SORRENTO VALLEY ASSOCIATES, INC.

CONSULTING ENGINEERS
COMPUTER APPLICATIONS

APP-L-CACHE

Operators Manual

RELEASE LEVEL 2.0

800857-B

11722 SORRENTO VALLEY ROAD, SAN DIEGO, CA 92121
TELEPHONE (714) 452-0101

BULLETIN 9-8-82

RE: 128K APP-L-CACHE AND VISICALC

The 128K version of the APP-L-CACHE has a minor bug in the software that doesn't allow access to the 16K language card and the main memory of the Apple. Adding 1 line to the basic program corrects this problem.

Apple	Operator Boot VC-256 disk.	Remark Loads basic program.
VC256 greeting	Ctrl reset	Exits to Applesoft.
]	List	
]	5 Poke-16241,60	Inits App-L-CACHE
]	Save Hello, S6, D1	

Now when you boot your VC256:disk with your 128K APP-L-CACHE installed you will have 144K available and not 111. The Hello program supercedes the VC-256 basic program.

Table of Contents

2.0 Description

2.1 Overview

What is an App-I-cache? - The App-I-cache is essentially a peripheral board that fits into your Apple II computer which contains up to 256 kilo-bytes of random access memory similar in nature to the 48k of ram already installed in your Apple. This gives you up to **304k bytes** of ram on board!

How do I use it? - The App-I-cache memory system allows you to address this extra memory as either a **ram disk** or (following the protocol outlined here) in 64 4k byte "banks".

What is a Ram Disk? - In this lies the elegance of the App-I-cache: namely allowing the computer to perform as if this extra memory was another mini-disk drive (a "pseudo disk drive", so to speak). There are no moving parts in the App-I-cache so software supplied on your distribution diskettes allow your operating system (for example Dos) to "see" the App-I-cache as a mini drive. And because there are no moving parts the App-I-cache is **FAST**, up to 25 times faster than a mini drive!

What do I get? - The SVA App-I-cache memory system is composed of the following parts:

A Main Board - with 64k bytes of ram on board, cabling and rom based software which allows it to respond as a language card and to emulate a mini-disk drive.

An Optional Memory Board - which adds additional memory when attached to the main board, yielding a maximum total of 256k bytes of ram available to the Apple II computer.

Distribution Software - which initializes the App-I-cache and interfaces it to the appropriate operating system such as Dos, Pascal, CP/M or Visicalc; requiring less than 300ma from the Apple's power supply.

How does it work? - The App-I-cache is actually two systems in one. When inserted into slot 0 of your 48k Apple II computer, one part of the App-I-cache is a direct replacement for a 16k language card. This means that, with few exceptions, any software currently running with a language card will run identically with the App-I-cache.

In addition, the App-I-cache provides up to 240k bytes of additional random access memory. This can be accessed indi-vidually or, more conveniently, as a group emulating a disk drive. You now can read and write to the App-I-cache without the significant delays normally associated with a floppy disk drive.

Some Applications - Adding the App-I-cache to your single mini-drive system will give you not only all the features of a language card, but the equivalent of up to two additional mini drive volumes "on line," running at speeds greater than a hard disk!

2.2 Applications

The following chart outlines some applications of the App-I-cache available with each configuration:

TABLE 2.1

Operation	64k	128k	192k	256k systems
Sort data files	X	X	X	X
Store & retrieve data (note 1)	X	X	X	X
Multisegment programs	X	X	X	X
Mini disk backup (note 4)			X	X
Total systems disk (note 2)			X	X
High speed compilation (note 3)			X	X

Note: UCSD PASCAL environment:
(1) up to 25 times faster
(2) 476 blocks available for file storage
(3) appx. 2 times faster

DOS 3.3 environment
(4) resembles two DOS volumes

Also, advanced application users can address the full 256k of ram and also may streamline data management by addressing special utility routines provided onboard in the 2k rom.

2.3 Operating System Support

The App-I-cache supports the following operating systems with some special features.

Pascal Versions 1.0 & 1.1 - The App-I-cache facilitates both program development and run-time data management with hard disk speed and versatility. With 256k of RAM, the 3 disk Pascal

operating system can be loaded into the App-I-cache at startup time along with your current development files. After this the App-I-cache can automatically be set to any volume number and also can be made the ROOT volume. You now may edit, compile or assemble at faster than hard disk speeds. You'll truly be amazed at the speed and power of Pascal program development and housekeeping with your App-I-cache and one mini drive!

Dos 3.3 - App-I-cache supports all Dos commands. The App-I-cache may be configured to resemble one or two mini disk drives, allowing a broad range of program applications.

CP/M - Your App-I-cache will support all CP/M operating system commands except operations on the boot track (0). It is configured to be drive E: with 238k bytes of memory.

Visicalc - The App-I-cache will support Visicalc with some unique features to fully use the additonal memory available. You can create an expanded worksheet and save these large files on to up to eight mini diskettes.

3.0 Installation

This section gives step-by-step instructions for installing your App-I-cache. Please read these instructions carefully before proceeding. You also may use this outline as a checklist.

Step 1: Inventory - You will need the following items before proceeding:

1) An APPLE II computer with a full 48k of ram onboard and at least one floppy disk drive in operation.

2) The App-I-cache Memory System Kit, including:

   a) A Main board (with an optional Memory board attached).

   b) Two ribbon cables with connectors.

   c) A chip puller (resembling a large pair of tweezers).

(Please see figure 2.1)

System upgrade - If you previously had acquired only the 64k Main board and now are adding a Memory board (upgrading to 128, 192 or 256k), refer to the installation section included with your upgrade kit.

Step 2: Open Your Apple

1) - Turn off the computer and disconnect the power cord. This is very important as it is almost certain that your computer and/or your peripheral cards will be damaged if tampered with when the computer's power is on. The **POWER** lamp on the lower left of the Apple keyboard is not illuminated when the power is off.

