# 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.

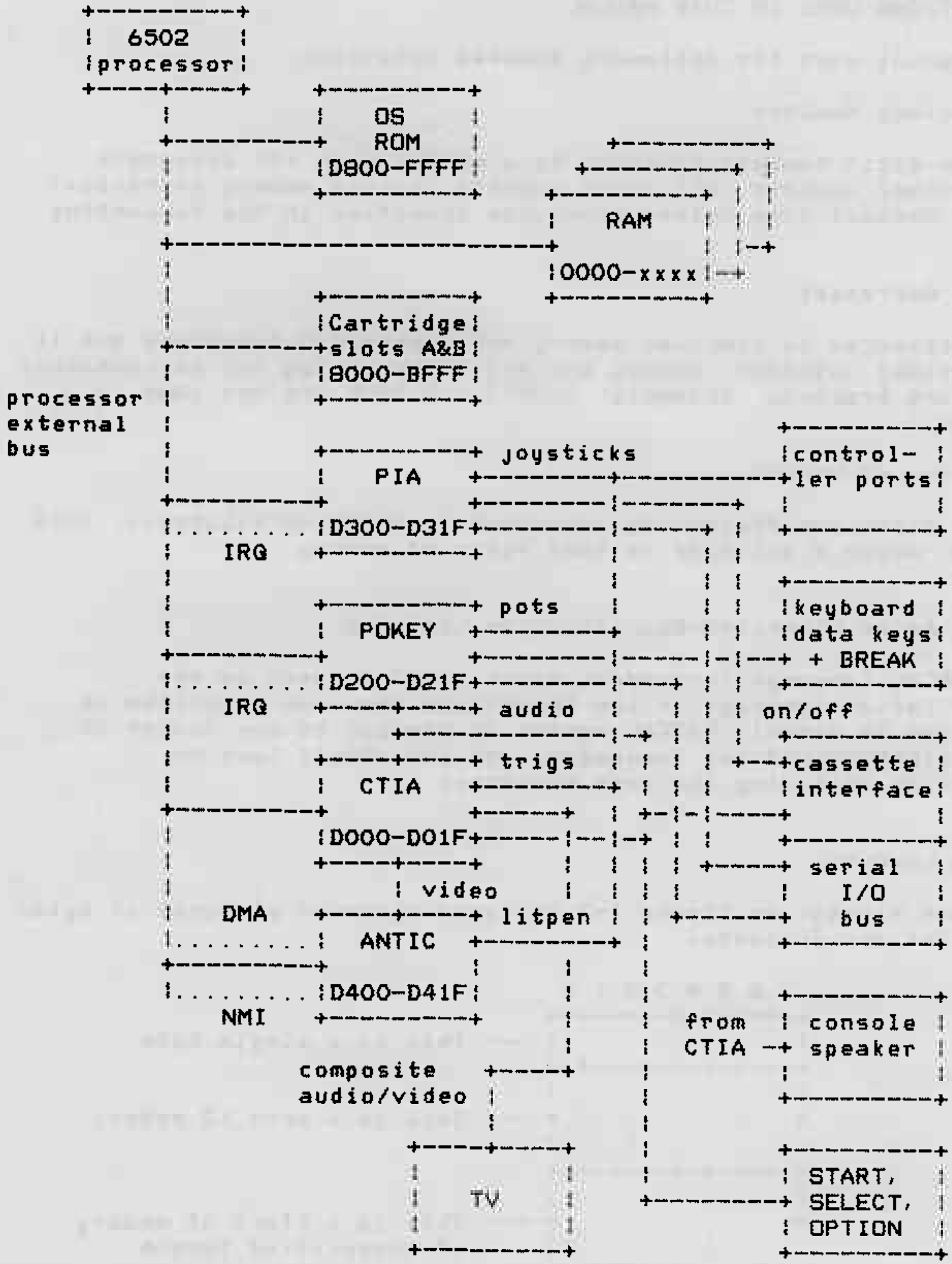


Figure 1-1. ATARI Home Computer Block Diagram

## 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.

## 3 CONFIGURATIONS

The ATARI 400 and ATARI 800 Home Computers support a wide variety of configurations, each with a unique operating environment:

- o Cartridge(s) may or may not be inserted
- o Memory can be optionally added to the ATARI 800 Computer console in 16K increments
- o Many different peripheral devices can be attached to the serial I/O bus.

