT TEXT RECORD" CIO COMMAND CODE
0007                          GETCAR  =       $7           ;"GET CHARACTER" CIO COMMAND CODE
000B                          PUTCAR  =       $B           ;"PUT CHARACTER" CIO COMMAND CODE
0000                          INIMLL  =       $00          ;INITIAL MEM LO LOW BYTE
0007                          INIMLH  =       $07          ;INITIAL MEM LO HIGH BYTE
                              ; GOOD  =       $1           ;GOOD STATUS CODE
                              ; WRITE =       $57          ;WRITE COMMAND
                              ; READ  =       $52          ;READ COMMAND
                              ; STATC =       $53          ;STATUS COMMAND
0000                          SEX     =       $0           ;SCREEN EDITOR IOCB INDEX
007D                          CLS     =       $7D          ;CLEAR SCREEN CODE
0092                          CTRLC   =       $92          ;KEYBOARD CODE FOR 'CONTROL C'
0088                          EOF     =       $88          ;CASSETTE END OF FILE CODE
0000                          LIRG    =       $0           ;LONG IRG TYPE CODE
                              ;
0004                          BUFFH   =       (CASBUF+3)/256
0000                          BUFFL   =       (-256)*BUFFH+CASBUF+3 ;BUFFER POINTER
                              ;
                              ;
                              ;
                              ; THE FOLLOWING EQUATES ARE IN THE CARTRIDGE ADDRESS SPACE.
                              ;
                              ;
                              ; "B" CARTRIDGE ADDR'S ARE 8000-9FFF (36K CONFIG. ONLY)
                              ; "A" CART. ADDR'S ARE A000-BFFF (36K CONFIG. ONLY)
                              ;
                              ; "A" CART. ADDR'S ARE B000-BFFF (48K CONFIG. ONLY)
                              ;
                                       *=$BFFA
BFFA                          CARTCS: .RES    2            ;CARTRIDGE COLD START ADDRESS.
BFFC                          CART:   .RES    1            ;CARTRIDGE AVAILABLE FLAG BYTE.
BFFD                          CARTFG: .RES    1            ;CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
BFFE                          CARTAD: .RES    2            ;2-BYTE CARTRIDGE START VECTOR
                              ;
                              ;
                              ;        CARTRIDGE FLAG ACTION DEFINITIONS
                              ;
                              ;
                              ;        BIT             ACTION IF SET
                              ;
                              ;        7               SPECIAL -- DON'T POWER-UP, JUST RUN CARTRIDGE
                              ;        6-3             NONE
                              ;        2               RUN CARTRIDGE
                              ;        1               NONE
                              ;        0               BOOT DOS
                              ;
                              ;
```

```
                      ;       *****
                      ;       NOTE
                      ;       *****
                      ;
                      ;       1. IF BIT2 IS 0, GOTO BLACKBOARD MODE.
                      ;       2. IF BIT0 SET, THE DISK WILL BE BOOTED BEFORE ANY
                      ;         OTHER ACTION.
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;       POWER-UP VECTOR
                      ;
                      ;****************************
                      ;       *=$FFFC
                      ;
                      ; PVECT .WORD  PWRUP         POWER-UP VECTOR
                      ;*************************
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;       ENTRY POINT VECTOR
                      ;
                              *=BLKBDV
                      ;
E471  4C 23 F2                JMP     SIGNON      ;BLACK BOARD VECTOR
                      ;
                              *=WARMSV
                      ;
E474  4C 1B F1                JMP     RESET       ;WARM START VECTOR
                      ;
                              *=COLDSV
                      ;
E477  4C 25 F1                JMP     PWRUP       ;COLD START VECTOR (9000 FOR RAM VECTOR WRIT
                      ;
                              *=$9000
9000  20 0C 90                JSR     $900C
9003  4C 25 F1                JMP     PWRUP       ;(TO HANDLE RAM VECTOR WRITING)
9006  20 0C 90                JSR     $900C
9009  4C 1B F1                JMP     RESET
                      ;
                      ;
                      ;
                              *=MONORG
                      ;
                      ;
                      ;
```

```
                        ;
                        ;       HANDLER TABLE ENTRIES
                        ;
F0E3  50                TBLENT: .BYTE   'P'
F0E4  30 E4                     .WORD   PRINTV
F0E6  43                        .BYTE   'C'
F0E7  40 E4                     .WORD   CASETV
F0E9  45                        .BYTE   'E'
F0EA  00 E4                     .WORD   EDITRV
F0EC  53                        .BYTE   'S'
F0ED  10 E4                     .WORD   SCRENV
F0EF  4B                        .BYTE   'K'
F0F0  20 E4                     .WORD   KEYBDV
                        ;
                        ;
                        ;TBLLEN  =       IDENT-TBLENT-1  HANDLER TABLE LENGTH.  "MOVED TO LINE 8
                        ;
                        ;       ***** PRINT MESSAGES *****
                        ;
                        ;
F0F2  7D 41 54 41       IDENT:  .BYTE   CLS,'ATARI COMPUTER - MEMO PAD',CR
F0F6  52 49 20 43
F0FA  4F 4D 50 55
F0FE  54 45 52 20
F102  2D 20 4D 45
F106  4D 4F 20 50
F10A  41 44 9B
                        ;
00F0                    IDENTH  =       IDENT/256
00F2                    IDENTL  =       (-256)*IDENTH+IDENT ;SYSTEM I.D. MSG POINTER
                        ;
000E                    TBLLEN  =       IDENT-TBLENT-1 ;HANDLER TABLE LENGTH
F10D  42 4F 4F 54       DERR5:  .BYTE   'BOOT ERROR',CR
F111  20 45 52 52
F115  4F 52 9B
                        ;
00F1                    DERRH   =       DERR5/256
000D                    DERRL   =       (-256)*DERRH+DERR5 ;DISK ERROR MSG POINTER
                        ;
                        ;
                        ;
                        ;
                        ;       DEVICE/FILENAME SPECIFICATIONS
                        ;
F118  45 3A 9B          OPNEDT: .BYTE   'E:',CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
                        ;
00F1                    OPNH    =       OPNEDT/256
0018                    OPNL    =(-256)*OPNH+OPNEDT ;SCREEN EDITOR OPEN POINTER
F11B
                        ;
                        ;
                        ;
                        ;
                        ;******************************************************************
```

```
                          ;       RESET BUTTON ROUTINE STARTS HERE
                          ;*****************************************************************
                          ;
F11B  78                  RESET:  SEI                     ;DISABLE IRQ INTERRUPTS
F11C  AD 44 02                    LDA     COLDST          ;WERE WE IN MIDDLE OF COLDSTART?
F11F  D0 04                       BNE     PWRUP           ;YES, GO TRY IT AGAIN
F121  A9 FF                       LDA     #$FF
F123  D0 03                       BNE     PWRUP1          ;SET WARM START FLAG
                          ;
                          ;
                          ;
                          ;***************************************************************
                          ;       POWER UP ROUTINES START HERE
                          ;***************************************************************
                          ;
F125  78                  PWRUP:  SEI                     ;DISABLE IRQ INTERRUPTS
F126  A9 00                       LDA     #0              ;CLEAR WARMSTART FLAG
F128  85 08               PWRUP1: STA     WARMST
F12A  D8                          CLD                     ;CLEAR DECIMAL FLAG.
F12B  A2 FF                       LDX     #$FF
F12D  9A                          TXS                     ;SET STACK POINTER
F12E  20 3F F2                    JSR     SPECL           ;CARTRIDGE SPECIAL CASE?
F131  20 77 F2                    JSR     HARDI           ;DO HARDWARE INITIALIZATION
F134  A5 08                       LDA     WARMST          ;IS IT WARMSTART?
F136  D0 28                       BNE     ZOSRAM          ;YES, ONLY ZERO OS RAM
                          ;
F138  A9 00               ZERORM: LDA     #0
F13A  A0 08                       LDY     #WARMST
F13C  85 04                       STA     RAMLO
F13E  85 05                       STA     RAMLO+1         ;INITIALIZE RAM POINTER
F140  91 04               CLRRAM: STA     (RAMLO),Y       ;CLEAR MEMORY LOC.
F142  C8                          INY
F143  C0 00                       CPY     #0              ;AT END OF PAGE?
F145  D0 F9                       BNE     CLRRAM
F147  E6 05                       INC     RAMLO+1         ;YES, INCR PAGE POINTER
F149  A6 05                       LDX     RAMLO+1
F14B  E4 06                       CPX     TRAMSZ          ;AT END OF MEM?
F14D  D0 F1                       BNE     CLRRAM          ;NO.
                          ;
                          ; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
F14F  AD 72 E4                    LDA     BLKBDV+1
F152  85 0A                       STA     DOSVEC          ;USE BLACKBOARD VECTOR
F154  AD 73 E4                    LDA     BLKBDV+2        ;FOR DOSVEC
F157  85 0B                       STA     DOSVEC+1
F159  A9 FF                       LDA     #$FF
F15B  8D 44 02                    STA     COLDST          ;SET TO SHOW IN MIDDLE OF COLDSTART
F15E  D0 13                       BNE     ESTSCM          ;GO AROUND ZOSRAM
                          ;
                          ; CLEAR OS RAM (FOR WARMSTART)
F160  A2 00               ZOSRAM: LDX     #0
F162  8A                          TXA
F163  9D 00 02            ZOSRM2: STA     $200,X          ;CLEAR PAGES 2 AND 3
F166  9D 00 03                    STA     $300,X
F169  CA                          DEX
```

```
F16A  D0 F7                      BNE    ZOSRM2
F16C  A2 10                      LDX    #INTZBS
F16E  95 00            ZOSRM3:   STA    0,X          ;CLEAR ZERO PAGE LOCATIONS INTZBS-7F
F170  E8                         INX
F171  10 FB                      BPL    ZOSRM3
                       ;
                       ; ESTABLISH SCREEN MARGINS
F173  A9 02            ESTSCM:   LDA    #LEDGE
F175  85 52                      STA    LMARGN
F177  A9 27                      LDA    #REDGE
F179  85 53                      STA    RMARGN
                       ;
                       ;
                       ; MOVE VECTOR TABLE FROM ROM TO RAM
F17B  A2 25            OPSYS:    LDX    #$25
F17D  BD 80 E4         MOVVEC:   LDA    VCTABL,X     ;ROM TABLE
F180  9D 00 02                   STA    INTABS,X     ;TO RAM
F183  CA                         DEX
F184  10 F7                      BPL    MOVVEC
F186  20 8A F2                   JSR    OSRAM        ;DO O.S. RAM SETUP
F189  58                         CLI                 ;ENABLE IRQ INTERRUPTS
                       ;
                       ;
                       ;         LINK HANDLERS
                       ;
F18A  A2 0E                      LDX    #TBLLEN
F18C  BD E3 F0         NXTENT:   LDA    TBLENT,X     ;READ HANDLER TABLE ENTRY
F18F  9D 1A 03                   STA    HATABS,X     ;PUT IN TABLE
F192  CA                         DEX
F193  10 F7                      BPL    NXTENT       ;DONE WITH ALL ENTRIES?
                       ;
                       ;
                       ;
                       ;
                       ;
                       ; INTERROGATE CARTRIDGE ADDR. SPACE TO SEE WHICH CARTRIDGES THERE ARE
                       ;
F195  A2 00                      LDX    #0
F197  86 07                      STX    TSTDAT       ;CLEAR "B" CART. FLAG
F199  86 06                      STX    TRAMSZ       ;CLEAR "A" CART. FLAG
F19B  AE E4 02                   LDX    RAMSIZ
F19E  E0 90                      CPX    #$90         ;RAM IN "B" CART. SLOT?
F1A0  B0 0A                      BCS    ENDBCK
F1A2  AD FC 9F                   LDA    CART-$2000   ;NO,
F1A5  D0 05                      BNE    ENDBCK       ;CART. PLUGGED INTO "B" SLOT?
F1A7  E6 07                      INC    TSTDAT       ;YES, SET "B" CART. FLAG
F1A9  20 3C F2                   JSR    CBINI        ;INITIALIZE CARTRIDGE "B"
                       ;
F1AC  AE E4 02         ENDBCK:   LDX    RAMSIZ
F1AF  E0 B0                      CPX    #$B0         ;RAM IN "A" CART. SLOT?
F1B1  B0 0A                      BCS    ENDACK
F1B3  AE FC BF                   LDX    CART         ;NO,
F1B6  D0 05                      BNE    ENDACK       ;CART. PLUGGED INTO "A" SLOT?
F1B8  E6 06                      INC    TRAMSZ       ;YES, SET "A" CART. FLAG
```

```
F1BA  20 39 F2                   JSR     CAINI         ;INITIALIZE CARTRIDGE "A"
                         ;
                         ;
                         ; OPEN SCREEN EDITOR
                         ;
F1BD  A9 03              ENDACK: LDA     #3
F1BF  A2 00                      LDX     #SEX
F1C1  9D 42 03                   STA     ICCOM,X       ;OPEN I/O COMMAND
F1C4  A9 18                      LDA     #OPNL
F1C6  9D 44 03                   STA     ICBAL,X
F1C9  A9 F1                      LDA     #OPNH
F1CB  9D 45 03                   STA     ICBAH,X       ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
F1CE  A9 0C                      LDA     #$C
F1D0  9D 4A 03                   STA     ICAX1,X       ;SET UP OPEN FOR INPUT/OUTPUT
F1D3  20 56 E4                   JSR     CIOV          ;GO TO CIO
                         ;
F1D6  10 03                      BPL     SCRNOK        ;BR IF NO ERROR
F1D8  4C 25 F1                   JMP     PWRUP         ;RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
F1DB  E8                 SCRNOK: INX                   ;SCREEN OK, SO WAIT FOR VBLANK TO
F1DC  D0 FD                      BNE     SCRNOK        ;BRING UP THE DISPLAY
F1DE  C8                         INY
F1DF  10 FA                      BPL     SCRNOK
                         ;
                         ;
                         ; DO CASSETTE BOOT
F1E1  20 B2 F3                   JSR     CSBOOT        ;CHECK, BOOT, AND INIT
                         ;
                         ; CHECK TO SEE IF EITHER CARTRIDGE WANTS DISK BOOT
F1E4  A5 06                      LDA     TRAMSZ        ;CHECK BOTH CARTRIDGES
F1E6  05 07                      ORA     TSTDAT        ;
F1E8  F0 12                      BEQ     NOCART        ;NEITHER CARTRIDGE LIVES
F1EA  A5 06                      LDA     TRAMSZ        ;"A" CART?
F1EC  F0 03                      BEQ     NOA1          ;NO
F1EE  AD FD BF                   LDA     CARTFG        ;GET CARTRIDGE MODE FLAG
F1F1  A6 07              NOA1:   LDX     TSTDAT        ;"B" CART?
F1F3  F0 03                      BEQ     NOB1          ;NO
F1F5  0D FD 9F                   ORA     CARTFG-$2000 ;ADD OTHER FLAG
F1F8  29 01              NOB1:   AND     #1            ;DOES EITHER CART WANT BOOT?
F1FA  F0 03                      BEQ     NOBOOT        ;NO
                         ;
                         ; DO DISK BOOT
F1FC  20 CF F2           NOCART: JSR     BOOT          ;CHECK, BOOT, AND INIT
                         ;
                         ; GO TO ONE OF THE CARTRIDGES IF THEY SO DESIRE
F1FF  A9 00              NOBOOT: LDA     #0
F201  8D 44 02                   STA     COLDST        ;RESET TO SHOW DONE WITH COLDSTART
F204  A5 06                      LDA     TRAMSZ        ;"A" CART?
F206  F0 0A                      BEQ     NOA2          ;NO
F208  AD FD BF                   LDA     CARTFG        ;GET CARTRIDGE MODE FLAG
F20B  29 04                      AND     #4            ;DOES IT WANT TO RUN?
F20D  F0 03                      BEQ     NOA2          ;NO
F20F  6C FA BF                   JMP     (CARTCS)      ;RUN "A" CARTRIDGE
F212  A5 07              NOA2:   LDA     TSTDAT        ;"B" CART?
F214  F0 0A                      BEQ     NOCAR2        ;NO
```

```
F216  AD FD 9F             LDA    CARTFG-$2000 ;GET "B" MODE FLAG
F219  29 04                AND    #4           ;DOES IT WANT TO RUN?
F21B  F0 DF                BEQ    NOCART       ;NO
F21D  6C FA 9F             JMP    (CARTCS-$2000) ;RUN "B" CARTRIDGE
                  ;
                  ; NO CARTRIDGES, OR NEITHER WANTS TO RUN,
                  ; SO GO TO DOSVEC (DOS, CASSETTE, OR BLACKBOARD)
F220  6C 0A 00    NOCAR2:  JMP    (DOSVEC)
                  ;
                  ; PRINT SIGN-ON MESSAGE
F223  A2 F2       SIGNON:  LDX    #IDENTL
F225  A0 F0                LDY    #IDENTH
F227  20 85 F3             JSR    PUTLIN         ;GO PUT SIGN-ON MSG ON SCREEN
                  ;
                  ;
                  ;
                  ;        BLACKBOARD ROUTINE
F22A  20 30 F2    BLACKB:  JSR    BLKB2          ;"JSR EGETCH"
F22D  4C 2A F2             JMP    BLACKB         ;FOREVER
F230  AD 05 E4    BLKB2:   LDA    EDITRV+5       ;HIGH BYTE
F233  48                   PHA
F234  AD 04 E4             LDA    EDITRV+4       ;LOW BYTE
F237  48                   PHA
F238  60                   RTS                   ;SIMULATES "JMP (EDITRV)"
                  ;
                  ;
                  ; CARTRIDGE INITIALIZATION INDIRECT JUMPS
F239  6C FE BF    CAINI:   JMP    (CARTAD)
F23C  6C FE 9F    CBINI:   JMP    (CARTAD-$2000)
```

```
                              .PAGE
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;                 S U B R O U T I N E S
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; CHECK FOR HOW MUCH RAM & SPECIAL CARTRIDGE CASE.
                        ; IF SPECIAL CARTRIDGE CASE, DON'T GO BACK -- GO TO CART.
                        ;
F23F  AD FC BF          SPECL:  LDA     CART         ;CHECK FOR RAM OR CART
F242  D0 13                     BNE     ENSPE2       ;GO IF NOTHING OR MAYBE RAM
F244  EE FC BF                  INC     CART         ;NOW DO RAM CHECK
F247  AD FC BF                  LDA     CART         ;IS IT ROM?
F24A  D0 08                     BNE     ENSPEC       ;NO
F24C  AD FD BF                  LDA     CARTFG       ;YES,
F24F  10 03                     BPL     ENSPEC       ;BIT SET?
F251  6C FE BF                  JMP     (CARTAD)     ;YES, GO RUN CARTRIDGE
                        ;
                        ; CHECK FOR AMOUNT OF RAM
                        ;
                        ;
F254  CE FC BF          ENSPEC: DEC     CART         ;RESTORE RAM IF NEEDED
F257  A0 00             ENSPE2: LDY     #0
F259  84 05                     STY     RAMLO+1
F25B  A9 10                     LDA     #$10
F25D  85 06                     STA     TRAMSZ       ;SET RAM POINTER TO 4K.
F25F  B1 05             HOWMCH: LDA     (RAMLO+1),Y ;READ RAM LOCATION
F261  49 FF                     EOR     #$FF         ;INVERT IT.
F263  91 05                     STA     (RAMLO+1),Y ;WRITE INVERTED DATA.
F265  D1 05                     CMP     (RAMLO+1),Y ;READ RAM AGAIN
F267  D0 0D                     BNE     ENDRAM
F269  49 FF                     EOR     #$FF         ;CONVERT IT BACK
F26B  91 05                     STA     (RAMLO+1),Y ;RESTORE ORIGINAL RAM DATA
F26D  A5 06                     LDA     TRAMSZ
```

```
F26F  18                 CLC
F270  69 10              ADC     #$10
F272  85 06              STA     TRAMSZ      ;INCR. RAM POINTER BY 4K.
F274  D0 E9              BNE     HOWMCH      ;GO FIND HOW MUCH RAM.
F276  60         ENDRAM: RTS
                 ;
                 ;
                 ;
                 ;
                 ;       HARDWARE INITIALIZATION
                 ;
                 ;
F277  A9 00      HARDI:  LDA     #0
F279  AA                 TAX
F27A  9D 00 D0   CLRCHP: STA     $D000,X
F27D  9D 00 D4           STA     $D400,X
F280  9D 00 D2           STA     $D200,X
F283  9D 00 D3           STA     $D300,X
F286  E8                 INX
F287  D0 F1              BNE     CLRCHP
F289  60                 RTS
                 ;
                 ;
                 ;       O.S. RAM SETUP
                 ;
F28A  C6 11      OSRAM:  DEC     BRKKEY      ;TURN OFF BREAK KEY FLAG
F28C  A9 54              LDA     #.LOW.BRKKY2
F28E  8D 36 02           STA     BRKKY
F291  A9 E7              LDA     #.HIGH.BRKKY2
F293  8D 37 02           STA     BRKKY+1
F296  A5 06              LDA     TRAMSZ      ;READ RAM SIZE IN TEMP. REG.
F298  8D E4 02           STA     RAMSIZ      ;SAVE IT IN RAM SIZE.
F29B  8D E6 02           STA     MEMTOP+1    ; INIT. MEMTOP ADDR HI BYTE
F29E  A9 00              LDA     #0
F2A0  8D E5 02           STA     MEMTOP      ;INIT. MEMTOP ADDR LO BYTE
F2A3  A9 00              LDA     #INIMLL
F2A5  8D E7 02           STA     MEMLO
F2A8  A9 07              LDA     #INIMLH
F2AA  8D E8 02           STA     MEMLO+1     ;INITIALIZE MEMLO ADDR VECTOR
F2AD  20 0C E4           JSR     EDITRV+$C   ;EDITOR INIT.
F2B0  20 1C E4           JSR     SCRENV+$C   ;SCREEN INIT.
F2B3  20 2C E4           JSR     KEYBDV+$C   ;KEYBOARD INIT.
F2B6  20 3C E4           JSR     PRINTV+$C   ;PRINTER HANDLER INIT
F2B9  20 4C E4           JSR     CASETV+$C   ;CASSETTE HANDLER INIT
F2BC  20 6E E4           JSR     CIOINV      ;CIO INIT.
F2BF  20 65 E4           JSR     SIOINV      ;SIO INIT.
F2C2  20 6B E4           JSR     INTINV      ;INTERRUPT HANDLER INIT.
F2C5  AD 1F D0           LDA     CONSOL
F2C8  29 01              AND     #$1
F2CA  D0 02              BNE     NOKEY       ;GAME START KEY DEPRESSED?
F2CC  E6 4A              INC     CKEY        ;YES, SET KEY FLAG.
F2CE  60         NOKEY:  RTS
                 ;
                 ;
```

```
                              ; DO BOOT OF DISK
                              ;
F2CF  A5 08                   BOOT:    LDA   WARMST
F2D1  F0 0A                            BEQ   NOWARM        ;WARM START?
F2D3  A5 09                            LDA   BOOT?         ;YES,
F2D5  29 01                            AND   #1
F2D7  F0 03                            BEQ   NOINIT        ;VALID BOOT?
F2D9  20 7E F3                         JSR   DINI          ;YES, RE-INIT. DOS SOFTWARE
F2DC  60                      NOINIT:  RTS
F2DD  A9 01                   NOWARM:  LDA   #1
F2DF  8D 01 03                         STA   DUNIT         ;ASSIGN DISK DRIVE NO.
F2E2  A9 53                            LDA   #STATC
F2E4  8D 02 03                         STA   DCOMND        ;SET UP STATUS COMMAND
F2E7  20 53 E4                         JSR   DSKINV        ;GO DO DISK STATUS
F2EA  10 01                            BPL   DOBOOT        ;IS STATUS FROM SIO GOOD?
F2EC  60                               RTS                 ;NO, GO BACK WITH BAD BOOT STATUS
                              ;
F2ED  A9 00                   DOBOOT:  LDA   #0
F2EF  8D 0B 03                         STA   DAUX2
F2F2  A9 01                            LDA   #1
F2F4  8D 0A 03                         STA   DAUX1         ;SET SECTOR # TO 1.
F2F7  A9 00                            LDA   #BUFFL
F2F9  8D 04 03                         STA   DBUFLO
F2FC  A9 04                            LDA   #BUFFH
F2FE  8D 05 03                         STA   DBUFHI        ;SET UP BUFFER ADDR
F301  20 9D F3                SECT1:   JSR   GETSEC        ;GET SECTOR
F304  10 08                            BPL   ALLSEC        ;STATUS O.K. ?
F306  20 81 F3                BADDSK:  JSR   DSKRDE        ;NO, GO PRINT DISK READ ERROR
F309  A5 4B                            LDA   CASSBT
F30B  F0 E0                            BEQ   DOBOOT        ;CASSETTE BOOT?
F30D  60                               RTS                 ;YES, QUIT
F30E  A2 03                   ALLSEC:  LDX   #3
F310  BD 00 04                RDBYTE:  LDA   CASBUF+3,X    ;READ A BUFFER BYTE
F313  9D 40 02                         STA   DFLAGS,X      ;STORE IT
F316  CA                               DEX
F317  10 F7                            BPL   RDBYTE        ;DONE WITH 4 BYTE TRANSFER ?
F319  AD 42 02                         LDA   BOOTAD        ;YES,
F31C  85 04                            STA   RAMLO
F31E  AD 43 02                         LDA   BOOTAD+1
F321  85 05                            STA   RAMLO+1       ;PUT BOOT ADDR INTO Z. PAGE RAM
F323  AD 04 04                         LDA   CASBUF+7
F326  85 0C                            STA   DOSINI        ;ESTABLISH DOS INIT ADDRESS
F328  AD 05 04                         LDA   CASBUF+8
F32B  85 0D                            STA   DOSINI+1
F32D  A0 7F                   MVBUFF:  LDY   #$7F          ;YES, SET BYTE COUNT
F32F  B9 00 04                MVNXB:   LDA   CASBUF+3,Y
F332  91 04                            STA   (RAMLO),Y     ;MOVE A BYTE FROM SECTOR BUFFER TO BOOT ADDR
F334  88                               DEY
F335  10 F8                            BPL   MVNXB         ;DONE ?
F337  18                               CLC                 ;YES,
F338  A5 04                            LDA   RAMLO
F33A  69 80                            ADC   #$80
F33C  85 04                            STA   RAMLO
F33E  A5 05                            LDA   RAMLO+1
```

```
F340  69 00             ADC    #0
F342  85 05             STA    RAMLO+1     ;INCR BOOT LOADER BUFFER POINTER.
F344  CE 41 02          DEC    DBSECT      ;DECR # OF SECTORS.
F347  F0 11             BEQ    ENBOOT      ;MORE SECTORS ?
F349  EE 0A 03          INC    DAUX1       ;YES, INCR SECTOR #
F34C  20 9D F3  SECTX:  JSR    GETSEC      ;GO GET SECTOR.
F34F  10 DC             BPL    MVBUFF      ;STATUS O.K. ?
F351  20 81 F3          JSR    DSKRDE      ;NO, GO PRINT DISK READ ERROR
F354  A5 4B             LDA    CASSBT
F356  D0 AE             BNE    BADDSK      ;IF CASSETTE, QUIT.
F358  F0 F2             BEQ    SECTX       ;IF DISK, TRY SECTOR AGAIN.
F35A  A5 4B     ENBOOT: LDA    CASSBT
F35C  F0 03             BEQ    XBOOT       ;CASSETTE BOOT ?
F35E  20 9D F3          JSR    GETSEC      ;YES, GET EOF RECORD, BUT DON'T USE IT.
F361  20 6C F3  XBOOT:  JSR    BLOAD       ;GO EXECUTE BOOT LOADER
F364  B0 A0             BCS    BADDSK      ;IF BAD BOOT, DO IT OVER AGAIN
F366  20 7E F3          JSR    DINI        ;GO INIT. SOFTWARE
F369  E6 09             INC    BOOT?       ;SHOW BOOT SUCCESS
F36B  60                RTS
F36C  18        BLOAD:  CLC
F36D  AD 42 02          LDA    BOOTAD
F370  69 06             ADC    #6
F372  85 04             STA    RAMLO
F374  AD 43 02          LDA    BOOTAD+1
F377  69 00             ADC    #0
F379  85 05             STA    RAMLO+1     ;PUT START ADDR OF BOOTLOADER INTO RAM
F37B  6C 04 00          JMP    (RAMLO)
F37E  6C 0C 00  DINI:   JMP    (DOSINI)
                ;
                ;
                ;
                ;
                ; DISPLAY DISK READ ERROR MSG
                ;
F381  A2 0D     DSKRDE: LDX    #DERRL
F383  A0 F1             LDY    #DERRH
                ;
                ;
                ;
                ; PUT LINE ON SCREEN AT PRESENT CURSOR POSITION
                ;
                ;   X-REG -- LO BYTE, BEGIN ADDR OF LINE
                ;   Y-REG -- HI BYTE, BEGIN ADDR OF LINE
                ;
F385  8A        PUTLIN: TXA
F386  A2 00             LDX    #SEX
F388  9D 44 03          STA    ICBAL,X
F38B  98                TYA
F38C  9D 45 03          STA    ICBAH,X     ;SET UP ADDR OF BEGIN OF LINE
F38F  A9 09             LDA    #PUTTXT
F391  9D 42 03          STA    ICCOM,X     ;"PUT TEXT RECORD" COMMAND
F394  A9 FF             LDA    #$FF
F396  9D 48 03          STA    ICBLL,X     ;SET BUFFER LENGTH
F399  20 56 E4          JSR    CIOV        ;PUT LINE ON SCREEN
```

```
F39C  60                RTS
                  ;
                  ;
                  ;
                  ;
                  ; GET SECTOR FROM DISK 0
                  ;
F39D  A5 4B       GETSEC: LDA     CASSBT
F39F  F0 03               BEQ     DISKM      ;CASSETTE BOOT ?
F3A1  4C 7A E4            JMP     RBLOKV     ;YES, GO TO READ BLOCK ROUTINE
F3A4  A9 52       DISKM:  LDA     #READ
F3A6  8D 02 03            STA     DCOMND     ;SET READ SECTOR COMMAND
F3A9  A9 01               LDA     #1
F3AB  8D 01 03            STA     DUNIT      ;SET DRIVE NO. TO DRIVE 0
F3AE  20 53 E4            JSR     DSKINV     ;GET SECTOR
F3B1  60                  RTS
                  ;
                  ;
                  ;
                  ; DO CHECK FOR CASSETTE BOOT & IF SO, DO BOOT
                  ;
F3B2  A5 08       CSBOOT: LDA     WARMST     ;WARMSTART?
F3B4  F0 0A               BEQ     CSBOT2     ;NO
F3B6  A5 09               LDA     BOOT?      ;GET BOOT FLAG
F3B8  29 02               AND     #2         ;WAS CASSETTE BOOT SUCCESFULL?
F3BA  F0 03               BEQ     NOCSB2     ;NO
F3BC  20 E1 F3            JSR     CINI       ;YES, INIT CASSETTE SOFTWARE
F3BF  60          NOCSB2: RTS
                  ;
F3C0  A5 4A       CSBOT2: LDA     CKEY
F3C2  F0 1C               BEQ     NOCSBT     ;"C" KEY FLAG SET ?
F3C4  A9 80               LDA     #$80       ;YES,
F3C6  85 3E               STA     FTYPE      ;SET LONG IRG TYPE
F3C8  E6 4B               INC     CASSBT     ;SET CASSETTE BOOT FLAG
F3CA  20 7D E4            JSR     CSOPIV     ;OPEN CASSETTE FOR INPUT
F3CD  20 01 F3            JSR     SECT1      ;DO BOOT & INIT.
F3D0  A9 00               LDA     #0
F3D2  85 4B               STA     CASSBT     ;RESET CASSETTE BOOT FLAG
F3D4  85 4A               STA     CKEY       ;CLEAR KEY FLAG
F3D6  06 09               ASL     BOOT?      ;SHIFT BOOT FLAG (NOW=2 IF SUCCESS)
F3D8  A5 0C               LDA     DOSINI
F3DA  85 02               STA     CASINI     ;MOVE INIT ADDRESS FOR CASSETTE
F3DC  A5 0D               LDA     DOSINI+1
F3DE  85 03               STA     CASINI+1
F3E0  60          NOCSBT: RTS
                  ;
F3E1  6C 02 00    CINI:   JMP     (CASINI)   ;INIT CASSETTE
                  ;*******************************************************************
                  ;
                  ;
                  ; SPARE BYTE OR MODULE TOO LONG FLAG
                  ;
F3E4              CRNTP7  =*
                  ;
```

```
    4022                                   *=$14
    4023  0014  00            MONSPR: .BYTE   KBDORG-CRNTP7 ;^GMONITP TOO LONG
    4024                      ;
```

```
                                .PAGE
                                .TITLE  'DISPLAY HANDLER  -- 10-30-78 --  DISPLC'
                        ;
                        ; HANDLER DEPENDENT EQUATES
                        ;
      007D              CLRCOD  =       $7D         ;CLEAR SCREEN ATASCI CODE
      009F              CNTL1   =       $9F         ;POKEY KEY CODE FOR ^1
                        ;
      0068              FRMADR  =       SAVADR
      0066              TOADR   =       MLTTMP
                        ;
```

```
                                 .PAGE
                         ;
                         ;
                                 *=EDITRV
                         ;
                         ; SCREEN EDITOR HANDLER ENTRY POINT
                         ;
E400  FB F3              EDITOR: .WORD   EOPEN-1
E402  33 F6                      .WORD   RETUR1-1     ;(CLOSE)
E404  3D F6                      .WORD   EGETCH-1
E406  A3 F6                      .WORD   EOUTCH-1
E408  33 F6                      .WORD   RETUR1-1     ;(STATUS)
E40A  3C F6                      .WORD   NOFUNC-1     ;(SPECIAL)
E40C  4C E4 F3                   JMP     PWRONA
E40F  00                         .BYTE   0            ;ROM FILLER BYTE
                         ;
                                 *=SCRENV
                         ;
                         ; DISPLAY HANDLER ENTRY POINT
                         ;
E410  F5 F3              DISPLA: .WORD   DOPEN-1
E412  33 F6                      .WORD   RETUR1-1     ;(CLOSE)
E414  92 F5                      .WORD   GETCH-1
E416  B6 F5                      .WORD   OUTCH-1
E418  33 F6                      .WORD   RETUR1-1     ;(STATUS)
E41A  FB FC                      .WORD   DRAW-1       ;(SPECIAL)
E41C  4C E4 F3                   JMP     PWRONA
E41F  00                         .BYTE   0            ;ROM FILLER BYTE
                         ;
                                 *=KEYBDV
                         ;
                         ;
                         ; KEYBOARD HANDLER ENTRY POINT
                         ;
E420  33 F6              KBDHND: .WORD   RETUR1-1
E422  33 F6                      .WORD   RETUR1-1     ;(CLOSE)
E424  E1 F6                      .WORD   KGETCH-1
E426  3C F6                      .WORD   NOFUNC-1     ;(OUTCH)
E428  33 F6                      .WORD   RETUR1-1     ;(STATUS)
E42A  3C F6                      .WORD   NOFUNC-1     ;(SPECIAL)
E42C  4C E4 F3                   JMP     PWRONA
E42F  00                         .BYTE   0            ;ROM FILLER BYTE
                         ;
                         ;
                         ; INTERRUPT VECTOR TABLE ENTRY
                                 *=VCTABL-INTABS+VKEYBD
E488  BE FF                      .WORD   PIRQ5        ;KEYBOARD IRQ INTERRUPT VECTOR
```

```

                                          *=KBDORG
                                  ;
       F3E4  A9 FF                PWRONA: LDA     #$FF
       F3E6  8D FC 02                     STA     CH
       F3E9  AD E6 02                     LDA     MEMTOP+1
       F3EC  29 F0                        AND     #$F0          ;INSURE 4K PAGE BOUNDARY
       F3EE  85 6A                        STA     RAMTOP
       F3F0  A9 40                        LDA     #$40          ;DEFAULT.TO UPPER CASE ALPHA AT PWRON
       F3F2  8D BE 02                     STA     SHFLOK
       F3F5  60                           RTS                   ;POWER ON COMPLETED
```

```
                                .PAGE
                         ;
                         ;
                         ; BEGIN DISPLAY HANDLER OPEN PROCESSING
                         ;
F3F6  A5 2B              DOPEN:  LDA     ICAX2Z      ;GET AUX 2 BYTE
F3F8  29 0F                      AND     #$F
F3FA  D0 08                      BNE     OPNCOM      ;IF MODE ZERO, CLEAR ICAX1Z
F3FC  A5 2A              EOPEN:  LDA     ICAX1Z      ;CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
F3FE  29 0F                      AND     #$F
F400  85 2A                      STA     ICAX1Z
F402  A9 00                      LDA     #0
F404  85 57              OPNCOM: STA     DINDEX
F406  A9 E0                      LDA     #$E0        ;INITIALIZE GLOBAL VBLANK RAM
F408  8D F4 02                   STA     CHBAS
F40B  A9 02                      LDA     #2
F40D  8D F3 02                   STA     CHACT
F410  8D 2F 02                   STA     SDMCTL      ;TURN OFF DMA NEXT VBLANK
F413  A9 01                      LDA     #SUCCES
F415  85 4C                      STA     DSTAT       ;CLEAR STATUS
F417  A9 C0                      LDA     #$C0        ;DO IRQEN
F419  05 10                      ORA     POKMSK
F41B  85 10                      STA     POKMSK
F41D  8D 0E D2                   STA     IRQEN
F420  A9 00                      LDA     #0
F422  8D 93 02                   STA     TINDEX      ;TEXT INDEX MUST ALWAYS BE 0
F425  85 64                      STA     ADRESS
F427  85 7B                      STA     SWPFLG
F429  8D F0 02                   STA     CRSINH      ;TURN CURSOR ON AT OPEN
F42C  A0 0E                      LDY     #14         ;CLEAR TAB STOPS
F42E  A9 01                      LDA     #1          ;INIT TAB STOPS TO EVERY 8 CHARACTERS
F430  99 A3 02           CLRTBS: STA     TABMAP,Y
F433  88                         DEY
F434  10 FA                      BPL     CLRTBS
F436  A2 04                      LDX     #4          ;LOAD COLOR REGISTERS
F438  BD C1 FE           DOPEN8: LDA     COLRTB,X
F43B  9D C4 02                   STA     COLOR0,X
F43E  CA                         DEX
F43F  10 F7                      BPL     DOPEN8
F441  A4 6A                      LDY     RAMTOP      ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
F443  88                         DEY
F444  8C 95 02                   STY     TXTMSC+1
F447  A9 60                      LDA     #$60
F449  8D 94 02                   STA     TXTMSC
F44C  A6 57                      LDX     DINDEX
F44E  BD 69 FE                   LDA     ANCONV,X    ;CONVERT IT TO ANTIC CODE
F451  D0 04                      BNE     DOPENA      ;IF ZERO, IT IS ILLEGAL
F453  A9 91              OPNERR: LDA     #BADMOD     ;SET ERROR STATUS
F455  85 4C                      STA     DSTAT
F457  85 51              DOPENA: STA     HOLD1
F459  A5 6A                      LDA     RAMTOP      ;SET UP AN INDIRECT POINTER
F45B  85 65                      STA     ADRESS+1
F45D  BC 45 FE                   LDY     ALOCAT,X    ;ALLOCATE N BLOCKS OF 40 BYTES
F460  A9 28              DOPEN1: LDA     #40
```

```
F462  20 21 F9           JSR     DBSUB
F465  88                 DEY
F466  D0 F8              BNE     DOPEN1
F468  AD 6F 02           LDA     GPRIOR      ;CLEAR GTIA MODES
F46B  29 3F              AND     #$3F
F46D  85 67              STA     OPNTMP+1
F46F  A8                 TAY
F470  E0 08              CPX     #8          ;TEST IF 320X1
F472  90 17              BCC     NOT8
F474  8A                 TXA                 ;GET 2 LOW BITS
F475  6A                 ROR     A
F476  6A                 ROR     A
F477  6A                 ROR     A
F478  29 C0              AND     #$C0        ;NOW 2 TOP BITS
F47A  05 67              ORA     OPNTMP+1
F47C  A8                 TAY
F47D  A9 10              LDA     #16         ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
F47F  20 21 F9           JSR     DBSUB
F482  E0 0B              CPX     #11         ;TEST MODE 11
F484  D0 05              BNE     NOT8        ;IF MODE = 11
F486  A9 06              LDA     #6          ;PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
F488  8D C8 02           STA     COLOR4
F48B  8C 6F 02   NOT8:   STY     GPRIOR      ;STORE NEW PRIORITY
F48E  A5 64              LDA     ADRESS      ;SAVE MEMORY SCAN COUNTER ADDRESS
F490  85 58              STA     SAVMSC
F492  A5 65              LDA     ADRESS+1
F494  85 59              STA     SAVMSC+1
F496  AD 0B D4   VBWAIT: LDA     VCOUNT      ;WAIT FOR NEXT VBLANK BEFORE MESSING
F499  C9 7A              CMP     #$7A        ;WITH THE DISPLAY LIST
F49B  D0 F9              BNE     VBWAIT
F49D  20 1F F9           JSR     DBDEC       ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
F4A0  BD 75 FE           LDA     PAGETB,X    ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
F4A3  F0 06              BEQ     NOMOD       ;OF CROSSING A 256 BYTE PAGE BOUNDARY
F4A5  A9 FF              LDA     #$FF        ;IF SO, DROP DOWN A PAGE
F4A7  85 64              STA     ADRESS
F4A9  C6 65              DEC     ADRESS+1
F4AB  A5 64      NOMOD:  LDA     ADRESS      ;SAVE END OF DISPLAY LIST FOR LATER
F4AD  85 68              STA     SAVADR
F4AF  A5 65              LDA     ADRESS+1
F4B1  85 69              STA     SAVADR+1
F4B3  20 13 F9           JSR     DBDDEC      ;(DOUBLE BYTE DOUBLE DECREMENT)
F4B6  A9 41              LDA     #$41        ;(ANTIC) WAIT FOR VBLANK AND JMP TO TOP
F4B8  20 17 F9           JSR     STORE
F4BB  86 66              STX     OPNTMP
F4BD  A9 18              LDA     #24         ;INITIALIZE BOTSCR
F4BF  8D BF 02           STA     BOTSCR
F4C2  A5 57              LDA     DINDEX      ;DISALLOW MIXED MODE IF MODE.GE.9
F4C4  C9 09              CMP     #9
F4C6  B0 2D              BCS     NOTMXD
F4C8  A5 2A              LDA     ICAX1Z      ;TEST MIXED MODE
F4CA  29 10              AND     #$10
F4CC  F0 27              BEQ     NOTMXD
F4CE  A9 04              LDA     #4
F4D0  8D BF 02           STA     BOTSCR
```