2) - Remove the cover by pulling up on its rear edge until the fasteners at the corners pop apart. Now slide the cover back until the cover is clear and may be lifted off the computer.

Step 3: Find Slot 0 - Familiarize yourself with the inside of your Apple II. Notice the big silver or gold colored box on the left-hand side; this is the power supply. Eight long and narrow slots are located at the rearmost part of the enclosure. These are the peripheral slots. Slot 0 is the leftmost of these slots and it is here that the App-I-cache must be installed. Further, notice a 4-

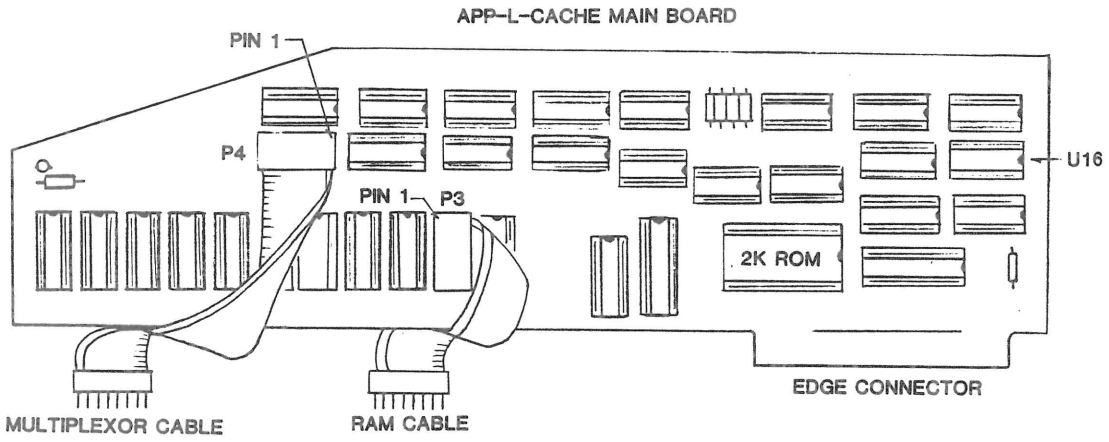


Figure 3.1

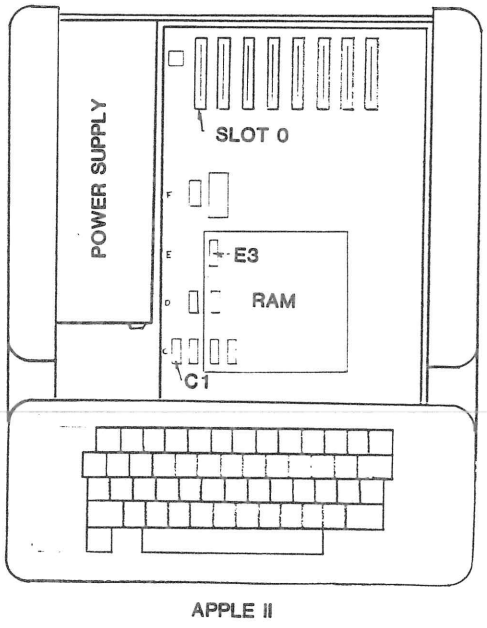


Figure 3.2

inch by 4-inch white square painted around an array of "chips", located directly behind the keyboard. This is the computer's main memory (or ram) (please see figure 3.2).

Step 4: Remove Any Card in Slot 0 - Since the App-I-cache must reside in SLOT 0, you must remove any peripheral card that is in it now. Some possible cards in this slot are a language card (directly replaced by your App-I-cache) or an Apple Firmware Card (this means the BASIC on this card is no longer available). Be sure to touch the power supply to discharge any static electricity you may have before you touch anything inside your Apple. If you don't have a card in Slot 0 then please continue.

Step 5: Remove Two Chips - Before you can install the App- I-cache you must remove two chips inside your APPLE II. Later you will insert plugs into the two empty sockets.

1) Locate the Sram chip - Again locate the 4-inch by 4-inch white square inside the computer. You must remove the left rear memory chip in this area (location E3) (please see figure 3.2).

Remove the Chip

a) - Touch the power supply cover to discharge any static electricity you may be carrying.

b) - Use the chip extractor (it looks like tweezers) included in the App-I-cache package to grasp the chip at both ends (please see figure 3.3). There is a slot under the chip at each end where the extractor prongs will fit. Wiggle the chip very slightly to loosen it. Apply no more upward or sideways force than necessary to remove the chip so that it will not be damaged.

2) Locate the multiplexor chip - Locate the front left corner of the 4-inch by 4-inch white square inside the computer. You must remove the second chip to the left of this corner in location C1. It is a 74LS153 multiplexor (please see figure 3.2).

Remove The Chip - Refer again to steps 1a and 1b above.

Step 6: Install the App-I-cache - You now are ready to install your App-I-cache.

1) Examine the App-I-cache - Set the App-I-cache in front of you, face up. Familiarize yourself with the ram cable, the multiplexor cable and the edge connector (please see figure 3.1).