The OS accounts for all of these variables without requiring a change in the resident OS itself (see Section 2). The machine configuration is checked when power is first turned on and then is not checked again, unless system reset is used. A general discussion of some of the valid configurations follows.

### PROGRAM ENVIRONMENTS

The OS allows one of four program types to be in control at any point in time:

- o The OS blackboard (ATARI Memo Pad) program
- o A cartridge-resident program
- o A diskette-booted program
- o A cassette-booted program

Control choice is based upon information in the cartridge(s), upon whether or not a disk drive is attached, and upon operator keyboard inputs. The exact algorithms are discussed in detail in Section 7.

#### Blackboard Mode

In blackboard mode, the screen is established as a 24 x 40 text screen. Anything entered from the keyboard goes to the screen without being examined, although all of the screen editing functions are supported. Blackboard mode is the lowest priority environment. You go there only by command from a higher priority environment, or by default, if there is no other reasonable environment for the OS to enter. For example, typing BYE in BASIC causes the OS to enter the blackboard mode. The blackboard mode can be exited by pressing the [SYSTEM RESET] key if it was entered from a higher environment.

Cartridge

An inserted cartridge normally provides the main control after initialization is complete (for example: ATARI BASIC, SUPER BREAKOUT[TM], BASKETBALL, COMPUTER CHESS, and others. All these cartridge programs interface directly with you in some way). Although a cartridge can provide a supporting function for some other program environment, this has not yet been done. Some cartridges (particularly keyboard-oriented ones) can change environments by entering special commands (such as "BYE") to go to blackboard mode or "DOS" to enter the disk utility. Other cartridges cannot change environments. Note that a hardware interlock prevents the removal or insertion of a cartridge with the power on; this feature causes the entire system to reinitialize with every cartridge change.

Diskette Boot

The diskette may or may not be booted when the system powers up with diskette-bootable software. This paragraph assumes that a diskette boot did occur. See Section 7 for boot condition explanations.

The diskette-booted software can take control as the Disk Utility Program (DUP) does under certain conditions, or can provide a supporting function as the File Management System (FMS) does. This environment is so flexible that it is difficult to generalize on its capabilities and restrictions. The only machine requirement (other than the disk drive) is that sufficient RAM be installed to support the program being booted.

Cassette-Boot

The cassette-boot environment is similar to the diskette-boot environment, although the cassette is limited as an I/O device. It is slow and can access only one file at a time in sequence. Note that the cassette-boot facility has no relation to the use of cassettes to store high-level language programs (e.g., programs written in ATARI BASIC), nor to the use of cassettes to store data.

## RAM EXPANSION

Although you can expand RAM noncontiguously in the ATARI 800 Home Computer, the OS will only recognize RAM that is contiguous starting from location 0. Installation directions are provided with the purchased RAM modules. RAM can be added until it totals 48K. After 32K, additional RAM overlays first the right-cartridge addresses (32K to 40K) and then the left-cartridge addresses (40K to 48K). Note that in cases of conflict, the inserted cartridge has higher priority and disables the conflicting RAM in 8K increments. See Section 4 for a detailed discussion of system memory.

As a result of power-up, the OS will generate two pointers that define the lowest available RAM location and the highest available RAM location. The OS and diskette or cassette-booted software will determine the location of the lowest available RAM, while the number of RAM modules and the current screen mode will determine the highest available RAM.

## PERIPHERAL DEVICES

Peripheral devices of several types can be added to the system using standard cables to either the serial bus or the connectors at the front of the computer console. The most common types deal with either transmission of bytes of data (usually serial bus) or transmission of sense information (usually game controllers).

### Game Controllers

The OS periodically senses (50 or 60 times per second) the standard game controllers (Paddles and Joysticks) and the values read are stored in RAM. You can plug in, remove, and rearrange these controllers at will without affecting system operation, because the system will always try to read all of these controllers.
The Driving Controllers are read, but not decoded, by the OS. Special instructions are required to read the keyboard controller (see Section 11).