```
F4D3  A2 02                       LDX     #2            ;ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
F4D5  A9 02             DOPEN2:   LDA     #2
F4D7  20 17 F9                    JSR     STORE
F4DA  CA                          DEX
F4DB  10 F8                       BPL     DOPEN2
F4DD  A4 6A                       LDY     RAMTOP        ;RELOAD MSC FOR TEXT
F4DF  88                          DEY
F4E0  98                          TYA
F4E1  20 17 F9                    JSR     STORE
F4E4  A9 60                       LDA     #$60
F4E6  20 17 F9                    JSR     STORE
F4E9  A9 42                       LDA     #$42
F4EB  20 17 F9                    JSR     STORE
F4EE  18                          CLC
F4EF  A9 0C                       LDA     #MXDMDE-NUMDLE ;POINT X AT MIXED MODE TABLE
F4F1  65 66                       ADC     OPNTMP
F4F3  85 66                       STA     OPNTMP
F4F5  A4 66             NOTMXD:   LDY     OPNTMP
F4F7  BE 51 FE                    LDX     NUMDLE,Y      ;GET NUMBER OF DISPLAY LIST ENTRIES
F4FA  A5 51             DOPEN3:   LDA     HOLD1         ;STORE N DLE'S
F4FC  20 17 F9                    JSR     STORE
F4FF  CA                          DEX
F500  D0 F8                       BNE     DOPEN3
F502  A5 57                       LDA     DINDEX        ;DO THE MESSY 320X1 PROBLEM
F504  C9 08                       CMP     #8
F506  90 1C                       BCC     DOPEN5
F508  A2 5D                       LDX     #93           ;GET REMAINING NUMBER OF DLE'S
F50A  A5 6A                       LDA     RAMTOP        ;RELOAD MEMORY SCAN COUNTER
F50C  38                          SEC
F50D  E9 10                       SBC     #$10
F50F  20 17 F9                    JSR     STORE
F512  A9 00                       LDA     #0
F514  20 17 F9                    JSR     STORE
F517  A9 4F                       LDA     #$4F          ;(ANTIC) RELOAD MSC CODE
F519  20 17 F9                    JSR     STORE
F51C  A5 51             DOPEN4:   LDA     HOLD1         ;DO REMAINING DLE'S
F51E  20 17 F9                    JSR     STORE
F521  CA                          DEX
F522  D0 F8                       BNE     DOPEN4
F524  A5 59             DOPEN5:   LDA     SAVMSC+1      ;POLISH OFF DISPLAY LIST
F526  20 17 F9                    JSR     STORE
F529  A5 58                       LDA     SAVMSC
F52B  20 17 F9                    JSR     STORE
F52E  A5 51                       LDA     HOLD1
F530  09 40                       ORA     #$40
F532  20 17 F9                    JSR     STORE
F535  A9 70                       LDA     #$70          ;24 BLANK LINES
F537  20 17 F9                    JSR     STORE
F53A  A9 70                       LDA     #$70
F53C  20 17 F9                    JSR     STORE
F53F  A5 64                       LDA     ADRESS        ;SAVE DISPLAY LIST ADDRESS
F541  8D 30 02                    STA     SDLSTL
F544  A5 65                       LDA     ADRESS+1
F546  8D 31 02                    STA     SDLSTL+1
```

```
F549  A9 70              LDA    #$70          ;ADD LAST BLANK LINE ENTRY
F54B  20 17 F9           JSR    STORE         ;POSITION ADRESS=SDLSTL-1
F54E  A5 64              LDA    ADRESS        ;STORE NEW MEMTOP
F550  8D E5 02           STA    MEMTOP
F553  A5 65              LDA    ADRESS+1
F555  8D E6 02           STA    MEMTOP+1
F558  A5 68              LDA    SAVADR
F55A  85 64              STA    ADRESS
F55C  A5 69              LDA    SAVADR+1
F55E  85 65              STA    ADRESS+1
F560  AD 31 02           LDA    SDLSTL+1
F563  20 17 F9           JSR    STORE
F566  AD 30 02           LDA    SDLSTL
F569  20 17 F9           JSR    STORE
F56C  A5 4C              LDA    DSTAT         ;IF ERROR OCURRED ON ALLOCATION, OPEN THE ED
F56E  10 07              BPL    DOPEN9
F570  48                 PHA                  ;SAVE STATUS
F571  20 FC F3           JSR    EOPEN         ;OPEN THE EDITOR
F574  68                 PLA                  ;RESTORE STATUS
F575  A8                 TAY                  ;AND RETURN IT TO CIO
F576  60                 RTS
F577  A5 2A     DOPEN9:  LDA    ICAX1Z        ;TEST CLEAR INHIBIT BIT
F579  29 20              AND    #$20
F57B  D0 0B              BNE    DOPEN7
F57D  20 B9 F7           JSR    CLRSCR        ;CLEAR SCREEN
F580  8D 90 02           STA    TXTROW        ;AND HOME TEXT CURSOR (AC IS ZERO)
F583  A5 52              LDA    LMARGN
F585  8D 91 02           STA    TXTCOL
F588  A9 22     DOPEN7:  LDA    #$22          ;EVERYTHING ELSE IS SET UP
F58A  0D 2F 02           ORA    SDMCTL        ;SO TURN ON DMACTL
F58D  8D 2F 02           STA    SDMCTL
F590  4C 21 F6           JMP    RETUR2
                ;
                ;
F593  20 96 FA  GETCH:   JSR    RANGE         ;GETCH DOES INCRSR, GETPLT DOESN'T
F596  20 A2 F5           JSR    GETPLT
F599  20 32 FB           JSR    INATAC        ;CONVERT INTERNAL CODE TO ATASCII
F59C  20 D4 F9           JSR    INCRSB
F59F  4C 34 F6           JMP    RETUR1
F5A2  20 47 F9  GETPLT:  JSR    CONVRT        ;CONVERT ROW/COLUMN TO ADRESS
F5A5  B1 64              LDA    (ADRESS),Y
F5A7  2D A0 02           AND    DMASK
F5AA  46 6F     SHIFTD:  LSR    SHFAMT        ;SHIFT DATA DOWN TO LOW BITS
F5AC  B0 03              BCS    SHIFT1
F5AE  4A                 LSR    A
F5AF  10 F9              BPL    SHIFTD        ;(UNCONDITIONAL)
F5B1  8D FA 02  SHIFT1:  STA    CHAR
F5B4  C9 00              CMP    #0            ;RESTORE FLAGS ALSO
F5B6  60                 RTS
                ;
                ;
                ;
F5B7  8D FB 02  OUTCH:   STA    ATACHR
F5BA  20 96 FA           JSR    RANGE
```

```
                        ;       JSR     OFFCRS
F5BD  AD FB 02          OUTCHA: LDA     ATACHR      ;TEST FOR CLEAR SCREEN
F5C0  C9 7D                     CMP     #CLRCOD
F5C2  D0 06                     BNE     OUTCHE
F5C4  20 B9 F7                  JSR     CLRSCR
F5C7  4C 21 F6                  JMP     RETUR2
F5CA  AD FB 02          OUTCHE: LDA     ATACHR      ;TEST FOR CARRIAGE RETURN
F5CD  C9 9B                     CMP     #CR
F5CF  D0 06                     BNE     OUTCHB
F5D1  20 30 FA                  JSR     DOCRWS      ;DO CR
F5D4  4C 21 F6                  JMP     RETUR2
F5D7  20 E0 F5          OUTCHB: JSR     OUTPLT
F5DA  20 D8 F9                  JSR     INCRSR
F5DD  4C 21 F6                  JMP     RETUR2
                        ;
                        ;
F5E0  AD FF 02          OUTPLT: LDA     SSFLAG      ;*****LOOP HERE IF START/STOP FLAG IS NON-0
F5E3  D0 FB                     BNE     OUTPLT
F5E5  A2 02                     LDX     #2
F5E7  B5 54             CRLOOP: LDA     ROWCRS,X    ;SAVE CURSOR LOCATION FOR DRAW LINE TO DRAW
F5E9  95 5A                     STA     OLDROW,X
F5EB  CA                        DEX
F5EC  10 F9                     BPL     CRLOOP
F5EE  AD FB 02                  LDA     ATACHR      ;CONVERT ATASCII(ATACHR) TO INTERNAL(CHAR)
F5F1  A8                        TAY                 ;SAVE ATACHR
F5F2  2A                        ROL     A
F5F3  2A                        ROL     A
F5F4  2A                        ROL     A
F5F5  2A                        ROL     A
F5F6  29 03                     AND     #3
F5F8  AA                        TAX                 ;X HAS INDEX INTO ATAINT
F5F9  98                        TYA                 ;RESTORE ATACHR
F5FA  29 9F                     AND     #$9F        ;STRIP OFF COLUMN ADDRESS
F5FC  1D F6 FE                  ORA     ATAINT,X    ;OR IN NEW COLUMN ADDRESS
F5FF  8D FA 02          OUTCH2: STA     CHAR
F602  20 47 F9                  JSR     CONVRT
F605  AD FA 02                  LDA     CHAR
F608  46 6F             SHIFTU: LSR     SHFAMT      ;SHIFT UP TO PROPER POSITION
F60A  B0 04                     BCS     SHIFT2
F60C  0A                        ASL     A
F60D  4C 08 F6                  JMP     SHIFTU
F610  2D A0 02          SHIFT2: AND     DMASK
F613  85 50                     STA     TMPCHR      ;SAVE SHIFTED DATA
F615  AD A0 02                  LDA     DMASK       ;INVERT MASK
F618  49 FF                     EOR     #$FF
F61A  31 64                     AND     (ADRESS),Y  ;MASK OFF OLD DATA
F61C  05 50                     ORA     TMPCHR      ;OR IN NEW DATA
F61E  91 64                     STA     (ADRESS),Y
F620  60                        RTS
                        ;
                        ;
F621  20 A2 F5          RETUR2: JSR     GETPLT      ;DO CURSOR ON THE WAY OUT
F624  85 5D                     STA     OLDCHR
F626  A6 57                     LDX     DINDEX      ;GRAPHICS HAVE INVISIBLE CURSOR
```

```
F628  D0 0A                   BNE    RETUR1
F62A  AE F0 02                LDX    CRSINH      ;TEST CURSOR INHIBIT
F62D  D0 05                   BNE    RETUR1
F62F  49 80                   EOR    #$80        ;TOGGLE MSB
F631  20 FF F5                JSR    OUTCH2      ;DISPLAY IT
F634  A4 4C          RETUR1:  LDY    DSTAT       ;RETURN TO CIO WITH STATUS IN Y
F636  A9 01                   LDA    #SUCCES
F638  85 4C                   STA    DSTAT       ;SET STATUS= SUCCESSFUL COMPLETION
F63A  AD FB 02                LDA    ATACHR      ;PUT ATACHR IN AC FOR RETURN TO CIO
F63D  60             NOFUNC:  RTS                ;(NON-EXISTENT FUNCTION RETURN POINT)
                     ;
                     ;
                     ;
                     ;  END OF DISPLAY HANDLER
                     ;
```

```
                                  PAGE
                          ;
                          ;
                          ;
                          ;
F63E  20 B3 FC            EGETCH: JSR     SWAP
F641  20 88 FA                    JSR     ERANGE
F644  A5 6B                       LDA     BUFCNT       ;ANYTHING IN THE BUFFER?
F646  D0 34                       BNE     EGETC3       ;YES
F648  A5 54                       LDA     ROWCRS       ;NO, SO SAVE BUFFER START ADDRESS
F64A  85 6C                       STA     BUFSTR
F64C  A5 55                       LDA     COLCRS
F64E  85 6D                       STA     BUFSTR+1
F650  20 E2 F6            EGETC1: JSR     KGETCH ;LET'S FILL OUR BUFFER
F653  84 4C                       STY     DSTAT        SAVE KEYBOARD STATUS
F655  AD FB 02                    LDA     ATACHR       ;TEST FOR CR
F658  C9 9B                       CMP     #CR
F65A  F0 12                       BEQ     EGETC2
F65C  20 AD F6                    JSR     DOSS         ;NO, SO PRINT IT
F65F  20 B3 FC                    JSR     SWAP         ;JSR DOSS DID SWAP SO SWAP BACK
F662  A5 63                       LDA     LOGCOL       ;BEEP IF NEARING LOGICAL COL 120
F664  C9 71                       CMP     #113
F666  D0 03                       BNE     EGETC6
F668  20 0A F9                    JSR     BELL
F66B  4C 50 F6            EGETC6: JMP     EGETC1
F66E  20 E4 FA            EGETC2: JSR     OFFCRS       ;GET BUFFER COUNT
F671  20 00 FC                    JSR     DOBUFC
F674  A5 6C                       LDA     BUFSTR       ;RETURN A CHARACTER
F676  85 54                       STA     ROWCRS
F678  A5 6D                       LDA     BUFSTR+1
F67A  85 55                       STA     COLCRS
F67C  A5 6B               EGETC3: LDA     BUFCNT
F67E  F0 11                       BEQ     EGETC5
F680  C6 6B               EGETC7: DEC     BUFCNT       ;AND RETURN TILL BUFCNT=0
F682  F0 0D                       BEQ     EGETC5
F684  A5 4C                       LDA     DSTAT        ;IF ERR, LOOP ON EGETC7 UNTIL BUFR IS EMPTIE
F686  30 F8                       BMI     EGETC7
F688  20 93 F5                    JSR     GETCH
F68B  8D FB 02                    STA     ATACHR
F68E  4C B3 FC                    JMP     SWAP         ;AND RETURN WITHOUT TURNING CURSOR BACK ON
F691  20 30 FA            EGETC5: JSR     DOCRWS       ;DO REAL CARRIAGE RETURN
F694  A9 9B                       LDA     #CR          ;AND RETURN EOL
F696  8D FB 02                    STA     ATACHR
F699  20 21 F6                    JSR     RETUR2       ;TURN ON CURSOR THEN SWAP
F69C  84 4C                       STY     DSTAT        ;SAVE KEYBOARD STATUS
F69E  4C B3 FC                    JMP     SWAP         ;AND RETURN THROUGH RETUR1
                          ;
F6A1  6C 64 00            JSRIND: JMP     (ADRESS)     ;JSR TO THIS CAUSES JSR INDIRECT
                          ;
F6A4  8D FB 02            EOUTCH: STA     ATACHR       ;SAVE ATASCII VALUE
F6A7  20 B3 FC                    JSR     SWAP
F6AA  20 88 FA                    JSR     ERANGE
F6AD  20 E4 FA            DOSS:   JSR     OFFCRS       ;TURN OFF CURSOR
F6B0  20 8D FC                    JSR     TSTCTL       ;TEST FOR CONTROL CHARACTERS (Z=1 IF CTL)
```

```
F6B3  F0 09             BEQ    EOUTC5
F6B5  0E A2 02  EOUTC6: ASL    ESCFLG       ;ESCFLG ONLY WORKS ONCE
F6B8  20 CA F5          JSR    OUTCHE
F6BB  4C B3 FC  ERETN:  JMP    SWAP         ;AND RETURN THROUGH RETUR1
F6BE  AD FE 02  EOUTC5: LDA    DSPFLG       ;DO DSPFLG AND ESCFLG
F6C1  0D A2 02          ORA    ESCFLG
F6C4  D0 EF             BNE    EOUTC6       ;IF NON-0 DISPLAY RATHER THAN EXECUTE IT
F6C6  0E A2 02          ASL    ESCFLG
F6C9  E8                INX                 ;PROCESS CONTROL CHARACTERS
F6CA  BD C6 FE          LDA    CNTRLS,X     ;GET DISPLACEMENT INTO ROUTINE
F6CD  85 64             STA    ADRESS
F6CF  BD C7 FE          LDA    CNTRLS+1,X   ;GET HIGH BYTE
F6D2  85 65             STA    ADRESS+1
F6D4  20 A1 F6          JSR    JSRIND       ;DO COMPUTED JSR
F6D7  20 21 F6          JSR    RETUR2       ;DO CURSOR
F6DA  4C B3 FC          JMP    SWAP         ;ALL DONE SO RETURN THROUGH RETUR1
                ;
                ;
                ;
                ;
                ; END SCREEN EDITOR.
                ;
                ;
                ; BEGIN KEYBOARD HANDLER
                ;
                ;
                ;
                ;
F6DD  A9 FF     KGETC2: LDA    #$FF
F6DF  8D FC 02          STA    CH
F6E2  A5 2A     KGETCH: LDA    ICAX1Z       ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MO
F6E4  4A                LSR    A
F6E5  B0 62             BCS    GETOUT
F6E7  A9 80             LDA    #BRKABT
F6E9  A6 11             LDX    BRKKEY       ;TEST BREAK
F6EB  F0 58             BEQ    K7           ;IF BREAK, PUT BRKABT IN DSTAT AND CR IN ATA
F6ED  AD FC 02          LDA    CH
F6F0  C9 FF             CMP    #$FF
F6F2  F0 EE             BEQ    KGETCH
F6F4  85 7C             STA    HOLDCH       ;SAVE CH FOR SHIFT LOCK PROC
F6F6  A2 FF             LDX    #$FF         ;"CLEAR" CH
F6F8  8E FC 02          STX    CH
F6FB  20 D8 FC          JSR    CLICK        ;DO KEYBOARD AUDIO FEEDBACK (A IS OK)
F6FE  AA        KGETC3: TAX                 ;DO ASCCON
F6FF  E0 C0             CPX    #$C0         ;TEST FOR CTL & SHIFT TOGETHER
F701  90 02             BCC    ASCC01
F703  A2 03             LDX    #3           ;BAD CODE
F705  BD FE FE  ASCC01: LDA    ATASCI,X
F708  8D FB 02          STA    ATACHR       ;DONE
F70B  C9 80             CMP    #$80         ;DO NULLS
F70D  F0 CE             BEQ    KGETC2
F70F  C9 81             CMP    #$81         ;CHECK ATARI KEY
F711  D0 0B             BNE    KGETC1
F713  AD B6 02          LDA    INVFLG
```

```
F716  49 80                  EOR    #$80
F718  8D B6 02               STA    INVFLG
F71B  4C DD F6               JMP    KGETC2     ;DONT RETURN A VALUE
F71E  C9 82        KGETC1:   CMP    #$82       ;CAPS/LOWER
F720  D0 07                  BNE    K1
F722  A9 00                  LDA    #0         ;CLEAR SHFLOK
F724  8D BE 02               STA    SHFLOK
F727  F0 B4                  BEQ    KGETC2
F729  C9 83        K1:       CMP    #$83       ;SHIFT CAPS/LOWER
F72B  D0 07                  BNE    K2
F72D  A9 40                  LDA    #$40
F72F  8D BE 02               STA    SHFLOK     ;SHIFT BIT
F732  D0 A9                  BNE    KGETC2
F734  C9 84        K2:       CMP    #$84       ;CNTL CAPS/LOWER
F736  D0 07                  BNE    K3
F738  A9 80                  LDA    #$80       ;CNTL BIT
F73A  8D BE 02               STA    SHFLOK
F73D  D0 9E                  BNE    KGETC2
F73F  C9 85        K3:       CMP    #$85       ;DO EOF
F741  D0 0A                  BNE    K6
F743  A9 88                  LDA    #EOFERR
F745  85 4C        K7:       STA    DSTAT
F747  85 11                  STA    BRKKEY     ;RESTORE BREAK
F749  A9 9B        GETOUT:   LDA    #CR        ;PUT CR IN ATACHR
F74B  D0 26                  BNE    K8         ;(UNCONDITIONAL)
F74D  A5 7C        K6:       LDA    HOLDCH     ;PROCESS SHIFT LOCKS
F74F  C9 40                  CMP    #$40       ;REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
F751  B0 15                  BCS    K5         ;OVER LOCK
F753  AD FB 02               LDA    ATACHR     ;TEST FOR ALPHA
F756  C9 61                  CMP    #$61       ;LOWER CASE A
F758  90 0E                  BCC    K5         ;NOT ALPHA IF LT
F75A  C9 7B                  CMP    #$7B       ;LOWER CASE Z+1
F75C  B0 0A                  BCS    K5         ;NOT ALPHA IF GE
F75E  AD BE 02               LDA    SHFLOK     ;DO SHIFT/CONTROL LOCK
F761  F0 05                  BEQ    K5         ;IF NO LOCK, DONT RE-DO IT
F763  05 7C                  ORA    HOLDCH
F765  4C FE F6               JMP    KGETC3     ;DO RETRY
F768  20 8D FC     K5:       JSR    TSTCTL     ;DONT INVERT MSB OF CONTROL CHARACTERS
F76B  F0 09                  BEQ    K4
F76D  AD FB 02               LDA    ATACHR
F770  4D B6 02               EOR    INVFLG
F773  8D FB 02     K8:       STA    ATACHR
F776  4C 34 F6     K4:       JMP    RETUR1     ALL DONE
                   ;
                   ;
```

```
                                      .PAGE
                               ;
                               ;
                               ; CONTROL CHARACTER PROCESSORS
                               ;
F779  A9 80                    ESCAPE: LDA    #$80         ;SET ESCAPE FLAG
F77B  8D A2 02                         STA    ESCFLG
F77E  60                               RTS
F77F  C6 54                    CRSRUP: DEC    ROWCRS
F781  10 06                            BPL    COMRET
F783  AE BF 02                         LDX    BOTSCR       ;WRAPAROUND
F786  CA                               DEX
F787  86 54                    UPDNCM: STX    ROWCRS
F789  4C 5C FC                 COMRET: JMP    STRBEG       ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F78C  E6 54                    CRSRDN: INC    ROWCRS
F78E  A5 54                            LDA    ROWCRS
F790  CD BF 02                         CMP    BOTSCR
F793  90 F4                            BCC    COMRET
F795  A2 00                            LDX    #0
F797  F0 EE                            BEQ    UPDNCM       ;(UNCONDITIONAL)
F799  C6 55                    CRSRLF: DEC    COLCRS
F79B  A5 55                            LDA    COLCRS
F79D  30 04                            BMI    CRSRL1       ;(IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
F79F  C5 52                            CMP    LMARGN
F7A1  B0 04                            BCS    COMRE1
F7A3  A5 53                    CRSRL1: LDA    RMARGN
F7A5  85 55                    LFRTCM: STA    COLCRS
F7A7  4C DD FB                 COMRE1: JMP    DOLCOL       ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F7AA  E6 55                    CRSRRT: INC    COLCRS
F7AC  A5 55                            LDA    COLCRS
F7AE  C5 53                            CMP    RMARGN
F7B0  90 F5                            BCC    COMRE1
F7B2  F0 F3                            BEQ    COMRE1       ;(CAUSE BLE)
F7B4  A5 52                            LDA    LMARGN
F7B6  4C A5 F7                         JMP    LFRTCM       ;UNCONDITIONAL TO COMMON STORE
F7B9  20 F3 FC                 CLRSCR: JSR    PUTMSC
F7BC  A0 00                            LDY    #0
F7BE  98                               TYA                 ;PUT 0 IN THE AC
F7BF  91 64                    CLRSC2: STA    (ADRESS),Y   ;(AC IS ZERO)
F7C1  C8                               INY
F7C2  D0 FB                            BNE    CLRSC2
F7C4  E6 65                            INC    ADRESS+1
F7C6  A6 65                            LDX    ADRESS+1
F7C8  E4 6A                            CPX    RAMTOP
F7CA  90 F3                            BCC    CLRSC2
F7CC  A9 FF                            LDA    #$FF         ;CLEAN UP LOGICAL LINE BIT MAP
F7CE  99 B2 02                 CLRSC3: STA    LOGMAP,Y     ;(Y IS ZERO AFTER CLRSC2 LOOP)
F7D1  C8                               INY
F7D2  C0 04                            CPY    #4
F7D4  90 F8                            BCC    CLRSC3
F7D6  20 E4 FC                 HOME:   JSR    COLCR        ;PLACE COLCRS AT LEFT EDGE
F7D9  85 63                            STA    LOGCOL
F7DB  85 6D                            STA    BUFSTR+1
F7DD  A9 00                            LDA    #0
```

```
F7DF  85 54               STA    ROWCRS
F7E1  85 56               STA    COLCRS+1
F7E3  85 6C               STA    BUFSTR
F7E5  60                  RTS
                 ;
F7E6  A5 63      BS:      LDA    LOGCOL       ;BACKSPACE
F7E8  C5 52               CMP    LMARGN
F7EA  F0 21               BEQ    BS1
F7EC  A5 55      BSA:     LDA    COLCRS       ;LEFT EDGE?
F7EE  C5 52               CMP    LMARGN
F7F0  D0 03               BNE    BS3          ;NO
F7F2  20 73 FC            JSR    DELTIM       ;YES, SEE IF LINE SHOULD BE DELETED
F7F5  20 99 F7   BS3:     JSR    CRSRLF
F7F8  A5 55               LDA    COLCRS
F7FA  C5 53               CMP    RMARGN
F7FC  D0 07               BNE    BS2
F7FE  A5 54               LDA    ROWCRS
F800  F0 03               BEQ    BS2
F802  20 7F F7            JSR    CRSRUP
F805  A9 20      BS2:     LDA    #$20         ;MAKE BACKSPACE DESTRUCTIVE
F807  8D FB 02            STA    ATACHR
F80A  20 E0 F5            JSR    OUTPLT
F80D  4C DD FB   BS1:     JMP    DOLCOL       ;AND RETURN
F810  20 AA F7   TAB:     JSR    CRSRRT       ;BEGIN SEARCH
F813  A5 55               LDA    COLCRS       ;TEST FOR NEW LINE
F815  C5 52               CMP    LMARGN
F817  D0 0A               BNE    TAB1         ;NO
F819  20 34 FA            JSR    DOCR         ;DO CARRIAGE RETURN
F81C  20 20 FB            JSR    LOGGET       ;CHECK IF END OF LOGICAL LINE
F81F  90 02               BCC    TAB1         ;NO, CONTINUE
F821  B0 07               BCS    TAB2         ;(UNCONDITIONAL)
F823  A5 63      TAB1:    LDA    LOGCOL       ;CHECK FOR TAB STOP
F825  20 25 FB            JSR    BITGET
F828  90 E6               BCC    TAB          ;NO, SO KEEP LOOKING
F82A  4C DD FB   TAB2:    JMP    DOLCOL       ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F82D  A5 63      SETTAB:  LDA    LOGCOL
F82F  4C 06 FB            JMP    BITSET       ;SET BIT IN MAP AND RETURN
F832  A5 63      CLRTAB:  LDA    LOGCOL
F834  4C 12 FB            JMP    BITCLR       ;CLEAR " " " " "
F837  20 9D FC   INSCHR:  JSR    PHACRS
F83A  20 A2 F5            JSR    GETPLT       ;GET CHARACTER UNDER CURSOR
F83D  85 7D               STA    INSDAT
F83F  A9 00               LDA    #0
F841  8D BB 02            STA    SCRFLG
F844  20 FF F5   INSCH4:  JSR    OUTCH2       ;STORE DATA
F847  A5 63               LDA    LOGCOL       ;SAVE LOGCOL: IF AFTER INCRSA LOGCOL IS
F849  48                  PHA                 ;< THAN IT IS NOW, END LOOP
F84A  20 DC F9            JSR    INCRSA       ;SPECIAL INCRSR ENTRY POINT
F84D  68                  PLA
F84E  C5 63               CMP    LOGCOL
F850  B0 0C               BCS    INSCH3       ;QUIT
F852  A5 7D      INSCH1:  LDA    INSDAT       ;KEEP GOING
F854  48                  PHA
F855  20 A2 F5            JSR    GETPLT
```

```
F858  85 7D              STA     INSDAT
F85A  68                 PLA
F85B  4C 44 F8           JMP     INSCH4
F85E  20 A8 FC  INSCH3:  JSR     PLACRS
F861  CE BB 02  INSCH6:  DEC     SCRFLG
F864  30 04              BMI     INSCH5       ;IF SCROLL OCCURRED
F866  C6 54              DEC     ROWCRS       ;MOVE CURSOR UP
F868  D0 F7              BNE     INSCH6       ;(UNCOND) CONTINUE UNTIL SCRFLG IS MINUS
F86A  4C DD FB  INSCH5:  JMP     DOLCOL       ;COLVERT ROW AND COL TO LOGCOL AND RETURN
                ;
                ;
F86D  20 9D FC  DELCHR:  JSR     PHACRS
F870  20 47 F9  DELCH1:  JSR     CONVRT       ;GET DATA TO THE RIGHT OF THE CURSOR
F873  A5 64              LDA     ADRESS
F875  85 68              STA     SAVADR       ;SAVE ADRESS TO KNOW WHERE TO PUT DATA
F877  A5 65              LDA     ADRESS+1
F879  85 69              STA     SAVADR+1
F87B  A5 63              LDA     LOGCOL
F87D  48                 PHA
F87E  20 D4 F9           JSR     INCRSB       ;PUT CURSOR OVER NEXT CHARACTER
F881  68                 PLA
F882  C5 63              CMP     LOGCOL       ;TEST NEW LOGCOL AGAINST OLD LOGCOL
F884  B0 10              BCS     DELCH2       ;IF OLD.GE.NEW THEN QUIT
F886  A5 54              LDA     ROWCRS       ;IS ROW OFF SCREEN?
F888  CD BF 02           CMP     BOTSCR
F88B  B0 09              BCS     DELCH2       ;YES, SO QUIT
F88D  20 A2 F5           JSR     GETPLT       ;GET DATA UNDER CURSOR
F890  A0 00              LDY     #0
F892  91 68              STA     (SAVADR),Y   ;PUT IT IN PREVIOUS POSITION
F894  F0 DA              BEQ     DELCH1       ;AND LOOP (UNCONDITIONAL)
F896  A0 00     DELCH2:  LDY     #0
F898  98                 TYA
F899  91 68              STA     (SAVADR),Y   ;CLEAR THE LAST POSITION
F89B  20 68 FC           JSR     DELTIA       ;TRY TO DELETE A LINE
F89E  20 A8 FC           JSR     PLACRS
F8A1  4C DD FB           JMP     DOLCOL       ;AND RETURN
F8A4  38        INSLIN:  SEC                  ;NORMAL INSLIN PUTS "1" INTO BIT MAP
F8A5  20 7B FB  INSLIA:  JSR     EXTEND       ;ENTRY POINT FOR C=0
F8A8  A5 52              LDA     LMARGN       ;DO CARRIAGE RETURN (NO LF)
F8AA  85 55              STA     COLCRS
F8AC  20 47 F9           JSR     CONVRT       ;GET ADDRESS
F8AF  A5 64              LDA     ADRESS       ;SET UP TO=40+FROM (FROM = CURSOR)
F8B1  85 68              STA     FRMADR
F8B3  18                 CLC
F8B4  69 28              ADC     #40
F8B6  85 66              STA     TOADR
F8B8  A5 65              LDA     ADRESS+1
F8BA  85 69              STA     FRMADR+1
F8BC  69 00              ADC     #0
F8BE  85 67              STA     TOADR+1
F8C0  A6 54              LDX     ROWCRS       ;SET UP LOOP COUNTER
F8C2  E0 17              CPX     #23
F8C4  F0 08              BEQ     INSLI2
F8C6  20 4E FB  INSLI1:  JSR     MOVLIN
```

```
F8C9  E8                 INX
F8CA  E0 17              CPX     #23
F8CC  D0 F8              BNE     INSLI1
F8CE  20 9B FB  INSLI2:  JSR     CLRLIN      ;CLEAR CURRENT LINE
F8D1  4C DD FB           JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
F8D4  20 DD FB  DELLIN:  JSR     DOLCOL      ;GET BEGINNING OF LOG LINE (HOLD1)
F8D7  A4 51     DELLIA:  LDY     HOLD1       ;SQUEEZE BIT MAP
F8D9  84 54              STY     ROWCRS      ;PUT CURSOR THERE
F8DB  A4 54     DELLIB:  LDY     ROWCRS
F8DD  98        DELLI1:  TYA
F8DE  38                 SEC
F8DF  20 23 FB           JSR     LO2GET      ;GET NEXT BIT
F8E2  08                 PHP
F8E3  98                 TYA
F8E4  18                 CLC
F8E5  69 78              ADC     #120
F8E7  28                 PLP
F8E8  20 04 FB           JSR     BITPUT      ;WRITE IT OVER PRESENT BIT
F8EB  C8                 INY
F8EC  C0 18              CPY     #24
F8EE  D0 ED              BNE     DELLI1      ;LOOP
F8F0  AD B4 02           LDA     LOGMAP+2    ;SET LSB
F8F3  09 01              ORA     #1
F8F5  8D B4 02           STA     LOGMAP+2
F8F8  A5 52     DELLI2:  LDA     LMARGN      ;DELETE LINE OF DATA USING PART OF SCROLL
F8FA  85 55              STA     COLCRS      ;CR NO LF
F8FC  20 47 F9           JSR     CONVRT
F8FF  20 B7 FB           JSR     SCROL1
F902  20 20 FB           JSR     LOGGET      ;TEST NEXT LINE FOR CONTINUATION
                ;  IS IT A NEW LOG LINE?
F905  90 D4              BCC     DELLIB      ;NO SO DELETE ANOTHER
F907  4C DD FB           JMP     DOLCOL      ;YES SO DOLCOL AND RETURN
F90A  A0 20     BELL:    LDY     #$20
F90C  20 D8 FC  BELL1:   JSR     CLICK
F90F  88                 DEY
F910  10 FA              BPL     BELL1
F912  60                 RTS
```

```
                                     .PAGE
                         ;
                         ;
                         ; ROUTINES
                         ;
                         ;
                         ; DOUBLE BYTE DECREMENT OF INDIRECT POINTER
                         ; INCLUDING DB SUBTRACT AND DB DOUBLE DECREMENT
                         ;
F913  A9 02              DBDDEC: LDA     #2
F915  D0 0A                      BNE     DBSUB         ;(UNCONDITIONAL)
                         ;
                         ; STORE DATA INDIRECT AND DECREMENT POINTER
                         ; (PLACED HERE TO SAVE JMP DBDEC AFTER STORE)
F917  A4 4C              STORE:  LDY     DSTAT         ;RETURN ON ERROR
F919  30 2B                      BMI     STROK
F91B  A0 00                      LDY     #0
F91D  91 64              STORE1: STA     (ADRESS),Y
                         ;       JMP     DBDEC         DECREMENT AND RETURN
                         ;
F91F  A9 01              DBDEC:  LDA     #1
F921  8D 9E 02           DBSUB:  STA     SUBTMP
F924  A5 4C                      LDA     DSTAT         ;RETURN ON ERROR
F926  30 1E                      BMI     STROK
F928  A5 64                      LDA     ADRESS
F92A  38                         SEC
F92B  ED 9E 02                   SBC     SUBTMP
F92E  85 64                      STA     ADRESS
F930  B0 02                      BCS     DBSUB1
F932  C6 65                      DEC     ADRESS+1
F934  A5 0F              DBSUB1: LDA     APPMHI+1      ;MAKE SURE NOTHING EVER OVERWRITES APPMHI
F936  C5 65                      CMP     ADRESS+1
F938  90 0C                      BCC     STROK         ;OK
F93A  D0 06                      BNE     STRERR        ;ERROR
F93C  A5 0E                      LDA     APPMHI
F93E  C5 64                      CMP     ADRESS
F940  90 04                      BCC     STROK
F942  A9 93              STRERR: LDA     #SCRMEM       ;SHOW MEM TOO SMALL FOR SCREEN ERROR
F944  85 4C                      STA     DSTAT
F946  60                 STROK:  RTS
                         ;
                         ;
                         ;
                         ; CONVERT ROW/COLUMN CURSOR INTO REAL ADDRESS (FROM SAVMSC ON UP)
                         ;
F947  A5 54              CONVRT: LDA     ROWCRS        ;SAVE CURSOR
F949  48                         PHA
F94A  A5 55                      LDA     COLCRS
F94C  48                         PHA
F94D  A5 56                      LDA     COLCRS+1
F94F  48                         PHA
F950  20 F3 FC                   JSR     PUTMSC
F953  A5 54                      LDA     ROWCRS        ;PUT 10*ROWCRS INTO MLTTMP
F955  85 66                      STA     MLTTMP
```

```
     4785  F957  A9 00                       LDA     #0
     4786  F959  85 67                       STA     MLTTMP+1
     4787  F95B  A5 66                       LDA     MLTTMP        ;QUICK X8
     4788  F95D  0A                          ASL     A
     4789  F95E  26 67                       ROL     MLTTMP+1
     4790  F960  85 51                       STA     HOLD1         ;(SAVE 2X VALUE)
     4791  F962  A4 67                       LDY     MLTTMP+1      ;""
     4792  F964  8C 9F 02                    STY     HOLD2         ;""
     4793  F967  0A                          ASL     A
     4794  F968  26 67                       ROL     MLTTMP+1
     4795  F96A  0A                          ASL     A
     4796  F96B  26 67                       ROL     MLTTMP+1
     4797  F96D  18                          CLC                   ;ADD IN 2X
     4798  F96E  65 51                       ADC     HOLD1
     4799  F970  85 66                       STA     MLTTMP
     4800  F972  A5 67                       LDA     MLTTMP+1
     4801  F974  6D 9F 02                    ADC     HOLD2
     4802  F977  85 67                       STA     MLTTMP+1
     4803  F979  A6 57                       LDX     DINDEX        ;NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINIS
     4804  F97B  BC 81 FE                    LDY     DHLINE,X      ;MULTIPLY
     4805  F97E  88                 CONVR1:  DEY                   ;LOOP N TIMES
     4806  F97F  30 07                       BMI     CONVR2
     4807  F981  06 66                       ASL     MLTTMP
     4808  F983  26 67                       ROL     MLTTMP+1
     4809  F985  4C 7E F9                    JMP     CONVR1
     4810  F988  BC A5 FE           CONVR2:  LDY     DIV2TB,X      ;NOW DIVIDE HCRSR TO ACCOUNT FOR PARTIAL BYT
     4811  F98B  A5 55                       LDA     COLCRS
     4812  F98D  A2 07                       LDX     #7            ;* TRICKY *
     4813  F98F  88                 CONVR3:  DEY
     4814  F990  30 0A                       BMI     CONVR4
     4815  F992  CA                          DEX
     4816  F993  46 56                       LSR     COLCRS+1
     4817  F995  6A                          ROR     A
     4818  F996  6E A1 02                    ROR     TMPLBT        ;SAVE LOW BITS FOR MASK
     4819  F999  4C 8F F9                    JMP     CONVR3
     4820  F99C  C8                 CONVR4:  INY                   ;SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
     4821  F99D  18                          CLC
     4822  F99E  65 66                       ADC     MLTTMP        ;ADD SHIFTED COLCRS TO MLTTMP
     4823  F9A0  85 66                       STA     MLTTMP
     4824  F9A2  90 02                       BCC     CONVR5
     4825  F9A4  E6 67                       INC     MLTTMP+1
     4826  F9A6  38                 CONVR5:  SEC                   ;* TRICKY *
     4827  F9A7  6E A1 02           CONVR6:  ROR     TMPLBT        ;SLIDE A "1" UP AGAINST LOW BITS (CONTINUE T
     4828  F9AA  18                          CLC
     4829  F9AB  CA                          DEX                   ;AND FINISH SHIFT SO LOW BITS ARE
     4830  F9AC  10 F9                       BPL     CONVR6        ;RIGHT JUSTIFIED.
     4831  F9AE  AE A1 02                    LDX     TMPLBT        ;TMPLBT IS NOW THE INDEX INTO DMASKTB
     4832  F9B1  A5 66                       LDA     MLTTMP        ;PREPARE FOR RETURN
     4833  F9B3  18                          CLC
     4834  F9B4  65 64                       ADC     ADRESS
     4835  F9B6  85 64                       STA     ADRESS
     4836  F9B8  85 5E                       STA     OLDADR        ;REMEMBER THIS ADDRESS FOR CURSOR
     4837  F9BA  A5 67                       LDA     MLTTMP+1
     4838  F9BC  65 65                       ADC     ADRESS+1
```

```
F9BE  85 65              STA    ADRESS+1
F9C0  85 5F              STA    OLDADR+1
F9C2  BD B1 FE           LDA    DMASKT,X
F9C5  8D A0 02           STA    DMASK
F9C8  85 6F              STA    SHFAMT
F9CA  68                 PLA
F9CB  85 56              STA    COLCRS+1
F9CD  68                 PLA
F9CE  85 55              STA    COLCRS
F9D0  68                 PLA
F9D1  85 54              STA    ROWCRS
F9D3  60                 RTS
                 ;
                 ;
                 ; INCREMENT CURSOR AND DETECT BOTH END OF LINE AND END OF SCREEN
                 ;
F9D4  A9 00      INCRSB: LDA    #0          ;NON-EXTEND ENTRY POINT
F9D6  F0 02              BEQ    INCRSC
F9D8  A9 9B      INCRSR: LDA    #$9B        ;SPECIAL CASE ELIMINATOR
F9DA  85 7D      INCRSC: STA    INSDAT
F9DC  E6 63      INCRSA: INC    LOGCOL      ;(INSCHR ENTRY POINT)
F9DE  E6 55              INC    COLCRS
F9E0  D0 02              BNE    INCRS2      ;DO HIGH BYTE
F9E2  E6 56              INC    COLCRS+1
F9E4  A5 55      INCRS2: LDA    COLCRS      ;TEST END OF LINE
F9E6  A6 57              LDX    DINDEX
F9E8  DD 8D FE           CMP    COLUMN,X    ;TEST TABLED VALUE FOR ALL SCREEN MODES
F9EB  F0 0B              BEQ    INC2A       ;DO CR IF EQUAL
F9ED  E0 00              CPX    #0          ;MODE 0?
F9EF  D0 06              BNE    INCRS3      ;IF NOT, JUST RETURN
F9F1  C5 53              CMP    RMARGN      ;TEST AGAINST RMARGN
F9F3  F0 02              BEQ    INCRS3      ;EQUAL IS OK
F9F5  B0 01              BCS    INC2A       ;IF GREATER THAN, DO CR
F9F7  60         INCRS3: RTS
F9F8  E0 08      INC2A:  CPX    #8          ;CHECK MODE
F9FA  90 04              BCC    DOCR1       ;NOT 320X1 SO DO IT
F9FC  A5 56              LDA    COLCRS+1    ;TEST MSD
F9FE  F0 F7              BEQ    INCRS3      ;ONLY AT 64 SO DON'T DO IT
FA00  A5 57      DOCR1:  LDA    DINDEX      ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
FA02  D0 30              BNE    DOCR
FA04  A5 63              LDA    LOGCOL      ;TEST LINE OVERRUN
FA06  C9 51              CMP    #81
FA08  90 0A              BCC    DOCR1B      ;IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
FA0A  A5 7D              LDA    INSDAT
FA0C  F0 26              BEQ    DOCR        ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
FA0E  20 30 FA           JSR    DOCRWS      ;LOG LINE OVERFLOW IS SPECIAL CASE
FA11  4C 77 FA           JMP    INCRS1      ;RETURN
FA14  20 34 FA   DOCR1B: JSR    DOCR        ;GET IT OVER WITH
FA17  A5 54              LDA    ROWCRS
FA19  18                 CLC                ;TEST LOGICAL LINE BIT MAP
FA1A  69 78              ADC    #120
FA1C  20 25 FB           JSR    BITGET
FA1F  90 08              BCC    DOCR1A      ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L
FA21  A5 7D              LDA    INSDAT      ;DON'T EXTEND IF INSDAT IS ZERO
```