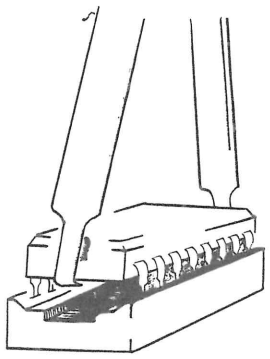


Figure 3.3

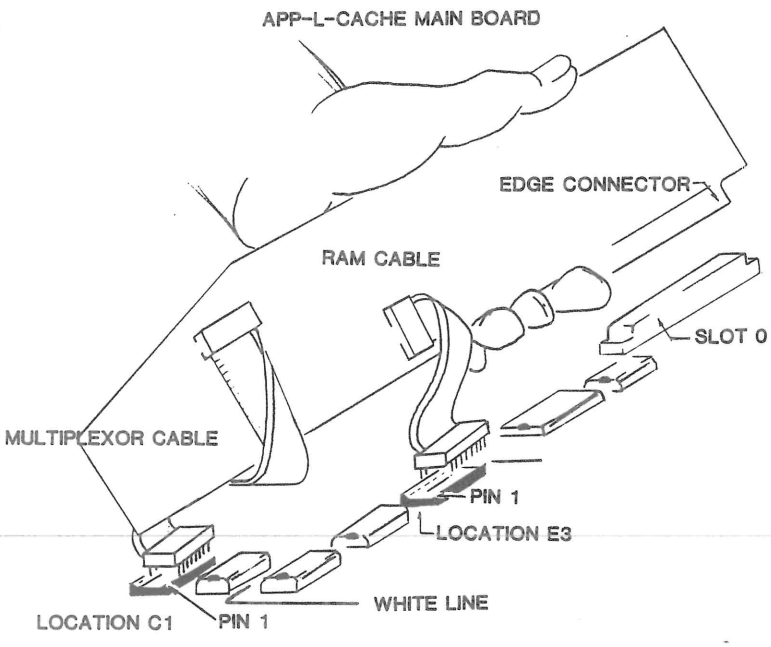


Figure 3.4

The **ram** cable is the short cable in the center of the board, attached to plug P3, with the ribbon at the right side of the connector and pin (1) at the left top edge (please see figure 3.1).

The **multiplexor** cable is the long cable at the top front edge of the board attached to plug P4, with the ribbon at the bottom of the connector and pin (1) at the upper right edge (please see figure 3.1).

The **edge connector** is the gold plated tab at the bottom right of the board. This will plug into Slot 0 (please see figure 3.1).

Plug in the cables

a) - If the cables are not attached to the Main board, then take a moment to plug them in. Make certain that the **ram** (short) cable is in P3 and the **multiplexor** (long) cable is in P4 with pin (1) oriented as shown in figure 3.1.

b) - Hold the App-I-cache above Slot 0 with your left hand. With your right hand fold the **multiplexor** cable under the App-I-cache and plug it into the now-empty multiplexor socket outside the lower left of the 4-inch by 4-inch white square as per figure 3.4. Note that **pin (1)** is toward the computer's keyboard. (Pin 1 is indicated by a white dot on the cover of the plug).

c) - Again with your right hand, fold the short **ram** cable under the App-I-cache and plug it into the now-empty ram socket just inside the upper left corner of the 4-inch by 4-inch white square. Note that **pin (1)** is again toward the computer's keyboard (please see figure 3.4).

Plug in the App-I-cache - Position the App-I-cache so the **edge connector** is just resting on top of Slot 0. Now press firmly down on the App-I-cache. You may rock the card from end to end but **do not twist** the card in the slot.

Step 7: Double Check All Connections - Verify that all connections are correct. Check that the ribbon cables are plugged in the **correct direction** and that all the little gold colored pins are in their sockets.

Step 8: Replace Cover - When you are satisfied that the plugs and card are correctly placed and installed, slide the cover of your Apple II toward the keyboard. When the cover is in place, press down on the rear corners of the cover until the fasteners snap closed.

Step 9: Power Up - Your Apple II now is ready to run. You can switch on the power and bring up Pascal, Dos, CP/M or Visicalc (please see the appropriate section in chapter 4: Operation).

4.0 The Pascal Environment

4.1 Overview

The App-I-cache may be configured to resemble a floppy disk drive with any valid PASCAL unit number. "CACHE:" and "App-I-cache" will be used interchangeably in this description to refer to the unit and volume. The App-I-cache volume contains:

476 blocks with the 256k system, 348 blocks with the 192k system, 220 blocks with the 128k system or 92 blocks with the 64k system

available for file storage.

This section is dedicated to instructing you on the use of Pascal and the App-I-cache. Under the Pascal environment App-I-cache use can range from simply a standard language card replacement (ignoring the extra ram on board) to a high speed program development tool. The following material is presented in more or less a step-by-step manner terminating with section section 4.3 **The Pascal Boot Procedure.** Numerous options are available to allow you to run the App-I-cache as effectively as a hard disk for program development and disk emulation. These are outlined under section 4.4 **Pascal Configure Options.** Constructing diskettes CACHE1: and CACHE2: as outlined below will give you the tools required to explore the App-I- cache/Pascal world.

4.2 Building the Diskettes

The App-I-cache requires two specially configured mini-floppy diskettes for operation in the Pascal environment. This section describes the creation of these diskettes.

Due to size, 64k and 128k users might find it convenient to configure the system disks as required for their particular application.

4.2.1 Diskette CACHE1:

This floppy disk will be used for **program operation** only. It will boot the computer and can configure your App-I-cache to automatically copy over the PASCAL system files, set the CACHE: unit number, make CACHE: the ROOT volume and/or copy any additional application files resident on this (or other) diskettes to the App-I-cache, thus providing you with high speed program and data access.

Format CACHE1: - The first step in creating diskette CACHE1: (or any other App- I-cache bootable diskette) is to write special App-I-cache boot code onto a standard Pascal formatted mini diskette (see your Apple Pascal users manual for the standard formatting procedure).

Patch the boot code on CACHE1: by executing **BPATCH** (located on your App-I-cache diskette ONE:). Insert your new diskette CACHE1: into a mini drive and type the unit number of the drive ('4' for drive 1, '5' for drive 2). BPATCH will tell you that the diskette is not currently patched, type 'Y' <ret> to patch the boot code on CACHE1:.

Build CACHE1:

Set up CACHE1: like this:

	source of file
CACHE1:	
SYSTEM.PASCAL	your PASCAL disk APPLE1:
SYSTEM.MISCINFO	APPLE1:
SYSTEM.LIBRARY	APPLE1:
SYSTEM.STARTUP	ONE:
RAMDRIVER	ONE:
STARTUP.OPTIONS	ONE:
SYSTEM.APPLE	APPLE1:

You will find that the order of the files on CACHE1: is important, as explained under the section 4.4 **Configure Options.**

What CACHE1: Does - When you power up the computer with CACHE1: in drive #4, the files

SYSTEM.APPLE
SYSTEM.PASCAL and
SYSTEM.MISCINFO

first initialize the language card portion of the App-I-cache, then boot the PASCAL operating system, which executes the file SYSTEM.STARTUP (as supplied with your App-I-cache).
SYSTEM.STARTUP in turn loads the contents of the file RAMDRIVER into memory and, based on the contents of the file STARTUP.OPTIONS, will automatically transfer files and set the ROOT volume and CACHE: unit number.