### Program Recorder

The ATARI 410[TM] Program Recorder is a special peripheral. It uses the serial bus to send and receive data, but does not conform to the protocol of the other peripherals that use the serial bus. The Program Recorder must also be the last device on the serial bus, because it does not have a serial bus extender connector as the other peripherals do. There can never be more than one Program Recorder connected to any system for the same reason. The system cannot sense the presence or absence of the Program Recorder, so it can be connected and disconnected at will.

## Serial Bus Devices

A serial bus device conforms to the serial I/O bus protocol as defined in Section 9, but this does not include the Program Recorder. Each serial bus device has two identical connectors: a serial bus input, and a serial bus extender. Either connector can be used for either purpose. Peripherals can be "daisychained" by cabling them together in a sequential fashion. There are usually no restrictions on the cabling order because each device has a unique identifier. Where restrictions exist, they will be mentioned in Section 5.

## 4 SYSTEM MEMORY UTILIZATION

Memory in the system is decoded in the full 64K range of the 6502 microcomputer and there are no provisions for additional mapping to extend memory. Memory is divided into four basic regions (with some overlap possible): RAM, cartridge area, I/O region and the resident OS ROM. The regions and their address boundaries are listed below (all addresses are in hexadecimal):

```
0000-1FFF = RAM (minimum required for operation)
2000-7FFF = RAM expansion area
8000-9FFF = Cartridge B, Cartridge A (half of 16K size) or RAM
A000-BFFF = Cartridge A or RAM
C000-CFFF = Unused
D000-D7FF = Hardware I/O decodes
D800-DFFF = Floating Point Package (OS)
E000=FFFF = Resident Operating System ROM
```

Figure 4-1  6502 System Memory Map

This section will break these regions into even smaller functional divisions and provide detailed explanations of their usage.

### RAM REGION

The OS and the control program share the RAM region. The RAM region can be further subdivided into the following sub regions for discussion purposes:

```
Page 0 = 6502 page zero address mode region.
Page 1 = 6502 stack region.
Pages 2-4 = OS database and user workspace.
Pages 5-6 = User program workspace.
Pages 7-XX = Bootable software area/free RAM.*
Pages XX-top of RAM = Screen display list and data.*
```

* Note that XX is a function of the screen graphics mode and the amount of RAM installed.

The paragraphs that follow describe how the OS uses RAM subregions, and presents user program recomendations.

## Page 0

The architecture of the 6502 microcomputer instruction set and addressing modes gives page 0 special significance. References to addresses in that page (0000 to 00FF) are faster, require fewer instruction bytes, and provide the only mechanism for hardware indirect addressing. Page 0 should be used sparingly so that all possible users can have a portion of it. The OS permanently takes the lower half of page 0 (0000 to 007F). This portion can never be used by any outer environment unless the OS is completely disabled and all interrupts to the OS are eliminated.

The upper half of page 0 (0080 to 00FF) is available to outer environments with the following restriction: the floating point package, if used, requires 00D4 through 00FF.

## Page 1

Page 1 is the 6502 hardware stack region; JSR instructions, PHA instructions, and interrupts all cause data bytes to be written to page 1. Conversely RTS, PLA, and RTI instructions all cause data bytes to be read from page 1. The 256 byte stack is adequate for normal subroutine calls plus interrupt process nesting, so no restrictions have been made on page 1 usage. It is obvious that a stack of this size is totally inadequate for deeply recursive processes or for nested processes with large local environments to be saved. So, for sophisticated applications, software maintained stacks must be implemented.

The 6502 stack pointer is initialized at power-up or system reset to point to location 01FF. The stack then pushes downward toward 0100. The stack will wrap around from 0100 to 01FF if a stack overflow condition occurs, because of the nature of the 6502's 8-bit stack pointer register.

## OS Data Base

Locations 0200 through 047F are allocated by the OS for working variables, tables and data buffers. Portions of this region can be used only after you determine that nonconflict with the OS is guaranteed. For example, the printer and cassette buffers could be used if I/O operations to these devices are impossible within the controlling environment. The amount of work involved in determining nonconflict seems to be completely out of line with the benefits to be gained (except for a few trivial cases) and it is recommended that pages 2 through 4 not be used except by the OS.