```
FA23  F0 04                BEQ   DOCR1A       ;(INSCHR SPECIAL CASE)
FA25  18                   CLC                ;INSERT "0" INTO BIT MAP
FA26  20 A5 FB             JSR   INSLIA
FA29  4C DD FB   DOCR1A:   JMP   DOLCOL       ;CONVERT ROW AND COL TO LOGCOL AND RETURN
FA2C  A9 00      NOSCRL:   LDA   #0           ;DOCR WITHOUT SCROLL
FA2E  F0 02                BEQ   NOSCR1       ;(UNCONDITIONAL)
FA30  A9 9B      DOCRWS:   LDA   #$9B         ;DOCR WITH SCROLLING (NORMAL MODE)
FA32  85 7D      NOSCR1:   STA   INSDAT
FA34  20 E4 FC   DOCR:     JSR   COLCR        ;PLACE COLCRS AT LEFT EDGE
FA37  A9 00                LDA   #0
FA39  85 56                STA   COLCRS+1
FA3B  E6 54                INC   ROWCRS
FA3D  A6 57      DOCR2:    LDX   DINDEX
FA3F  A0 18                LDY   #24          ;SET UP SCROLL LOOP COUNTER
FA41  24 7B                BIT   SWPFLG
FA43  10 05                BPL   DOCR2A       ;BRANCH IF NORMAL
FA45  A0 04                LDY   #4
FA47  98                   TYA
FA48  D0 03                BNE   DOCR2B       ;(UNCONDITIONAL)
FA4A  BD 99 FE   DOCR2A:   LDA   NOROWS,X     ;GET NO OF ROWS
FA4D  C5 54      DOCR2B:   CMP   ROWCRS
FA4F  D0 26                BNE   INCRS1
FA51  8C 9D 02             STY   HOLD3
FA54  8A                   TXA                ;DON'T SCROLL IF MODE <> 0
FA55  D0 20                BNE   INCRS1
FA57  A5 7D                LDA   INSDAT       ;OR IF INSDAT = 0
FA59  F0 1C                BEQ   INCRS1
                 ;         LDA   INSDAT       IF INSDAT <> $9B THEN ROLL IN A 0
FA5B  C9 9B                CMP   #$9B         ;TO EXTEND BOTTOM LOGICAL LINE
FA5D  38                   SEC
FA5E  F0 01                BEQ   DOCR4B
FA60  18                   CLC
FA61  20 AC FB   DOCR4B:   JSR   SCROLL       ;LOOP BACK TO HERE IF >1 SCROLLS
FA64  EE BB 02             INC   SCRFLG
FA67  C6 6C                DEC   BUFSTR       ;ROWS MOVE UP SO BUFSTR SHOULD TOO
FA69  CE 9D 02             DEC   HOLD3
FA6C  AD B2 02             LDA   LOGMAP
FA6F  38                   SEC                ;FOR PARTIAL LINES, ROLL IN A "1"
FA70  10 EF                BPL   DOCR4B       ;AGAIN IF PARTIAL LOGICAL LINE
FA72  AD 9D 02             LDA   HOLD3        ;PLACE CURSOR AT NEW LINE NEAR THE BOTTOM
FA75  85 54                STA   ROWCRS
FA77  4C DD FB   INCRS1:   JMP   DOLCOL       ;COLVERT ROW AND COL TO LOGCOL AND RETURN
                 ;
                 ;
                 ; SUBEND: SUBTRACT ENDPT FROM ROWAC OR COLAC. (X=0 OR 2)
                 ;
FA7A  38         SUBEND:   SEC
FA7B  B5 70                LDA   ROWAC,X
FA7D  E5 74                SBC   ENDPT
FA7F  95 70                STA   ROWAC,X
FA81  B5 71                LDA   ROWAC+1,X
FA83  E5 75                SBC   ENDPT+1
FA85  95 71                STA   ROWAC+1,X
FA87  60                   RTS
```

```
                          ;
                          ;
                          ; RANGE: DO CURSOR RANGE TEST. IF ERROR, POP STACK TWICE AND JMP RETURN
                          ;        (ERANGE IS EDITOR ENTRY POINT AND TEST IF EDITOR IS OPEN.
                          ;         IF IT ISNT IT OPENS THE EDITOR AND CONTINUES)
                          ;
FA88  AD BF 02            ERANGE: LDA    BOTSCR      ;IF BOTSCR=4
FA8B  C9 04                       CMP    #4
FA8D  F0 07                       BEQ    RANGE       ;THEN IT IS IN MIXED MODE AND OK
FA8F  A5 57                       LDA    DINDEX      ;IF MODE = 0
FA91  F0 03                       BEQ    RANGE       ;THEN IT IS IN EDITOR MODE AND OK
FA93  20 FC F3                    JSR    EOPEN       ;IF NOT, OPEN EDITOR
FA96  A9 27               RANGE:  LDA    #39         ;***** RANGE CHECK RMARGN ***** SET UP AC
FA98  C5 53                       CMP    RMARGN      ;***** RANGE CHECK RMARGN ***** COMPARE
FA9A  B0 02                       BCS    RANGE3      ;***** RANGE CHECK RMARGN ***** BRANCH GE
FA9C  85 53                       STA    RMARGN      ;***** RANGE CHECK RMARGN ***** BAD SO STORE
FA9E  A6 57               RANGE3: LDX    DINDEX
FAA0  BD 99 FE                    LDA    NOROWS,X    ;CHECK ROWS
FAA3  C5 54                       CMP    ROWCRS
FAA5  90 2A                       BCC    RNGERR      ;(ERROR IF TABLE.GE.ROWCRS)
FAA7  F0 28                       BEQ    RNGERR
FAA9  E0 08                       CPX    #8          ;CHECK FOR 320X1
FAAB  D0 0A                       BNE    RANGE1      ;SPECIAL CASE IT
FAAD  A5 56                       LDA    COLCRS+1
FAAF  F0 13                       BEQ    RNGOK       ;IF HIGH BYTE IS 0, COL IS OK
FAB1  C9 01                       CMP    #1
FAB3  D0 1C                       BNE    RNGERR      ;IF >1, BAD
FAB5  F0 04                       BEQ    RANGE2      ;IF 1, GO CHECK LOW BYTE
FAB7  A5 56               RANGE1: LDA    COLCRS+1    ;FOR OTHERS, NON-ZERO HIGH BYTE IS BAD
FAB9  D0 16                       BNE    RNGERR
FABB  BD 8D FE            RANGE2: LDA    COLUMN,X    ;CHECK LOW BYTE
FABE  C5 55                       CMP    COLCRS
FAC0  90 0F                       BCC    RNGERR
FAC2  F0 0D                       BEQ    RNGERR
FAC4  A9 01               RNGOK:  LDA    #SUCCES     ;SET STATUS OK
FAC6  85 4C                       STA    DSTAT
FAC8  A9 80                       LDA    #BRKABT     ;PREPARE BREAK ABORT STATUS
FACA  A6 11                       LDX    BRKKEY      ;CHECK BREAK KEY FLAG
FACC  85 11                       STA    BRKKEY      ;'CLEAR' BREAK
FACE  F0 06                       BEQ    RNGER2      ;IF BREAK, QUIT IMMEDIATELY AND RETURN TO CI
FAD0  60                          RTS
FAD1  20 D6 F7            RNGERR: JSR    HOME        ;ON RANGE ERROR, BRING CURSOR BACK
FAD4  A9 8D                       LDA    #CRSROR     ;SHOW CURSOR OVERRANGE ERROR
FAD6  85 4C               RNGER2: STA    DSTAT
FAD8  68                  RNGER1: PLA                ;RESTORE STACK (THIS ROUTINE IS ALWAYS 1 LEV
FAD9  68                          PLA                ;AWAY FROM RETURN TO CIO)
FADA  A5 7B                       LDA    SWPFLG      ;IF SWAPPED, SWAP BACK
FADC  10 03                       BPL    RETUR3
FADE  20 B9 FC                    JSR    SWAPA       ;AND DONT DO RETUR1
FAE1  4C 34 F6            RETUR3: JMP    RETUR1      ;RETURN TO CIO
                          ;
                          ;
                          ;
                          ; OFFCRS: RESTORE OLD DATA UNDER CURSOR SO IT CAN BE MOVED
```

```
                     ;
FAE4  A0 00          OFFCRS: LDY    #0
FAE6  A5 5D                  LDA    OLDCHR
FAE8  91 5E                  STA    (OLDADR),Y
FAEA  60                     RTS
                     ;
                     ;
                     ;
                     ; BITMAP ROUTINES:
                     ;
                     ; BITCON: PUT MASK IN BITMSK AND INDEX IN X
                     ; BITPUT: PUT CARRY INTO BITMAP
                     ; BITROL: ROL CARRY INTO BOTTOM OF BITMAP (SCROLL)
                     ; BITSET: SET PROPER BIT
                     ; BITCLR: CLEAR PROPER BIT
                     ; BITGET: RETURN CARRY SET IF BIT IS THERE
                     ; LOGGET: DO BITGET FOR LOGMAP INSTEAD OF TABMAP
                     ;
FAEB  48             BITCON: PHA
FAEC  29 07                  AND    #7
FAEE  AA                     TAX                ;GET MASK
FAEF  BD B9 FE               LDA    MASKTB,X
FAF2  85 6E                  STA    BITMSK
FAF4  68                     PLA                ;PROCESS INDEX
FAF5  4A                     LSR    A
FAF6  4A                     LSR    A
FAF7  4A                     LSR    A
FAF8  AA                     TAX
FAF9  60                     RTS
                     ;
                     ;
FAFA  2E B4 02       BITROL: ROL    LOGMAP+2
FAFD  2E B3 02               ROL    LOGMAP+1
FB00  2E B2 02               ROL    LOGMAP
FB03  60                     RTS
                     ;
                     ;
FB04  90 0C          BITPUT: BCC    BITCLR      ;AND RETURN
                     ; OTHERWISE FALL THROUGH TO BITSET AND RETURN
FB06  20 EB FA       BITSET: JSR    BITCON
FB09  BD A3 02               LDA    TABMAP,X
FB0C  05 6E                  ORA    BITMSK
FB0E  9D A3 02               STA    TABMAP,X
FB11  60                     RTS
                     ;
FB12  20 EB FA       BITCLR: JSR    BITCON
FB15  A5 6E                  LDA    BITMSK
FB17  49 FF                  EOR    #$FF
FB19  3D A3 02               AND    TABMAP,X
FB1C  9D A3 02               STA    TABMAP,X
FB1F  60                     RTS
                     ;
FB20  A5 54          LOGGET: LDA    ROWCRS
FB22  18             LO1GET: CLC
```

```
FB23  69 78       LO2GET: ADC     #120
FB25  20 EB FA    BITGET: JSR     BITCON
FB28  18                  CLC
FB29  BD A3 02            LDA     TABMAP,X
FB2C  25 6E               AND     BITMSK
FB2E  F0 01               BEQ     BITGE1
FB30  38                  SEC
FB31  60          BITGE1: RTS
                  ;
                  ;
                  ;
                  ;
                  ; INATAC: INTERNAL(CHAR) TO ATASCII(ATACHR) CONVERSION
                  ;
FB32  AD FA 02    INATAC: LDA     CHAR
FB35  A4 57               LDY     DINDEX       ;IF GRAPHICS MODES
FB37  C0 03               CPY     #3
FB39  B0 0F               BCS     INATA1       ;THEN DON'T CHANGE CHAR
FB3B  2A                  ROL     A
FB3C  2A                  ROL     A
FB3D  2A                  ROL     A
FB3E  2A                  ROL     A
FB3F  29 03               AND     #3
FB41  AA                  TAX
FB42  AD FA 02            LDA     CHAR
FB45  29 9F               AND     #$9F
FB47  1D FA FE            ORA     INTATA,X
FB4A  8D FB 02    INATA1: STA     ATACHR
FB4D  60                  RTS
                  ;
                  ;
                  ;
                  ; MOVLIN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOADR
                  ;         DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
                  ;         BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40
                  ;
FB4E  A9 02       MOVLIN: LDA     #LINBUF/256 ;SET UP ADRESS=LINBUF=$247
FB50  85 65               STA     ADRESS+1
FB52  A9 47               LDA     #LINBUF.AND.$FF
FB54  85 64               STA     ADRESS
FB56  A0 27               LDY     #39
FB58  B1 66       MOVLI1: LDA     (TOADR),Y    ;SAVE TO DATA
FB5A  85 50               STA     TMPCHR
FB5C  B1 68               LDA     (FRMADR),Y   ;STORE DATA
FB5E  91 66               STA     (TOADR),Y
FB60  A5 50               LDA     TMPCHR
FB62  91 64               STA     (ADRESS),Y
FB64  88                  DEY
FB65  10 F1               BPL     MOVLI1
FB67  A5 65               LDA     ADRESS+1     ;SET UP FRMADR=LAST LINE
FB69  85 69               STA     FRMADR+1
FB6B  A5 64               LDA     ADRESS
FB6D  85 68               STA     FRMADR
FB6F  18                  CLC                  ;ADD 40 TO TOADR
```

```
FB70  A5 66                 LDA    TOADR
FB72  69 28                 ADC    #40
FB74  85 66                 STA    TOADR
FB76  90 02                 BCC    MOVLI2
FB78  E6 67                 INC    TOADR+1
FB7A  60           MOVLI2:  RTS
                   ;
                   ;
                   ;
                   ; EXTEND: EXTEND BIT MAP FROM ROWCRS (EXTEND LOGICAL LINE
                   ;
FB7B  08           EXTEND:  PHP                   ;SAVE CARRY
FB7C  A0 17                 LDY    #23
FB7E  98           EXTEN1:  TYA
FB7F  20 22 FB              JSR    LO1GET
FB82  08                    PHP
FB83  98                    TYA
FB84  18                    CLC
FB85  69 79                 ADC    #121
FB87  28                    PLP
FB88  20 04 FB              JSR    BITPUT
FB8B  88           EXTEN3:  DEY
FB8C  30 04                 BMI    EXTEN4
FB8E  C4 54                 CPY    ROWCRS
FB90  B0 EC                 BCS    EXTEN1
FB92  A5 54        EXTEN4:  LDA    ROWCRS
FB94  18                    CLC
FB95  69 78                 ADC    #120
FB97  28                    PLP
FB98  4C 04 FB              JMP    BITPUT     ;STORE NEW LINE'S BIT AND RETURN
                   ;
                   ;
                   ;
                   ; CLRLIN: CLEAR LINE CURSOR IS ON
                   ;
FB9B  A5 52        CLRLIN:  LDA    LMARGN
FB9D  85 55                 STA    COLCRS
FB9F  20 47 F9              JSR    CONVRT
FBA2  A0 27                 LDY    #39
FBA4  A9 00                 LDA    #0
FBA6  91 64        CLRLI1:  STA    (ADRESS),Y
FBA8  88                    DEY
FBA9  10 FB                 BPL    CLRLI1
FBAB  60                    RTS
                   ;
                   ;
                   ;
                   ;
                   ; SCROLL: SCROLL SCREEN
                   ;
FBAC  20 FA FA     SCROLL:  JSR    BITROL     ;ROLL IN CARRY
FBAF  A5 58                 LDA    SAVMSC     ;SET UP WORKING REGISTERS
FBB1  85 64                 STA    ADRESS
FBB3  A5 59                 LDA    SAVMSC+1
```

```
FBB5  85 65             STA     ADRESS+1
FBB7  A0 28    SCROL1:  LDY     #40          ;LOOP
FBB9  B1 64             LDA     (ADRESS),Y
FBBB  A6 6A             LDX     RAMTOP       ;TEST FOR LAST LINE
FBBD  CA                DEX
FBBE  E4 65             CPX     ADRESS+1
FBC0  D0 08             BNE     SCROL2
FBC2  A2 D7             LDX     #$D7
FBC4  E4 64             CPX     ADRESS
FBC6  B0 02             BCS     SCROL2
FBC8  A9 00             LDA     #0           ;YES SO STORE ZERO DATA FOR THIS ENTIRE LINE
FBCA  A0 00    SCROL2:  LDY     #0
FBCC  91 64             STA     (ADRESS),Y
FBCE  E6 64             INC     ADRESS
FBD0  D0 E5             BNE     SCROL1
FBD2  E6 65             INC     ADRESS+1
FBD4  A5 65             LDA     ADRESS+1
FBD6  C5 6A             CMP     RAMTOP
FBD8  D0 DD             BNE     SCROL1
FBDA  4C DD FB          JMP     DOLCOL       ;AND RETURN
                ;
                ;
                ; DOLCOL: DO LOGICAL COLUMN FROM BITMAP AND COLCRS
                ;
FBDD  A9 00    DOLCOL:  LDA     #0           ;START WITH ZERO
FBDF  85 63             STA     LOGCOL
FBE1  A5 54             LDA     ROWCRS
FBE3  85 51             STA     HOLD1
FBE5  A5 51    DOLCO1:  LDA     HOLD1        ;ADD IN ROW COMPONENT
FBE7  20 22 FB          JSR     LO1GET
FBEA  B0 0C             BCS     DOLCO2       ;FOUND BEGINNING OF LINE
FBEC  A5 63             LDA     LOGCOL       ;ADD 40 AND LOOK BACK ONE
FBEE  18                CLC
FBEF  69 28             ADC     #40
FBF1  85 63             STA     LOGCOL
FBF3  C6 51             DEC     HOLD1        ;UP ONE LINE
FBF5  4C E5 FB          JMP     DOLCO1
FBF8  18       DOLCO2:  CLC                  ;ADD IN COLCRS
FBF9  A5 63             LDA     LOGCOL
FBFB  65 55             ADC     COLCRS
FBFD  85 63             STA     LOGCOL
FBFF  60                RTS
                ;
                ;
                ;
                ; DOBUFC: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
                ;         BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
                ;
FC00  20 9D FC DOBUFC:  JSR     PHACRS
FC03  A5 63             LDA     LOGCOL
FC05  48                PHA
FC06  A5 6C             LDA     BUFSTR       ;START
FC08  85 54             STA     ROWCRS
FC0A  A5 6D             LDA     BUFSTR+1
```

```
FC0C  85 55              STA     COLCRS
FC0E  A9 01              LDA     #1
FC10  85 6B              STA     BUFCNT
FC12  A2 17     DOBUF1:  LDX     #23        ;NORMAL
FC14  A5 7B              LDA     SWPFLG     ;IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREE
FC16  10 02              BPL     DOB1
FC18  A2 03              LDX     #3
FC1A  E4 54     DOB1:    CPX     ROWCRS     ;TEST IF CRSR IS AT LAST SCREEN POSITION
FC1C  D0 0B              BNE     DOBU1A
FC1E  A5 55              LDA     COLCRS
FC20  C5 53              CMP     RMARGN
FC22  D0 05              BNE     DOBU1A
FC24  E6 6B              INC     BUFCNT     ;YES, SO FAKE INCRSR TO AVOID SCROLLING
FC26  4C 39 FC           JMP     DOBUF2
FC29  20 D4 F9  DOBU1A:  JSR     INCRSB
FC2C  E6 6B              INC     BUFCNT
FC2E  A5 63              LDA     LOGCOL
FC30  C5 52              CMP     LMARGN
FC32  D0 DE              BNE     DOBUF1     ;NOT YET EOL
FC34  C6 54              DEC     ROWCRS     ;BACK UP ONE INCRSR
FC36  20 99 F7           JSR     CRSRLF
FC39  20 A2 F5  DOBUF2:  JSR     GETPLT     ;TEST CURRENT COLUMN FOR NON-ZERO DATA
FC3C  D0 17              BNE     DOBUF4     ;QUIT IF NON-ZERO
FC3E  C6 6B              DEC     BUFCNT     ;DECREMENT COUNTER
FC40  A5 63              LDA     LOGCOL     ;BEGINNING OF LOGICAL LINE YET?
FC42  C5 52              CMP     LMARGN
FC44  F0 0F              BEQ     DOBUF4     ;YES, SO QUIT
FC46  20 99 F7           JSR     CRSRLF     ;BACK UP CURSOR
FC49  A5 55              LDA     COLCRS     ;IF LOGCOL=RMARGN, GO UP 1 ROW
FC4B  C5 53              CMP     RMARGN
FC4D  D0 02              BNE     DOBUF3
FC4F  C6 54              DEC     ROWCRS
FC51  A5 6B     DOBUF3:  LDA     BUFCNT
FC53  D0 E4              BNE     DOBUF2     ;LOOP UNLESS BUFCNT JUST WENT TO ZERO
FC55  68        DOBUF4:  PLA
FC56  85 63              STA     LOGCOL
FC58  20 A8 FC           JSR     PLACRS
FC5B  60                 RTS
                ;
                ;
                ;
                ;
                ; STRBEG: MOVE BUFSTR TO BEGINNING OF LOGICAL LINE.
                ;
FC5C  20 DD FB  STRBEG:  JSR     DOLCOL     ;USE DOLCOL TO POINT HOLD1 AT BOL
FC5F  A5 51              LDA     HOLD1
FC61  85 6C              STA     BUFSTR
FC63  A5 52              LDA     LMARGN
FC65  85 6D              STA     BUFSTR+1
FC67  60                 RTS
                ;
                ;
                ;
                ;
```

```
                        ;
                        ; DELTIM: TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
                        ;
FC68  A5 63             DELTIA: LDA     LOGCOL          ;IF LOGCOL<>LMARGN
FC6A  C5 52                     CMP     LMARGN          ;THEN DONT MOVE UP ONE
FC6C  D0 02                     BNE     DELTIB          ;LINE BEFORE TESTING DELTIM
FC6E  C6 54                     DEC     ROWCRS
FC70  20 DD FB          DELTIB: JSR     DOLCOL
FC73  A5 63             DELTIM: LDA     LOGCOL          ;TEST FOR EXTENSION
FC75  C5 52                     CMP     LMARGN
FC77  F0 13                     BEQ     DELTI3          ;NO
FC79  20 47 F9                  JSR     CONVRT
FC7C  A5 53                     LDA     RMARGN          ;SET UP COUNT
FC7E  38                        SEC
FC7F  E5 52                     SBC     LMARGN
FC81  A8                        TAY
FC82  B1 64             DELTI1: LDA     (ADRESS),Y
FC84  D0 06                     BNE     DELTI3          ;FOUND A NON-0 SO QUIT AND RETURN
FC86  88                        DEY
FC87  10 F9                     BPL     DELTI1
FC89  4C DB F8          DELTI2: JMP     DELLIB          ;DELETE A LINE AND RETURN
FC8C  60                DELTI3: RTS
                        ;
                        ;
                        ;
                        ; TSTCTL: SEARCH CNTRLS TABLE TO SEE IF ATACHR IS A CNTL CHAR
                        ;
FC8D  A2 2D             TSTCTL: LDX     #45             ;PREPARE TO SEARCH TABLE
FC8F  BD C6 FE          TSTCT1: LDA     CNTRLS,X
FC92  CD FB 02                  CMP     ATACHR
FC95  F0 05                     BEQ     TSTCT2
FC97  CA                        DEX
FC98  CA                        DEX
FC99  CA                        DEX
FC9A  10 F3                     BPL     TSTCT1
FC9C  60                TSTCT2: RTS
                        ;
                        ;
                        ;
                        ; PUSH ROWCRS,COLCRS AND COLCRS+1
                        ;
FC9D  A2 02             PHACRS: LDX     #2
FC9F  B5 54             PHACR1: LDA     ROWCRS,X
FCA1  9D B8 02                  STA     TMPROW,X
FCA4  CA                        DEX
FCA5  10 F8                     BPL     PHACR1
FCA7  60                        RTS
                        ;
                        ;
                        ; PULL COLCRS+1,COLCRS AND ROWCRS
                        ;
FCA8  A2 02             PLACRS: LDX     #2
FCAA  BD B8 02          PLACR1: LDA     TMPROW,X
FCAD  95 54                     STA     ROWCRS,X
```

```
FCAF  CA                          DEX
FCB0  10 F8                       BPL     PLACR1
FCB2  60                          RTS
                        ;
                        ;
                        ;
                        ; SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS
                        ;
FCB3  20 B9 FC          SWAP:     JSR     SWAPA         ;THIS ENTRY POINT DOES RETUR1
FCB6  4C 34 F6                    JMP     RETUR1
FCB9  AD BF 02          SWAPA:    LDA     BOTSCR
FCBC  C9 18                       CMP     #24
FCBE  F0 17                       BEQ     SWAP3
FCC0  A2 0B                       LDX     #11
FCC2  B5 54             SWAP1:    LDA     ROWCRS,X
FCC4  48                          PHA
FCC5  BD 90 02                    LDA     TXTROW,X
FCC8  95 54                       STA     ROWCRS,X
FCCA  68                          PLA
FCCB  9D 90 02                    STA     TXTROW,X
FCCE  CA                          DEX
FCCF  10 F1                       BPL     SWAP1
FCD1  A5 7B                       LDA     SWPFLG
FCD3  49 FF                       EOR     #$FF
FCD5  85 7B                       STA     SWPFLG
FCD7  60                SWAP3:    RTS
                        ;
                        ;
                        ; CLICK: MAKE CLICK THROUGH KEYBOARD SPEAKER
                        ;
FCD8  A2 7F             CLICK:    LDX     #$7F
FCDA  8E 1F D0          CLICK1:   STX     CONSOL
FCDD  8E 0A D4                    STX     WSYNC
FCE0  CA                          DEX
FCE1  10 F7                       BPL     CLICK1
FCE3  60                          RTS
                        ;
                        ;
                        ; COLCR: PUTS EITHER 0 OR LMARGN INTO COLCRS BASED ON MODE AND SWPFLG
                        ;
FCE4  A9 00             COLCR:    LDA     #0
FCE6  A6 7B                       LDX     SWPFLG
FCE8  D0 04                       BNE     COLCR1
FCEA  A6 57                       LDX     DINDEX
FCEC  D0 02                       BNE     COLCR2
FCEE  A5 52             COLCR1:   LDA     LMARGN
FCF0  85 55             COLCR2:   STA     COLCRS
FCF2  60                          RTS
                        ;
                        ;
                        ; PUTMSC: PUT SAVMSC INTO ADRESS
                        ;
FCF3  A5 58             PUTMSC:   LDA     SAVMSC        ;SET UP ADDRESS
FCF5  85 64                       STA     ADRESS
```

```
     5379  FCF7  A5 59                       LDA     SAVMSC+1
     5380  FCF9  85 65                       STA     ADRESS+1
     5381  FCFB  60                          RTS
     5382                            ;
```

```
                                  .PAGE
                           ;
                           ;
                           ; DRAW -- DRAW A LINE FROM OLDROW,OLDCOL TO NEWROW,NEWCOL
                           ; (THE AL MILLER METHOD FROM BASKETBALL)
FCFC  A2 00                DRAW:   LDX     #0
FCFE  A5 22                        LDA     ICCOMZ      ;TEST COMMAND: $11=DRAW $12=FILL
FD00  C9 11                        CMP     #$11
FD02  F0 08                        BEQ     DRAWA
FD04  C9 12                        CMP     #$12        ;TEST FILL
FD06  F0 03                        BEQ     DRAWB       ;YES
FD08  A0 84                        LDY     #NVALID     ;NO, SO RETURN INVALID COMMAND
FD0A  60                           RTS
FD0B  E8                   DRAWB:  INX
FD0C  8E B7 02             DRAWA:  STX     FILFLG
FD0F  A5 54                        LDA     ROWCRS      ;PUT CURSOR INTO NEWROW,NEWCOL
FD11  85 60                        STA     NEWROW
FD13  A5 55                        LDA     COLCRS
FD15  85 61                        STA     NEWCOL
FD17  A5 56                        LDA     COLCRS+1
FD19  85 62                        STA     NEWCOL+1
FD1B  A9 01                        LDA     #1
FD1D  85 79                        STA     ROWINC      ;SET UP INITIAL DIRECTIONS
FD1F  85 7A                        STA     COLINC
FD21  38                           SEC
FD22  A5 60                        LDA     NEWROW      ;DETERMINE DELTA ROW
FD24  E5 5A                        SBC     OLDROW
FD26  85 76                        STA     DELTAR
FD28  B0 0D                        BCS     DRAW1       ;DO DIRECTION AND ABSOLUTE VALUE
FD2A  A9 FF                        LDA     #$FF        ;BORROW WAS ATTEMPTED
FD2C  85 79                        STA     ROWINC      ;SET DIRECTION=DOWN
FD2E  A5 76                        LDA     DELTAR
FD30  49 FF                        EOR     #$FF        ;DELTAR = |DELTAR|
FD32  18                           CLC
FD33  69 01                        ADC     #1
FD35  85 76                        STA     DELTAR
FD37  38                   DRAW1:  SEC
FD38  A5 61                        LDA     NEWCOL      ;NOW DELTA COLUMN
FD3A  E5 5B                        SBC     OLDCOL
FD3C  85 77                        STA     DELTAC
FD3E  A5 62                        LDA     NEWCOL+1    ;TWO-BYTE QUANTITY
FD40  E5 5C                        SBC     OLDCOL+1
FD42  85 78                        STA     DELTAC+1
FD44  B0 16                        BCS     DRAW2       ;DIRECTION AND ABSOLUTE VALUE
FD46  A9 FF                        LDA     #$FF        ;BORROW WAS ATTEMPTED
FD48  85 7A                        STA     COLINC      ;SET DIRECTION = LEFT
FD4A  A5 77                        LDA     DELTAC
FD4C  49 FF                        EOR     #$FF        ;DELTAC = |DELTAC|
FD4E  85 77                        STA     DELTAC
FD50  A5 78                        LDA     DELTAC+1
FD52  49 FF                        EOR     #$FF
FD54  85 78                        STA     DELTAC+1
FD56  E6 77                        INC     DELTAC      ;ADD ONE FOR TWOS COMPLEMENT
FD58  D0 02                        BNE     DRAW2
```

```
FD5A  E6 78              INC    DELTAC+1
FD5C  A2 02     DRAW2:   LDX    #2          ;ZERO RAM FOR DRAW LOOP
FD5E  A0 00              LDY    #0
FD60  84 73              STY    COLAC+1
FD62  98        DRAW3A:  TYA
FD63  95 70              STA    ROWAC,X
FD65  B5 5A              LDA    OLDROW,X
FD67  95 54              STA    ROWCRS,X
FD69  CA                 DEX
FD6A  10 F6              BPL    DRAW3A
FD6C  A5 77              LDA    DELTAC      ;FIND LARGER ONE (ROW OR COL)
                ;        STA    COUNTR      (PREPARE COUNTR AND ENDPT)
                ;        STA    ENDPT
FD6E  E8                 INX                ;MAKE X 0
FD6F  A8                 TAY
FD70  A5 78              LDA    DELTAC+1
FD72  85 7F              STA    COUNTR+1
FD74  85 75              STA    ENDPT+1
FD76  D0 0B              BNE    DRAW3       ;AUTOMATICALLY LARGER IF MSD>0
FD78  A5 77              LDA    DELTAC
FD7A  C5 76              CMP    DELTAR      ;LOW COL >LOW ROW?
FD7C  B0 05              BCS    DRAW3       ;YES
FD7E  A5 76              LDA    DELTAR
FD80  A2 02              LDX    #2
FD82  A8                 TAY
FD83  98        DRAW3:   TYA                ;PUT IN INITIAL CONDITIONS
FD84  85 7E              STA    COUNTR
FD86  85 74              STA    ENDPT
FD88  48                 PHA                ;SAVE AC
FD89  A5 75              LDA    ENDPT+1     ;PUT LSB OF HIGH BYTE
FD8B  4A                 LSR    A           ;INTO CARRY
FD8C  68                 PLA                ;RESTORE AC
FD8D  6A                 ROR    A           ;ROR THE 9 BIT ACUMULATOR
FD8E  95 70              STA    ROWAC,X
FD90  A5 7E     DRAW4A:  LDA    COUNTR      ;TEST ZERO
FD92  05 7F              ORA    COUNTR+1
FD94  D0 03              BNE    DRAW11      ;IF COUNTER IS ZERO, LEAVE DRAW
FD96  4C 42 FE           JMP    DRAW10
FD99  18        DRAW11:  CLC                ;ADD ROW TO ROWAC (PLOT LOOP)
FD9A  A5 70              LDA    ROWAC
FD9C  65 76              ADC    DELTAR
FD9E  85 70              STA    ROWAC
FDA0  90 02              BCC    DRAW5
FDA2  E6 71              INC    ROWAC+1
FDA4  A5 71     DRAW5:   LDA    ROWAC+1     ;COMPARE ROW TO ENDPOINT
FDA6  C5 75              CMP    ENDPT+1     ;IF HIGH BYTE OF ROW IS .LT. HIGH
FDA8  90 14              BCC    DRAW6       ;BYTE OF ENDPT, BLT TO COLUMN
FDAA  D0 06              BNE    DRAW5A
FDAC  A5 70              LDA    ROWAC
FDAE  C5 74              CMP    ENDPT       ;LOW BYTE
FDB0  90 0C              BCC    DRAW6       ;ALSO BLT
FDB2  18        DRAW5A:  CLC                ;GE SO MOVE POINT
FDB3  A5 54              LDA    ROWCRS
FDB5  65 79              ADC    ROWINC
```

```
FDB7  85 54                 STA    ROWCRS
FDB9  A2 00                 LDX    #0           ;AND SUBTRACT ENDPT FROM ROWAC
FDBB  20 7A FA              JSR    SUBEND
FDBE  18          DRAW6:    CLC                 ;DO SAME FOR COLUMN (DOUBLE BYTE ADD)
FDBF  A5 72                 LDA    COLAC        ;ADD
FDC1  65 77                 ADC    DELTAC
FDC3  85 72                 STA    COLAC
FDC5  A5 73                 LDA    COLAC+1
FDC7  65 78                 ADC    DELTAC+1
FDC9  85 73                 STA    COLAC+1
FDCB  C5 75                 CMP    ENDPT+1      ;COMPARE HIGH BYTE
FDCD  90 27                 BCC    DRAW8
FDCF  D0 06                 BNE    DRAW6A
FDD1  A5 72                 LDA    COLAC        ;COMPARE LOW BYTE
FDD3  C5 74                 CMP    ENDPT
FDD5  90 1F                 BCC    DRAW8
FDD7  24 7A       DRAW6A:   BIT    COLINC       ;+ OR - ?
FDD9  10 10                 BPL    DRAW6B
FDDB  C6 55                 DEC    COLCRS       ;DO DOUBLE BYTE DECREMENT
FDDD  A5 55                 LDA    COLCRS
FDDF  C9 FF                 CMP    #$FF
FDE1  D0 0E                 BNE    DRAW7
FDE3  A5 56                 LDA    COLCRS+1
FDE5  F0 0A                 BEQ    DRAW7        ;DON'T DEC IF ZERO
FDE7  C6 56                 DEC    COLCRS+1
FDE9  10 06                 BPL    DRAW7        ;(UNCONDITIONAL)
FDEB  E6 55       DRAW6B:   INC    COLCRS       ;DO DOUBLE BYTE INCREMENT
FDED  D0 02                 BNE    DRAW7
FDEF  E6 56                 INC    COLCRS+1
FDF1  A2 02       DRAW7:    LDX    #2           ;AND SUBTRACT ENDPT FROM COLAC
FDF3  20 7A FA              JSR    SUBEND
FDF6  20 96 FA    DRAW8:    JSR    RANGE
FDF9  20 E0 F5              JSR    OUTPLT       ;PLOT POINT
FDFC  AD B7 02              LDA    FILFLG       ;TEST RIGHT FILL
FDFF  F0 2F                 BEQ    DRAW9
FE01  20 9D FC              JSR    PHACRS
FE04  AD FB 02              LDA    ATACHR
FE07  8D BC 02              STA    HOLD4
FE0A  A5 54       DRAW8A:   LDA    ROWCRS       ;SAVE ROW IN CASE OF CR
FE0C  48                    PHA
FE0D  20 DC F9              JSR    INCRSA       ;POSITION CURSOR ONE PAST DOT
FE10  68                    PLA                 ;RESTORE ROWCRS
FE11  85 54                 STA    ROWCRS
FE13  20 96 FA    DRAW8C:   JSR    RANGE
FE16  20 A2 F5              JSR    GETPLT       ;GET DATA
FE19  D0 0C                 BNE    DRAW8B       ;STOP IF NON-ZERO DATA IS ENCOUNTERED
FE1B  AD FD 02              LDA    FILDAT       ;FILL DATA
FE1E  8D FB 02              STA    ATACHR
FE21  20 E0 F5              JSR    OUTPLT       ;DRAW IT
FE24  4C 0A FE              JMP    DRAW8A       ;LOOP
FE27  AD BC 02    DRAW8B:   LDA    HOLD4
FE2A  8D FB 02              STA    ATACHR
FE2D  20 A8 FC              JSR    PLACRS
FE30  38          DRAW9:    SEC                 ;DO DOUBLE BYTE SUBTRACT
```

```
    5545  FE31  A5 7E                   LDA     COUNTR
    5546  FE33  E9 01                   SBC     #1
    5547  FE35  85 7E                   STA     COUNTR
    5548  FE37  A5 7F                   LDA     COUNTR+1
    5549  FE39  E9 00                   SBC     #0
    5550  FE3B  85 7F                   STA     COUNTR+1
    5551  FE3D  30 03                   BMI     DRAW10
    5552  FE3F  4C 90 FD                JMP     DRAW4A
    5553  FE42  4C 34 F6        DRAW10: JMP     RETUR1
```

```
                                      .PAGE
                                 ;
                                 ;
                                 ; TABLES
                                 ;
                                 ;
                                 ; MEMORY ALLOCATION
                                 ;
FE45  18 10 0A 0A                ALOCAT: .BYTE   24,16,10,10,16,28,52,100,196,196,196,196
FE49  10 1C 34 64
FE4D  C4 C4 C4 C4
                                 ;
                                 ;
                                 ; NUMBER OF DISPLAY LIST ENTRIES
                                 ;
FE51  17 17 0B 17                NUMDLE: .BYTE   23,23,11,23,47,47,95,95,97,97,97,97
FE55  2F 2F 5F 5F
FE59  61 61 61 61
FE5D  13 13 09 13                MXDMDE: .BYTE   19,19,9,19,39,39,79,79,65,65,65,65 ;(EXT OF NUMDLE)
FE61  27 27 4F 4F
FE65  41 41 41 41
                                 ;
                                 ;
                                 ; ANTIC CODE FROM INTERNAL MODE CONVERSION TABLE
                                 ;
                                 ;   INTERNAL         ANTIC CODE            DESCRIPTION
                                 ;       0                2              40X2X8  CHARACTERS
                                 ;       1                6              20X5X8      ""
                                 ;       2                7              20X5X16     ""
                                 ;       3                8              40X4X8  GRAPHICS
                                 ;       4                9              80X2X4     ""
                                 ;       5                A              80X4X4     ""
                                 ;       6                B              160X2X2    ""
                                 ;       7                D              160X4X2    ""
                                 ;       8                F              320X2X1    ""
                                 ;       9                SAME AS 8 BUT GTIA 'LUM' MODE
                                 ;       10               SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE
                                 ;       11               SAME AS 8 BUT GTIA 'COLOR' MODE
                                 ;
FE69  02 06 07 08                ANCONV: .BYTE   2,6,7,8,9,$A,$B,$D,$F,$F,$F,$F ;ZEROS FOR RANGE TEST IN
FE6D  09 0A 0B 0D
FE71  0F 0F 0F 0F
                                 ;
                                 ;
                                 ; PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF
                                 ; CROSSING A 256 BYTE PAGE BOUNDARY
                                 ;
FE75  00 00 00 00                PAGETB: .BYTE   0,0,0,0,0,0,0,1,1,1,1,1
FE79  00 00 00 01
FE7D  01 01 01 01
                                 ;
                                 ;
                                 ; THIS IS THE NUMBER OF LEFT SHIFTS NEEDED TO MULTIPLY
                                 ; COLCRS BY 10,20, OR 40.  (ROWCRS*10)/(2**DHLINE)
```

```
                                      ;
      FE81  02 01 01 00       DHLINE: .BYTE   2,1,1,0,0,1,1,2,2,2,2,2
      FE85  00 01 01 02
      FE89  02 02 02 02
                                      ;
                                      ;
                                      ; COLUMN: NUMBER OF COLUMNS
                                      ;
      FE8D  28 14 14 28       COLUMN: .BYTE   40,20,20,40,80,80,160,160,64,80,80,80 ;MODE 8 IS SPECIAL
      FE91  50 50 A0 A0
      FE95  40 50 50 50
                                      ;
                                      ;
                                      ;
                                      ; NOROWS: NUMBER OF ROWS
                                      ;
      FE99  18 18 0C 18       NOROWS: .BYTE   24,24,12,24,48,48,96,96,192,192,192,192
      FE9D  30 30 60 60
      FEA1  C0 C0 C0 C0
                                      ;
                                      ;
                                      ;
                                      ;
                                      ; DIV2TB: HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES
                                      ;
      FEA5  00 00 00 02       DIV2TB: .BYTE   0,0,0,2,3,2,3,2,3,1,1,1
      FEA9  03 02 03 02
      FEAD  03 01 01 01
                                      ;
                                      ;
                                      ; DMASKT: DISPLAY MASK TABLE
                                      ;
      FEB1  00 FF F0 0F       DMASKT: .BYTE   $00,$FF,$F0,$0F
      FEB5  C0 30 0C 03               .BYTE   $C0,$30,$0C,$03
                                      ;
                                      ; MASKTB: BIT MASK.   (ALSO PART OF DMASKTB! DO NOT SEPARATE)
                                      ;
      FEB9  80 40 20 10       MASKTB: .BYTE   $80,$40,$20,$10,$08,$04,$02,$01
      FEBD  08 04 02 01
                                      ;
                                      ;
                                      ;
                                      ;
      FEC1  28 CA 94 46       COLRTB: .BYTE   $28,$CA,$94,$46,$00
      FEC5  00
                                      ;
                                      ;
                                      ;
                                      ;
                                      ; CNTRLS: CONTROL CODES AND THEIR DISPLACEMENTS INTO THE
                                      ;         CONTROL CHARACTER PROCESSORS
                                      ;
      FEC6  1B                CNTRLS: .BYTE   $1B
      FEC7  79 F7                     .WORD   ESCAPE
```

```
FEC9  1C                            .BYTE   $1C
FECA  7F F7                         .WORD   CRSRUP
FECC  1D                            .BYTE   $1D
FECD  8C F7                         .WORD   CRSRDN
FECF  1E                            .BYTE   $1E
FED0  99 F7                         .WORD   CRSRLF
FED2  1F                            .BYTE   $1F
FED3  AA F7                         .WORD   CRSRRT
FED5  7D                            .BYTE   $7D
FED6  B9 F7                         .WORD   CLRSCR
FED8  7E                            .BYTE   $7E
FED9  E6 F7                         .WORD   BS
FEDB  7F                            .BYTE   $7F
FEDC  10 F8                         .WORD   TAB
FEDE  9B                            .BYTE   $9B
FEDF  30 FA                         .WORD   DOCRWS
FEE1  9C                            .BYTE   $9C
FEE2  D4 F8                         .WORD   DELLIN
FEE4  9D                            .BYTE   $9D
FEE5  A4 F8                         .WORD   INSLIN
FEE7  9E                            .BYTE   $9E
FEE8  32 F8                         .WORD   CLRTAB
FEEA  9F                            .BYTE   $9F
FEEB  2D F8                         .WORD   SETTAB
FEED  FD                            .BYTE   $FD
FEEE  0A F9                         .WORD   BELL
FEF0  FE                            .BYTE   $FE
FEF1  6D F8                         .WORD   DELCHR
FEF3  FF                            .BYTE   $FF
FEF4  37 F8                         .WORD   INSCHR
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; ATAINT: ATASCI TO INTERNAL TABLE
                        ;
FEF6  40 00 20 60       ATAINT: .BYTE   $40,$00,$20,$60
                        ;
                        ;
                        ; INTATA: INTERNAL TO ATASCI TABLE
                        ;
FEFA  20 40 00 60       INTATA: .BYTE   $20,$40,$00,$60
                        ;
                        ;
                        ; ATASCI: ATASCII CONVERSION TABLE
                        ;
FEFE  6C 6A 3B 80       ATASCI: .BYTE   $6C,$6A,$3B,$80,$80,$6B,$2B,$2A ;LOWER CASE
FF02  80 6B 2B 2A
FF06  6F 80 70 75               .BYTE   $6F,$80,$70,$75,$9B,$69,$2D,$3D
FF0A  9B 69 2D 3D
FF0E
FF0E  76 80 63 80               .BYTE   $76,$80,$63,$80,$80,$62,$78,$7A
FF12  80 62 78 7A
```