4.2.2 Diskette CACHE2:

This disk will be used in conjunction with CACHE1: for **program development.** It is the source for the automatic transfer of the editor, filer, compiler etc. to the App-I-cache, thus providing high speed access to the PASCAL operating system.

Build CACHE2:

Set up CACHE2: like this (special boot patch is not required here):

	source of file	
CACHE2:		
SYSTEM.EDITOR	your PASCAL disk	APPLE1:
SYSTEM.FILER		APPLE1:
SYSTEM.SYNTAX		APPLE1:
SYSTEM.COMPILER	your PASCAL disk	APPLE2:
SETOPTIONS.CODE	your App-I-cache disk	ONE:
BPATCH.CODE		ONE:
PBOOT.ALC		ONE:
CACHEDEMO.CODE		ONE:

The file order here is not important.

What CACHE2: does - After booting the computer with CACHE1:, you can insert CACHE2: into drive #4 to automatically transfer the contents of this disk including the PASCAL system files, any units, data, or workfiles you may be using into the App-I-cache. This gives you development performance unsurpassed on the APPLE II computer.

Besides the PASCAL system files CACHE2: contains **SETOPTIONS.CODE** (used to configure the file **STARTUP.OPTIONS**), **BPATCH.CODE** and **PBOOT.ALC** (used to patch 5" & 8" App-I-cache Pascal boot diskettes respectively) plus **CACHEDEMO.CODE** (demonstration program showing the relative speeds of a mini and the App-I-cache).

4.3 Pascal Boot Procedure

The following is a step-by-step description of the App-I-cache boot procedure under Pascal, showing the prompts and your responses to them. This is to help you become familiar with the App-I-cache under the PASCAL environment.

You should have created diskettes CACHE1: and CACHE2: as described above (recall that CACHE1: should have special boot code as written by **BPATCH**).

The file STARTUP.OPTIONS, as supplied with your App-I-cache, has fields SYSTEM & PROMPT set to TRUE, and UNIT NO set to 10. The App-I-cache will be configured as the ROOT volume (#10:) and will prompt the user for the source of the file transfers (as discussed in detail below).

Turn on the computer and insert CACHE1: into drive #4. The following display will appear on the screen and the appropriate response is shown:

S.V.A. App-I-cache STARTUP

TRANSFER FROM UNIT # ? (Type: '4' and then <ret>. If the volume can't be read then it will prompt again)

LOADING App-I-cache

READING... etc.

TRANSFER FROM UNIT # ? (Insert CACHE2: into a mini drive and type the drive number followed by <ret>)

READING... etc.

NEW DATE ? (<ret> for none; if required make sure CACHE1: is in drive #4 so that the new date may be written to the boot disk)

The command prompt line now will be present. You may go to the F(iler and L(ist '*' where you will find the App-I-cache is the ROOT volume, containing all the system files plus a demonstration program called CACHEDEMO. X(ecute this from the command level for a graphic demonstration of the relative speeds of the App-I-cache versus a mini drive.

4.4 Pascal Configure Options

What are they? - When you boot the Apple computer with disk CACHE1:, a number of options are available to you as determined by the file **STARTUP.OPTIONS.** This file has three fields: two boolean-like fields, **SYSTEM** and **PROMPT** (that can be thought of as either true or false), and a numeric field **UNIT NO** which represents the unit number you wish to assign to CACHE:.

By properly setting these options you may load your entire operating system onto the App-I-cache (including your current work file, special programs, units and data) plus configure the system ROOT volume and UNIT numbers all at boot time.

4.4.1 SYSTEM

The **SYSTEM** field specifies the ROOT volume.

TRUE - sets the ROOT volume to CACHE: allowing immediate access to system files resident on the App-I-cache. Your work file will grow here.

FALSE - ROOT volume is in unit #4 (usually the last disk booted in drive #4). This treats the App-I-cache as another volume and may be used for sorting data files and other runtime applications.

In **normal operation,** this field is set to **TRUE** for fast access to program development.

4.4.2 PROMPT

The **PROMPT** field prompts the user for the source (unit number) of files to be automatically transferred to the App-I-cache at boot time.

TRUE - prompts the user for the unit number of the source of transfer (the destination is always CACHE:).

FALSE - the program will default to #4: as the volume of the transfer source.

Normally, this field is set to **FALSE** unless automatic transfer from units other then #4: is desired.

Note that if both the **SYSTEM** and **PROMPT** fields are false, then **no** automatic transfer will take place.

4.4.3 UNIT NO

In addition to the SYSTEM and PROMPT options you also may select the unit number of the App-I-cache. This option is provided to allow execution of existing PASCAL software that ordinarily will run only on a predefined unit (e.g. #4: or #5:).

Under normal conditions the App-I-cache will default to unit 10.

If you select a unit other than 10 for the App-I-cache, the unit numbers are actually swapped (e.g. if you select 5 for the App-I-cache then 10 becomes your second mini drive).

You may select any valid Pascal unit number; these are (in order of system priority) 4,5,11,12,9,10.

In **normal** operation this field is set to **10.**

NOTE that if you select the App-I-cache to be volume #4: and you also select the SYSTEM option (making the App-I-cache the ROOT volume) you must transfer over SYSTEM.PASCAL and SYSTEM.MISCINFO to the App-I-cache at boot time because after executing the STARTUP program the operating system will need SYSTEM.PASCAL on volume #4:.