## User Workspace

Locations 0480 through 06FF are dedicated for outer environment use except when the floating point package is used. The floating point package uses locations 057E through 05FF.

## Boot Region

Page 7 is the start of the "boot region." When software is booted from either the diskette or the cassette, it can start at the lowest free memory address (that is 0700) and proceed upward (although it can also start at any address above 0700 and below the screen display list). The top of this region defines the start of the "free memory" region. When the boot process is complete, a pointer in the data base contains the address of the next available location above the software just booted. When no software has been booted, this pointer contains the value 0700.

## Screen Display List and Data

When the OS is handling the screen display, the display list that defines the screen characteristics and the current data that is contained on the screen are placed at the high address end of RAM. The bottom of this region defines the end of the free memory region and its location is a function of the screen mode currently in effect. A pointer in the data base contains the address of the last available location below the screen region.

## Free Memory Region

The free memory region is all the RAM between the end of the boot region and the start of the screen region. The outer level application is responsible for managing the free memory region.

## CARTRIDGES A AND B

There are two 8K regions reserved for plug-in cartridges. Cartridge B, that is the right-hand cartridge slot found only in the ATARI 800 Home Computer, has been allocated memory addresses 8000 through 9FFF. Cartridge A (the left-hand cartridge slot in the ATARI 800 Computer console, and the only slot in the ATARI 400 Computer console) has been allocated memory addresses A000 through BFFF and optionally 8000 through BFFF, for 16K cartridges. If a RAM module is plugged into the last slot such as to overlay any of these addresses, the RAM takes precedence as long as a cartridge is not inserted. However, if a cartridge is inserted, it will disable the entire conflicting RAM module in the last slot in 8K increments.

MAPPED I/O

The 6502 performs input/output operations by addressing the external support chips as memory; some chip registers are read/write while others are read-only or write-only (the ATARI Home Computer Hardware Manual gives descriptions of all of the external registers). While the entire address space from D000 to D7FF has been allocated for I/O decoding, only the following subregions are used:

```
D000-D01F = CTIA
D200-D21F = POKEY
D300-D31F = PIA
D400-D41F = ANTIC
```

Figure 4-2. Mapped I/O

RESIDENT OS AND FLOATING POINT PACKAGE ROM

The region from D800 through FFFF always contains the OS and the floating point package. Care should be taken to avoid using any entry points that are not guaranteed not to move, to allow for the possibility that another, but functionally compatible, OS can be generated in the future. The OS contains many vectored entry points at the end of the ROM and in RAM that will not move. The floating point package is not vectored, but all documented entry points will be fixed: Do not use undocumented routines found by scanning the listing. A list of the fixed ROM vectors can be found in Appendix J.

CENTRAL DATA BASE DESCRIPTION

See Appendix L.

MEMORY DYNAMICS

The free memory region is the area between the end of the boot region and the start of the screen region. As such, its limits are variable. MEMLO [02E7] defines the bottom of the free region, and MEMTOP [02E5] defines the top of the region. This section presents the conditions that cause the setup or alteration of these variables.

System Initialization Process

The OS determines the extent of the lowest block of contiguous RAM, and saves the limits. The Screen Editor is then opened, thus setting a new (and lower) value in MEMTOP. Diskette or cassette-booted software might be brought into memory, that would probably set a new (and higher) value in MEMLO (see Section 7). MEMLO and MEMTOP will define the maximum amount of free memory available when the application program finally gets control. That amount of free memory can later decrease, as described in the next paragraph.

Changing Screen Modes

The Display Handler interprets the variable APPMHI '[000E]' to contain the address below which MEMTOP cannot extend. This allows you to protect the portion of free memory space that you are using from being overwritten as a result of screen mode change. The display handler will set the screen for mode 0, update MEMTOP, and return an error status to you, if it determines that the screen memory will extend below APPMHI as a result of a screen mode change. In other cases the Display Handler effects the desired mode change and updates MEMTOP.