```
FF16  34 80 33 36        .BYTE   $34,$80,$33,$36,$1B,$35,$32,$31
FF1A  1B 35 32 31
FF1E
FF1E  2C 20 2E 6E        .BYTE   $2C,$20,$2E,$6E,$80,$6D,$2F,$81
FF22  80 6D 2F 81
FF26  72 80 65 79        .BYTE   $72,$80,$65,$79,$7F,$74,$77,$71
FF2A  7F 74 77 71
FF2E
FF2E  39 80 30 37        .BYTE   $39,$80,$30,$37,$7E,$38,$3C,$3E
FF32  7E 38 3C 3E
FF36  66 68 64 80        .BYTE   $66,$68,$64,$80,$82,$67,$73,$61
FF3A  82 67 73 61
FF3E
FF3E
FF3E  4C 4A 3A 80        .BYTE   $4C,$4A,$3A,$80,$80,$4B,$5C,$5E ;UPPER CASE
FF42  80 4B 5C 5E
FF46  4F 80 50 55        .BYTE   $4F,$80,$50,$55,$9B,$49,$5F,$7C
FF4A  9B 49 5F 7C
FF4E
FF4E  56 80 43 80        .BYTE   $56,$80,$43,$80,$80,$42,$58,$5A
FF52  80 42 58 5A
FF56  24 80 23 26        .BYTE   $24,$80,$23,$26,$1B,$25,$22,$21
FF5A  1B 25 22 21
FF5E
FF5E  5B 20 5D 4E        .BYTE   $5B,$20,$5D,$4E,$80,$4D,$3F,$81
FF62  80 4D 3F 81
FF66  52 80 45 59        .BYTE   $52,$80,$45,$59,$9F,$54,$57,$51
FF6A  9F 54 57 51
FF6E
FF6E  28 80 29 27        .BYTE   $28,$80,$29,$27,$9C,$40,$7D,$9D
FF72  9C 40 7D 9D
FF76  46 48 44 80        .BYTE   $46,$48,$44,$80,$83,$47,$53,$41
FF7A  83 47 53 41
FF7E
FF7E
FF7E  0C 0A 7B 80        .BYTE   $0C,$0A,$7B,$80,$80,$0B,$1E,$1F ;CONTROL
FF82  80 0B 1E 1F
FF86  0F 80 10 15        .BYTE   $0F,$80,$10,$15,$9B,$09,$1C,$1D
FF8A  9B 09 1C 1D
FF8E
FF8E  16 80 03 80        .BYTE   $16,$80,$03,$80,$80,$02,$18,$1A
FF92  80 02 18 1A
FF96  80 80 85 80        .BYTE   $80,$80,$85,$80,$1B,$80,$FD,$80
FF9A  1B 80 FD 80
FF9E
FF9E  00 20 60 0E        .BYTE   $00,$20,$60,$0E,$80,$0D,$80,$81
FFA2  80 0D 80 81
FFA6  12 80 05 19        .BYTE   $12,$80,$05,$19,$9E,$14,$17,$11
FFAA  9E 14 17 11
FFAE
FFAE  80 80 80 80        .BYTE   $80,$80,$80,$80,$FE,$80,$7D,$FF
FFB2  FE 80 7D FF
FFB6  06 08 04 80        .BYTE   $06,$08,$04,$80,$84,$07,$13,$01
FFBA  84 07 13 01
```

```
                                ;
                                ;
                                ;
                                ;
                                ;
      FFBE  AD 09 D2            PIRQ5:  LDA     KBCODE
      FFC1  CD F2 02                    CMP     CH1         ;TEST AGAINST LAST KEY PRESSED
      FFC4  D0 05                       BNE     PIRQ3       ;IF NOT, GO PROCESS KEY
      FFC6  AD F1 02                    LDA     KEYDEL      ;IF KEY DELAY BYTE > 0
      FFC9  D0 20                       BNE     PIRQ4       ;IGNORE KEY AS BOUNCE
      FFCB  AD 09 D2            PIRQ3:  LDA     KBCODE      ;RESTORE AC
      FFCE  C9 9F                       CMP     #CNTL1      ;TEST CONTROL 1 (SSFLAG)
      FFD0  D0 0A                       BNE     PIRQ1
      FFD2  AD FF 02                    LDA     SSFLAG
      FFD5  49 FF                       EOR     #$FF
      FFD7  8D FF 02                    STA     SSFLAG
      FFDA  B0 0F                       BCS     PIRQ4       ;(UNCONDITIONAL) MAKE ^1 INVISIBLE
      FFDC  8D FC 02            PIRQ1:  STA     CH
      FFDF  8D F2 02                    STA     CH1
      FFE2  A9 03                       LDA     #3
      FFE4  8D F1 02                    STA     KEYDEL      ;INITIALIZE KEY DELAY FOR DEBOUNCE
      FFE7  A9 00                       LDA     #0          ;CLEAR COLOR SHIFT BYTE
      FFE9  85 4D                       STA     ATRACT
      FFEB  A9 30               PIRQ4:  LDA     #$30
      FFED  8D 2B 02                    STA     SRTIMR
      FFF0  68                  PIRQ2:  PLA
      FFF1  40                          RTI
                                ;
                                ;
      FFF2  FF FF FF FF                 .BYTE   $FF,$FF,$FF,$FF,$FF,$FF
      FFF6  FF FF
                                ;
      FFF8                      CRNTPC  =*
                                        *=$14
      0014  00                  KBDSPR: .BYTE   $FFF8-CRNTPC ;^GDISPLC IS TOO LONG
      0015                              .END
```

ASSEMBLY ERRORS = 0

CROSS REFERENCE

LABEL	VALUE	REFERENCE							
ACK	0041	-1632	1902						
ACKREC	E9C6	1785	-1794						
ADDCOR	030E	-498	2515	2519	2523				
ADJ1	ED0C	2533	-2537						
ADJUST	ED04	2470	2473	-2532					
ADRESS	0064	-269	4120	4145	4171	4173	4182	4183	4184
		4186	4252	4254	4258	4260	4263	4265	4296
		4355	4357	4426	4443	4445	4570	4573	4574
		4653	4655	4681	4686	4748	4755	4758	4760
		4762	4766	4834	4835	4838	4839	5092	5094
		5101	5104	5106	5149	5161	5163	5165	5168
		5171	5175	5176	5178	5179	5287	5378	5380
ADRTAB	E6FE	-1292	1327						
AFP	D800	-574							
ALLPOT	D208	-659							
ALLSEC	F30E	3886	-3891						
ALOCAT	FE45	4146	-5562						
ANCONV	FE69	4139	-5593						
ANTIC	D400	-735	736	737	738	739	740	741	742
		743	744	745	746	747	748	749	750
APPEND	0001	-111							
APPMHI	000E	-179	4761	4765					
ASCC01	F705	4478	-4480						
ASCZER	0030	-764	1193						
ATACHR	02FB	-469	4308	4311	4316	4333	4372	4394	4417
		4421	4428	4481	4515	4526	4528	4606	5082
		5300	5527	5538	5542				
ATAINT	FEF6	4343	-5699						
ATAN	BE43	-602							
ATASCI	FEFE	4480	-5709						
ATEOF	F00B	3250	-3256						
ATRACT	004D	-248	1340	1395	1401	1403	5792		
AUDC1	D201	-667	2343	2353	2378				
AUDC2	D203	-669	2354						
AUDC3	D205	-671	2349						
AUDC4	D207	-673							
AUDCTL	D208	-674	2336						
AUDF1	D200	-666	2297						
AUDF2	D202	-668	2295						
AUDF3	D204	-670	1749	2203	2607	3200			
AUDF4	D206	-672	1751	2205	2609	3202			
B192HI	0000	-1641	1750						
B192LO	0028	-1640	1748						
B600HI	0005	-1643	2204						
B600LO	00CC	-1642	2202						
BAD	EA63	1919	-1925						
BADCOM	E9BF	1782	-1788	1809					
BADDSK	F306	-3887	3923	3929					
BADIOC	0086	-142	826						

BADMOD	0091	-154	4141						
BADST	EE9E	2914	2917	2963	-2965				
BEEP	F05B	3164	3197	-3314					
BEEP1	F05A	-3315	3349						
BEGIN	ED10	2255	-2558	2570					
BELL	F90A	4402	-4726	5687					
BELL1	F90C	-4727	4729						
BFENHI	0035	-224	1774	1886	1985	2147	2184		
BFENLO	0034	-223	1771	1883	1983	2145	2179		
BITCLR	FB12	4624	5038	-5046					
BITCON	FAEB	-5019	5040	5046	5056				
BITGE1	FB31	5060	-5062						
BITGET	FB25	4618	4890	-5056					
BITMSK	006E	-276	5023	5042	5047	5059			
BITPUT	FB04	4711	-5038	5129	5138				
BITROL	FAFA	-5032	5159						
BITSET	FB06	4622	-5040						
BLACKB	F22A	-3740	3741						
BLFILL	EEC1	2992	-2995	3015					
BLIM	028A	-398	3183	3235	3254				
BLKB2	F230	3740	-3742						
BLKBDV	E471	-77	3483	3601	3603				
BLOAD	F36C	3928	-3933						
BOOT	F2CF	3710	-3862						
BOOTAD	0242	-360	3896	3898	3934	3937			
BOOT?	0009	-176	3864	3931	3991	4007			
BOTSCR	02BF	-429	4193	4201	4542	4548	4664	4953	5335
BPTR	003D	-234	3182	3225	3234	3238	3246	3263	3265
		3274	3293						
BRKABT	0080	-136	2637	3188	4466	4983			
BRKKEY	0011	-184	1337	1957	2064	2558	2580	2643	3189
		3220	3831	4467	4509	4984	4985		
BRKKY	0236	-345	1292	3833	3835				
BRKKY2	E754	-1336	3832	3834					
BROKE	EDA0	1959	2066	2560	2581	-2632			
BS	F7E6	-4591	5673						
BS1	F80D	4593	-4608						
BS2	F805	4601	4603	-4605					
BS3	F7F5	4596	-4598						
BSA	F7EC	-4594							
BUFADR	0015	-187	2829	2838	2842	2863	2865		
BUFCNT	006B	-274	4386	4410	4412	5219	5229	5232	5240
		5249							
BUFFH	0004	-3421	3422	3883					
BUFFL	0000	-3422	3881						
BUFFUL	EECB	2989	-3000						
BUFRFL	0038	-227	2054	2111	2160				
BUFRHI	0033	-222	1773	1885	1980	1984	2142	2146	2182
		2624	2626						
BUFRLO	0032	-221	1769	1881	1951	1978	1982	2009	2133
		2140	2144	2177	2615	2617	2621	2623	
BUFSTR	006C	-275	4389	4391	4406	4408	4584	4588	4927
		5214	5216	5263	5265				

CAINI	F239	3669	-3750						
CART	BFFC	-3436	3658	3666	3781	3783	3784	3793	
CARTAD	BFFE	-3438	3750	3751	3788				
CARTCS	BFFA	-3435	3720	3726					
CARTFG	BFFD	-3437	3702	3705	3717	3723	3786		
CAS31	EC5E	2352	-2357						
CASBUF	03FD	-545	3101	3102	3237	3248	3253	3264	3295
		3301	3389	3391	3392	3421	3422	3892	3900
		3902	3905						
CASENT	EB80	1734	-2197						
CASET	0060	-1611	1732	2289	2351				
CASETV	E440	-59	3118	3513	3849				
CASFLG	030F	-499	1739	2050	2234				
CASINI	0002	-169	4009	4011	4014				
CASORG	EF41	-30	3097	3136					
CASRED	EBB3	2198	-2233						
CASSBT	004B	-245	3888	3922	3925	3975	4001	4005	
CASSET	0043	-124							
CASSPR	0014	-3398							
CAUX1	023C	-351	1763						
CAUX2	023D	-352	1765						
CBAUDH	02EF	-457	2602	2608	3144				
CBAUDL	02EE	-456	2601	2606	3142				
CBINI	F23C	3661	-3751						
CBUFH	0003	-3101	3102	3366					
CBUFHI	0002	-1669	1670	1772					
CBUFL	00FD	-3102	3368						
CBUFLO	003A	-1670	1768						
CCOMND	023B	-350	1760						
CDEVIC	023A	-349	1669	1670	1757				
CDTMA1	0226	-330	1528	2653	2655				
CDTMA2	0228	-331	1529						
CDTMF3	022A	-332	1455	3177	3179	3219	3222		
CDTMF4	022C	-334							
CDTMF5	022E	-336							
CDTMV1	0218	-323	1451	1452	1535	1537	1539	1540	1542
		1564	1566						
CDTMV2	021A	-324							
CDTMV3	021C	-325							
CDTMV4	021E	-326							
CDTMV5	0220	-327							
CDUBL	EF26	3072	-3075						
CH	02FC	-470	1478	4087	4462	4469	4474	5787	
CH1	02F2	-461	5776	5788					
CHACT	02F3	-463	1442	4110					
CHACTL	D401	-737	1443						
CHAR	02FA	-468	4302	4344	4346	5069	5079		
CHBAS	02F4	-464	1440	4108					
CHBASE	D409	-743	1441						
CHKDON	EABA	1995	-2002	2017					
CHKERR	008F	-151	2118						
CHKSNT	003B	-230	1947	1988	1994	2026			
CHKSUM	0031	-220	1946	1955	1991	2013	2015	2053	2115

		2136 2138 2619
CHKTIM	EAF9	-2064 2071
CHRORG	E000	-22
CICLO2	E53F	901 -903
CICLOS	E533	853 -898
CIERR1	E4D1	821 -826
CIERR2	E6B0	1173 -1183
CIERR3	E50F	-869
CIERR4	E511	842 -870 874 880 920
CIJUMP	E693	1152 -1158
CINI	F3E1	3994 -4014
CIO	E4C4	771 -815
CIOCHR	002F	-217 815 951 956 966 985 989 1019
		1026 1037 1083
CIOI1	E4A8	-787 798
CIOINT	E4A6	775 -786 1212
CIOINV	E46E	-76 774 3850
CIOORG	E4A6	-25 783
CIOPEN	E509	851 -864
CIOSPR	0014	-1215
CIOV	E456	-68 770 3683 3967
CIREAD	E569	857 -934
CIRT3	E62B	-1075 1080
CIRTN1	E61B	827 870 940 1009 -1065
CIRTN2	E61D	893 909 930 952 999 1060 -1069
CIST1	E559	916 -923
CISTSP	E54E	855 -914
CIWRIT	E5C9	858 -1003
CIX	00F2	-616
CKEY	004A	-244 3856 3997 4006
CKSTC	EE11	2808 -2810
CLICK	FCD8	4475 4727 -5355
CLICK1	FCDA	-5356 5359
CLOSE	000C	-93
CLOSEC	F02B	3119 -3286
CLRCHP	F27A	-3820 3825
CLRCOD	007D	-4030 4312
CLRLI1	FBA6	-5149 5151
CLRLIN	FB9B	4697 -5144
CLRRAM	F140	-3591 3594 3598
CLRSC2	F7BF	-4570 4572 4576
CLRSC3	F7CE	-4578 4581
CLRSCR	F7B9	4280 4314 -4567 5671
CLRTAB	F832	-4623 5683
CLRTBS	F430	-4125 4127
CLS	007D	-3416 3527
CLWRT	F038	3287 -3293
CMODE	EF1E	-3071 3087
CMPTAB	E6F6	-1282 1313
CNTL1	009F	-4031 5781
CNTRLS	FEC6	4442 4444 5299 -5660
COLAC	0072	-279 5440 5495 5497 5498 5500 5504
COLBK	D01A	-708

```
COLCR      FCE4          4582  4901 -5365
COLCR1     FCEE          5367 -5370
COLCR2     FCF0          5369 -5371
COLCRS     0055          -259  4390  4409  4552  4553  4558  4560  4561
                         4587  4594  4599  4610  4679  4719  4778  4780
                         4811  4816  4845  4847  4860  4862  4863  4875
                         4903  4970  4975  4978  5145  5202  5217  5226
                         5245  5371  5400  5402  5509  5510  5513  5515
                         5517  5519
COLDST     0244          -361  3565  3606  3714
COLDSV     E477           -79  3491
COLINC     007A          -284  5406  5428  5507
COLON      003A          -765
COLOR0     02C4          -436  4130
COLOR1     02C5          -437
COLOR2     02C6          -438
COLOR3     02C7          -439
COLOR4     02C8          -440  4169
COLPF0     D016          -704
COLPF1     D017          -705
COLPF2     D018          -706
COLPF3     D019          -707
COLPM0     D012          -700  1437
COLPM1     D013          -701
COLPM2     D014          -702
COLPM3     D015          -703
COLRSH     004F          -250  1407  1435
COLRTB     FEC1          4129 -5651
COLUMN     FE8D          4865  4977 -5616
COM1       E647          1095 -1102
COM2       E662          1099 -1116
COMENT     E63D           879   888   900   923   943  1012 -1093
COMFRM     E978         -1748  1789
COMMND     E974         -1743  1856
COMPLT     0043         -1634  1904
COMPUT     ECA3         -2468  2600
COMRE1     F7A7          4556 -4559  4563  4564
COMRET     F789          4541 -4545  4549
COMTAB     E6C9           850  1107 -1209
CONSOL     D01F          -713  1432  3325  3330  3853  5356
CONTIN     EC31          2307 -2331
CONVR1     F97E         -4805  4809
CONVR2     F988          4806 -4810
CONVR3     F98F         -4813  4819
CONVR4     F99C          4814 -4820
CONVR5     F9A6          4824 -4826
CONVR6     F9A7         -4827  4830
CONVRT     F947          4295  4345  4652  4680  4720 -4776  5146  5282
COS        BD73          -601
COUNT      ED3B         -2580  2591  2594  2604
COUNTR     007E          -288  5453  5463  5471  5472  5545  5547  5548
                         5550
CR         009B          -129  2991  3527  3539  3551  4317  4395  4420
```

		4510							
CRETRI	000D	-1672	1743	1802					
CRETRN	EBDF	2227	-2259						
CRETRY	0036	-225	1744	1788	1803				
CRITIC	0042	-239	1412	1729	1863				
CRLOOP	F5E7	-4329	4332						
CRNTP1	E6D5	-1213	1215						
CRNTP2	E90B	-1582	1584						
CRNTP3	EDEB	-2725	2727						
CRNTP4	EE78	-2872	2875						
CRNTP5	EF41	-3093	3097						
CRNTP6	F0E3	-3396	3398						
CRNTP7	F3E4	-4020	4023						
CRNTPC	FFF8	-5802	5804						
CRSINH	02F0	-459	1339	4122	4365				
CRSRDN	F78C	-4546	5665						
CRSRL1	F7A3	4554	-4557						
CRSRLF	F799	-4552	4598	5237	5244	5667			
CRSROR	008D	-149	4989						
CRSRRT	F7AA	-4560	4609	5669					
CRSRUP	F77F	-4540	4604	5663					
CSBOOT	F3B2	3694	-3989						
CSBOT2	F3C0	3990	-3997						
CSIDE	EF2E	3076	-3079						
CSIO	F0AC	-3370							
CSOPIV	E47D	-81	3132	4002					
CSTAT	0288	-396							
CTIA	D000	-681	682	683	684	685	686	687	688
		689	690	691	692	693	694	695	696
		697	698	699	700	701	702	703	704
		705	706	707	708	709	710	711	712
		713	714	715	716	717	718	719	720
		721	722	723	724	725	726	727	728
		729	730	731	732	733			
CTIMHI	0000	-1675	2446						
CTIMLO	0002	-1674	2445						
CTRLC	0092	-3417							
D	0044	-2901	3075						
DAUX1	030A	-494	1762	2960	3084	3880	3918		
DAUX2	030B	-495	1764	2210	2237	2259	3383	3878	
DBDDEC	F913	4188	-4740						
DBDEC	F91F	4178	-4751						
DBSECT	0241	-359	3916						
DBSUB	F921	4148	4165	4741	-4752				
DBSUB1	F934	4759	-4761						
DBUFHI	0305	-489	2181	2815	2864	3035	3367	3884	
DBUFLO	0304	-488	2176	2813	2862	3034	3369	3882	
DBUFSZ	0014	-2894	3077						
DBYTHI	0309	-493	2183	2820	2850	3049	3363		
DBYTLO	0308	-492	2178	2818	2848	3047	3365		
DCB	0300	-483							
DCOMND	0302	-486	1759	2799	2806	2824	2831	2959	3041
		3083	3361	3376	3872	3979			

DCTIM1	E8DD	1536 -1540
DCTIMR	E8D0	1409 1445 1454 -1535
DCTXF	E8EA	1538 1541 1543 -1546
DDEVIC	0300	-484 1731 1754 2288 2350 2797 3037 3204 3371
DECBF1	E66D	1124 -1126
DECBFL	E663	969 973 1040 1044 -1121
DEGFLG	00FB	-621
DEGON	0006	-624
DELAY0	EC8C	-2437 2441
DELAY1	EC8E	-2438 2439
DELCH1	F870	-4652 4669
DELCH2	F896	4662 4665 -4670
DELCHR	F86D	-4651 5689
DELETE	0021	-101
DELLI1	F8DD	-4703 4714
DELLI2	F8F8	-4718
DELLIA	F8D7	-4700
DELLIB	F8DB	-4702 4724 5291
DELLIN	F8D4	-4699 5679
DELTAC	0077	-282 5422 5425 5429 5431 5432 5434 5435 5437 5447 5452 5456 5496 5499
DELTAR	0076	-281 5410 5414 5418 5457 5459 5477
DELTI1	FC82	-5287 5290
DELTI2	FC89	-5291
DELTI3	FC8C	5281 5288 -5292
DELTIA	FC68	4673 -5274
DELTIB	FC70	5276 -5278
DELTIM	FC73	4597 -5279
DERR	E9F6	1821 -1842
DERR1	EA06	1791 1830 -1853
DERR5	F10D	-3539 3543 3544
DERRH	00F1	-3543 3544 3949
DERRL	000D	-3544 3948
DERROR	0090	-153 1910
DEVS1	E6A6	-1175 1180
DEVS2	E6B5	1176 -1188
DEVS3	E6C5	1195 -1197
DEVS4	E6C8	1185 -1201
DEVSRC	E69E	873 919 -1171
DFLAGS	0240	-358 3893
DHLINE	FE81	4804 -5609
DIGRT	00F1	-615
DINDEX	0057	-260 4106 4138 4194 4225 4363 4803 4864 4877 4905 4956 4963 5070 5368
DINI	F37E	3867 3930 -3941
DINIT	EDEA	2759 -2788
DIRECT	0002	-112
DISK	0044	-126
DISKID	0031	-2744 2796
DISKIV	E450	-66
DISKM	F3A4	3976 -3978
DISPLA	E410	-4056

DISPLY	0053	-122
DIV2TB	FEA5	4810 -5633
DLISTH	D403	-739 1424
DLISTL	D402	-738 1426
DMACTL	D400	-736 1428
DMASK	02A0	-416 4297 4351 4353 4842
DMASKT	FEB1	4841 -5640
DNACK	008B	-147 1914
DOB1	FC1A	5222 -5224
DOBOOT	F2ED	3874 -3877 3889
DOBU1A	FC29	5225 5228 -5231
DOBUF1	FC12	-5220 5235
DOBUF2	FC39	5230 -5238 5250
DOBUF3	FC51	5247 -5249
DOBUF4	FC55	5239 5243 -5251
DOBUFC	FC00	4405 -5211
DOCR	FA34	4613 4878 4883 4886 -4901
DOCR1	FA00	4874 -4877
DOCR1A	FA29	4891 4893 -4896
DOCR1B	FA14	4881 -4886
DOCR2	FA3D	-4905
DOCR2A	FA4A	4908 -4912
DOCR2B	FA4D	4911 -4913
DOCR4B	FA61	4923 -4925 4931
DOCRWS	FA30	4319 4419 4884 -4899 5677
DOINTP	ECE5	-2509 2512
DOLCO1	FBE5	-5191 5199
DOLCO2	FBF8	5193 -5200
DOLCOL	FBDD	4559 4608 4620 4648 4675 4698 4699 4725
		4896 4934 5182 -5187 5261 5278
DOPEN	F3F6	4056 -4099
DOPEN1	F460	-4147 4150
DOPEN2	F4D5	-4203 4206
DOPEN3	F4FA	-4221 4224
DOPEN4	F51C	-4237 4240
DOPEN5	F524	4227 -4241
DOPEN7	F588	4279 -4284
DOPEN8	F438	-4129 4132
DOPEN9	F577	4271 -4277
DOPENA	F457	4140 -4143
DOSINI	000C	-178 3901 3903 3941 4008 4010
DOSS	F6AD	4397 -4431
DOSVEC	000A	-177 3602 3604 3730
DOUBLE	0044	-1626
DRAW	FCFC	4061 -5388
DRAW1	FD37	5411 -5419
DRAW10	FE42	5474 5551 -5553
DRAW11	FD99	5473 -5475
DRAW2	FD5C	5426 5436 -5438
DRAW3	FD83	5455 5458 -5462
DRAW3A	FD62	-5441 5446
DRAW4A	FD90	-5471 5552
DRAW5	FDA4	5479 -5481

```
DRAW5A   FDB2       5484 -5488
DRAW6    FDBE       5483  5487 -5494
DRAW6A   FDD7       5503 -5507
DRAW6B   FDEB       5508 -5517
DRAW7    FDF1       5512  5514  5516  5518 -5520
DRAW8    FDF6       5502  5506 -5522
DRAW8A   FE0A      -5529  5540
DRAW8B   FE27       5536 -5541
DRAW8C   FE13      -5534
DRAW9    FE30       5525 -5544
DRAWA    FD0C       5391 -5397
DRAWB    FD0B       5393 -5396
DRAWLN   0011        -98
DRETRI   0001      -1673  1741
DRETRY   0037       -226  1742  1853
DRKMSK   004E       -249  1406  1436
DSKFMS   0018       -191
DSKIF    EDF0       2760 -2796
DSKINV   E453        -67  3873  3982
DSKORG   EDEA        -28  2727  2769
DSKRDE   F381       3887  3921 -3948
DSKSPR   0014      -2875
DSKTIM   0246       -365  2789  2798  2830
DSKUTL   001A       -192
DSPFLG   02FE       -472  4437
DSTAT    004C       -246  4113  4142  4270  4369  4371  4393  4414
                    4423  4508  4745  4753  4769  4982  4990
DSTATS   0303       -487  1794  1826  1865  2197  2817  2851  3045
                    3381
DT1      00FA      -3110  3251  3296
DTA      00FC      -3109  3271
DTIMLO   0306       -490  2396  2803  3051  3375
DUNIT    0301       -485  1755  3039  3373  3870  3981
DUNUSE   0307       -491
DVSTAT   02EA       -455  2736  2737  2939  3059
EDITOR   E400      -4043
EDITRV   E400        -55  3515  3742  3744  3845  4039
EEXP     00ED       -611
EGETC1   F650      -4392  4403
EGETC2   F66E       4396 -4404
EGETC3   F67C       4387 -4410
EGETC5   F691       4411  4413 -4419
EGETC6   F66B       4401 -4403
EGETC7   F680      -4412  4415
EGETCH   F63E       4045 -4384
ENBOOT   F35A       3917 -3925
ENDACK   F1BD       3665  3667 -3674
ENDBCK   F1AC       3657  3659 -3663
ENDDIF   EE69       2833 -2851
ENDPT    0074       -280  4941  4944  5454  5464  5466  5482  5486
                    5501  5505
ENDRAM   F276       3802 -3810
ENINTP   ECEB      -2514
```

ENSPE2	F257	3782	-3794					
ENSPEC	F254	3785	3787	-3793				
ENTVEC	002C	-810						
EOF	0088	-3418						
EOFERR	0088	-144	3257	4507				
EOL	009B	-766	967	990	1038	1055		
EOPEN	F3FC	4043	-4102	4273	4958			
EOT	00FE	-3111	3249	3304				
EOUTC5	F6BE	4433	-4437					
EOUTC6	F6B5	-4434	4439					
EOUTCH	F6A4	4046	-4428					
ERANGE	FA88	4385	4430	-4953				
ERETN	F6BB	-4436						
ERRFLG	023F	-356	1781	1818	1829	1842	1877	1922
ERROR	0045	-1635	1907					
ERRTN	E4C0	-802	803	804				
ERRTNH	00E4	791	-803	804	905			
ERRTNL	00C0	789	-804	907				
ESCAPE	F779	-4537	5661					
ESCFLG	02A2	-418	4434	4438	4440	4538		
ESIGN	00EF	-613						
ESTSCM	F173	3607	-3622					
EXP	DDC0	-595						
EXP10	DDCC	-596						
EXTEN1	FB7E	-5122	5133					
EXTEN3	FB8B	-5130						
EXTEN4	FB92	5131	-5134					
EXTEND	FB7B	4677	-5120					
FADD	DA66	-581						
FASC	D8E6	-576						
FCAX	F032	-3290	3298	3306				
FCHRFL	00F0	-614						
FDIV	DB28	-583						
FEOF	003F	-236	3162	3232	3256			
FILDAT	02FD	-471	5537					
FILFLG	02B7	-422	5397	5524				
FILLBF	EEC3	-2996	2999					
FILLIN	0012	-99						
FINDX	ECD6	-2497						
FLD0P	DD8D	-585						
FLD0R	DD89	-584						
FLD1P	DD9C	-587						
FLD1R	DD98	-586						
FLOPPY	0030	-1608						
FLPTR	00FC	-625						
FMOVE	DDB6	-590						
FMSZPG	0043	-241						
FMTD	EE4A	-2834						
FMUL	DADB	-582						
FNCNOT	0092	-155	1151					
FOMAT	0021	-2749	2800	2832				
FOOEY	EADE	2027	-2036					
FORMAT	0022	-102						

FPI	D9D2	-579							
FPREC	0006	-572	634	635					
FPSCR	05E6	-634	635	636					
FPSCR1	05EC	-635	637						
FPTR2	00FE	-626							
FR0	00D4	-606							
FR1	00E0	-608							
FR2	00E6	-609							
FRE	00DA	-607							
FREQ	0040	-237	3314	3336					
FRMADR	0068	-4033	4682	4687	5098	5105	5107		
FRMERR	008C	-148	2102						
FRX	00EC	-610							
FSCR	05E6	-636							
FSCR1	05EC	-637							
FSTOP	DDAB	-589							
FSTOR	DDA7	-588							
FSUB	DA60	-580							
FTYPE	003E	-235	3152	3382	4000				
GBX	EFE8	-3240	3244						
GBYTE	EFD6	3119	-3232	3255					
GETCAR	0007	-3407							
GETCH	F593	4058	-4290	4416					
GETCHR	0007	-90							
GETDAT	0040	-2751	2804						
GETOUT	F749	4465	-4510						
GETPLT	F5A2	4291	-4295	4361	4626	4639	4666	5238	5535
GETREC	0005	-89							
GETSEC	F39D	3885	3919	3927	-3975				
GLBABS	02E0	-447							
GOBACK	EB09	-2072							
GOERR	EF3D	3080	-3086						
GOHAND	E689	883	902	924	950	955	984	1027	1056
		-1151							
GOOD	EA65	1903	1905	1923	-1927				
GOODST	EE32	2822	-2824						
GOON	EB64	2151	-2159						
GOREAD	ED6F	2597	-2606						
GPRIOR	026F	-369	1429	4151	4170				
GRACTL	D01D	-711							
GRAFM	D011	-699							
GRAFP0	D00D	-695							
GRAFP1	D00E	-696							
GRAFP2	D00F	-697							
GRAFP3	D010	-698							
HARDI	F277	3583	-3818						
HATABS	031A	-512	513	1102	1104	1175	3642		
HDR	00FB	-3112							
HITCLR	D01E	-712							
HITIMR	ECCC	-2490	2494						
HITONE	0005	-1644	2296						
HOLD1	0051	-255	4143	4221	4237	4245	4700	4790	4798
		5190	5191	5198	5262				

HOLD2	029F	-415	4792	4801					
HOLD3	029D	-413	4915	4928	4932				
HOLD4	02BC	-426	5528	5541					
HOLD5	02BD	-427							
HOLDCH	007C	-286	4472	4512	4522				
HOME	F7D6	-4582	4988						
HOWMCH	F25F	-3798	3809						
HPOSM0	D004	-686							
HPOSM1	D005	-687							
HPOSM2	D006	-688							
HPOSM3	D007	-689							
HPOSP0	D000	-682							
HPOSP1	D001	-683							
HPOSP2	D002	-684							
HPOSP3	D003	-685							
HSCROL	D404	-740							
ICAX1	034A	-528	3682						
ICAX1Z	002A	-213	935	1004	3153	4102	4104	4197	4277
		4463							
ICAX2	034B	-529							
ICAX2Z	002B	-214	3070	3151	4099				
ICBAH	0345	-523	1072	3680	3962				
ICBAHZ	0025	-208	1073						
ICBAL	0344	-522	1070	3678	3960				
ICBALZ	0024	-207	959	1025	1071	1131	1133	1172	1192
ICBLH	0349	-527	1143						
ICBLHZ	0029	-212	1145						
ICBLL	0348	-526	1140	3966					
ICBLLZ	0028	-211	947	948	1016	1017	1020	1121	1122
		1125	1126	1141	1142	1144			
ICCOM	0342	-520	3676	3964					
ICCOMT	0017	-189	849	887	1106				
ICCOMZ	0022	-205	840	934	961	979	1003	1032	1050
		5389							
ICDNO	0341	-519							
ICDNOZ	0021	-204	1197						
ICHID	0340	-518	788	928					
ICHIDZ	0020	-203	864	904	914	929	1093	1189	
ICIDNO	002E	-216	816	927	1069	1084	1138	1164	
ICPTH	0347	-525	792						
ICPTHZ	0027	-210	892	906					
ICPTL	0346	-524	790						
ICPTLZ	0026	-209	890	908					
ICSPR	034C	-530							
ICSPRZ	002C	-215	216	217	810	889	891	1103	1105
		1109	1112	1113	1114	1159	1161		
ICSTA	0343	-521							
ICSTAZ	0023	-206	899	995	1065	1085	1153		
IDENT	F0F2	-3527	3535	3536	3538				
IDENTH	00F0	-3535	3536	3734					
IDENTL	00F2	-3536	3733						
IFP	D9AA	-577							
IHINIT	E6D5	1230	-1269						

IMASK	028B	-399							
INATA1	FB4A	5072	-5082						
INATAC	FB32	4292	-5069						
INBUFF	00F3	-617							
INC2A	F9F8	4866	4871	-4873					
INCBF1	E676	1132	-1134						
INCBFP	E670	960	1029	-1131					
INCRS1	FA77	4885	4914	4917	4919	-4934			
INCRS2	F9E4	4861	-4863						
INCRS3	F9F7	4868	4870	-4872	4876				
INCRSA	F9DC	4633	-4859	5531					
INCRSB	F9D4	4293	4659	-4855	5231				
INCRSC	F9DA	4856	-4858						
INCRSR	F9D8	4322	-4857						
INIMLH	0007	-3410	3843						
INIMLL	0000	-3409	3841						
INIT	EF41	3122	-3141						
INSCH1	F852	-4637							
INSCH3	F85E	4636	-4643						
INSCH4	F844	-4630	4642						
INSCH5	F86A	4645	-4648						
INSCH6	F861	-4644	4647						
INSCHR	F837	-4625	5691						
INSCLR	0020	-117							
INSDAT	007D	-287	4627	4637	4640	4858	4882	4892	4900
		4918							
INSLI1	F8C6	-4693	4696						
INSLI2	F8CE	4692	-4697						
INSLIA	F8A5	-4677	4895						
INSLIN	F8A4	-4676	5681						
INTABS	0200	-310	1232	1292	1293	1294	1295	1296	1297
		1298	1299	1329	1331	1694	3631	4081	
INTATA	FEFA	5081	-5704						
INTEMP	022D	-335	1557	1563					
INTINV	E46B	-75	1229	3852					
INTORG	E6D5	-26	1215	1268					
INTSPR	0014	-1584							
INTTBL	EC84	-2419							
INTZBS	0010	-182	3616						
INVFLG	02B6	-421	4486	4488	4527				
IOC1	E4D6	823	-830						
IOC1A	E4D8	-831	836						
IOC2	E4F3	847	-849						
IOC6	E514	866	-873						
IOC7	E519	-879							
IOCB	0340	-517	831	1076					
IOCBAS	0020	-202	832	1075					
IOCBSZ	0010	-200	201	795					
IOCFRE	00FF	-107	787	865	903	915			
IRQEN	D20E	-678	1306	1308	1324	1326	2000	2034	2332
		2374	4117						
IRQST	D20E	-664	1302	1318					
ISEOF	F00D	3233	-3257						

ISRODN	EA90	1697	-1975	2420	2425	2426			
ISRSIR	EB0F	1696	-2091	2419	2423	2424			
ISRTD	EACF	1698	-2026	2421	2427	2428			
JMPP	E735	1319	-1323						
JSRIND	F6A1	-4426	4446						
JTADRH	00EB	-2272	2273	2654					
JTADRL	00EC	-2273	2652						
JTIMER	EBEC	-2271	2272	2273					
JTIMR1	E8CA	1411	-1528						
JTIMR2	E8CD	1447	-1529						
JVECK	028C	-400	1330	1332	1335	1354	1359	1362	
K1	F729	4491	-4495						
K2	F734	4496	-4500						
K3	F73F	4501	-4505						
K4	F776	4525	-4529						
K5	F768	4514	4517	4519	4521	-4524			
K6	F74D	4506	-4512						
K7	F745	4468	-4508						
K8	F773	4511	-4528						
KBCODE	D209	-660	1477	5775	5780				
KBD	004B	-121							
KBDHND	E420	-4070							
KBDORG	F3E4	-32	4023	4084					
KBDSPR	0014	-5804							
KEYBDV	E420	-57	3353	3355	3519	3847	4065		
KEYDEL	02F1	-460	1463	1465	5778	5790			
KGETC1	F71E	4485	-4490						
KGETC2	F6DD	-4461	4483	4489	4494	4499	4504		
KGETC3	F6FE	-4476	4523						
KGETCH	F6E2	4072	4392	-4463	4471				
LBFEND	05FF	-638							
LBPR1	057E	-630							
LBPR2	057F	-631							
LBUFF	0580	-632	633						
LDPNTR	EB6A	1805	1838	-2175	2223	2250			
LEDGE	0002	-252	3622						
LENGTH	022F	-1212							
LFRTCM	F7A5	-4558	4566						
LINBUF	0247	-367	5091	5093					
LINZBS	0000	-166							
LIRG	0000	-3419							
LL	E72F	1317	-1320						
LMARGN	0052	-256	3623	4282	4555	4565	4592	4595	4611
		4678	4718	5144	5234	5242	5264	5275	5280
		5285	5370						
LO1GET	FB22	-5054	5123	5192					
LO2GET	FB23	4705	-5055						
LOCKFL	0023	-103							
LOG	DECD	-597							
LOG10	DED1	-598							
LOGCOL	0063	-268	4399	4583	4591	4617	4621	4623	4631
		4635	4657	4661	4859	4879	5188	5194	5197
		5201	5203	5212	5233	5241	5252	5274	5279

LOGGET		FB20	4614	4722	-5053					
LOGMAP		02B2	-420	4578	4715	4717	4929	5032	5033	5034
LOOPM		E71F	-1313	1321						
LOOPM2		E72A	1315	-1318						
LOTONE		0007	-1645	2294						
LPENH		0234	-343	1422						
LPENV		0235	-344	1420						
MOPF		D000	-714							
MOPL		D008	-722							
M1PF		D001	-715							
M1PL		D009	-723							
M2PF		D002	-716							
M2PL		D00A	-724							
M3PF		D003	-717							
M3PL		D00B	-725							
MASKTB		FEB9	5022	-5645						
MAXDEV		0021	-513	1094	1174					
MAXIOC		0080	-201	797	822					
MEMLO		02E7	-453	3842	3844					
MEMORY	M	0000	0							
MEMTOP		02E5	-452	3838	3840	4088	4259	4261		
MLTTMP		0066	-270	271	4034	4784	4786	4787	4789	4791
			4794	4796	4799	4800	4802	4807	4808	4822
			4823	4825	4832	4837				
MODATA		E9F0	1827	-1838						
MODEM		004D	-125							
MONORG		F0E3	-31	3398	3503					
MONSPR		0014	-4023							
MOTRGO		0034	-1664	2217	2244	3166	3206			
MOTRST		003C	-1665	1706	2262	2633	3290			
MOVLI1		FB58	-5096	5103						
MOVLI2		FB7A	5112	-5114						
MOVLIN		FB4E	4693	-5091						
MOVVEC		F17D	-3630	3633						
MVBUFF		F32D	-3904	3920						
MVNXB		F32F	-3905	3908						
MXDMDE		FE5D	4216	-5572						
MXDMOD		0010	-116							
N		004E	-2900	3071	3086					
NACK		004E	-1633							
NARG		0000	0							
NBUFSZ		0028	-2893	3073						
NCOMHI		003C	-1663	1709	2062					
NCOMLO		0034	-1662	1776						
NEWCOL		0061	-267	5401	5403	5420	5423			
NEWROW		0060	-266	5399	5408					
NLR		F005	3252	-3254						
NMIEN		D40E	-748	1270						
NMIRES		D40F	-749	1384						
NMIST		D40F	-750	1372	1376					
NOA1		F1F1	3701	-3703						
NOA2		F212	3716	3719	-3721					
NOB1		F1F8	3704	-3706						

NOBOOT	F1FF	3707	-3713		
NOCAR2	F220	3722	-3730		
NOCART	F1FC	3699	-3710	3725	
NOCKSM	003C	-231	1889	2150	2154
NOCLR	EAE9	2051	-2054		
NOCSB2	F3BF	3993	-3995		
NOCSBT	F3E0	3998	-4012		
NODAT	0000	-2750			
NOFUNC	F63D	4048	4073	4075	-4373
NOINIT	F2DC	3866	-3868		
NOISE1	EC45	2341	-2343	2346	
NOKEY	F2CE	3855	-3857		
NOMOD	F4AB	4180	-4184		
NONDEV	0082	-138	1183		
NORMAL	004E	-1625			
NOROWS	FE99	4912	4964	-5624	
NOSCR1	FA32	4898	-4900		
NOSCRL	FA2C	-4897			
NOT8	F48B	4156	4167	-4170	
NOTCAS	EC0C	2290	-2299		
NOTCST	E96B	1733	-1738		
NOTDER	EA52	1908	-1914		
NOTDON	EA81	-1957	1962		
NOTE	0026	-106			
NOTEND	EABE	1986	-2008		
NOTERR	EA00	1843	-1849		
NOTMXD	F4F5	4196	4199	-4219	
NOTOPN	0085	-141	805	1098	
NOTYET	EB3C	2112	-2131		
NOWARM	F2DD	3863	-3869		
NOWRPO	EA98	1979	-1982		
NSIGN	00EE	-612			
NTBRKO	EA88	1958	-1961		
NTBRK1	EB00	2065	-2068		
NTBRK2	ED17	2559	-2562		
NTFRAM	EB1D	2100	-2105		
NTOVRN	EB25	2106	-2111		
NTWRP1	EB50	2141	-2144		
NUMDLE	FE51	4216	4220	-5569	
NVALID	0084	-140	839	5394	
NWOK	EA56	1896	1912	-1917	
NXTENT	F18C	-3641	3644		
ODNHI	00EA	-2425	2426		
ODNLO	0090	-2426			
OFFCRS	FAE4	4404	4431	-5002	
OKTIM1	ED1F	2565	-2568		
OKTIMR	ED48	2584	-2588		
OLDADR	005E	-265	4836	4840	5004
OLDCHR	005D	-264	4362	5003	
OLDCOL	005B	-263	5421	5424	
OLDROW	005A	-262	4330	5409	5443
OPEN	0003	-88	841		
OPENC	EF4C	3119	-3151		

OPINP	EF5D	3133	3156	-3160					
OPNCOM	F404	4101	-4106						
OPNEDT	F118	-3551	3553	3554					
OPNERR	F453	-4141							
OPNH	00F1	-3553	3554	3679					
OPNIN	0004	-113	115						
OPNINO	000C	-115							
OPNL	0018	-3554	3677						
OPNOT	0008	-114	115						
OPNOUT	0002	-2892							
OPNRTN	EF8F	3165	-3190	3198					
OPNTMP	0066	-271	4153	4162	4191	4217	4218	4219	
OPOK	EFD3	3184	-3226						
OPOUT	EF95	3158	-3194						
OPSYS	F17B	-3629							
OSRAM	F28A	3634	-3831						
OUTCH	F5B7	4059	-4308						
OUTCH2	F5FF	-4344	4368	4630					
OUTCHA	F5BD	-4311							
OUTCHB	F5D7	4318	-4321						
OUTCHE	F5CA	4313	-4316	4435					
OUTPLT	F5E0	4321	-4326	4327	4607	5523	5539		
OVRRUN	008E	-150	2108						
P0PF	D004	-718							
P0PL	D00C	-726							
P1PF	D005	-719							
P1PL	D00D	-727							
P2PF	D006	-720							
P2PL	D00E	-728							
P3PF	D007	-721							
P3PL	D00F	-729							
PACTL	D302	-754	1272	1278	1345	1707	2218	2245	2263
		2634	3167	3207	3291				
PADDL0	0270	-371	1500						
PADDL1	0271	-372							
PADDL2	0272	-373							
PADDL3	0273	-374							
PADDL4	0274	-375	1502						
PADDL5	0275	-376							
PADDL6	0276	-377							
PADDL7	0277	-378							
PAGETB	FE75	4179	-5601						
PALFLG	0000	-17	1647	1653	2483	2488	2538	2541	3168
		3172	3209	3213	3317	3320	3340	3343	
PBCTL	D303	-755	1273	1279	1349	1710	1777	2063	2635
PBPNT	001D	-195	2974	2984	2990	3001	3014		
PBRK	EF8B	-3188	3221						
PBUFSZ	001E	-196	2955	2988	2998	3046	3082		
PBYTE	F010	3119	-3263						
PCOLR0	02C0	-432	1434						
PCOLR1	02C1	-433							
PCOLR2	02C2	-434							
PCOLR3	02C3	-435							

PDEVN	0040	-2896	3036						
PENH	D40C	-746	1421						
PENV	D40D	-747	1419						
PHACR1	FC9F	-5313	5316						
PHACRS	FC9D	4625	4651	5211	-5312	5526			
PHCHLO	EE7F	-2940	3002	3003					
PHCLOS	EEDC	2913	-3013						
PHINIT	EE78	2918	-2932						
PHOPEN	EE9F	2912	-2972						
PHPUT	EF14	2964	-3059						
PHSTAT	EE81	2916	-2954	2972					
PHSTLO	EE7D	-2939	2956	2957					
PHWRIT	EEA7	2915	-2982						
PIA	D300	-751	752	753	754	755			
PIRQ	E6F3	-1281	1577	1578					
PIRQ1	FFDC	5782	-5787						
PIRQ2	FFF0	-5795							
PIRQ3	FFCB	5777	-5780						
PIRQ4	FFEB	5779	5786	-5793					
PIRQ5	FFBE	4082	-5775						
PIRQH	00E6	1252	-1577	1578					
PIRQL	00F3	1254	-1578						
PLACR1	FCAA	-5323	5326						
PLACRS	FCA8	4643	4674	5253	-5322	5543			
PLOT	0050	-1627							
PLUS	ECF8	2518	-2520						
PLYARG	05E0	-633	634						
PLYEVL	DD40	-591							
PMBASE	D407	-742							
PNMI	E791	-1372	1579	1580					
PNMI1	E799	1373	-1375						
PNMIH	00E7	1256	-1579	1580					
PNMIL	0091	1258	-1580						
POINT	0025	-105							
POKEY	D200	-650	651	652	653	654	655	656	657
		658	659	660	661	662	663	664	665
		666	667	668	669	670	671	672	673
		674	675	676	677	678	679		
POKMSK	0010	-183	1307	1316	1325	1997	1999	2031	2033
		2303	2329	2331	2372	2373	4115	4116	
POKTAB	EDD0	2521	2524	-2699					
PORTA	D300	-752	1275	1347	1482	1489			
PORTB	D301	-753	1276	1351					
POT0	D200	-651	1499						
POT1	D201	-652							
POT2	D202	-653							
POT3	D203	-654							
POT4	D204	-655	1501						
POT5	D205	-656							
POT6	D206	-657							
POT7	D207	-658							
POTGO	D20B	-662	1505						
PRINTR	0050	-123							

PRINTV	E430	-58	3511	3848					
PRIOR	D01B	-709	1430						
PRMODE	EF1A	2983	3013	-3069					
PRNBUF	03C0	-533	2940	2986	2996				
PRNORG	EE78	-29	2875	2925					
PRNSPR	0014	-3097							
PRVOPN	0081	-137	869						
PSIOC	EF01	3043	-3045						
PTEMP	001F	-197	2982	2985					
PTIMOT	001C	-194	2933	3050	3060				
PTRIG0	027C	-383	1516						
PTRIG1	027D	-384	1513						
PTRIG2	027E	-385							
PTRIG3	027F	-386							
PTRIG4	0280	-387							
PTRIG5	0281	-388							
PTRIG6	0282	-389							
PTRIG7	0283	-390							
PTRLP	E8AC	-1509	1520						
PUTADR	EE6D	2827	2834	-2862					
PUTBC	EE43	2826	-2831						
PUTCAR	000B	-3408							
PUTCHR	000B	-92	886						
PUTCNT	EE21	2811	-2817						
PUTDAT	0080	-2752	2809						
PUTDTO	EE01	2801	-2803						
PUTLIN	F385	3735	-3958						
PUTMSC	FCF3	4567	4782	-5377					
PUTREC	0009	-91							
PUTSEC	0050	-2745							
PUTTXT	0009	-3406	3963						
PWRONA	F3E4	4049	4062	4076	-4086				
PWRUP	F125	3493	3497	3566	-3576	3686			
PWRUP1	F128	3568	-3578						
RADFLG	00FB	-622							
RADON	0000	-623							
RAMLO	0004	-170	3589	3590	3591	3595	3596	3795	3798
		3800	3801	3804	3897	3899	3906	3910	3912
		3913	3915	3936	3939	3940			
RAMSIZ	02E4	-451	3655	3663	3837				
RAMTOP	006A	-273	4090	4133	4144	4207	4229	4575	5166
		5180							
RANDOM	D20A	-661							
RANGE	FA96	4290	4309	4955	4957	-4959	5522	5534	
RANGE1	FAB7	4969	-4975						
RANGE2	FABB	4974	-4977						
RANGE3	FA9E	4961	-4963						
RBLOK	EFE9	3130	3236	-3241					
RBLOKV	E47A	-80	3129	3977					
RCI1	E5A7	963	968	-973					
RCI11	E5BF	-994							
RCI1A	E574	936	-943						
RCI1B	E571	-940	944						