4.4.4 Setting The Options

The source of the above options is the file STARTUP.OPTIONS. This file contains three fields:

SYSTEM
PROMPT and
UNIT NO

If the file STARTUP.OPTIONS is not present at boot time then both SYSTEM and PROMPT options are assumed to be FALSE and CACHE: will default to unit 10.

How to do it - You may change (or create, if not present) the file **STARTUP.OPTIONS** by executing **SETOPTIONS.CODE.** Be aware that this program accesses the default volume and you may have to transfer your new file STARTUP.OPTIONS to your boot disk once created or modified.

What to expect

The following table outlines the action of the boot time options:

Table 4.1

Boot type	Options SYSTEM	PROMPT	BOOT TRANSFER Source Volume	BOOT TRANSFER Root Volume After Transfer	
power	false	false	- - - - - none - - - - - -		
power	true	false	#4	CACHE:	
power	false	true	prompt	#4	
power	true	true	prompt	CACHE:	
reset	false	false	- - - - - none - - - - - -		
reset	true	false	#4	CACHE:	note (1)
reset	false	true	- - - - - none - - - - - -		
reset	true	true	- - - - - none - - - - - -		

note (1): Providing the App-I-cache does not contain SYSTEM.PASCAL

Explanation of Table 4.1

Boot type - column refers to a "cold" (power) boot versus a warm boot generated by reinitializing the system (e.g. pressing the RESET button). Unlike the regular treatment of volumes in the PASCAL system, the App-I-cache is NOT reinitialized once configured from a cold boot. This means that your system files, work files etc. won't be lost if the RESET button is pressed.

Options - are the option fields present in SYSTEM.OPTIONS, as outlined above.

BOOT TRANSFER - columns outline the following:

Source Volume of the transfer - is the floppy drive to transfer to the App-I-cache.

Root Volume After Transfer - is the volume number (CACHE: for App-I-cache or #4 for the floppy disk drive) that expects the file SYSTEM.PASCAL and associated system files to appear. Your workfile will grow on this volume.

In other words...

POWER ON boot

If the SYSTEM field is TRUE then files are transferred from one or more source volumes to the App-I-cache.

If PROMPT is TRUE then the user is prompted for the source volume.

RESET (or CTRL-C) boot

Reloads SYSTEM.APPLE from #4: and executes SYSTEM.STARTUP on #4: which in turn checks the file STARTUP.OPTIONS. If the SYSTEM option is TRUE and the App-I-cache contains SYSTEM. PASCAL, then no transfer of files will take place.

The Automatic Transfer: An Anomaly - The one exception to this orderly volume-to-volume transfer, as explained above, is the treatment of the four special files on your disk CACHE1:

SYSTEM.STARTUP
RAMDRIVER.
STARTUP.OPTIONS
SYSTEM.APPLE

(Notice these are the last files on the disk).

These files have no purpose save the boot and configuration of the PASCAL operating system. With this in mind, if any or all of these four special files are the **LAST** files on a floppy disk undergoing the automatic transfer of files at boot time, they will be **deleted** from the App-I-cache after they are transferred, thereby making an additional 47 blocks available. However, if placed ahead of **ANY** other file on the source disk, they will **NOT be deleted** after transfer. For example, if a disk contains:

SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.LIBRARY
SYSTEM.STARTUP
RAMDRIVER
STARTUP.OPTIONS and
SYSTEM.APPLE

in that order, then the startup program will effectively transfer only:

SYSTEM.PASCAL
SYSTEM.MISCINFO and
SYSTEM.LIBRARY

over to the App-I-cache (provided, of course, that the SYSTEM option in the file STARTUP.OPTIONS was TRUE).

However, by placing SYSTEM.LIBRARY at the end of the directory, i.e.:

SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.STARTUP
RAMDRIVER
STARTUP.OPTIONS
SYSTEM.APPLE and
SYSTEM.LIBRARY

in that order, then every file will transfer over to the App-I-cache because the four special files were 'hidden' behind SYSTEM.LIBRARY, which is not one of these four files.

4.5 Misc. Pascal Notes

(*) Blocks 0 & 1 - On a standard PASCAL floppy disk, blocks 0 and 1 are reserved for boot code. This is not required on the App-I-cache so they are mapped into nonexistent locations.

(*) Boot code - When using the App-I-cache in the Apple II computer, Pascal must be booted with a specially modified boot disk. This diskette can be made by executing **BPATCH** (from distribution diskette ONE:).

(*) PASCAL 1.0 (also known as II.1) **vs. 1.1** - The App-I-cache automatically differentiates between the two versions of the operating system at boot time. The boot procedure is identical for both. Just build the App-I-cache diskettes CACHE1: and CACHE2: on a 1.0 or 1.1 floppy disk as desired (clearly, however, each version has its own unique set of system files and are themselves not necessarily interchangeable).

(*) Volume to Volume T(ransfer - The App-I-cache is protected against a full volume transfer. That is, the F(iler command

T(ransfer #4:,CACHE: (and vice versa)

will not be executed but the message "BAD I/O OPERATION" will appear. This is because blocks 0 & 1 do not exist on the App-I-cache. To effectively transfer a volume with the App-I-cache, use the wildcard ("="), thereby moving one file at a time.

(*) Accounting for volume space - A breakdown of the actual space available on the App-I-cache for file storage in the Pascal environment follows:

Table 4.2

	256k	192k	128k	64k system
Total RAM	256k	192	128	64k
Language card	-16k	-16	-16	-16k
Available for CACHE:	240k	176	112	48k bytes
@ 2 blocks/k byte	480	352	224	96 blocks
Directory	-4	-4	-4	-4 blocks
Actual file space avail	476	348	220	92 blocks

(*) <RESET> - The App-I-cache (volume CACHE:) is FULLY PROTECTED from RAM reinitialization when pressing the RESET key. To reinitialize the Pascal system, you must replace your boot disk, type RESET twice (to get into BASIC) and then type "PR#6 <ret>" which boots Slot 1 (Pascal drive #4).

Important - Don't power down unless all else fails because the contents of the App-I-cache will be lost!