```
RCI2      E5AC        -979
RCI3      E587         949  -955   974
RCI4      E5C3         957   970   981   986  -998
RCI6      E5B2        -984   991
RDBAD     EE51       -2838  2845
RDBYTE    F310       -3892  3895
RDONLY    0087        -143  1008
READ      0052       -1616  3978
RECEIV    EAE0        1840  1891 -2048  2257
RECVDN    0039        -228  2055  2070  2122
RECVDS    EC60       -2371
RECVEN    EC1B        2061 -2320
REDGE     0027        -253  3624
RELONE    EAB1        1989 -1997
RENAME    0020        -100
RESET     F11B        3489  3499 -3564
RETUR1    F634        4044  4047  4057  4060  4070  4071  4074  4294
                      4364  4366 -4369  4529  4996  5334  5553
RETUR2    F621        4287  4315  4320  4323 -4361  4422  4447
RETUR3    FAE1        4994 -4996
RETURN    EA0D        1831  1851  1854 -1861  2265  2646
RIRGHI    0000       -1660  2241
RIRGLO    0078       -1655  2240
RMARGN    0053        -257  3625  4557  4562  4600  4869  4960  4962
                      5227  5246  5283
RNGER1    FAD8       -4991
RNGER2    FAD6        4986 -4990
RNGERR    FAD1        4966  4967  4973  4976  4979  4980 -4988
RNGOK     FAC4        4971 -4981
ROWAC     0070        -278  4940  4942  4943  4945  5442  5470  5476
                      5478  5480  5481  5485
ROWCRS    0054        -258  4329  4388  4407  4540  4544  4546  4547
                      4586  4602  4646  4663  4690  4701  4702  4776
                      4783  4849  4887  4904  4913  4933  4965  5053
                      5132  5134  5189  5215  5224  5236  5248  5277
                      5313  5324  5339  5342  5398  5444  5489  5491
                      5529  5533
ROWINC    0079        -283  5405  5413  5490
RRETRN    EB0E       -2081
RSIRG     000A       -1657  2236
RTCLOK    0012        -185  1393  1396  1398  1404  2573  2599  3315
                      3334  3347
S         0053       -2902  3079
SAVADR    0068        -272  4033  4185  4187  4262  4264  4654  4656
                      4668  4672
SAVIO     0316        -505  2571  2590  2592
SAVMSC    0058        -261  4172  4174  4241  4243  5160  5162  5377
                      5379
SBUFSZ    001D       -2895  3081
SCOLLP    E80E       -1433  1439
SCREDT    0045        -120
SCRENV    E410         -56  3517  3846  4052
SCRFLG    02BB        -425  4629  4644  4926
```

SCRMEM	0093	-156	4768						
SCRNOK	F1DB	3685	-3687	3688	3690				
SCROL1	FBB7	4721	-5164	5177	5181				
SCROL2	FBCA	5169	5172	-5174					
SCROLL	FBAC	4925	-5159						
SDLSTH	0231	-339	1423						
SDLSTL	0230	-338	1425	4253	4255	4266	4268		
SDMCTL	022F	-337	1427	4111	4285	4286			
SECT1	F301	-3885	4003						
SECTX	F34C	-3919	3924						
SEND	EA6B	-1940	2225	2443					
SENDDS	EC5F	1861	1964	-2370	2632				
SENDEN	EBF2	1692	1943	2207	-2284				
SENDEV	E468	-74	1691	3205					
SENDIN	EC8A	1779	1807	-2436					
SERIN	D20D	-663	2114	2131					
SEROUT	D20D	-677	1952	1992	2010				
SETBSZ	EF34	3074	3078	-3082					
SETDCB	EEE6	2961	3004	-3034					
SETLOP	E8F7	-1561	1562						
SETTAB	F82D	-4621	5685						
SETVBL	E8ED	1224	1226	-1556					
SETVBV	E45C	-70	1225	2660	3178	3217			
SETVBX	EDB9	2215	2242	2253	2447	-2652			
SEX	0000	-3415	3675	3959					
SFH	EF64	-3163							
SHFAMT	006F	-277	4298	4347	4843				
SHFLOK	02BE	-428	4092	4493	4498	4503	4520		
SHIFT1	F5B1	4299	-4302						
SHIFT2	F610	4348	-4351						
SHIFTD	F5AA	-4298	4301						
SHIFTU	F608	-4347	4350						
SIDWAY	0053	-1624							
SIGNON	F223	3485	-3733						
SIN	BD81	-600							
SIO	E959	1686	-1726						
SIOINT	E944	1689	-1706						
SIOINV	E465	-73	1688	3851					
SIOORG	E944	-27	1584	1702					
SIOSB	F095	3242	-3361	3394					
SIOSPR	0014	-2727							
SIOV	E459	-69	1685	2821	2962	3005	3384		
SIRHI	00EB	-2423	2424						
SIRLO	000F	-2424							
SIZEM	D00C	-694							
SIZEP0	D008	-690							
SIZEP1	D009	-691							
SIZEP2	D00A	-692							
SIZEP3	D00B	-693							
SKCTL	D20F	-679	1716	2300	2324				
SKRES	D20A	-676	2097	2326					
SKSTAT	D20F	-665	1459	1469	2096	2568	2588	2611	2613
SOUNDR	0041	-238	1715	2340					

```
SPACE       0020       -2899  2995
SPECIA      EF4B        3119 -3145
SPECIL      000E         -95   846   848
SPECL       F23F        3582 -3781
SQR         BEB1        -603
SRETRN      EB34        2116 -2121  2156
SRSTA       0040       -3103  3377
SRTIM2      0006       -1219  1475
SRTIMR      022B        -333  1467  1472  1476  1526  5794
SRTIR0      EB9B        2211 -2214
SRTIR1      EBC1        2238 -2241
SRTIR2      EBE9        2260 -2265
SSFLAG      02FF        -473  1338  4326  5783  5785
SSKCTL      0232        -340  1714  2285  2299  2321  2323  2612
STACK     S 0000           0
STACKP      0318        -507  1727  2640
STATC       0053       -2748  2810  2825  2958  3042  3871
STATIS      000D         -94
STATU       F028        3119 -3280
STATUS      0030        -219  1846  1849  1864  1894  1911  1915  1917
                        1927  1941  2060  2074  2103  2109  2119  2638
STATVH      0002       -2736  2737  2814
STATVL      00EA       -2737  2812
STICK0      0278        -379  1487  1491  1509
STICK1      0279        -380
STICK2      027A        -381
STICK3      027B        -382
STIMER      D209        -675
STLOOP      E877       -1482  1494
STORE       F917        4190  4204  4210  4212  4214  4222  4232  4234
                        4236  4238  4242  4244  4247  4249  4251  4257
                        4267  4269 -4745
STORE1      F91D       -4748
STRBEG      FC5C        4545 -5261
STRERR      F942        4764 -4768
STRIG0      0284        -391  1498
STRIG1      0285        -392
STRIG2      0286        -393
STRIG3      0287        -394
STRL        E890       -1497  1504
STROK       F946        4746  4754  4763  4767 -4770
STTMOT      EC75        1815  2252 -2396
SUBBFL      E677         998  1059 -1138
SUBEND      FA7A       -4939  5493  5521
SUBTMP      029E        -414  4752  4757
SUCCES      0001        -134   898  1850  1895  1940  2059  2993  3016
                        3226  3239  3266  3280  3289  4112  4370  4981
SUSUAL      EB38       -2124  2148  2162
SV7H        00E8       -1523  1524
SV7L        0073       -1524
SWAP        FCB3        4384  4398  4418  4424  4429  4436  4448 -5333
SWAP1       FCC2       -5339  5346
SWAP3       FCD7        5337 -5350
```

SWAPA	FCB9	4995	5333	-5335					
SWPFLG	007B	-285	4121	4907	4993	5221	5347	5349	5366
SWSTA	0080	-3104	3380						
SYIRQ	E706	1243	-1301						
SYIRQ2	E71B	1304	-1310						
SYIRQ8	E762	1322	-1343						
SYIRQ9	E76F	1346	-1349						
SYIRQA	E77A	1350	-1353						
SYIRQB	E78F	1235	1236	1237	1240	1241	1242	-1364	
SYRTI	E790	1234	-1365						
SYRTI2	E78B	1358	-1362						
SYSVB1	E7BA	1394	1397	-1399					
SYSVB2	E7D6	1410	-1412						
SYSVB3	E7E5	1417	-1419						
SYSVB4	E832	1446	-1448						
SYSVB6	E8C3	1471	-1525						
SYSVB7	E873	1468	1473	-1480	1523	1524	1527		
SYSVBA	E844	1453	-1456						
SYSVBB	E834	-1449	1457						
SYSVBL	E7AE	1227	1247	-1393					
SYSVBV	E45F	-71							
SYVB6A	E857	1461	1464	-1467					
TAB	F810	-4609	4619	5675					
TAB1	F823	4612	4615	-4617					
TAB2	F82A	4616	-4620						
TABMAP	02A3	-419	4125	5041	5043	5049	5050	5058	
TBLENT	F0E3	-3510	3538	3641					
TBLLEN	000E	-3538	3640						
TDHI	00EA	-2427	2428						
TDLO	00CF	-2428							
TEMP	023E	-354	1667	1668	1901				
TEMP1	0312	-502	2478	2496					
TEMP2	0314	-503							
TEMP3	0315	-504	2578	2596					
TEMPHI	0002	-1667	1668	1884					
TEMPLO	003E	-1668	1880						
TIMER1	030C	-497	2472	2474	2477	2481	2574	2575	
TIMER2	0310	-500	2468	2469	2471	2475	2479		
TIMFLG	0317	-506	2068	2220	2247	2274	2564	2583	2662
TIMIT	EBA5	-2220	2221						
TIMIT1	EBCB	-2247	2248						
TIMOUT	008A	-146	1918	2073					
TINDEX	0293	-409	4119						
TMPCHR	0050	-254	4352	4356	5097	5100			
TMPCOL	02B9	-424							
TMPLBT	02A1	-417	4818	4827	4831				
TMPROW	02B8	-423	5314	5323					
TMPX1	029C	-412							
TOADR	0066	-4034	4685	4689	5096	5099	5109	5111	5113
TONE1	0002	-3113	3196						
TONE2	0001	-3114	3163						
TOUT	EB0A	2069	-2073	2586					
TOUT1	ED44	2566	-2585						

```
TRAMSZ    0006        -171  3597  3654  3668  3697  3700  3715  3797
                      3805  3808  3836
TRIG0     D010        -730  1497
TRIG1     D011        -731
TRIG2     D012        -732
TRIG3     D013        -733
TRNRCD    0089        -145   994
TSTAT     0319        -508  1845  1928
TSTCT1    FC8F       -5299  5305
TSTCT2    FC9C        5301 -5306
TSTCTL    FC8D        4432  4524 -5298
TSTDAT    0007        -172  3653  3660  3698  3703  3721
TWICE     EE4F       -2836  2840
TXTCOL    0291        -408  4283
TXTMSC    0294        -410  4135  4137
TXTOLD    0296        -411
TXTROW    0290        -407  4281  5341  5344
UNLOCK    0024        -104
UPDNCM    F787       -4544  4551
USAREA    0480        -548
VBATRA    E7C8        1402 -1406
VBREAK    0206        -314  1361
VBWAIT    F496       -4175  4177
VCOUNT    D40B        -745  2572  2598  4175
VCTABL    E480         -24  1232  1694  3630  4081
VDELAY    D01C        -710
VDSLST    0200        -311  1232  1374
VECTBL    E400         -23
VIMIRQ    0216        -322  1281
VINTER    0204        -313  1352
VKEYBD    0208        -315  1293  4081
VPRCED    0202        -312  1348
VSCROL    D405        -741
VSERIN    020A        -316  1299  1309  1694
VSEROC    020E        -318  1297
VSEROR    020C        -317  1298
VTIMR1    0210        -319  1296
VTIMR2    0212        -320  1295
VTIMR4    0214        -321  1294
VVBLKD    0224        -329  1522
VVBLKI    0222        -328  1385
WAIT      EA1A       -1876  2449
WAITER    EC9B        1820 -2447
WAITTM    EF7C       -3179  3180
WARMST    0008        -175  3578  3584  3588  3862  3989
WARMSV    E474         -78  1379  3487
WATCOM    E9D7        1795 -1815
WCI1      E605        1034  1039 -1044
WCI1A     E5D4        1005 -1012
WCI1B     E5D1       -1009  1013
WCI2      E60A       -1050
WCI3      E5E5        1018 -1024  1045
WCI4      E5EB        1021 -1027
```

WCI5	E615	1028	1041	1052	-1059	
WDLR	EFC6	-3220	3223			
WFAK	F087	3337	-3350			
WFAK1	F08C	3350	-3353			
WFL	F060	-3324	3335			
WIRGHI	0000	-1659	2214			
WIRGLO	00B4	-1654	2213			
WMODE	0289	-397	3161	3191	3195	3286
WOK	EA3D	-1901				
WRITE	0057	-1617	2807			
WRITEC	0057	-2898	3069			
WRONLY	0083	-139	939			
WSIOSB	F0D2	3272	3297	3305	-3389	
WSIRG	000F	-1656	2209			
WSYNC	D40A	-744	1560	5357		
WTLR	F046	3294	-3299			
XBOOT	F361	3926	-3928			
XITVBL	E905	1228	1248	1418	-1571	
XITVBV	E462	-72				
XMTDON	003A	-229	1948	1961	2029	
XXIT	E7E2	1413	-1418			
ZERIT	EC6D	-2378	2381			
ZERORM	F138	-3587				
ZIOCB	0020	-199				
ZOSRAM	F160	3585	-3610			
ZOSRM2	F163	-3612	3615			
ZOSRM3	F16E	-3617	3619			
ZTBUF	F04A	-3301	3303			
ZTEMP1	00F5	-618				
ZTEMP3	00F9	-620				
ZTEMP4	00F7	-619				

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 <u>ATARI 400/800 Basic Reference Manual</u>.

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.

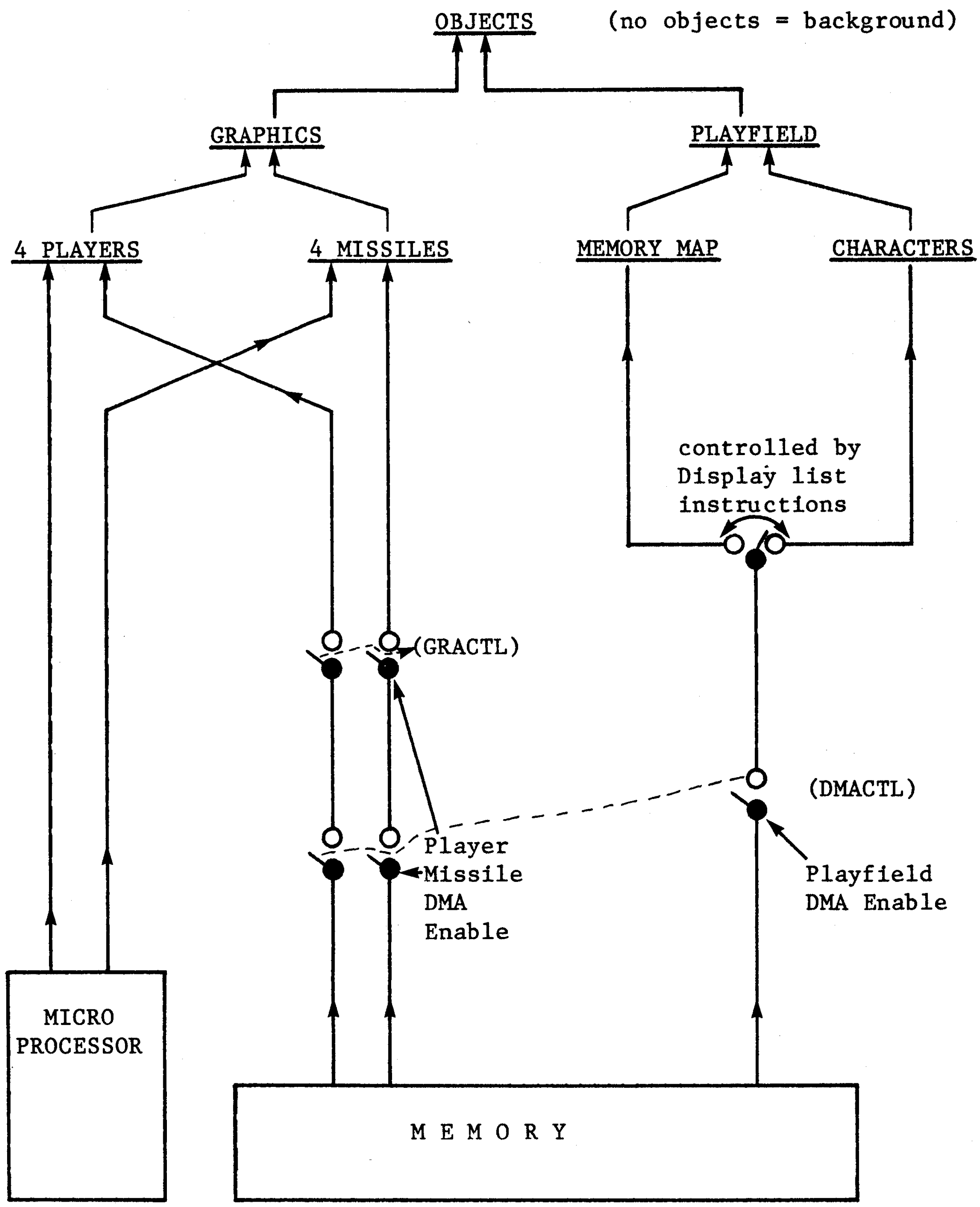


Figure II.2 <u>O B J E C T D I S P L A Y S O U R C E S</u>

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.

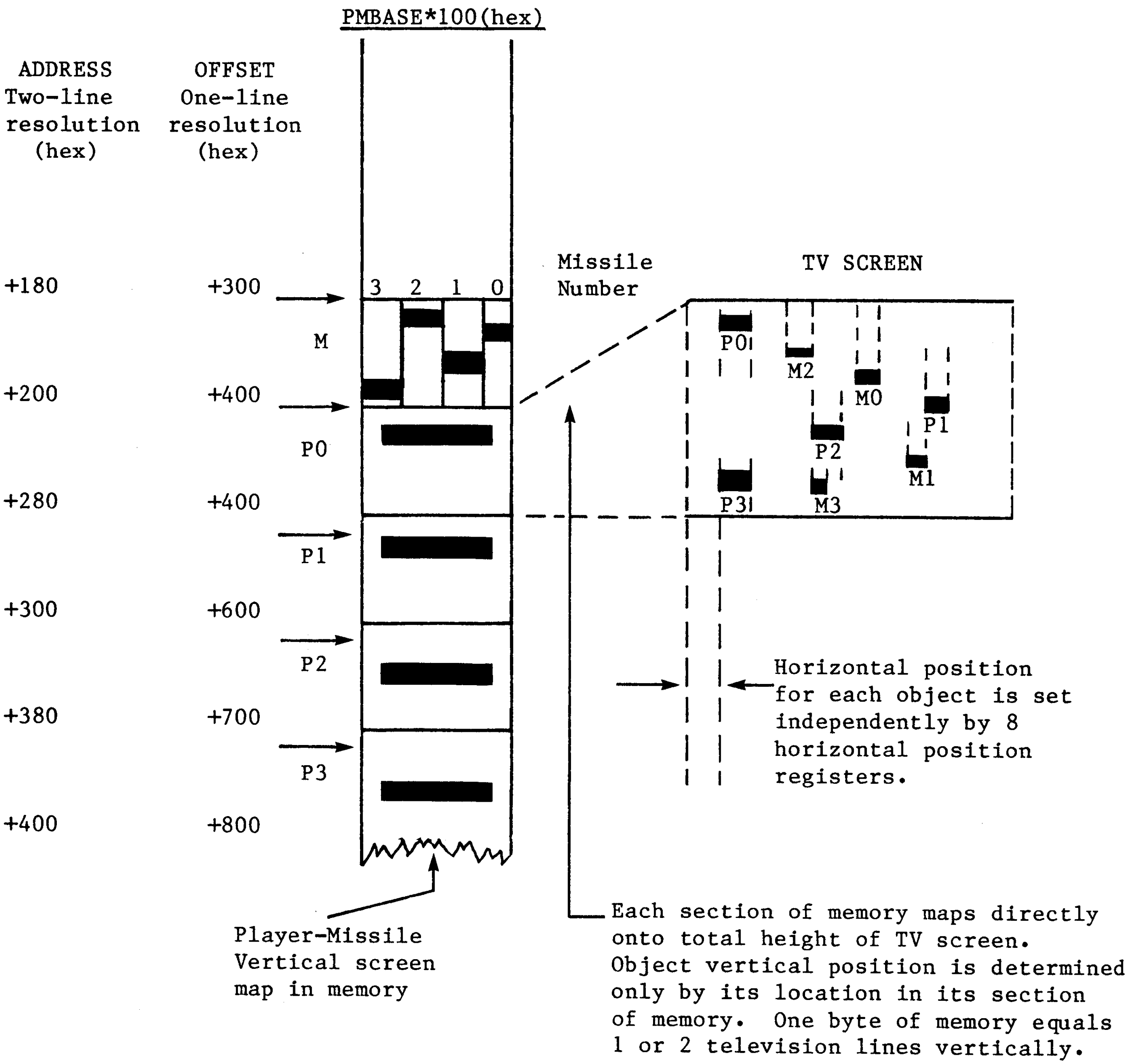


Figure II.2 P L A Y E R - M I S S I L E D M A

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.

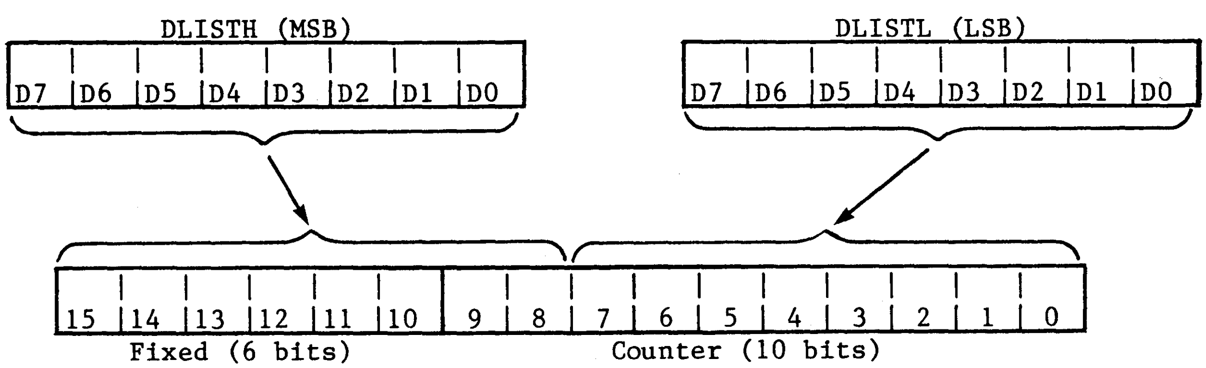
There are three different playfield widths: narrow (128 color clocks), standard (160 color clocks), and wide (192 color clocks). The width is selected by storing into DMACTL. The advantage of a narrower width is that less RAM is required and fewer machine cycles are stolen for DMA. The OS graphics modes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

$\boxed{\text{Opcode}}$ ------Single Byte Display Instruction

$\left.\begin{array}{l}\boxed{\text{Opcode}}\\ \boxed{\text{Operand}}\\ \boxed{\text{Operand}}\end{array}\right\}$----Triple Byte Display Instruction

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
CHR (40,2,8)	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2	(also 3 byte)
" (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
" (40,4,16)	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5	Instructions
" (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
" (160,2,1)	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC	Instructions
" (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).

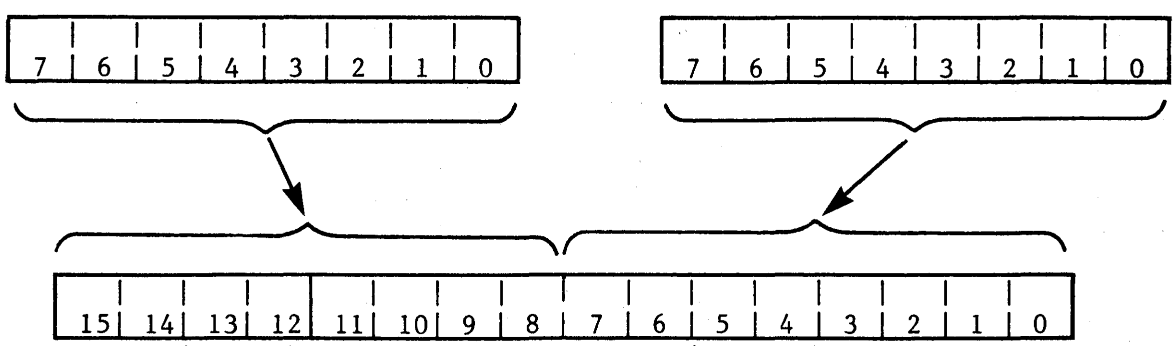
Memory Scan Counter: This counter is not directly accessible by the programmer. It is loaded with the value in the last 2 bytes of a 3 byte (non-Jump) instruction.

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.

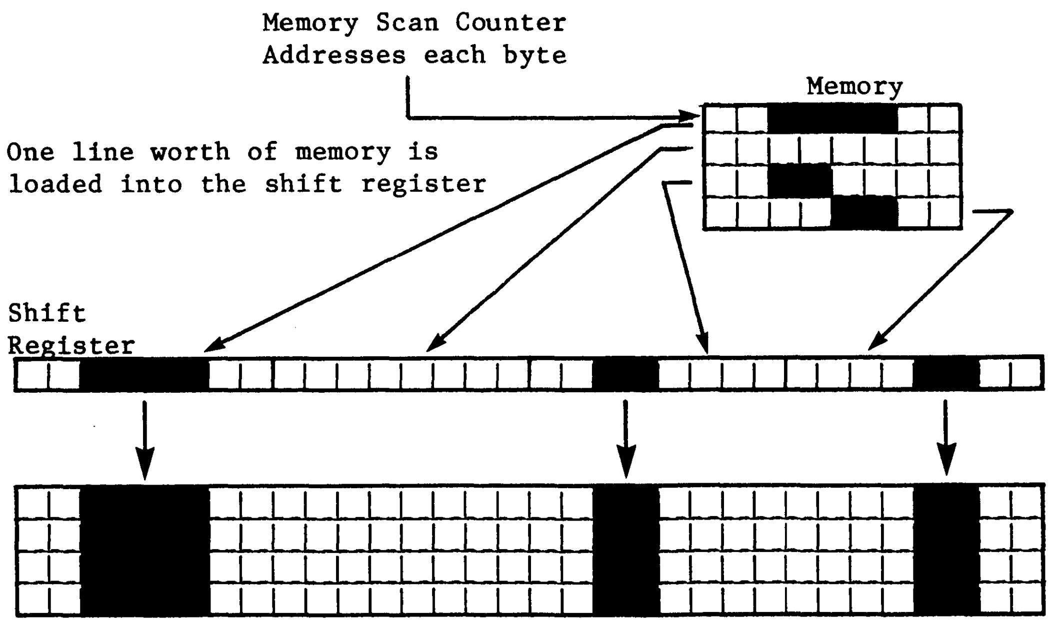
MSB third byte of 3 byte byte instruction

LSB Second byte of 3 byte instruction



Fixed (4 bits) Counter (12 bits)

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.



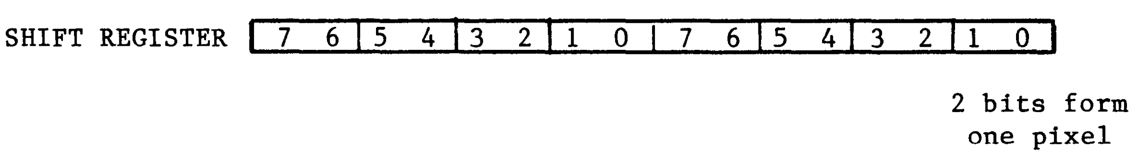
Shift register data is displayed for four TV scan lines in this example.

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.



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.

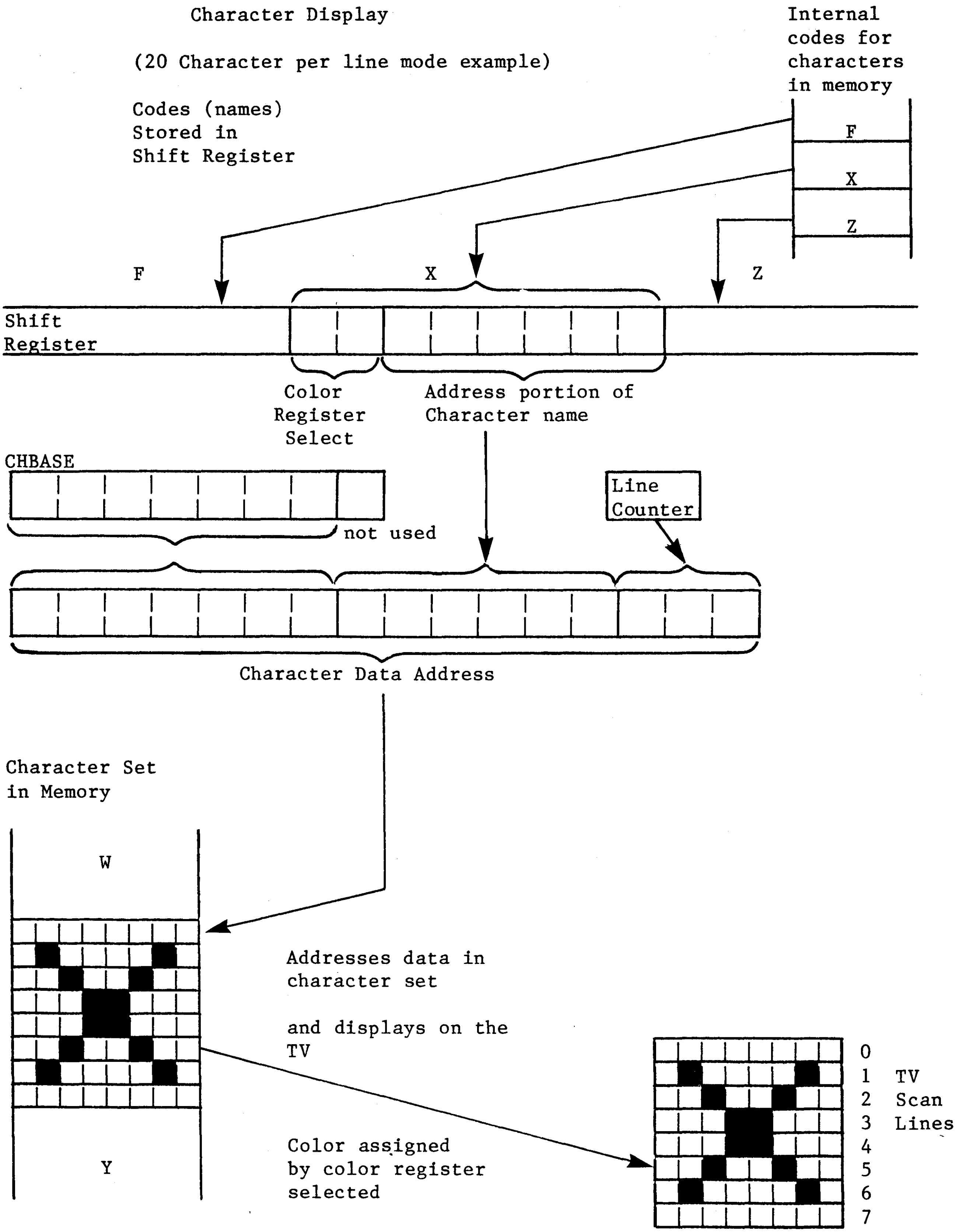
Character names (codes) are fetched by the memory scan counter, and are placed in a shift register. On any given line of display the shift register rotates, changing only the name portion of the character address, as shown below.

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
Line
Counter
not used
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).

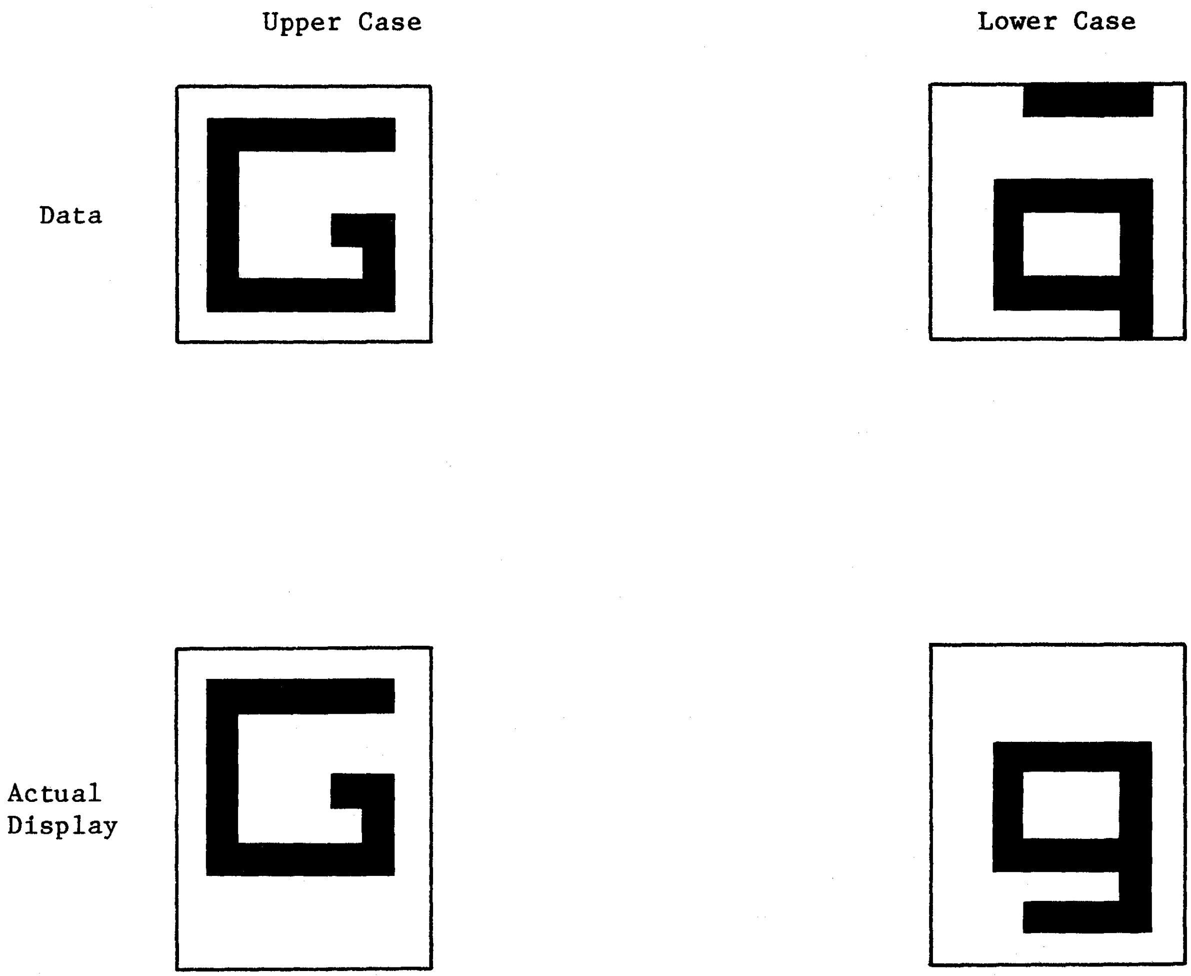
In IR modes 2 and 3, bit 7 of the character name is used for inverse video or blanking. This is controlled by CHACTL (Character Control). If bit 2 of CHACTL is a one then all of the characters will be displayed upside down, regardless of mode. If CHACTL bit 1 is set, then each character which has bit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



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.

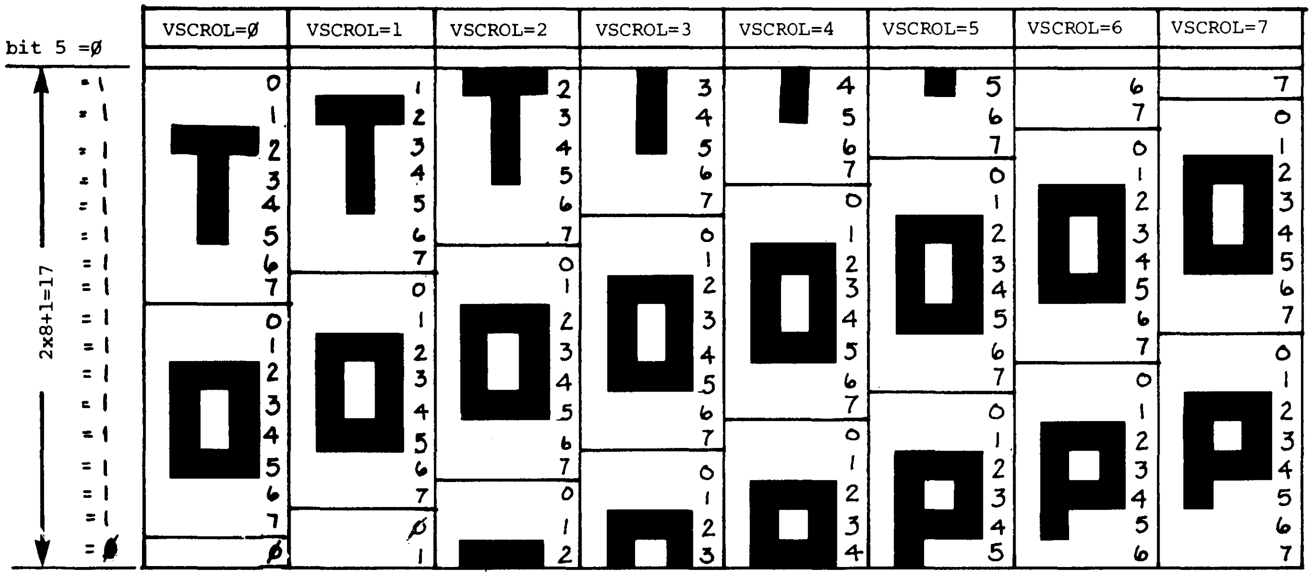
Vertical Scrolling: A zone of playfield on the screen can be scrolled upward by using VSCROL and bit 5 of the display list instruction. The display blocks at the upper and lower boundaries of the zone must have a variable vertical size. In particular, the first display block within that zone must be 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	
	70	24 blank lines
	70	
	42	reload memory scan counter with 7C40
	40	IR mode 2
	7C	
	2	
	2	
	2	
	.	
	.	23 more IR mode 2 instructions
	.	
	2	
	2	
	2	
	41	Jump back to 7C20 and
	20	wait for end of vertical blank.
	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.

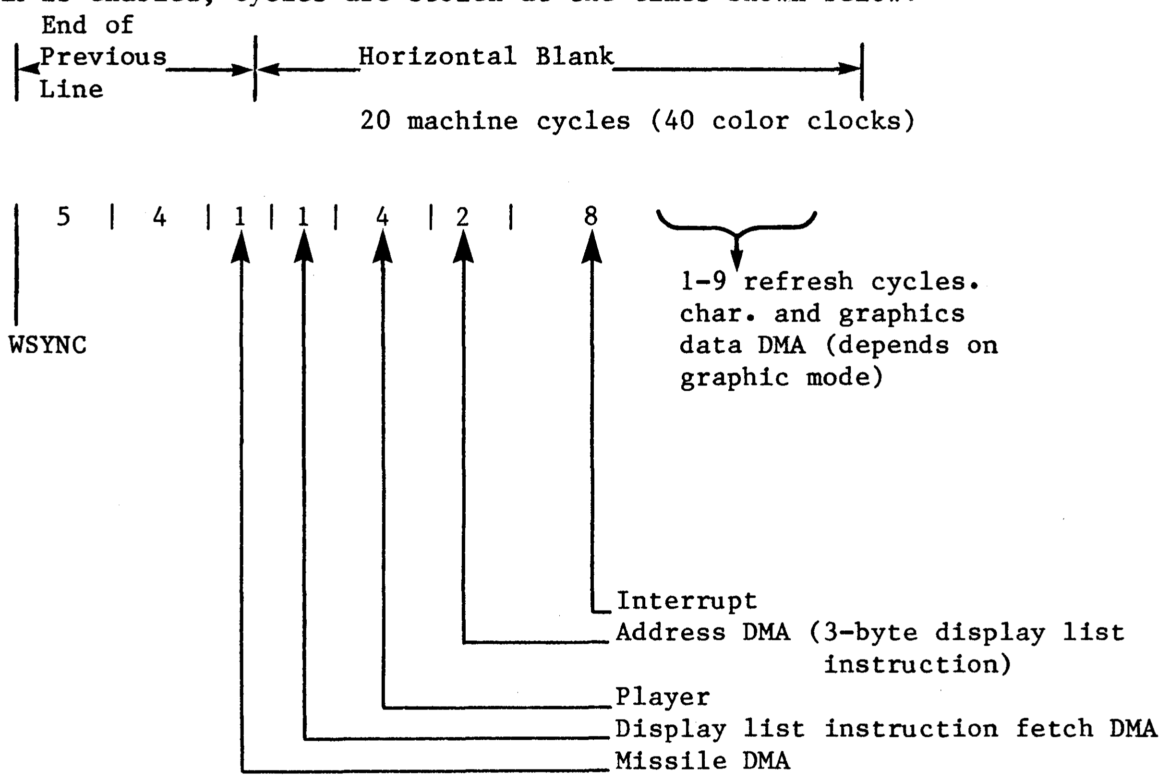
ANTIC does horizontal scrolling of an even number of color clocks by delaying the time at which it DMA's the data. To do an odd number of color clocks (which involves half of a machine cycle), ANTIC has a one color clock internal delay.

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

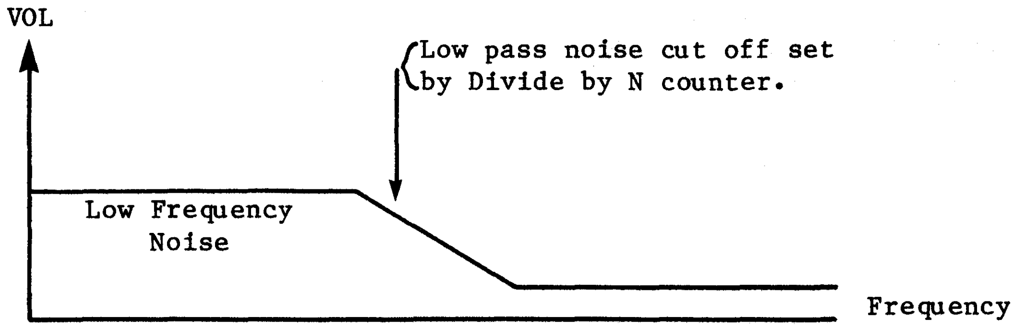
Audio: There are 4 semi-independent audio channels, each with its own frequency, noise, and volume control. Each has an 8 bit "divide by N" frequency divider, controlled by an 8 bit register (AUDFX). (See audio-serial port block diagram.) Each channel also has an 8 bit control register (AUDCX) which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

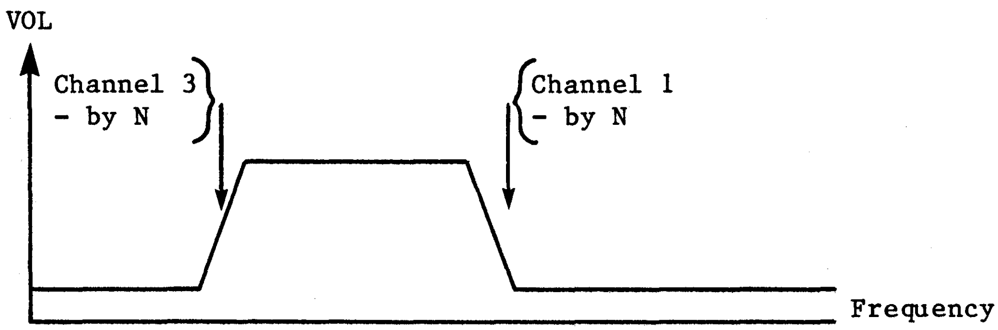
Poly Noise Counters: There are 3 polynomial counters (17 bit, 5 bit and 4 bit) used to generate random noise. The 17 bit poly counter can be reduced to 9 bits (AUDCTL bit 7). These counters are all clocked by 1.79 MHZ. Their outputs, however, can be sampled independently by the four audio channels at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

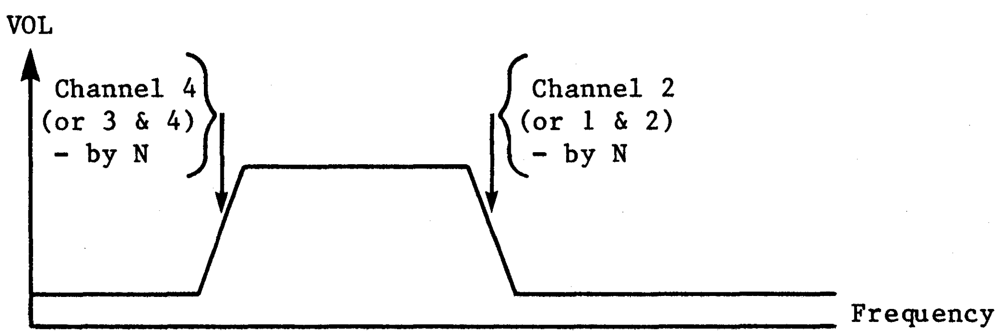
Audio Noise Filters:



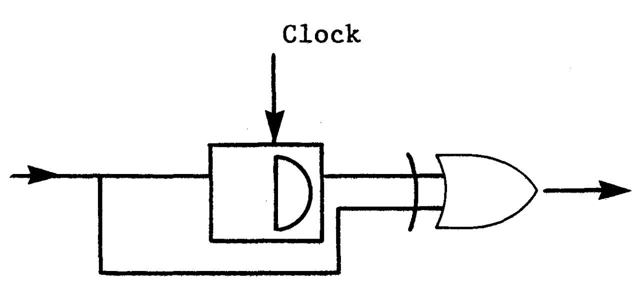
Any channel noise output (without high pass filter)



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.

<u>Serial Output Clock</u>: 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.)

<u>Serial Input</u>: 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.

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

<u>Bi-Directional Clock</u>: 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.

<u>Asynchronous Serial Input</u>: 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	D6	D5	D4	D3	D2	D1	D0

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).

D7	D6	D5	D4	D3	D2	D1	D0		
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):

not used

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

D7	D6	D5	D4	D3	D2	*	CHBASE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Base Address — Char Name — Line Counter

20 Character Modes

D7	D6	D5	D4	D3	D2	D1	*	CHBASE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Base Address — Char Name — Line Counter

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

D7	D6	D5	D4	D3	not used	PMBASE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Base Address — Player-Missile Select — Player-Missile Scan Counters

Two Line Resolution

D7	D6	D5	D4	D3	D2	*	PMBASE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Base Address — Player-Missile Select — Player-Missile Scan Counter

---

* = Not Used

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

not used	D3	D2	D1	D0

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.

not used	D2	D1	D0

8 line display modes

not used	D3	D2	D1	D0

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	D5	D4	D3	D2	D1	D0

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.

Higher Priority ↑

D3=1	D2=1	D1=1	D0=1
PF0	PF0	P0	P0
PF1	PF1	P1	P1
P0	PF2	PF0	P2
P1	PF3 + P5	PF1	P3
P2	P0	PF2	PF0
P3	P1	PF3 + 5	PF1
PF2	P2	P2	PF2
PF3 + P5	P3	P3	PF3 + 5
BAK	BAK	BAK	BAK

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.

D7	D6	D5	D4	D3	D2	D1	D0
L	R	L	R	L	R	L	R
M3		M2		M1		M0	

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.

D7	D6	D5	D4	D3	D2	D1	D0
M3		M2		M1		M0	

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.

D7	D6	D5	D4	D3	D2	D1	D0

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:

$$\text{Fout} = \text{Fin}/2\text{N}$$

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:

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

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	Divisor "N" set by audio frequency register.
0	0	0	0	0					- 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	AUDF 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)

not used

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).

D7	D6	D5	D4	D3	D2	D1	D0

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.

D7	D6	D5	D4	D3	D2	D1	D0

(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)

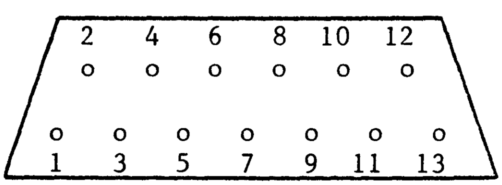
SKRES (Reset above Status Register)(D20A): This write address resets bits 7, 6, and 5 of the Serial Port-Keyboard Status Register to 1.

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).

D7	D6	D5	D4	D3	D2	D1	D0

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

D7	D6	D5	D4	D3	D2	D1	D0
Right	Left	Back	Fwd.	Right	Left	Back	Fwd.
Stick1 (Jack 2)				Stick0 (Jack 1)			

0=Switch pressed
1=Switch not pressed

Paddle Operation

D7	D6	D5	D4	D3	D2	D1	D0
PTRIG3	PTRIG2			PTRIG1	PTRIG0		

0=Switch pressed
1=Switch not pressed

Keyboard Controller Operation

D7	D6	D5	D4	D3	D2	D1	D0

D0 — Top Row, D1 — 2nd Row, D2 — 3rd Row, D3 — 4rd Row: Jack 1

D4 — Top Row, D5 — 2nd Row, D6 — 3rd Row, D7 — 4th Row: Jack 2

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: 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.

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

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

Joystick Operation

D7	D6	D5	D4	D3	D2	D1	D0
Right	Left	Back	Fwd.	Right	Left	Back	Fwd.
Stick3 (Jack 4)				Stick2 (Jack 3)			

0=Switch pressed
1=Switch not pressed

Paddle Operation

D7	D6	D5	D4	D3	D2	D1	D0
PTRIG7	PTRIG6			PTRIG5	PTRIG4		

0=Switch pressed
1=Switch not pressed

Keyboard Controller Operation

D7	D6	D5	D4	D3	D2	D1	D0
4th Row	3rd Row	2nd Row	Top Row	4rd Row	3rd Row	2nd Row	Top Row
Jack 4				Jack 3			

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
7	6	5	4	3	2	1	0

Pot number: (row above)

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).

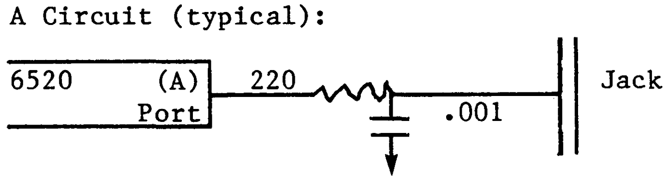
D7	D6	D5	D4	D3	D2	D1	D0	
LP8	7	6	5	4	3	2	1	0 LP0 not read. Two line resolution supplied.

OS SHADOW: LPENV (hex 235)

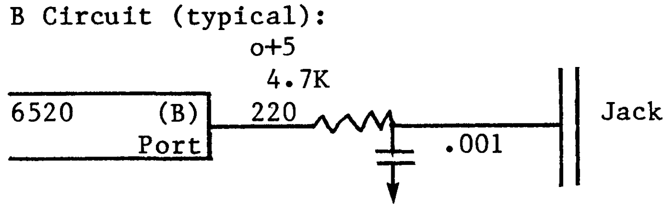
Front Panel (Controller) Jacks as I/O Parts:

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

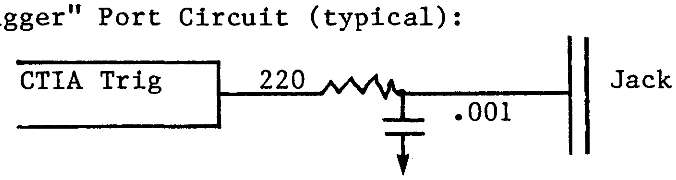
Port A Circuit (typical):



Port B Circuit (typical):



"Trigger" Port Circuit (typical):



Controller Port Pinout:

Male (console) Female (connector)

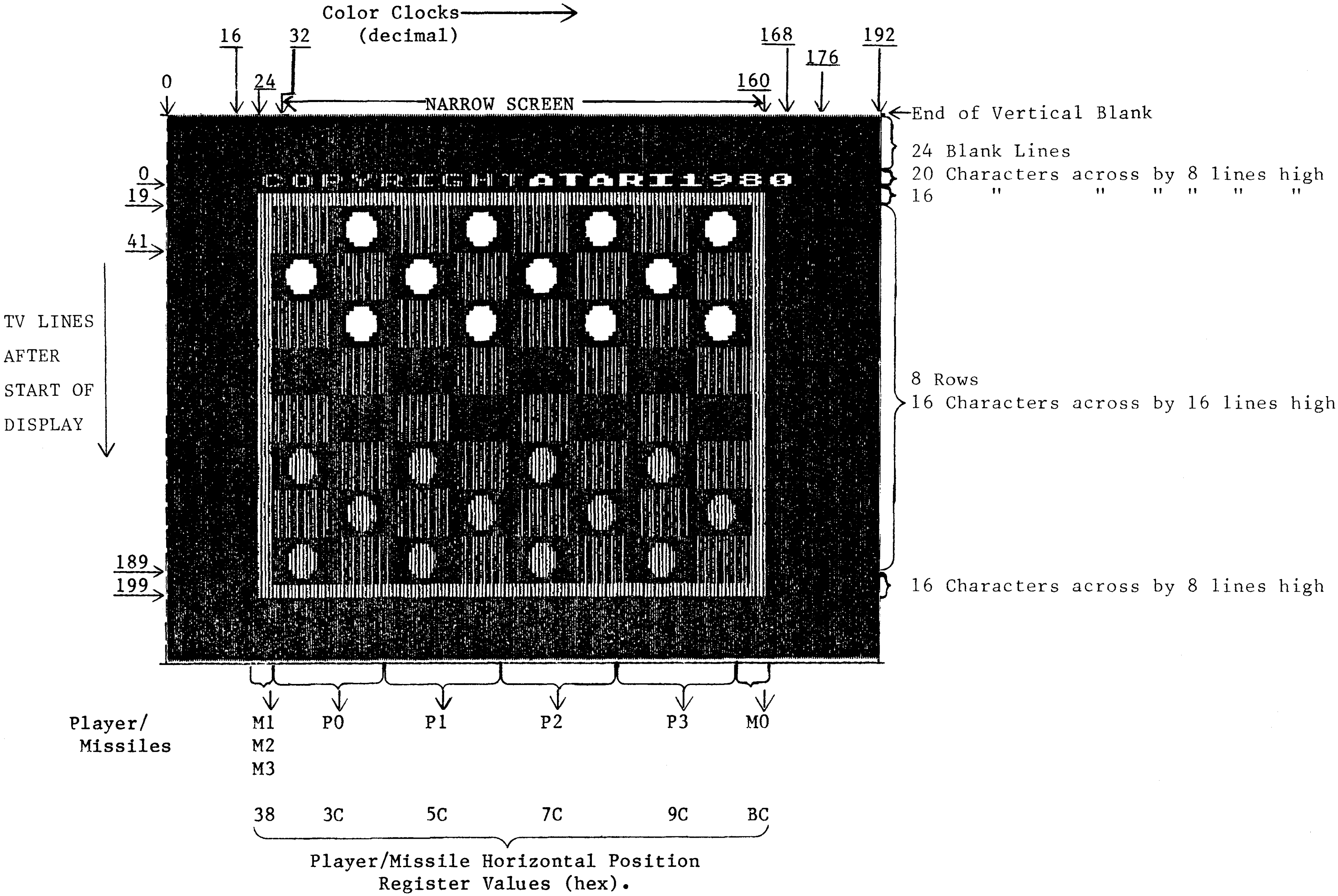
	Controllers				
PIN	JOYSTICK	PADDLE (POT)	KEYBOARD	HARDWARE REGISTERS	OS VARIABLES
1	Forward		Top Row*	Bit 0 or 4**	Bit 0***
2	Back		2nd Row*	Bit 1 or 5**	Bit 1***
3	Left	A(Left)Trigger	3rd Row*	Bit 2 or 6**	PTRIG0,2,4,6 Bit 2***
4	Right	B(Right)Trigger	Bottom Row*	Bit 3 or 7**	PTRIG1,3,5,7 Bit 3***
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 3/31/80"
              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 ; RAM 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                   PAGE
               ;
               ;INITIALIZATION CODE -- START EXECUTION HERE
               ;
5652                 *=   PMB+$700
               ;
               ;INIT OS'S DMACTL VARIABLE
               ;INSTRUCTION FETCH DMA ENABLE, P/M 2 LINE RES, P/M DMA ENABLE, STANDARD SCREEN (160 CLOCKS)
               ;
5700 A92E            LDA  #$2E
5702 8D2F02          STA  SDMCTL
               ;
               ;CLEAR RAM
               ;
5705 A900            LDA  #0
5707 AA              TAX
               INITLP
5708 9D0050          STA  PMB,X
570B 9D0051          STA  PMB+$100,X
570E 9D0052          STA  PMB+$200,X
5711 9D0053          STA  PMB+$300,X
5714 9D0054          STA  PMB+$400,X
5717 E8              INX
5718 D0EE            BNE  INITLP
               ;
               ;INITIALIZE MISSILE GRAPHICS FOR BORDERS
               ;
571A A90E            LDA  #$0E
571C A05E            LDY  #$5E
571E 999451    LQPZ  STA  GRM03+$14,Y
5721 88              DEY
5722 D0FA            BNE  LQPZ
               ;
               ;INITIALIZE TOP AND BOTTOM BORDERS.
               ;
5724 A010            LDY  #16
5726 A9C4            LDA  #CLBOR+BORDER
5728 991354    TBLP  STA  TOPBRD-1,Y
572B 99A354          STA  BOTBRD-1,Y
572E 88              DEY
572F D0F7            BNE  TBLP      ;CONTINUE UNTIL Y=0
               ;
               ;INITIALIZE PLAYER GRAPHICS FOR SQUARES (CHECKER BOARD) Y=0
               ;
5731 A9F0            LDA  #$F0
5733 A20A      IN2   LDX  #10
5735 991852    IN3   STA  GRP0+$18,Y
5738 999852          STA  GRP1+$18,Y
573B 991853          STA  GRP2+$18,Y
573E 999853          STA  GRP3+$18,Y
               ;
```

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 A901    2650        LDA  #CHECKER+CLP0 ;HUMAN PIECES ON SQUARES 0-11
5791 9D0050  2660        STA  BOARD,X
5794 A981    2670        LDA  #CHECKER+CLP1 ;COMPUTER PIECES ON SQUARES 20-31
5796 9D1450  2680        STA  BOARD+20,X
5799 CA      2690        DEX
579A 10F3    2700        BPL  BRDLP
             2710 ;
```

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

```
              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 ;
```

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

```
                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 Trigger Buttons
D012	COLPM0	Player-Missile 0	TRIG2	
D011	GRAFM	Graphics All Missiles	TRIG1	
D010	GRAFP3	Graphics Player 3	TRIG0	
D00F	GRAFP2	Graphics Player 2	P3PL	Read Player to Player Collisions
D00E	GRAFP1	Graphics Player 1	P2PL	
D00D	GRAFP0	Graphics Player 0	P1PL	
D00C	SIZEM	Size All Missiles	P0PL	
D00B	SIZEP3	Size Player 3	M3PL	Read Missile To Player Collisions
D00A	SIZEP2	Size Player 2	M2PL	
D009	SIZEP1	Size Player 1	M1PL	
D008	SIZEP0	Size Player 0	M0PL	
D007	HPOSM3	Horz. Posit. Missile 3	P3PF	Read Player To Playfield Collisions
D006	HPOSM2	Horz. Posit. Missile 2	P2PF	
D005	HPOSM1	Horz. Posit. Missile 1	P1PF	
D004	HPOSM0	Horz. Posit. Missile 0	P0PF	
D003	HPOSP3	Horz. Posit. Player 3	M3PF	Read Missile To Playfield Collisions
D002	HPOSP2	Horz. Posit. Player 2	M2PF	
D001	HPOSP1	Horz. Posit. Player 1	M1PF	
D000	HPOSP0	Horz. Posit. Player 0	M0PF	

ANTIC ADDRESSES				
	WRITE		READ	
Address	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	Read the value of each POT
D206	AUDF4	Audio Channel 4 Frequency	POT 6	
D205	AUDC3	Audio Channel 4 Control	POT 5	
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			

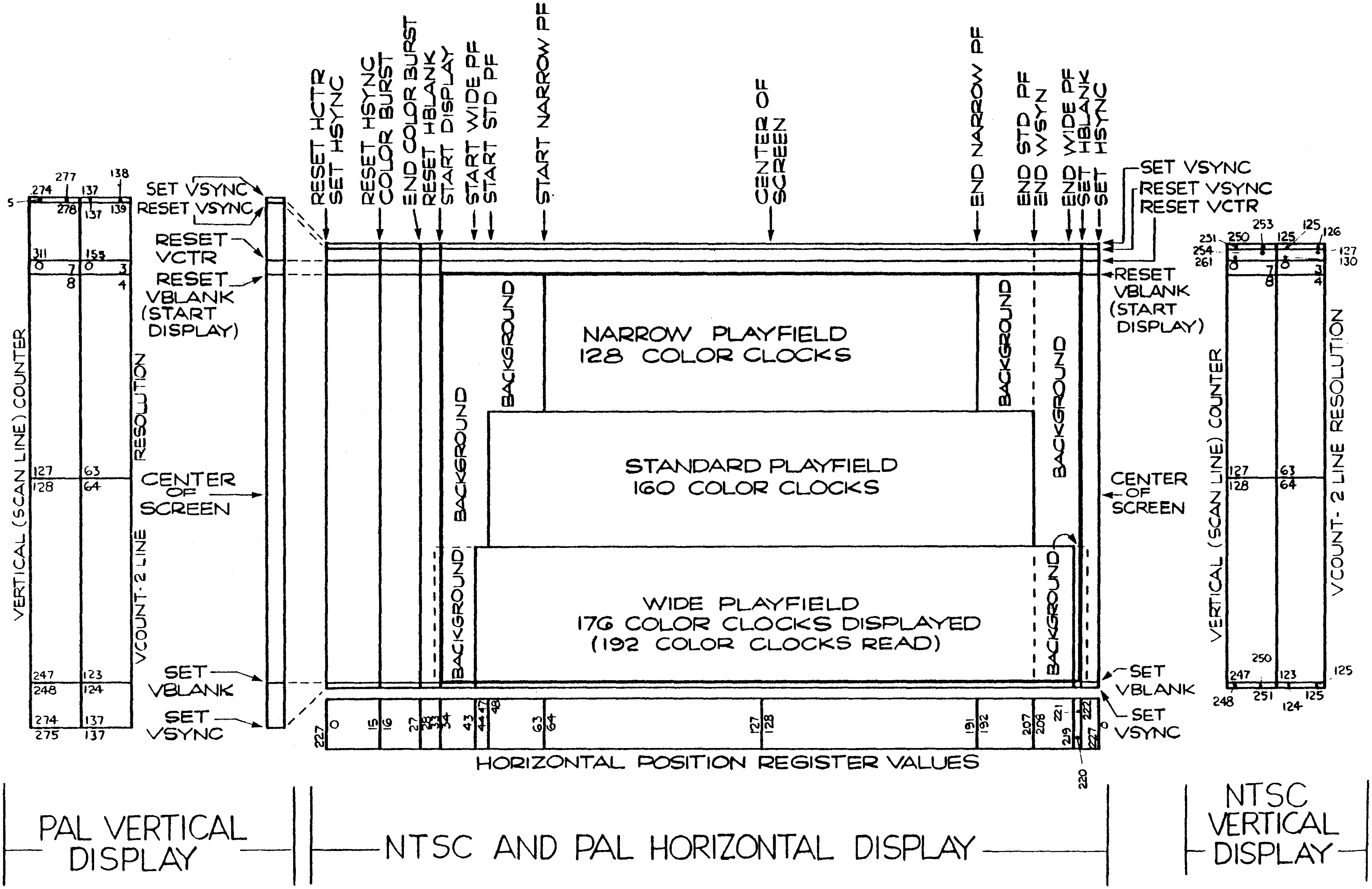
Hardware Register				OS Shadow		
		Address			Address	
Name	Description	Hex	Dec	Name	Hex	Dec
M3PF	Missile 3 to Playfield Collisions	D003	53251			
M3PL	Missile 3 to Player	D00B	53259			
NMIEN	Non-Maskable Interrupt (NMI) Enable	D40E	54286	Set to $40 by IRQ code		
NMIRES	NMI reset	D40F	54287	written to by NMI code		
NMIST	NMI Status	D40F	54287	read by NMI code		
P0PF	Player 0 to Playfield Collisions	D004	53252			
P0PL	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		

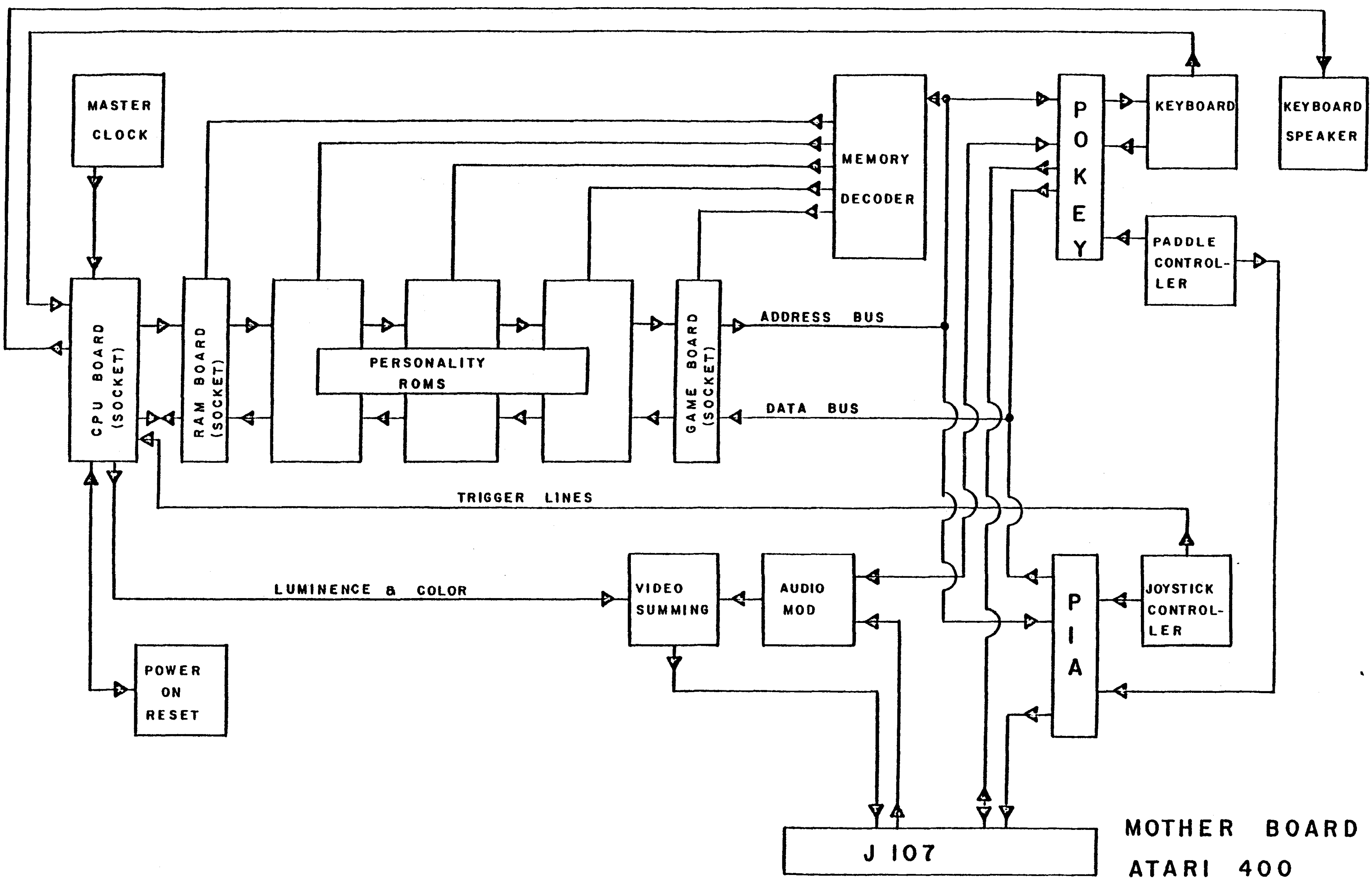
VI. FIGURES

A. MEMORY MAP

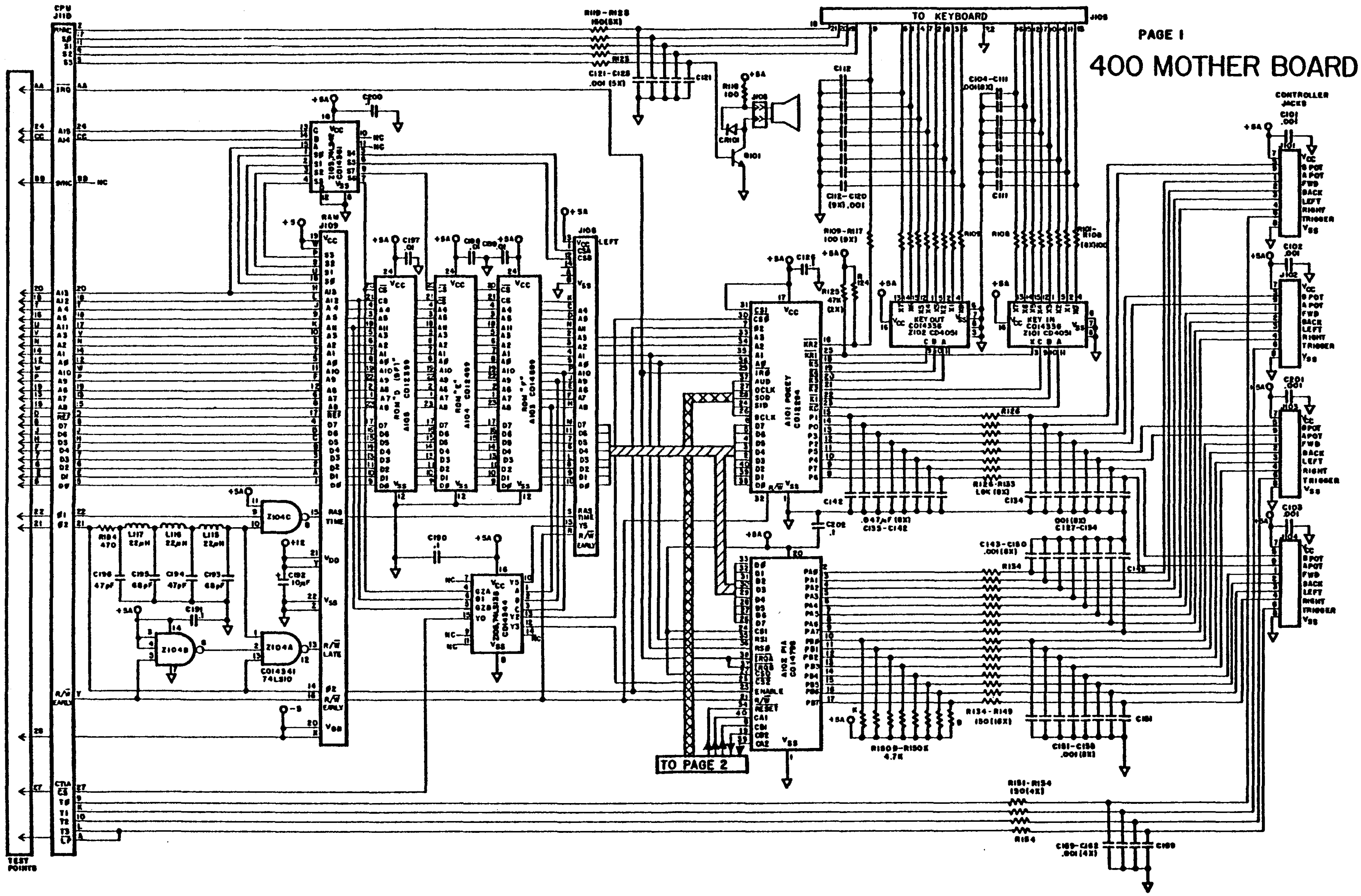
ADDRESS	FUNCTION	SIZE
FFFF D800	Operating System And Math Routines	10K
D000-D7FF	Hardware Addresses	2K
CFFF	Reserved for Future O.S. expansion	4K
BFFF 8000	ROM Cartridge (Colleen left and right slot and Candy single slot all address to this space)	16K
7FFF 2000	RAM Expansion *	
1FFF	RAM initially supplied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

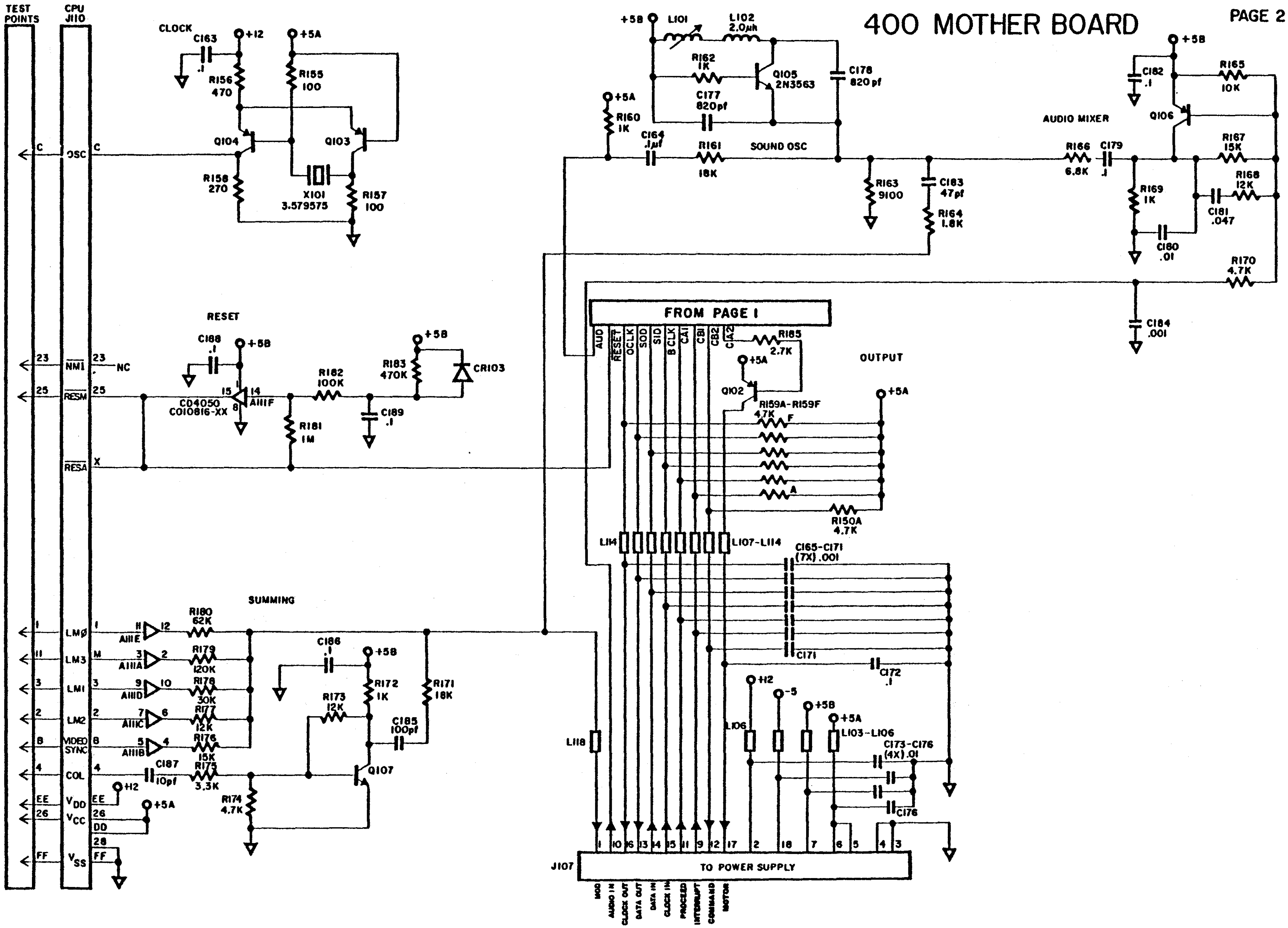

RESET HCTR
SET HSYNC
RESET HSYNC
COLOR BURST
END COLOR BURST
RESET HBLANK
START DISPLAY
START WIDE PF
START STD PF
START NARROW PF
CENTER OF SCREEN
END NARROW PF
END STD PF
END WSYN
END WIDE PF
SET HBLANK
SET HSYNC
SET VSYNC
RESET VSYNC
RESET VCTR
RESET VBLANK (START DISPLAY)
CENTER OF SCREEN
SET VBLANK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

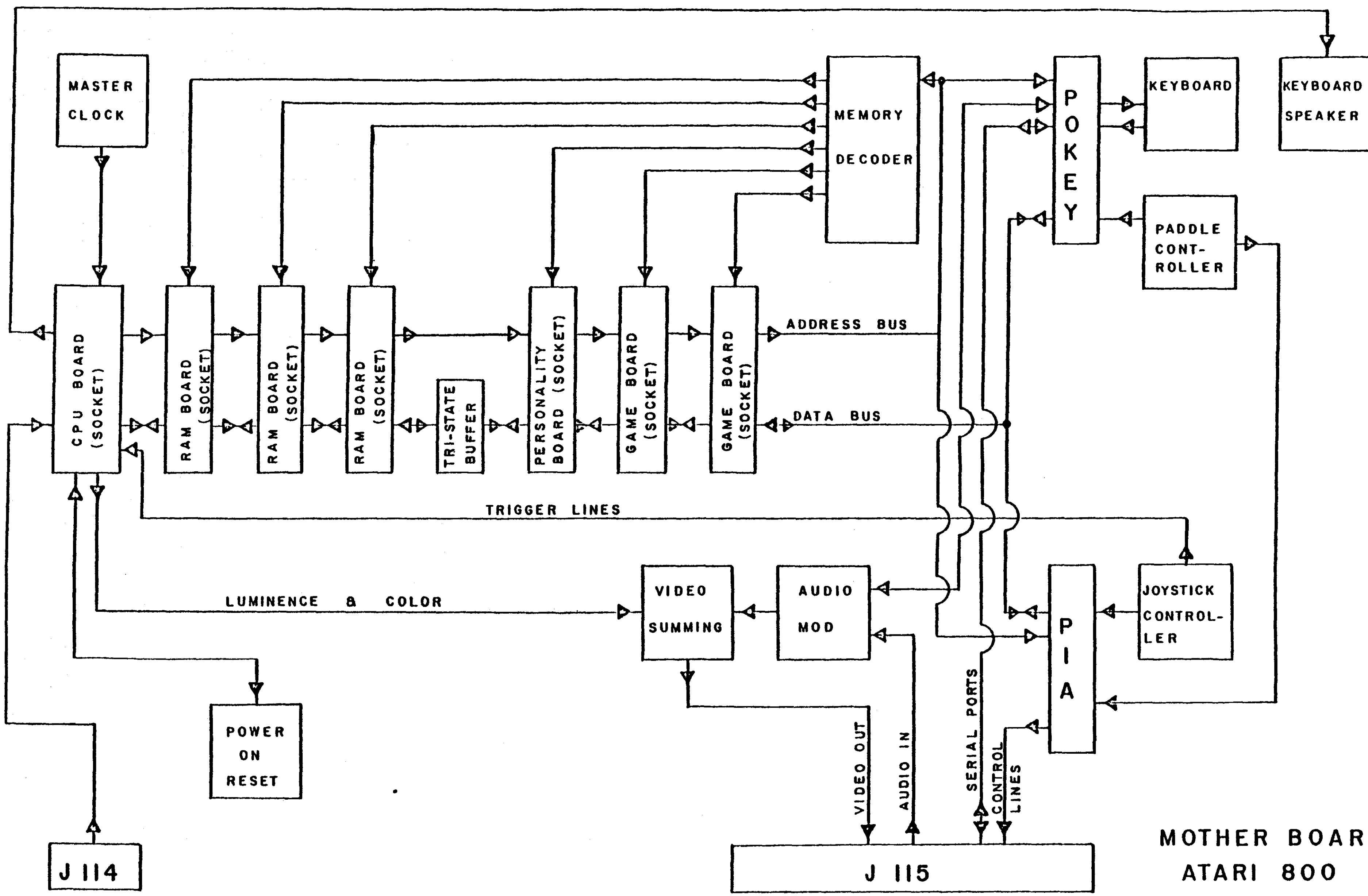

MASTER CLOCK
MEMORY DECODER
POKEY
KEYBOARD
KEYBOARD SPEAKER
PADDLE CONTROLLER
CPU BOARD (SOCKET)
RAM BOARD (SOCKET)
PERSONALITY ROMS
GAME BOARD (SOCKET)
ADDRESS BUS
DATA BUS
TRIGGER LINES
LUMINENCE & COLOR
VIDEO SUMMING
AUDIO MOD
PIA
JOYSTICK CONTROLLER
POWER ON RESET
J 107
MOTHER BOARD
ATARI 400