(*) Update Often - As with all random access memory, a loss of power results in a loss of the data stored. W(rite your workfile to a floppy disk from time to time to minimize the effects of this possibility.

(*) Source Files - All App-I-cache Pascal sources have been included on your distribution disk ONE:. These are provided to assist the advanced application user in understanding and configuring the App-I-cache on startup.

4.6 Pascal With SVA 8" Floppy Drives

The App-I-cache is completely compatible with all 8-inch floppy drive systems marketed by Sorrento Valley Associates, Inc. to operating under Pascal:

1) **Write special boot code to your SVA 8" boot diskette.** This is achieved by the following steps:

a) First change your 8" booter code file PBOOT to PBOOT.8 for clarity.

b) Now transfer PBOOT.ALC to your 8" disk and change its name to PBOOT.

c) Now execute WRITEBOOTER (supplied with your 8" Pascal distribution diskette). This will write the code named PBOOT (formally PBOOT.ALC, you recall) to the boot track of your 8" diskette.

d) Now change the file PBOOT.8 back to PBOOT for future use.

2) **Build the 8" Disk**

To build the App-I-cache bootable 8" diskette simply transfer the files

```
RAMDRIVER
STARTUP.OPTIONS &
SYSTEM.STARTUP
```

to your typical bootable 8" floppy (recall that these files will be automatically deleted from the App-I-cache, as will SYSTEM.APPLE, if they are the **LAST** files on the floppy disk).

By placing your system and application files ahead of these four special file you can boot and load the App-I-cache from 8" drives.

5.0 The Dos Environment

5.1 Overview

Your App-I-cache will support all Dos 3.3 commands and may be configured to resemble up to two mini drives, (S1,D1 & S1,D2) thereby allowing the use of most commercial software.

The App-I-cache will speed up any I/O intensive Dos 3.3 routines including file management, sorting and program chaining. No disk controller card may be present in slot 1 for the App-I-cache to be properly addressed as a Dos 3.3 volume. This is due to the software assignment of slot 1 to the App-I-cache.

5.2 Dos Boot Procedure

To initialize the App-I-cache as a Dos 3.3 volume you need Applesoft Basic available and the files **RAMINIT** and **RAMINIT.OBJ** on a diskette (as supplied on the App-I-cache distribution diskette **TWO:**).

5.2.1 Boot on an Applesoft Machine

If your Apple computer has the Applesoft Basic in rom then:

a) - First boot any standard Dos 3.3 diskette (you can use your App-I-cache distribution diskette **TWO:**)

b) - Next type **RUN RAMINIT** (an Applesoft Basic program on diskette **TWO:**)

c) - Now type "Y" in response to the request to initialize. This will access the file **RAMINIT.OBJ**, patch the Dos and configure the App-I-cache as two volumes in slot 1:, drive 1 (and drive 2 with the 192 & 256k systems).

d) - Try typing "CATALOG S1,D1". This will confirm the presence of a valid Dos directory (volume 055).

5.2.2 Boot on an Integer Machine

If your Apple computer has Integer Basic in rom then you must first load Applesoft into the language card portion of your App-I-cache before you can run RAMINIT. This is done by booting with your Dos system master diskette. The Dos master diskette contains a HELLO program which automatically loads Applesoft into the language card.

5.3 Copying Dos Files

There are three ways of moving user programs and data to and from the 192k and 256k (full Dos volume) App-I-cache. These are:

1) - A **file-by-file** transfer using the FID program,

2) - An **image** copy using the App-I-cache **IMAGE COPY** utility program and

3) - An **image** copy at boot time, incorporated into the Dos **HELLO** program using the **FAST READ** and **FAST WRITE** binary files (as in the program **SAMPLE CACHE-LOAD HELLO** provided on TWO: the Dos distribution diskette).

An **image** copy is an identical track-by-track and sector-by-sector copy of a diskette. This can only be used with the full Dos volume on the **192k and 256k** App-I-cache (S1,D1) and a mini dirve. An image copy will not work with expanded Dos volumes such as with the SVA ZVX4 8" controller.

Although image copy always copies the whole disk -- regardless of how full it is - - it is actually **faster** than a file-by-file transfer. This is because the program does not have to stop and figure out which files to transfer and where they are on the disk.

5.3.1 The Image Copy: Two Examples (192k and 256k Only)

An image copy is ideal for one who wishes to boot the computer, initialize the App-I-cache and quickly copy over a Dos volume with a minimum of user interaction. This procedure can be used, for example, as the first part of a turnkey system or to streamline program development.

Example 1: IMAGE COPY Utility - This utility program (supplied on the Dos distribution diskette TWO:) allows you to read to and write from the 192k or 256k App-I-cache by doing a full diskette image copy from the mini drive of your choice.

Try the following example using FAST READ to transfer from slot 6, drive 1 to the App-I-cache slot 1, drive 1.

Computer	User	Comments
	PR#6	boot any Dos 3.3 diskette but not your Dos TWO: disk (note the App-I-cache is not initialized)
	remove Dos 3.3 boot disk	
	now insert TWO: into your mini drive	
	RUN IMAGE COPY	this will run the demonstration program
<main menu displayed>		
ENTER SELECTION:	1<cr>	Choose FAST READ program
<RAMINIT done> <FAST READ menu> SLOT (RET=6) ?	<cr>	for default source = S6
DRIVE (RET=1) ?	<cr>	for default source = D1
HIT RETURN	put disk to be copied into mini-drive 1 then hit <cr>	
READING: <35 dots>		35 dots, 1 dot per track transferred
]		program over
	CATALOG,S1,D1	Verify copy by cataloging App-I-cache

Fast Write - Runs in a manner similar to FAST READ, follow the example above selecting option 2. Be sure to insert an unused mini diskette when executing an image copy from the App-I-cache because the previous contents of the mini diskette will be destroyed.

Example 2: Boot Time Image Copy - Your Distribution diskette TWO: contains an example of a boot time image copy program: **SAMPLE CACHE-LOAD HELLO.**

This program was designed to be used as the HELLO program on a diskette you wish to copy into the App-I-cache at boot time.

Try the following demonstration:

Computer	User	Comments
	PR#6	boot any Dos 3.3 diskette but not your Dos TWO: disk (note the App-I-cache is not initialized)
	remove Dos 3.3 boot disk	
	now insert TWO: into your mini drive	
	RUN SAMPLE CACHE-LOAD HELLO	
		this will run the demonstration program: initing and copying disk TWO: into the App-I-cache
<35 dots>		prints 35 dots, one for each track transferred to the App-I-cache
<catalog of App-I-cache>		program does a catalog of S1,D1, note that the App-I-cache is now connected!

5.4 Misc Dos Notes

(*) Compatibility - App-I-cache supports most Dos 3.3 commands and can be configured to resemble up to two mini drives.

(*) Location - The App-I-cache uses slot 1, drive 1 (and drive 2 with the 192k and 256k systems). For this reason, no disk controller card may be in slot 1 when using the App-I-cache as a Dos 3.3 volume.

(*) Configuration - The following table shows the actual available storage under each App-I-cache configuration:

Table 4.1

App-I-cache Size	Drive 1 Sectors	Drive 1 Kbytes	Drive 2 Sectors	Drive 2 Kbytes
64k	176	44		
128k	448	108		
192k	544	136*	128	32
256k	544	136*	384	96

(*) Full sized Dos volume

(*) Volume matching - No volume matching is done against the App-I-cache during I/O operations. This is because you can't change floppy disks on the App-I-cache and thereby change the volume number. Drive 1 (and 2 with the 192 and 256k systems) uses volume number 055.

(*) App-I-cache With SVA 8" Floppy Drives - The App-I-cache is completely compatible with all 8-inch floppy drive systems marketed by Sorrento Valley Associates Inc. Simply transfer the files RAMINIT and RAMINIT.OBJ from your distribution disk TWO: to a Dos 3.3 bootable 8-inch disk. Once booted on an 8", RUN RAMINIT to initialize the App-I-cache.

6.0 The CP/M Environment

6.1 Overview

Your App-I-cache will support all CP/M operating system commands except operations on the boot track (0). It is configured to be drive E:.

To connect the App-I-cache you simply execute the CACHE program which is supplied on the App-I-cache distribution diskette THREE:.

All source code for the drivers is provided to allow custom installation of the App-I-cache for your particular hardware configuration, if required.

The version "B" App-I-cache software is TPA resident. This approach reduces the amount of available user memory by approximately 6k bytes.

6.2 Making a Boot diskette

Format a mini diskette for use as your CP/M boot diskette. Use the MicroSoft COPY program to copy the App-I-cache distribution diskette to the newly formatted boot diskette . Also put a 56k CP/M System and PIP.COM on this new disk for convenience(see example below).

Copying the App-I-cache files to new disk

Computer	User	Comments
	Install disks A: is 56k CP/M system disk B: is a blank disk	
	Boot CP/M System disk	
A>	**COPY B:=A:** <cr>	Copy App-I-cache distribution disk THREE: to a newly formatted disk in drive B:
	Install disks A: is App-I-cache dist. disk B: is App-I-cache new disk	
PRESS RETURN etc.	<cr>	Start copy

ANOTHER COPY N

INSERT etc.

Install
A: is CP/M system disk

<cr> return to CP/M

Copy 56k CP/M system to new disk

Computer	User	Comments
A>	**COPY B:=A:/S** <cr>	Copy 56k CP/M system to new disk
	Install disks A: is CP/M system disk B: is App-I-cache new disk	
PRESS RETURN etc.	<cr>	Start copy
ANOTHER COPY N		
INSERT etc.	Install A: is CP/M system disk B: is App-I-cache new disk	
	<cr>	return to CP/M

6.3 CP/M Boot Procedure

To initialize the App-I-cache boot your new CP/M 56k system disk that contains the file **CACHE.COM** provided on the App-I-cache distribution disk. Once booted type:

Computer	User	Comments
A>	**CACHE** <cr>	

The system will respond with the App-I-cache sign on message and indicate that drive E: is assigned to the App-I-cache and the latest version number. If the CACHE program has been previously run, the sign on message will indicate that the App-I-cache has already been connected.

6.4 Misc CP/M Notes

(*) Software Incompatibility - Do **NOT** run the format program **FORMAT.COM** on the App-I-cache. It will format your system disk by mistake. Also do **NOT** run **COPY.COM**. Use PIP.COM instead.

(*) Software Configuration - The App-I-cache is configured as Drive E: and Drive F: has been disabled. The 256k Drive E: provides 238K bytes of storage.

When accessed via memory management each 4K bank corresponds to one track. For example Bank 03CH is track 1 and bank 0 is track 62. There are 16 sectors/ track, 256 bytes/sector in each 4k bank (see chapter 5, Memory Management). Track 0 (the boot track) is not provided.

(*) Source code - Source code for the App-I-cache drivers is provided. Two programs may be assembled using MAC by Digital Research and combined using DDT to form CACHE. The first program is RAMCON.ASM which provides the CP/M patching and the driver relocation. The actual driver is RAMDRV.ASM. This program handles reads and writes to the App-I-cache and includes debug code for use during program modification. Note that DDT cannot be used to debug in the CP/M driver environment.

(*) CACHE.COM - must be on a disk that contains a 56k CP/M system to connect the App-I-cache.

(*) Interaction - The following outlines some methods of interacting with the App-I-cache.

Computer	User	Description
A>	E: <cr>	To select the App-I-cache drive
E>	A: <cr>	To return to the bootdrive
A>	PIP E:=A:*.* <cr>	To PIP some files to the App-I-cache
A>	STAT E: <cr>	To verify the App-I-cache size

(*) CP/M Memory Available

App-I-cache	CP/M Memory
256k	238k
192k	174k
128k	110k
64k	46k

(*) App-I-cache and ZVX4 CP/M Operation

The ZVX4 connect program is set up for four drives. A small change in the program will allow the user to have 5 drives on line with the 256K APP-L-CACHE on line as drive E.