400 MOTHER BOARD



400 MOTHER BOARD

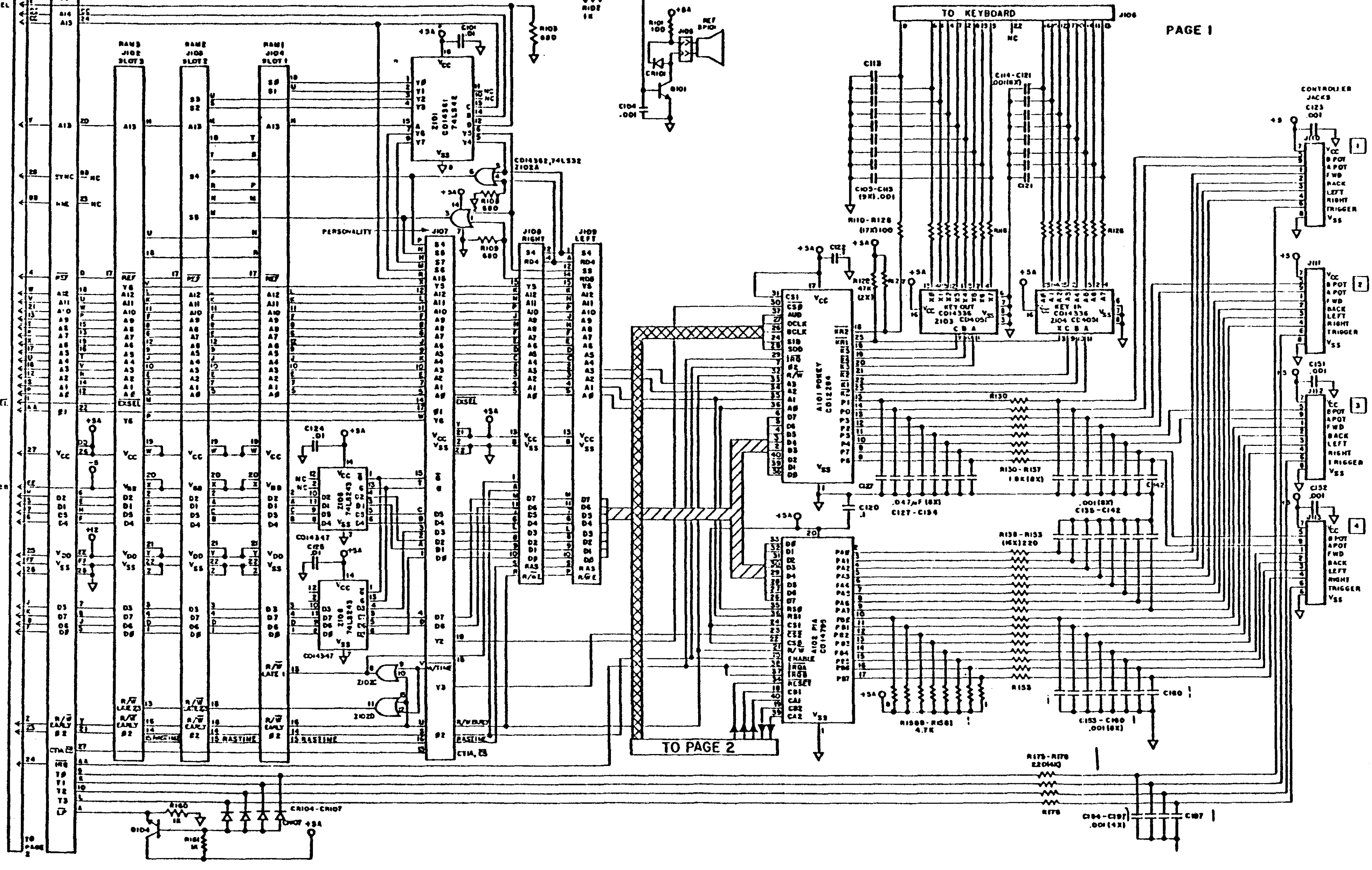




MOTHER BOARD
ATARI 800

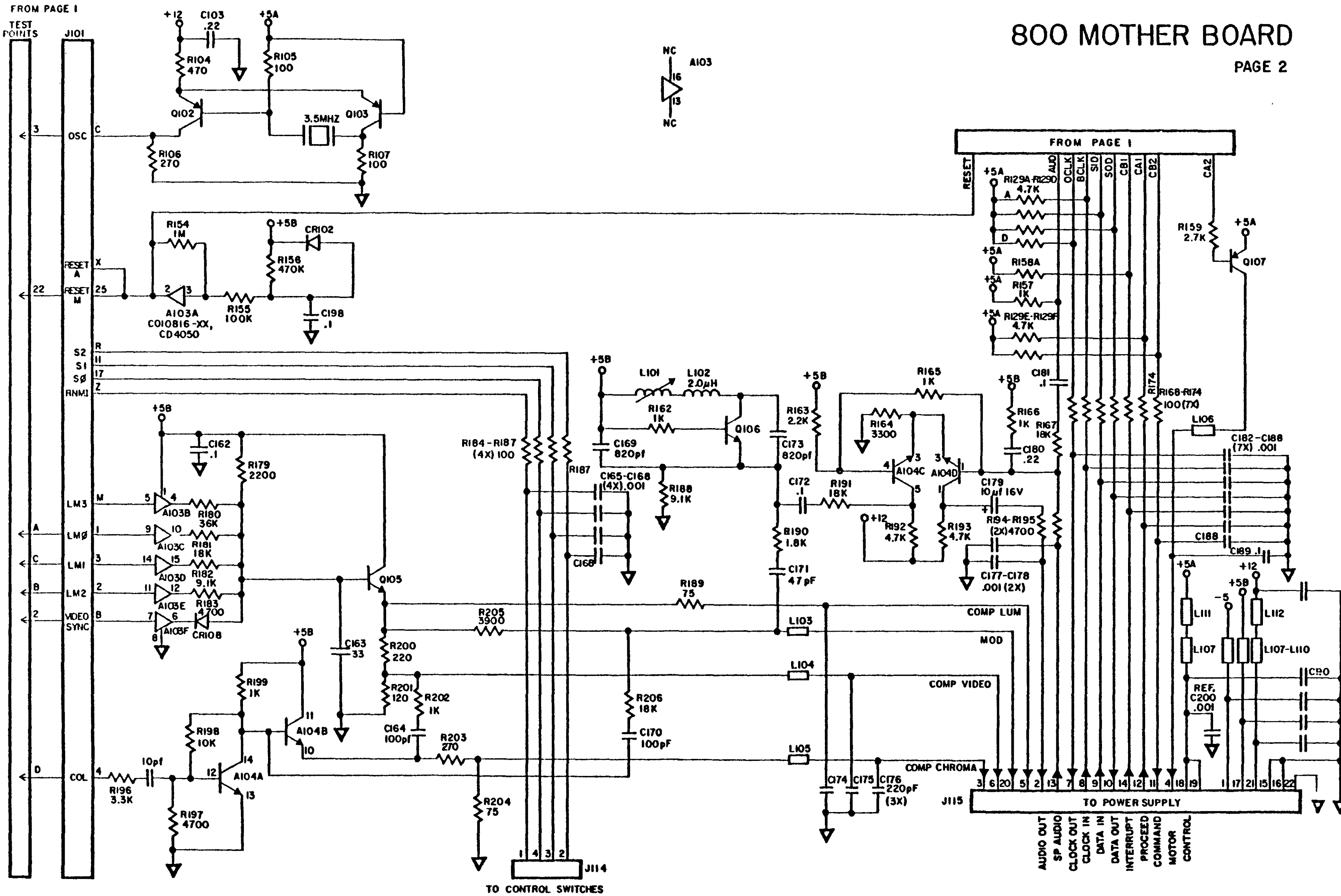
800 MOTHER BOARD

PAGE 1

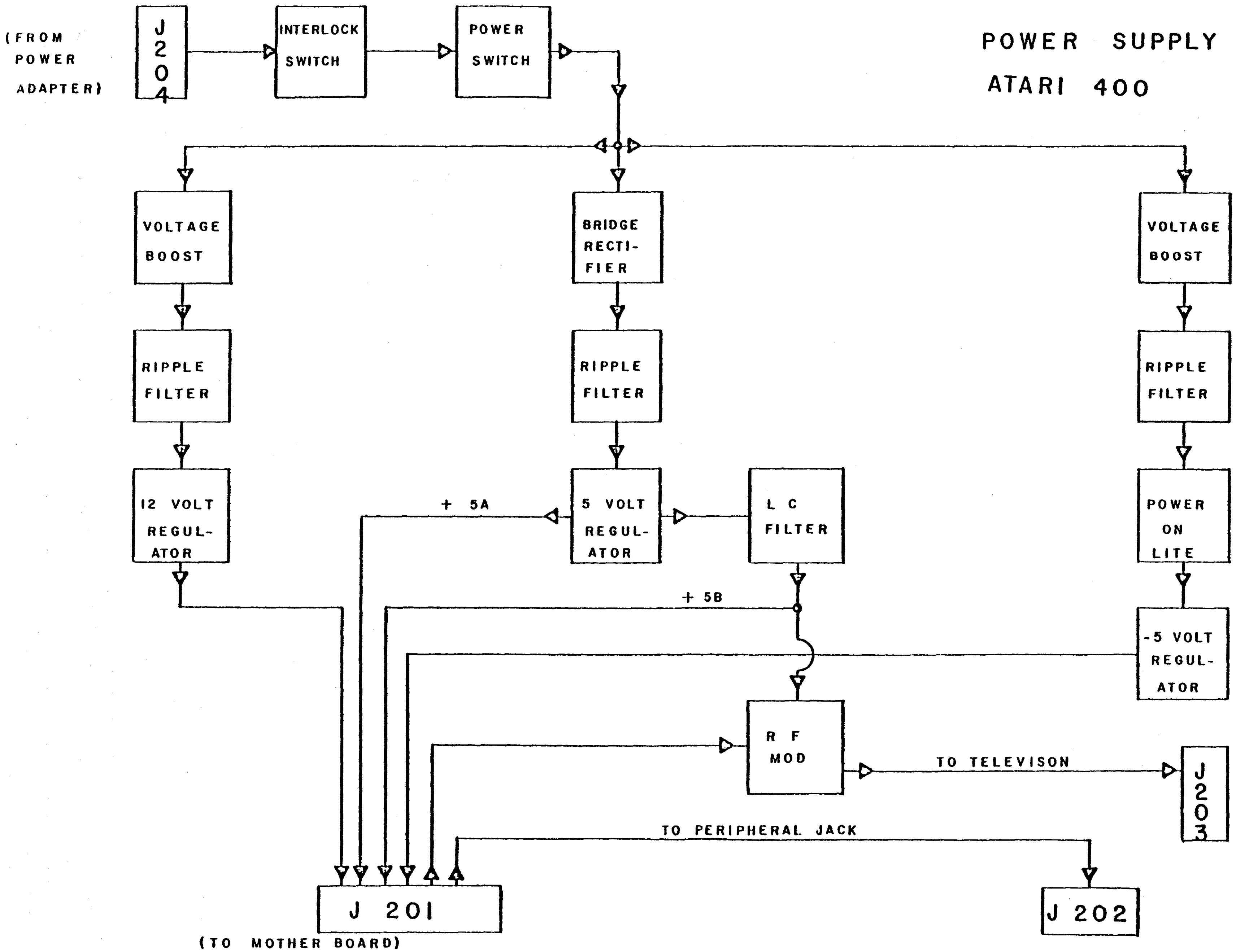


800 MOTHER BOARD

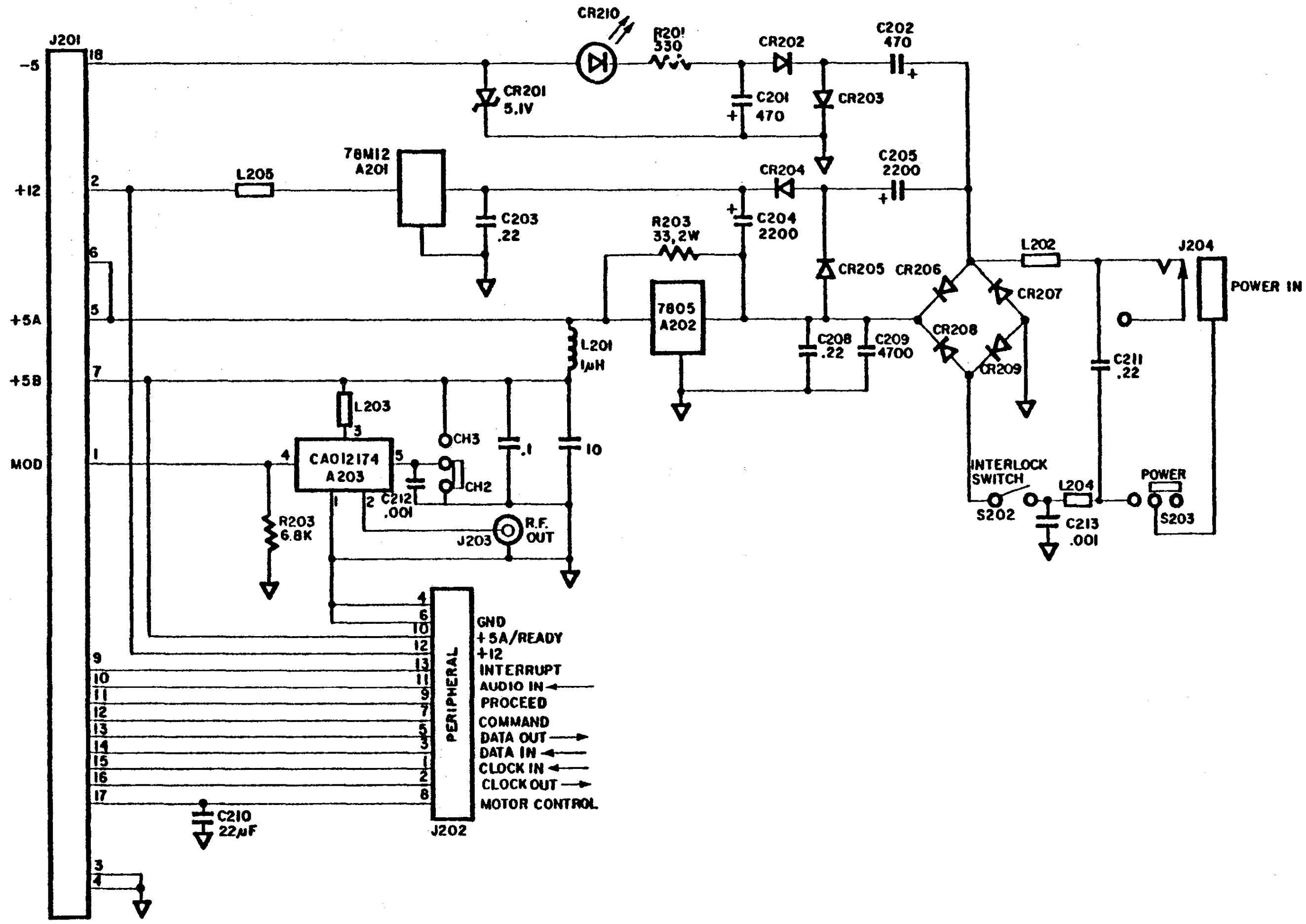
PAGE 2



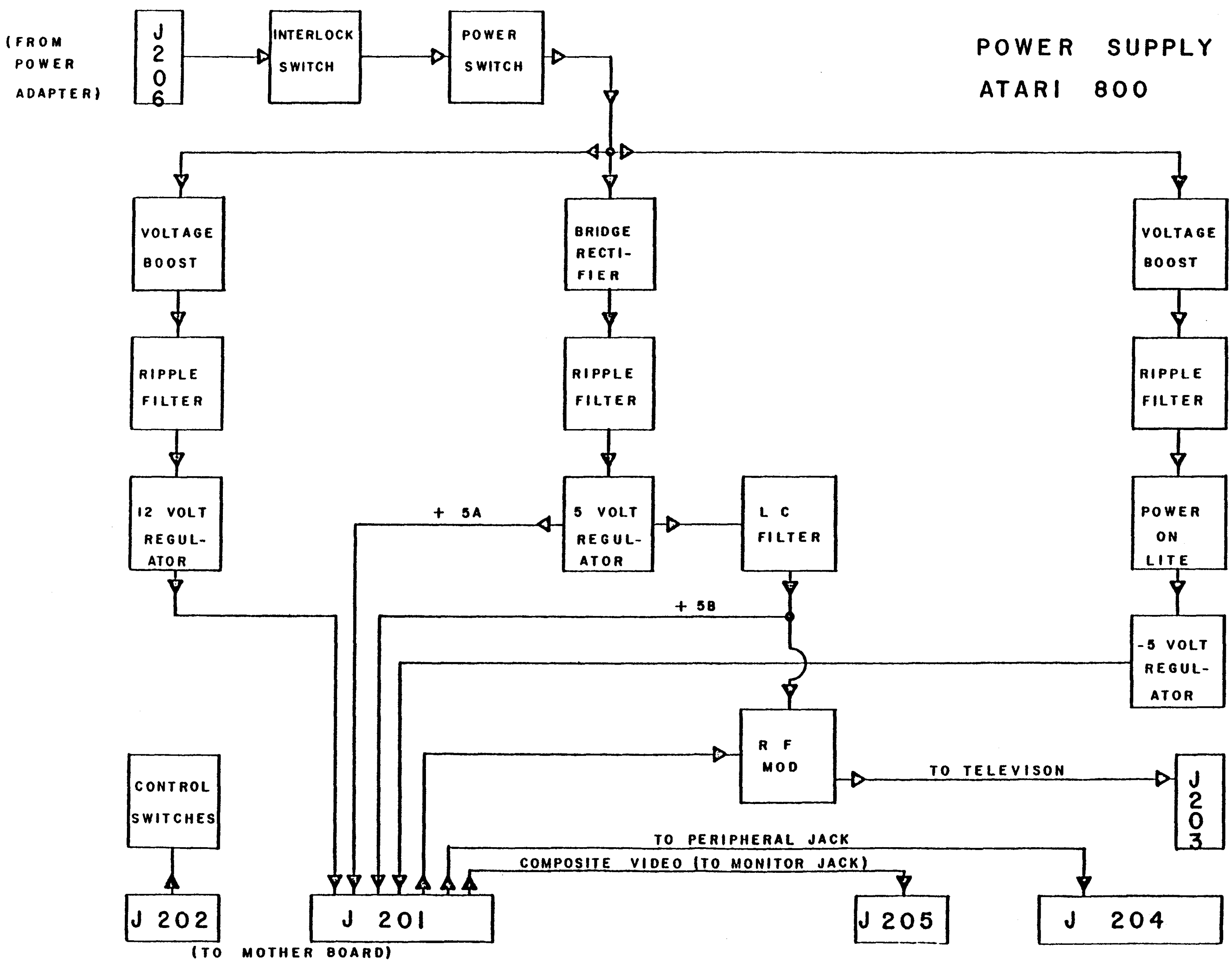
POWER SUPPLY BOARD
ATARI 400



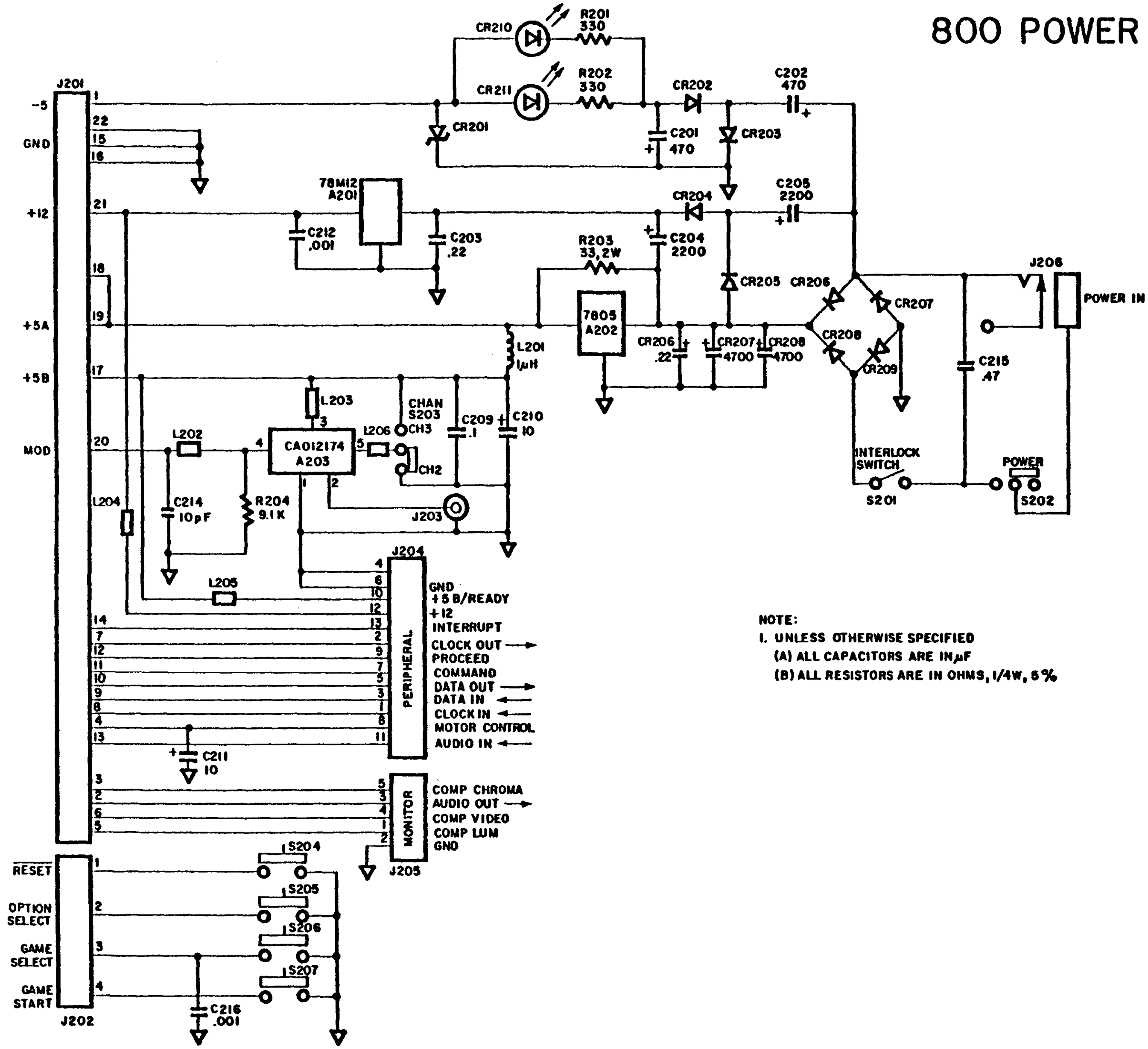
400 POWER SUPPLY



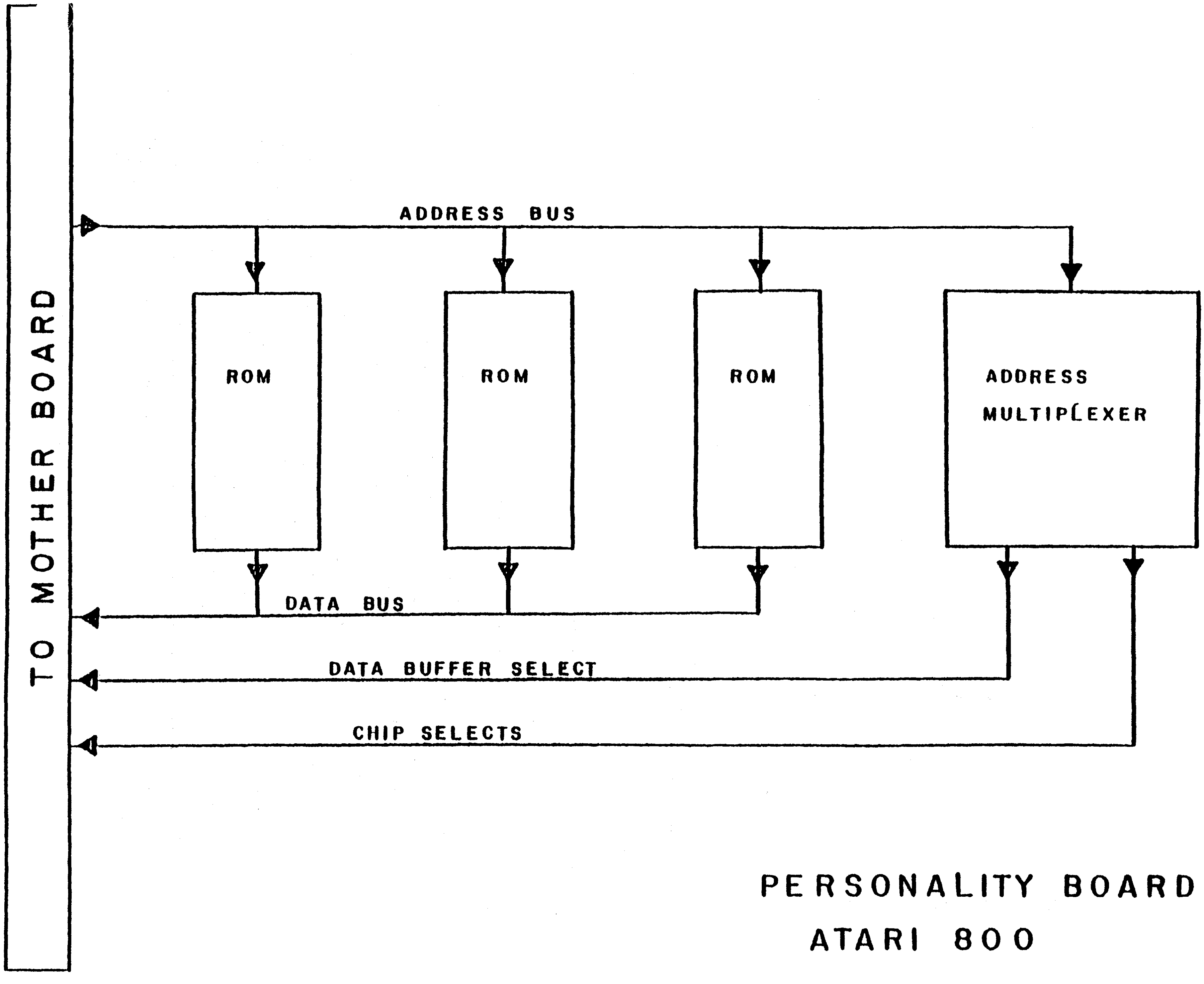
POWER SUPPLY BOARD
ATARI 800



800 POWER SUPPLY

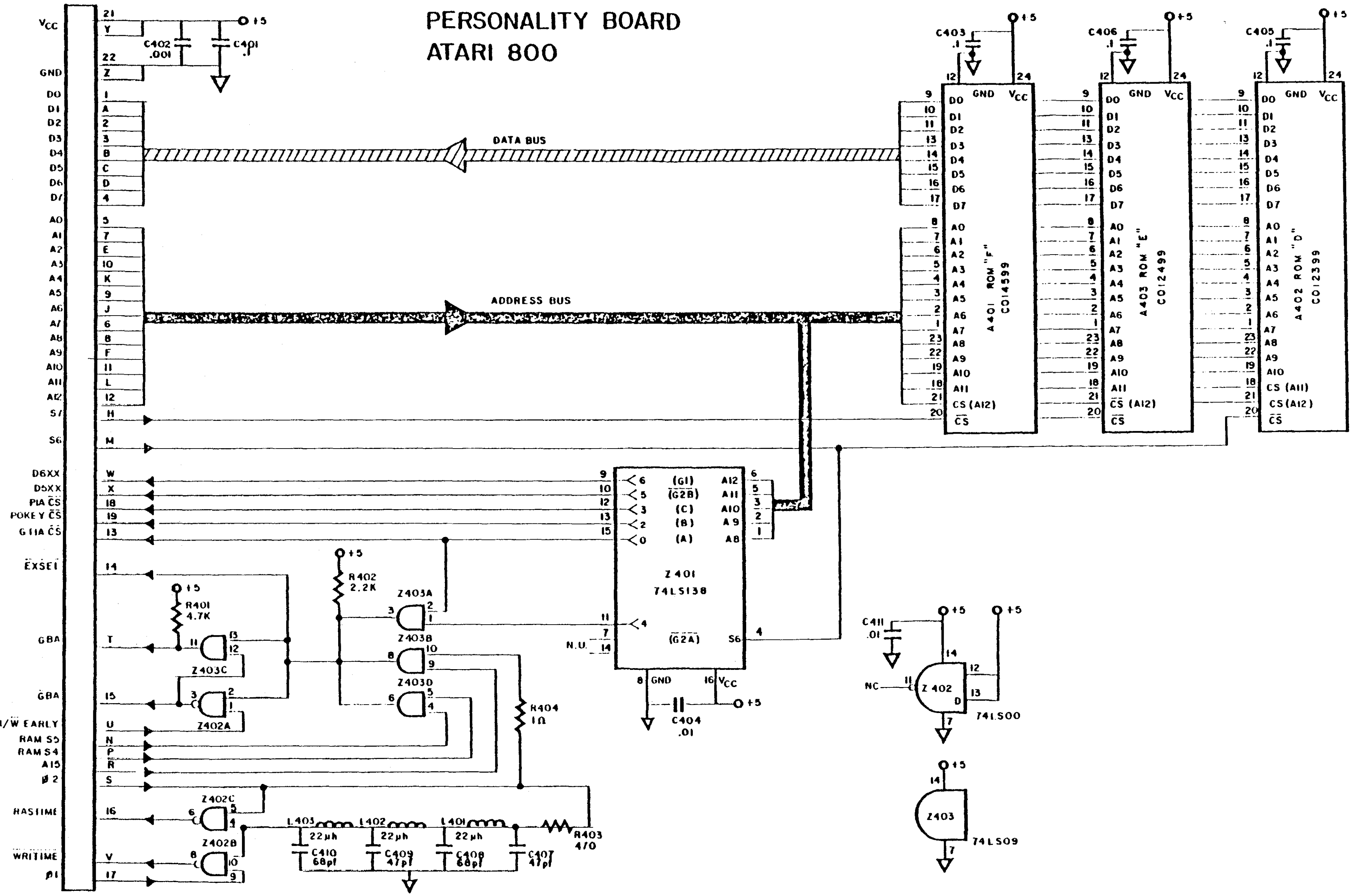


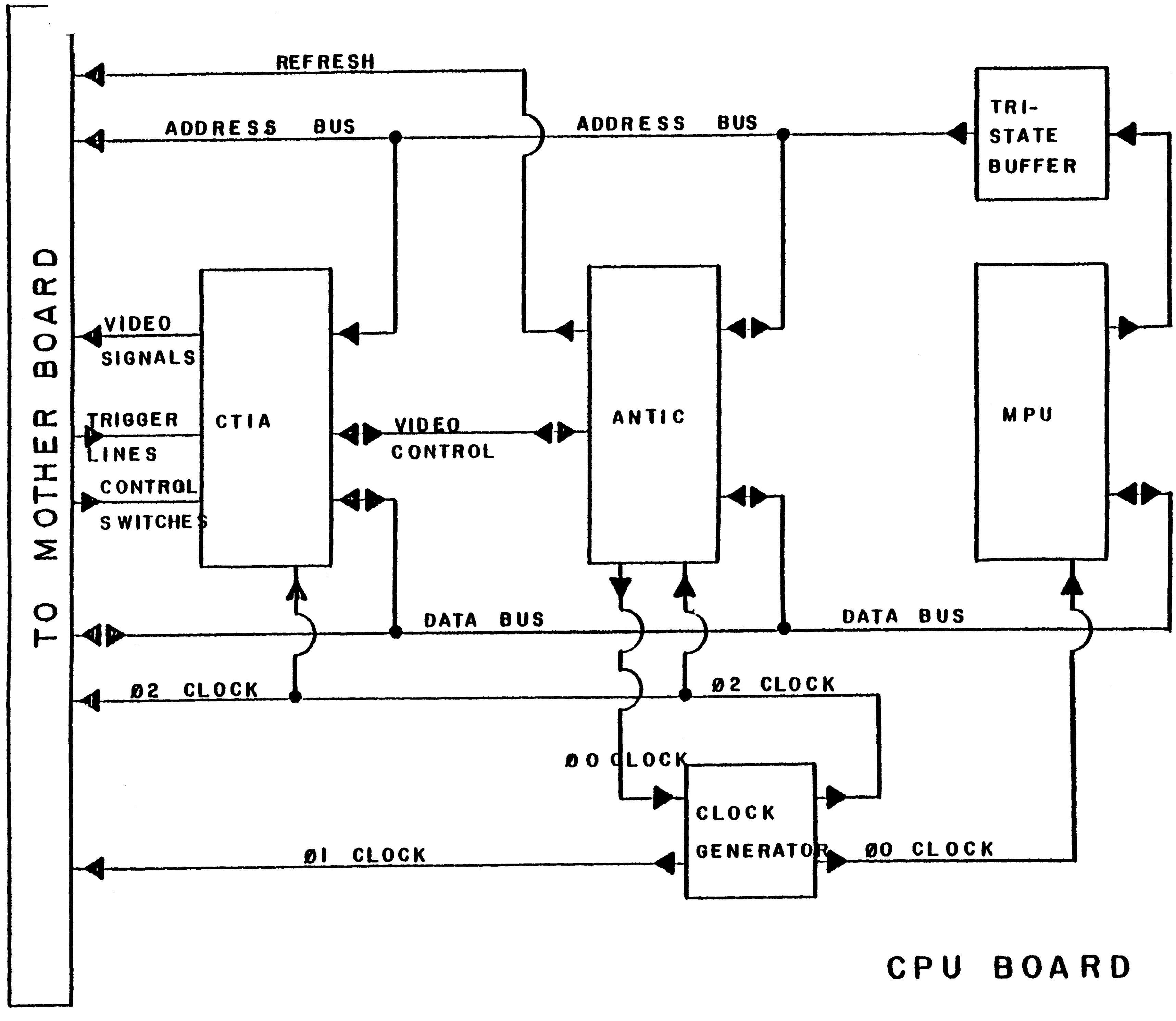
NOTE:
1. UNLESS OTHERWISE SPECIFIED
(A) ALL CAPACITORS ARE IN μF
(B) ALL RESISTORS ARE IN OHMS, 1/4W, 5%



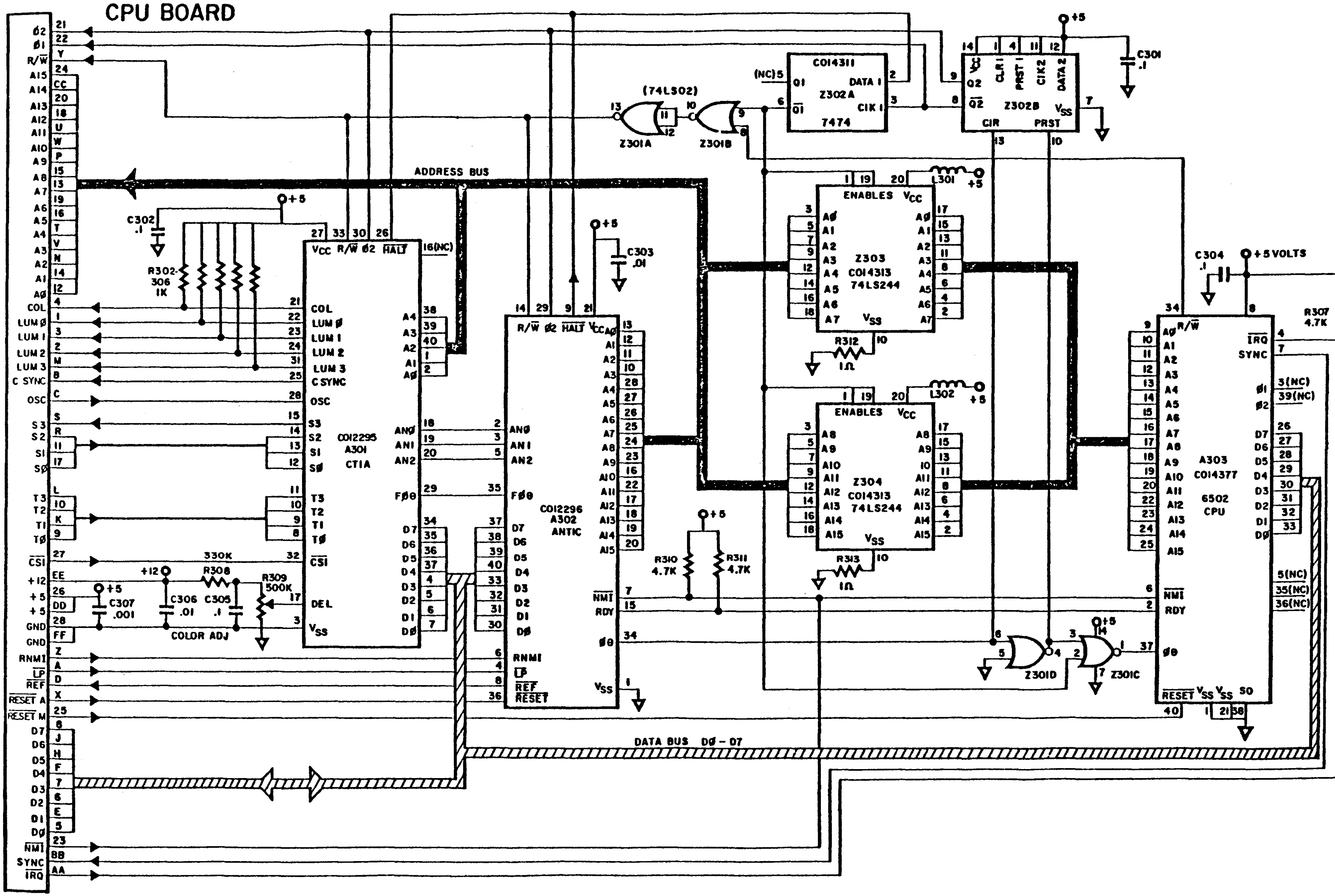
PERSONALITY BOARD
ATARI 800

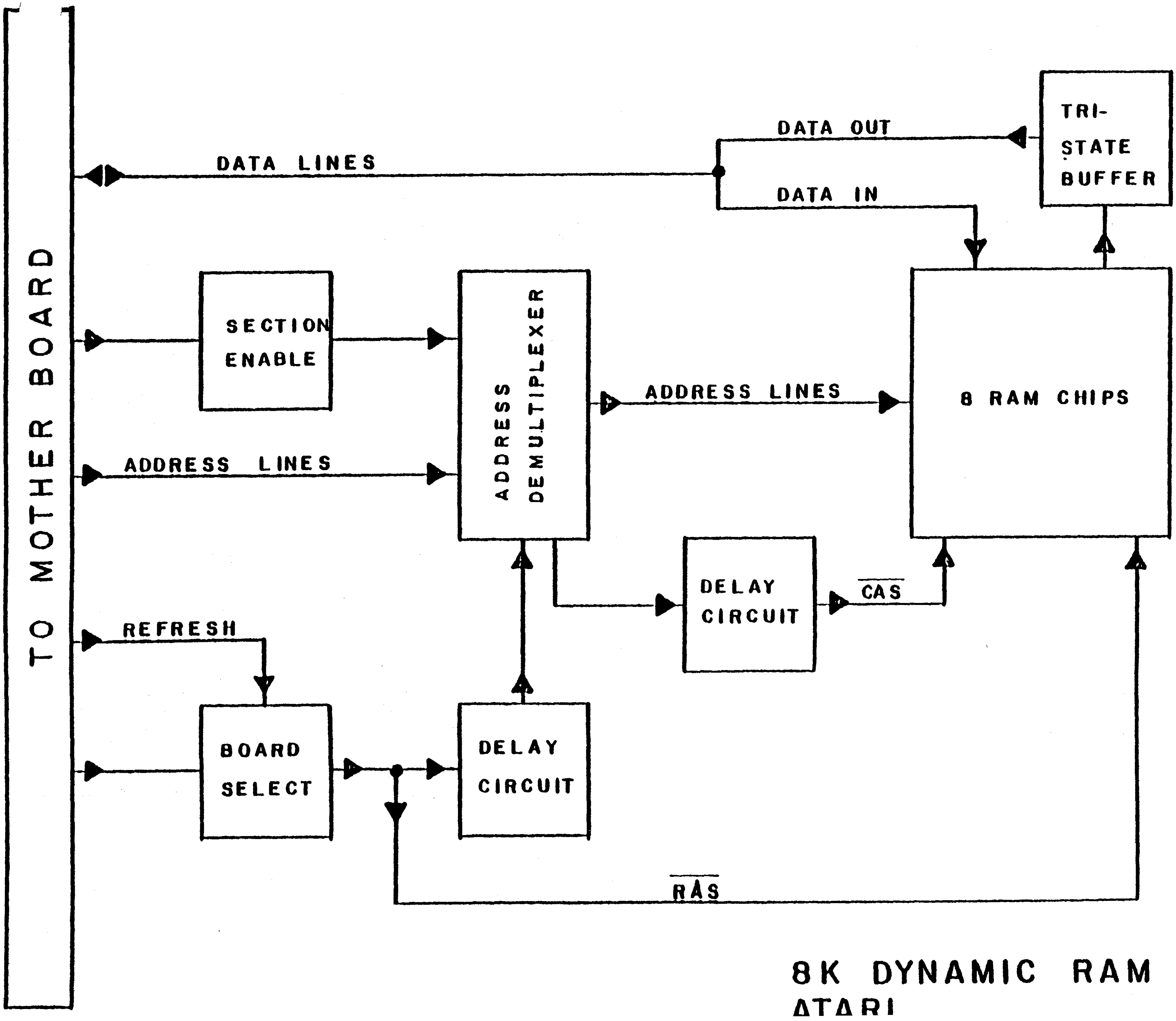
PERSONALITY BOARD
ATARI 800




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

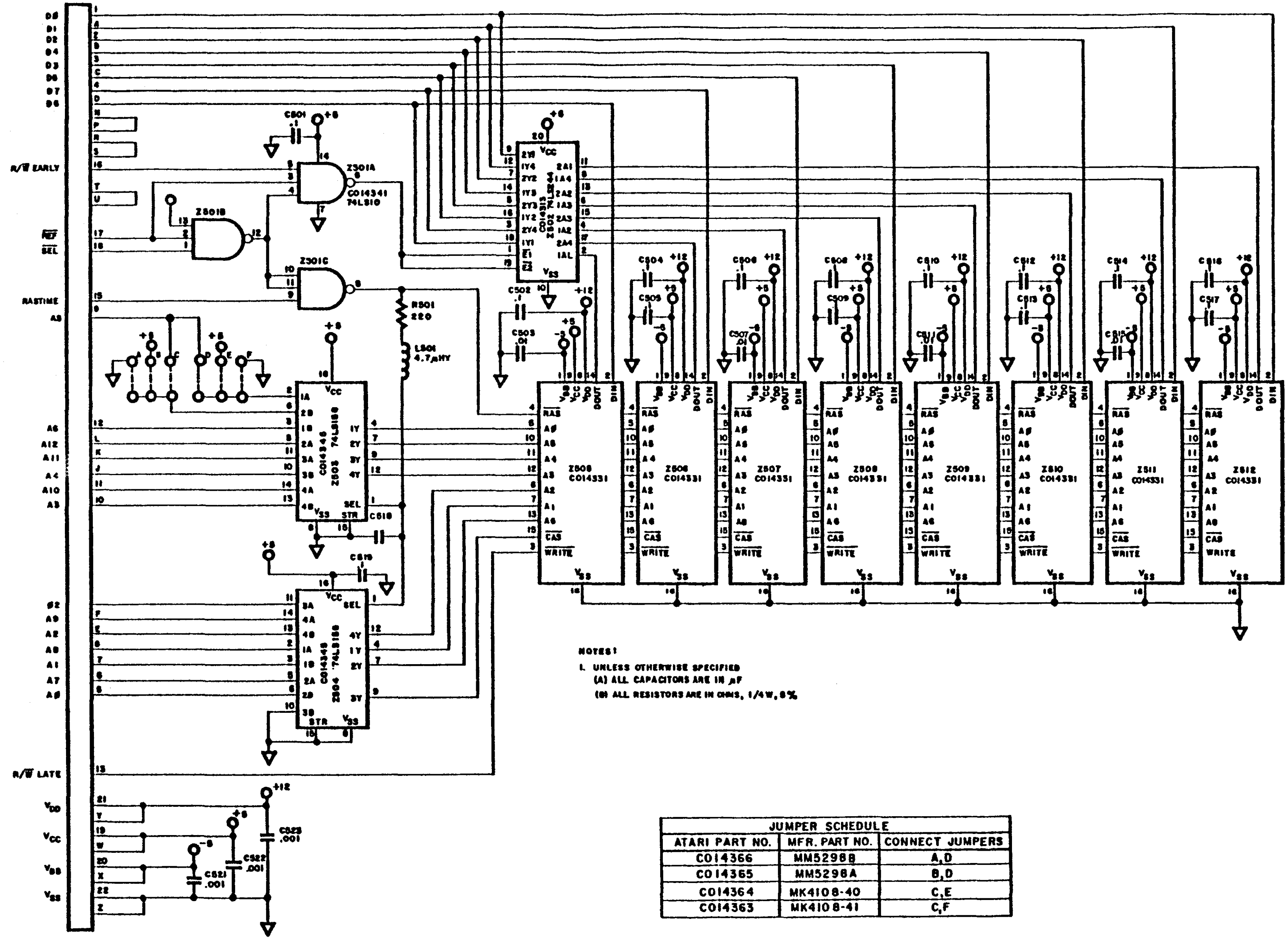
CPU BOARD


CPU BOARD
ADDRESS BUS
DATA BUS DØ - D7
C012295 A301 CTIA
C012296 A302 ANTIC
A303 C014377 6502 CPU
Z303 C014313 74LS244
Z304 C014313 74LS244
C014311 Z302A 7474
Z302B
(74LS02)
Z301A
Z301B
Z301C
Z301D
COLOR ADJ
+5 VOLTS



8K DYNAMIC RAM
ATARI

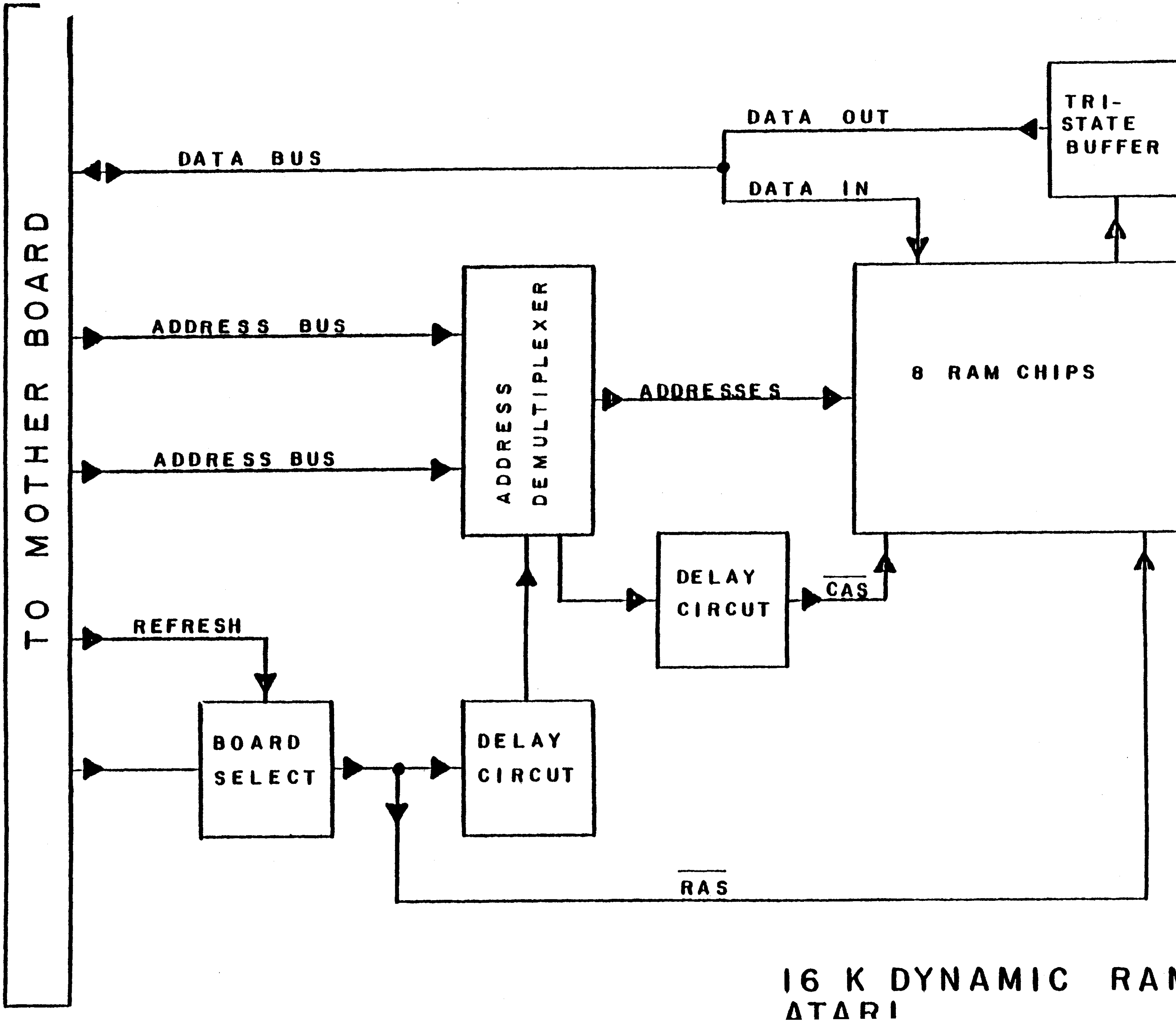
8K RAM



NOTES:

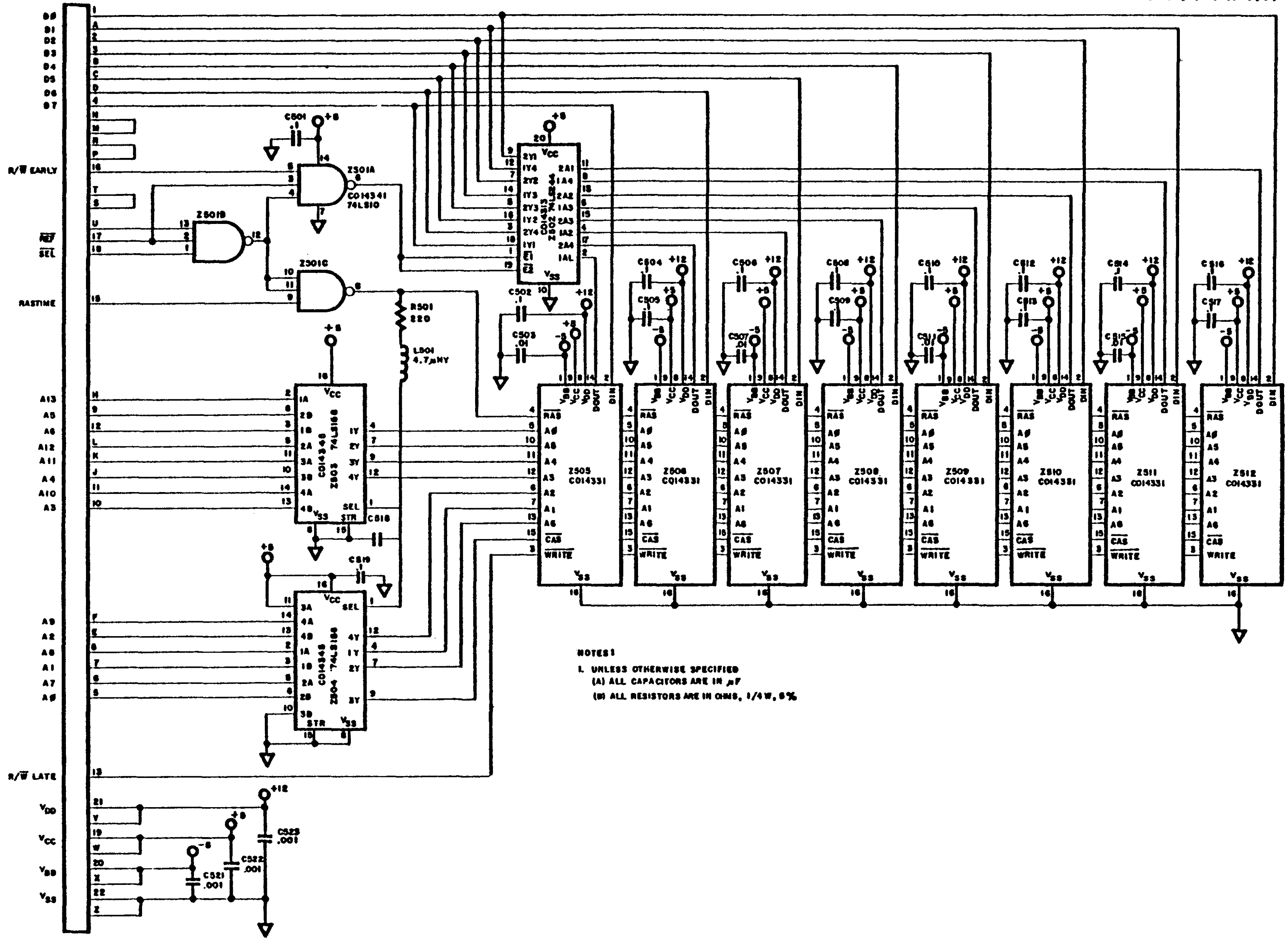
1. UNLESS OTHERWISE SPECIFIED
   (A) ALL CAPACITORS ARE IN μF
   (B) ALL RESISTORS ARE IN OHMS, 1/4W, 5%

JUMPER SCHEDULE		
ATARI PART NO.	MFR. PART NO.	CONNECT JUMPERS
CO14366	MM5298B	A,D
CO14365	MM5298A	B,D
CO14364	MK4108-40	C,E
CO14363	MK4108-41	C,F

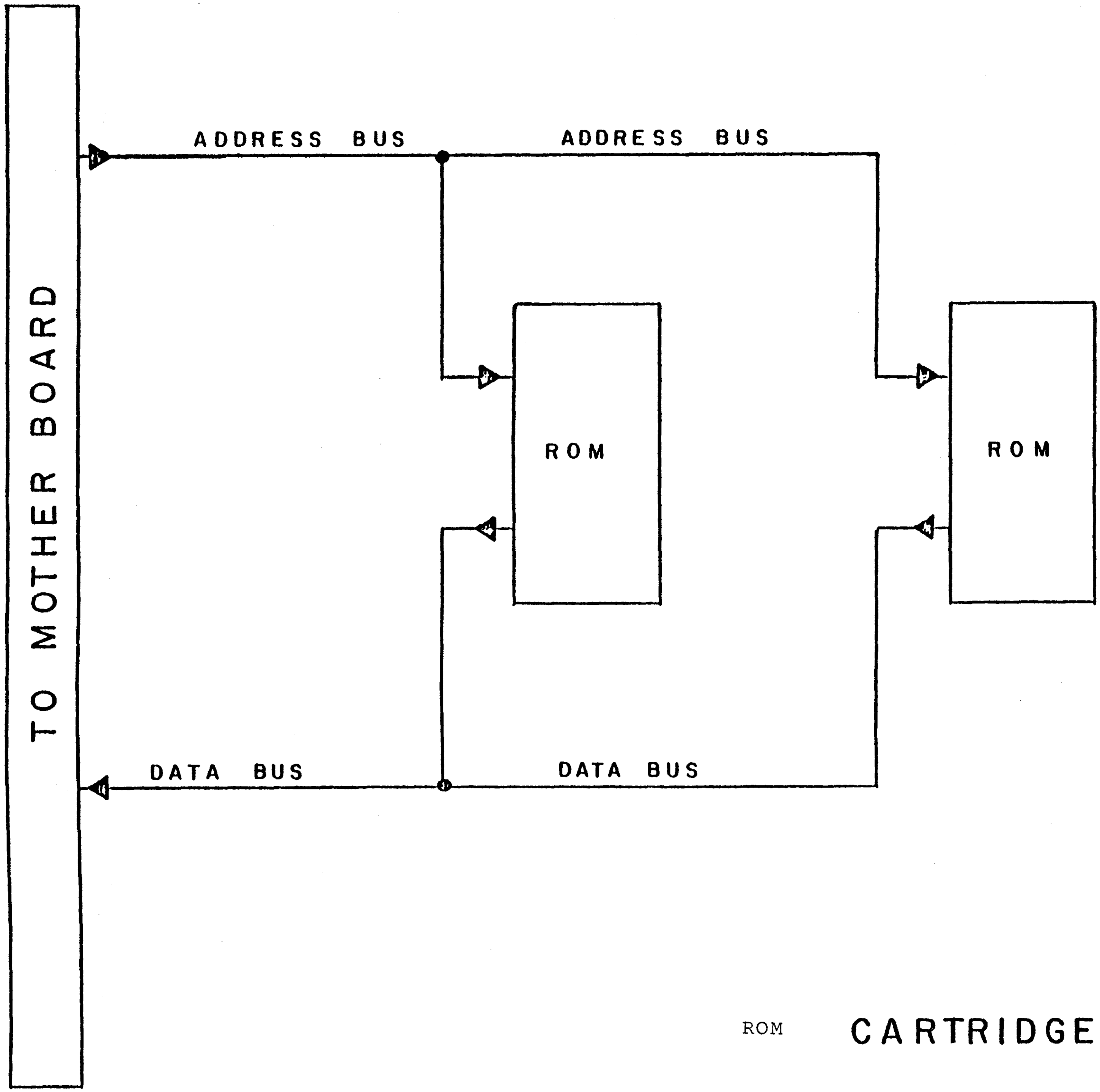


16 K DYNAMIC RAM
ATARI

16K RAM

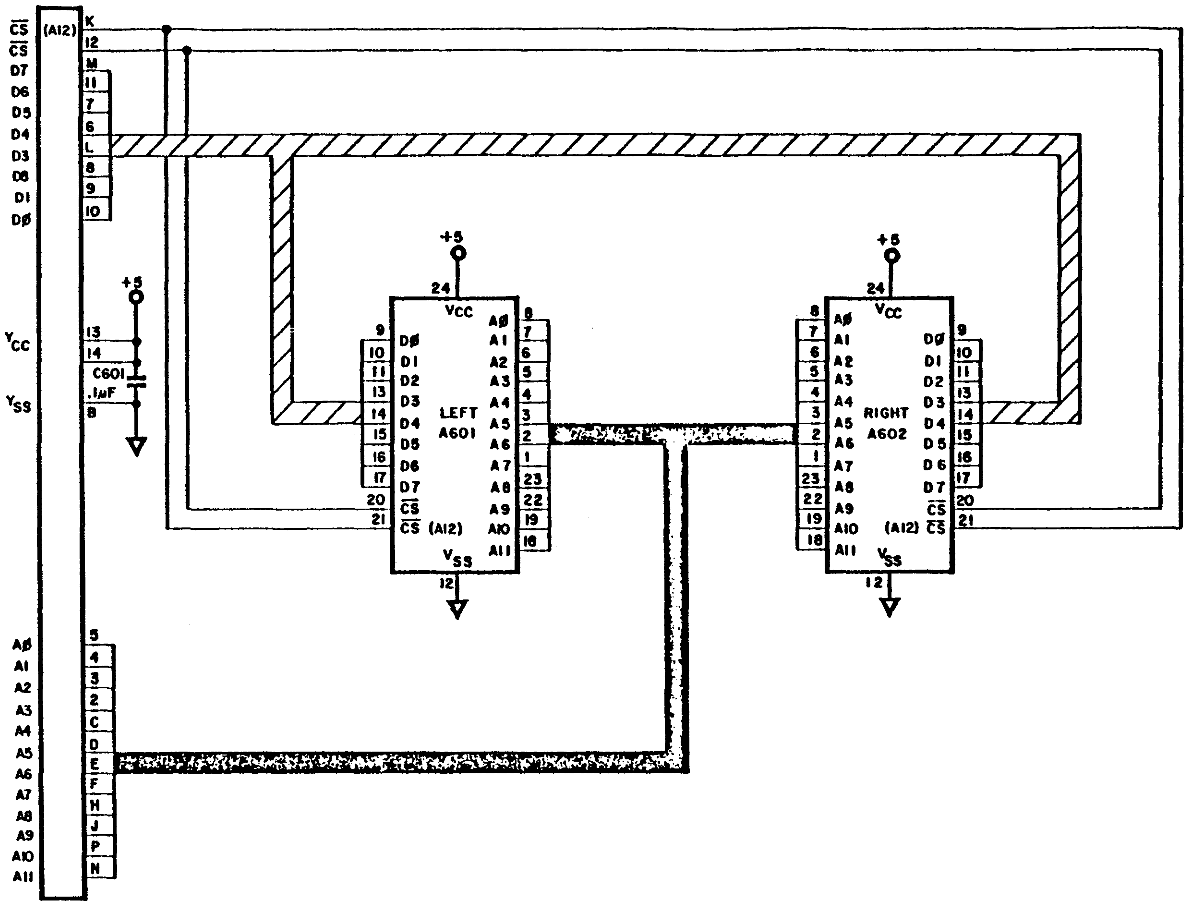


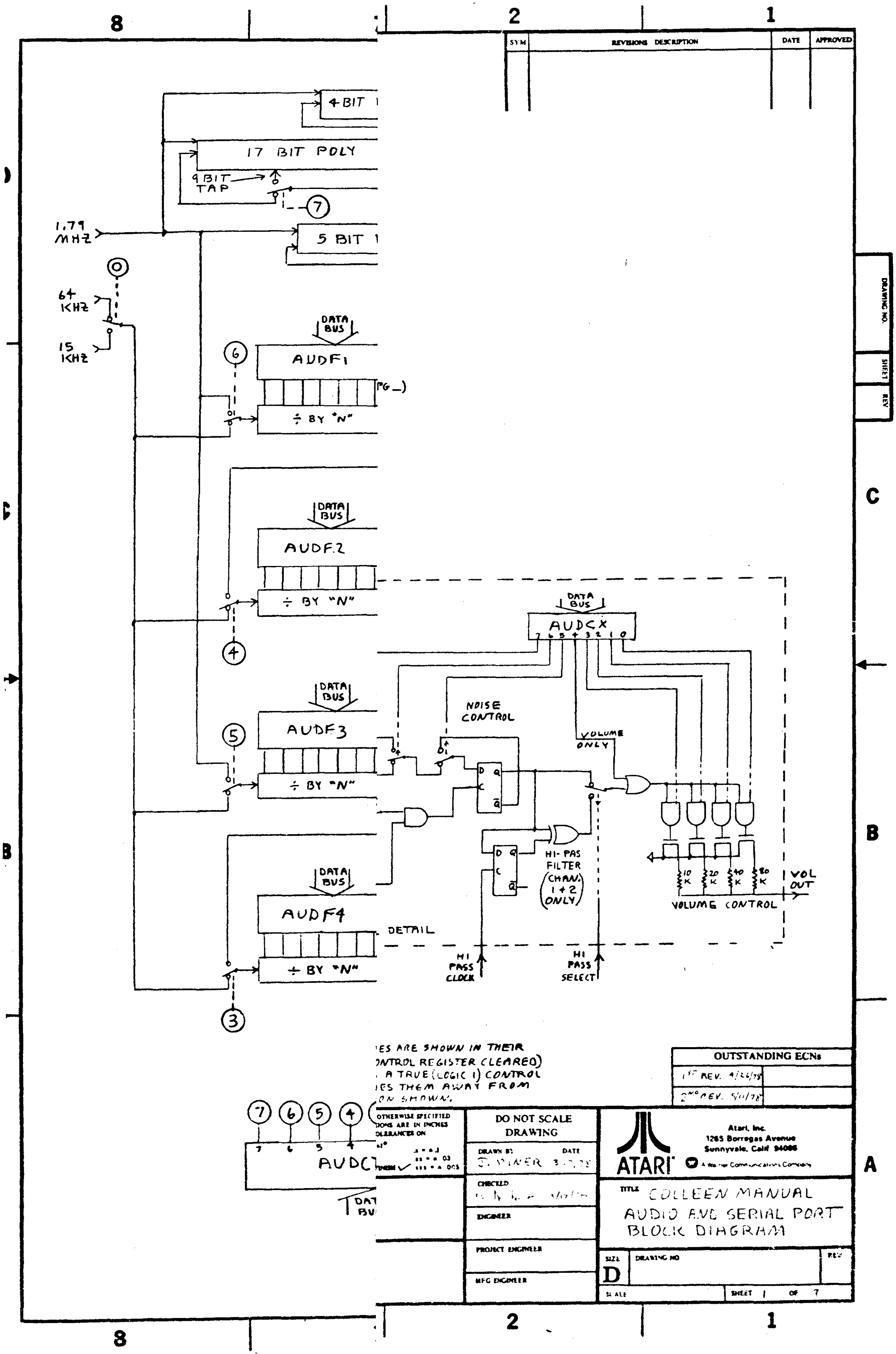
NOTES:
1. UNLESS OTHERWISE SPECIFIED
(A) ALL CAPACITORS ARE IN μF
(B) ALL RESISTORS ARE IN OHMS, 1/4W, 5%



ROM CARTRIDGE BOARD

SCHEMATIC ROM CARTRIDGE





REVISIONS DESCRIPTION
SYM
DATE
APPROVED
4 BIT
17 BIT POLY
9 BIT TAP
5 BIT
1.79 MHZ
64 KHZ
15 KHZ
DATA BUS
AUDF1
÷ BY "N"
AUDF2
AUDF3
AUDF4
AUDCX
NOISE CONTROL
VOLUME ONLY
HI-PAS FILTER (CHAN. 1+2 ONLY)
VOLUME CONTROL
VOL OUT
DETAIL
HI PASS CLOCK
HI PASS SELECT
ES ARE SHOWN IN THEIR
ONTROL REGISTER CLEARED)
A TRUE (LOGIC 1) CONTROL
ES THEM AWAY FROM
ON SHOWN.
AUDC
OUTSTANDING ECNs
1ST REV. 4/26/78
2ND REV. 5/1/78
DO NOT SCALE DRAWING
DRAWN BY
DATE
CHECKED
ENGINEER
PROJECT ENGINEER
MFG ENGINEER
ATARI
Atari, Inc.
1265 Borregas Avenue
Sunnyvale, Calif 94086
A Warner Communications Company
TITLE COLLEEN MANUAL
AUDIO AND SERIAL PORT
BLOCK DIAGRAM
SIZE D
DRAWING NO
REV
SCALE
SHEET 1 OF 7

APPENDIX A

USE OF PLAYER/MISSILE GRAPHICS

WITH BASIC

The ATARI® 400/800 TM 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.

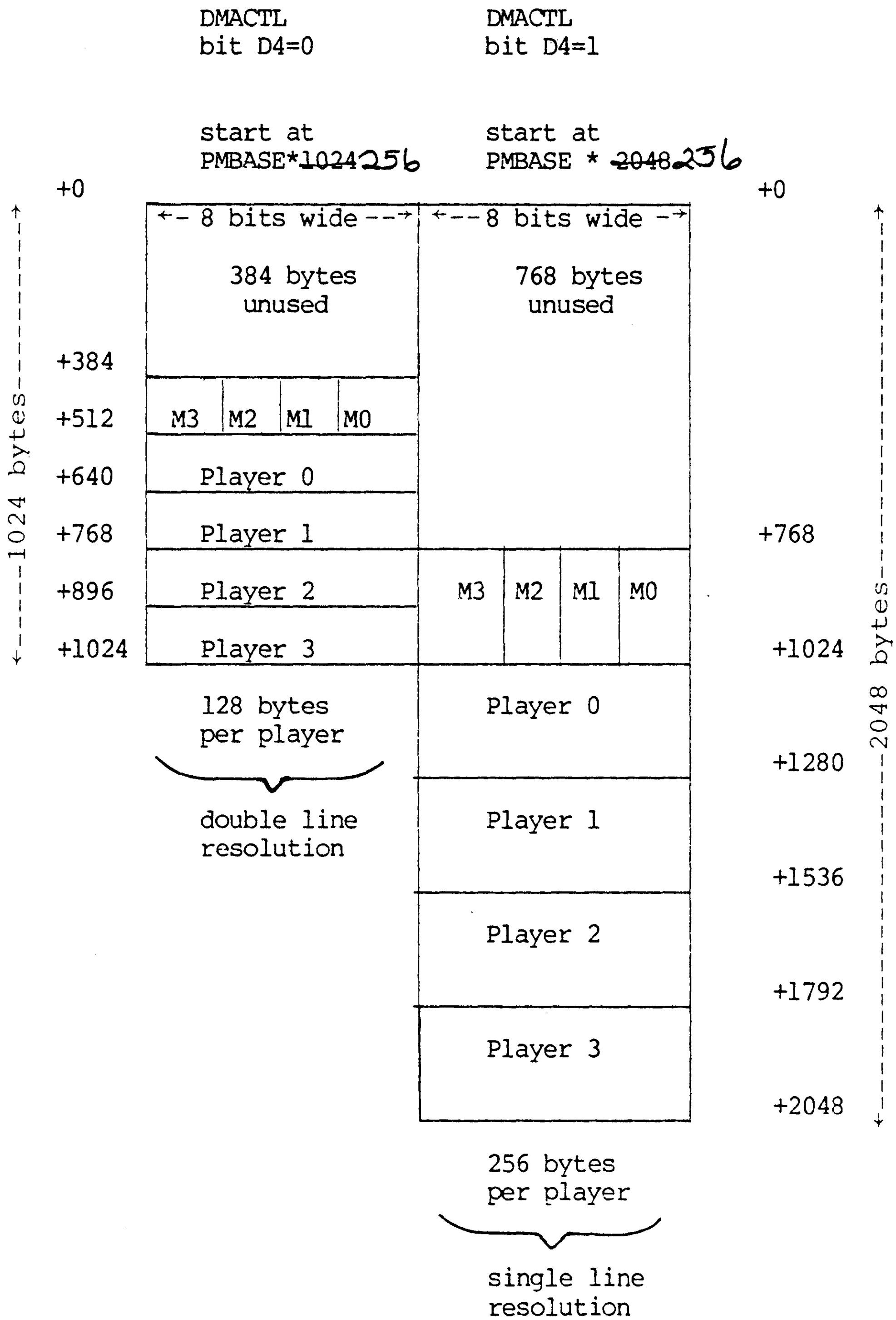
9. The vertical position of the player is determined by its location in memory. After the initial offset is obtained in step 8, its height may be defined. Its range on the playfield is from 32 to 223 in single line resolution and from 16 to 111 in double line resolution. By adding the desired height 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.



PLAYER-MISSILE

Memory
Configuration

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

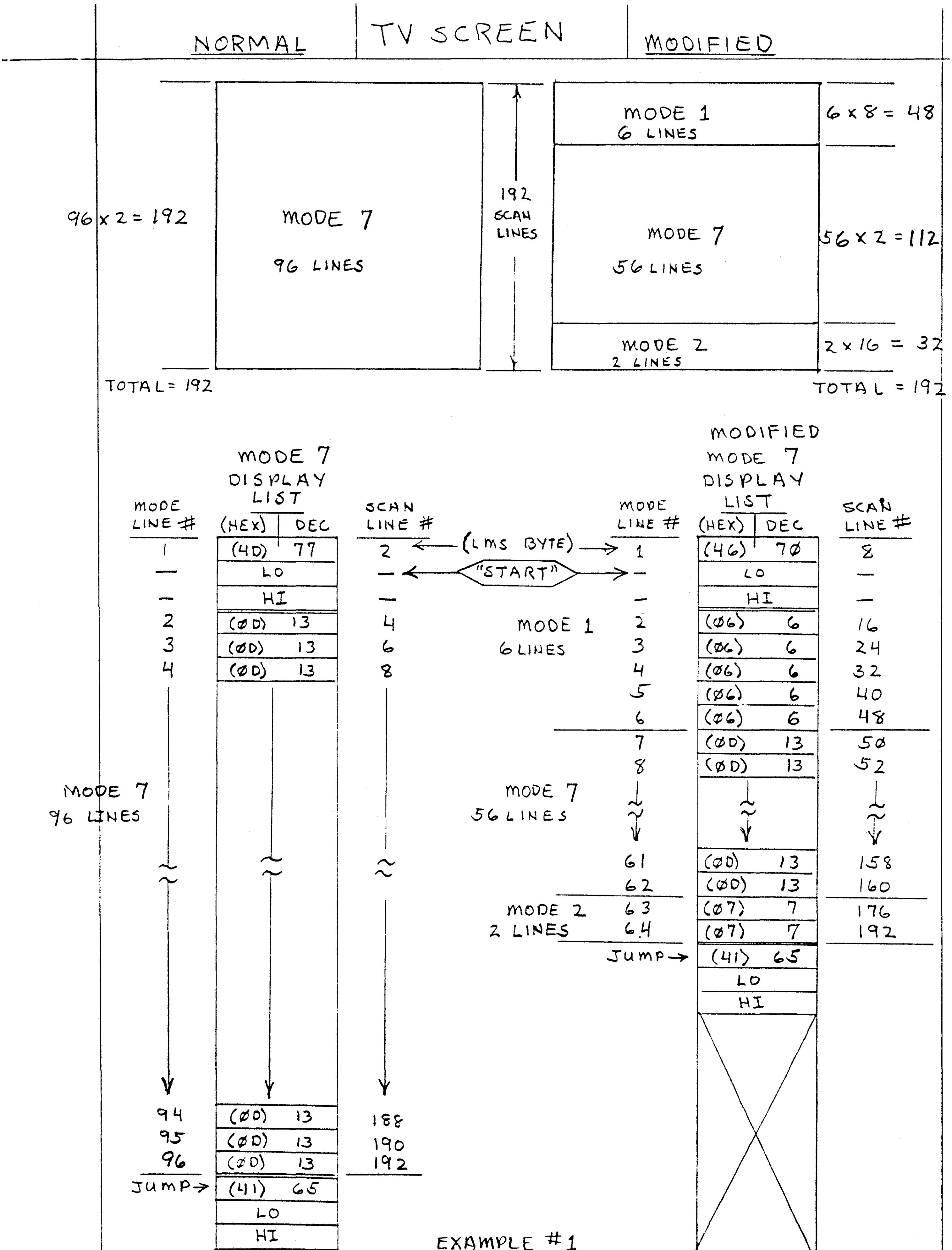
This procedure describes how to mix several graphic modes on the TV screen at the same time using BASIC commands. Each graphics mode has a different number of scan lines per "Mode Line" (one line of a graphics mode). The TV screen must consist of 192 scan lines, so when mixing modes, they must be combined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


TV SCREEN
NORMAL
MODIFIED
96 x 2 = 192
MODE 7
96 LINES
TOTAL = 192
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
MODE 7 DISPLAY LIST
MODE LINE #
(HEX) DEC
SCAN LINE #
(4D) 77
LO
HI
(ØD) 13
(LMS BYTE)
"START"
MODE 7
96 LINES
94
95
96
188
190
192
JUMP→
(41) 65
MODIFIED MODE 7 DISPLAY LIST
(46) 70
(Ø6) 6
MODE 1
6 LINES
50
52
MODE 7
56 LINES
61
62
158
160
MODE 2
2 LINES
63
64
(Ø7) 7
176
192
EXAMPLE #1

II. <u>PROCEDURE TO SET UP SCREEN IN MIXED MODES</u>:

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
	2	POKE START + 2,6
	3	POKE START + 3,6
MODE 1	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. <u>PROCEDURE TO PRINT AND PLOT IN MIXED MODES</u>, 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 <u>above</u> the target line.
      M_3 = the number of mode lines of 10 bytes of RAM per line <u>above</u> 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

MODE BYTE

REMARK	LEFT NIBBLE (HEX)		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.

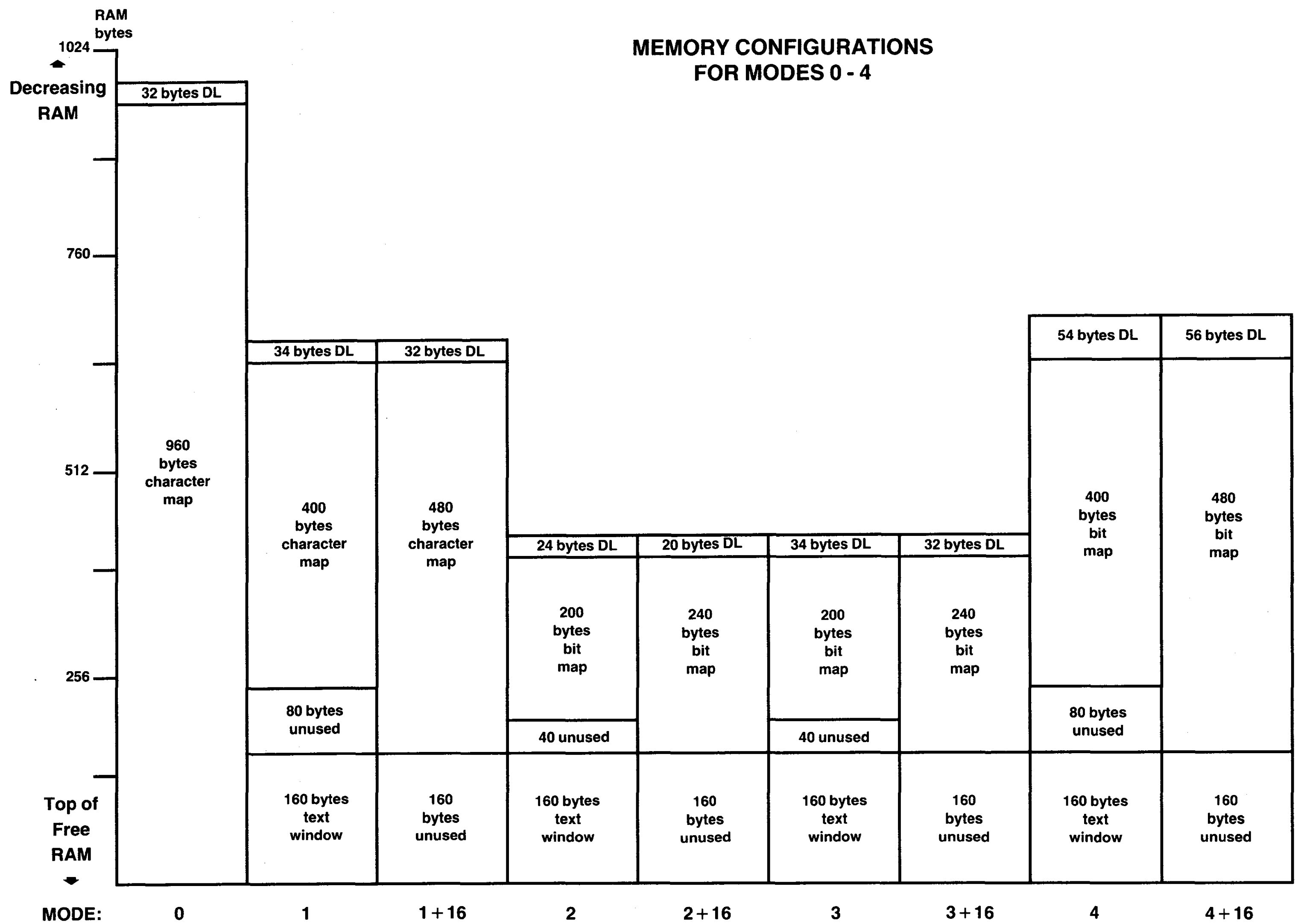
   BLANK - to output selected number of background lines.

TABLE #2

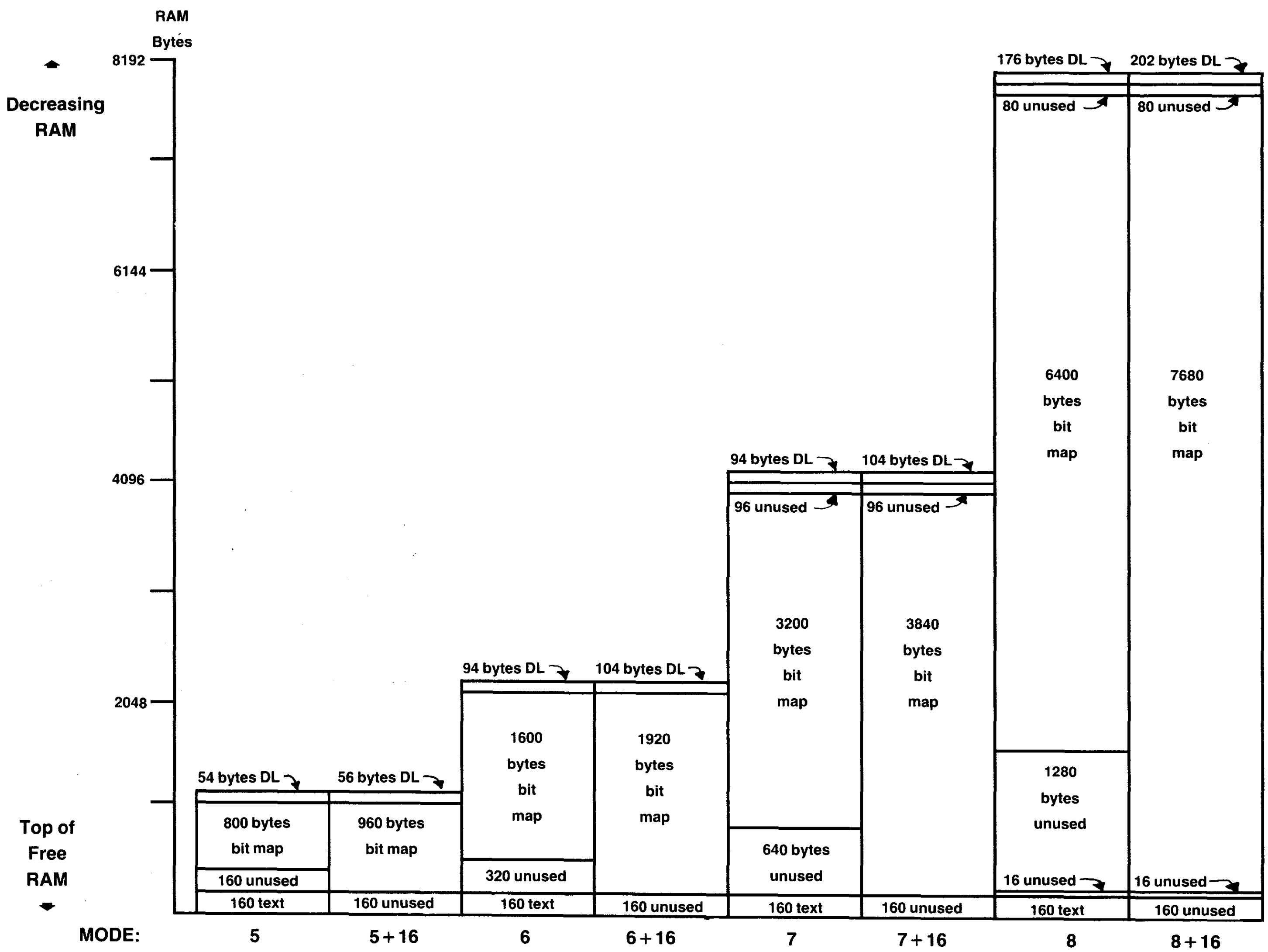
GRAPHICS MODES
RAM REQUIREMENTS

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
RAM bytes
1024
760
512
256
Decreasing RAM
Top of Free RAM
32 bytes DL
960 bytes character map
34 bytes DL
400 bytes character map
80 bytes unused
160 bytes text window
32 bytes DL
480 bytes character map
160 bytes unused
24 bytes DL
200 bytes bit map
40 unused
160 bytes text window
20 bytes DL
240 bytes bit map
160 bytes unused
34 bytes DL
200 bytes bit map
40 unused
160 bytes text window
32 bytes DL
240 bytes bit map
160 bytes unused
54 bytes DL
400 bytes bit map
80 bytes unused
160 bytes text window
56 bytes DL
480 bytes bit map
160 bytes unused
MODE: 0 1 1+16 2 2+16 3 3+16 4 4+16

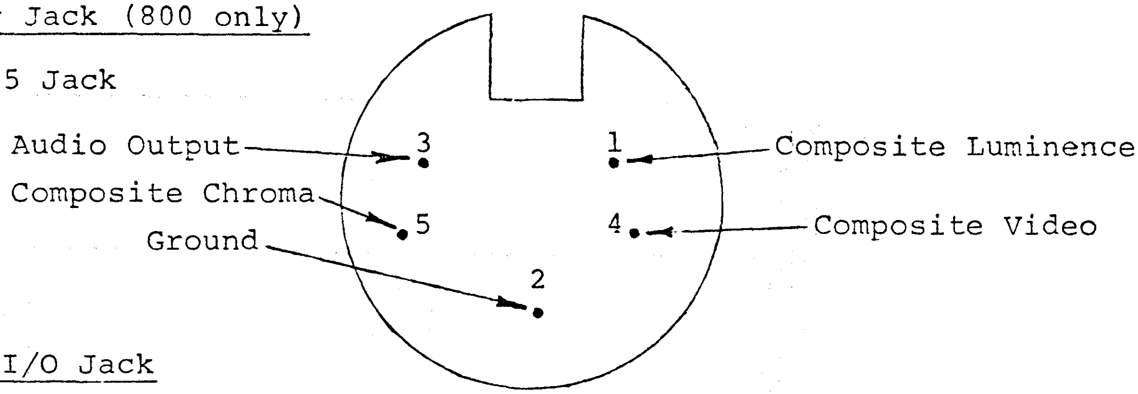
MEMORY CONFIGURATIONS
FOR MODES 5 - 8



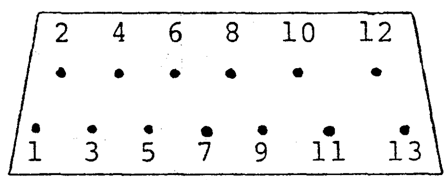
APPENDIX C: PINOUTS

Monitor Jack (800 only)

D.I.N. 5 Jack

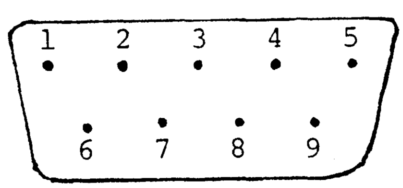


Serial I/O Jack



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