Program Patch - Boot 56K CP/M with DDZVX4C.COM

Computer	User	Description
A:>	DDT DDZVX4C.COM<CR>	Enter DDT
0D00 100		
-	S19B<CR>	Change 19B
019B 04	05<CR>	From 4 drives to 5 drives
019C 32	.<CR>	End change
	<CTRL>C	Exit DDT
A:>	Save 12 ZVX4C.COM	Saves changed program to system disk

Operation - Boot 56K CP/M with ZVX4C.COM and CACHE

Computer	User	Description
A:>	CACHE<CR>	CONNECTS APP-L-CACHE to
sign on message		system as drive E.
A:>	ZVX4C<CR>	Connects 8" drives as C
sign on message		and D.
A:>		

Please log on to each of the five drives to make sure they are connected.

7.0 The Visicalc Environment

A Visicalc memory expansion system for use with the SVA App-I-cache in conjunction with mini, SVA 8" or hard disk drives.

7.1 Overview

VC256 is a bootable Dos 3.3 mini diskette. It contains an Applesoft program that loads and patches your Visicalc software to allow Visicalc full usage of your App-I-cache ram.

The standard Visicalc limits the maximum file size to 34K. VC256 expands the maximium Visicalc file size to the limit of your App-I-cache ram. VC256/Visicalc will save/load a fully expanded Visicalc file on up to 8 mini diskette volumes.

The SVA AMS5000 (hard disk) and AMS8000 (two 8-inch floppy drives) are automatically detected and incorporated into the VC256-Visicalc system.

System Requirements - VC256 requires an Apple II with a full 48k of ram, an SVA App-I-cache (64k through 256k systems), at least one mini disk drive and a Visicalc diskette with a release level matching that on the App-I-cache VC256 distribution diskette.

7.2 VC256 & Mini Drives

The Boot Procedure - The following procedure describes the boot process to be used each time you run VC256 and Visicalc. Once booted Visicalc operates with all standard commands and the Memory Indicator will indicate the expanded storage of VC256.

DON'T TAKE CHANCES! - make a backup copy of the VC256: diskette before proceeding.

Computer	User	Description
	Install	Mini disk in slot 6 (other slots are ok too)
	Boot	Using THE COPY of the SVA VC256: diskette
VC-256 <current release level & date>		VC256 program heading
SLOT # OF MINI DRIVE W/Visicalc DISK ?		

6 <cr>	Enter the slot number containing your mini drive (type '5' for most systems).
PUT PROGRAM DISK IN SLOT 6 THEN <RET>	Now insert your Visicalc diskette into your mini drive
<cr>	Type the return key when diskette is in place
	The screen goes blank while Visicalc is being loaded
Visicalc worksheet displayed	You are now ready to use Visicalc. Note the Memory Indicator in the upper right hand corner.

Saving the Worksheet - The App-I-cache/Visicalc worksheet can grow up to 252k bytes in size, which is more than can be stored on a single mini diskette. Therefore VC256: allows Visicalc to save large worksheets to more than one diskette. The worksheet is distributed between the diskettes where the file names are numbered consecutively.

Note: the multiple-disk save/load uses a single drive for saving or loading files.

For example, you could save a full 252k byte worksheet to a Dos file **MYFILE** as follows:

Computer	User	Description
	/ S S	from Visicalc get into SAVE mode
STORAGE: FILE TO SAVE		prompt for file name
	MYFILE,S6,D1	give file name and destination
<writing...>		writing to MYFILE until disk fills up If the file is large this can take an appreciable length of time

Caution: Be sure not to touch the keyboard during a file transfer or Visicalc may treat a stray keystroke to mean there are no more files to be transferred.

RETURN TO CONTINUE ON DISK #2 — disk full, prompts for another disk

insert another disk
<cr>

MYFILE.DISC2
<writing...> — continues to write file to disk as MYFILE.DISC2

Two Sets of Backups - When using multiple-disk saves we recommend that you keep three copies of your Visicalc worksheet: a master and two backup copies. This is because you cannot "write over" an older file with the same name during the save procedure. Before you save your large file use the Delete command to remove the files on your destination disk sets. By alternating your backup copies you can be certain to always have a valid backup available.

Loading the Worksheet - Once saved (as above) a "multi-source" file can be loaded into the Visicalc worksheet in a manner similar to the way it was saved. Simply insert the diskette with the first part of the file (for example MYFILE.DISC1) into the mini drive. Visicalc will prompt for other parts of the file to be inserted into the drive.

7.3 VC256 with SVA drives

Operation with the expanded storage capability of SVA drives is similar to operation with minis. During the boot process additional prompts are automatically provided to select the expanded storage to be used.

A typical system would consist of an Apple mini and an SVA AMS5000 hard disk or the AMS8000 8" floppy disk with ZVX4 controller. Since the VC256 and VC256.OBJ files are standard DOS files they may be moved to the SVA drives. The SVA drives would then provide the VC256 boot and Visicalc worksheet data storage. The mini would be used for loading the Visicalc disk.

Note: Multiple disk storage **cannot** be used with the hard disk because you cannot 'remove' or 'insert' a hard disk platter!

Computer	User	Description
	Boot AMS5000 hard disk	If available
	RUN VC256 <cr>	From mini, SVA 8", or SVA hard disk.
VC-256 <current version and date>		VC256 program heading
CONNECT HARD DISK (Y/N)?		The following hard disk prompts appear only if you have an SVA hard disk on line
	Y<cr>	Type response as desired

Caution: Do not use the Visicalc INIT command on hard disk volumes. Be sure that your hard disk has already been 'INIT'ed under Dos 3.3

Note: You may select 2 hard disk volumes to be D1 and D2 Visicalc will save/load on 2 drives per slot.(ie 2 hard disk volumes)

VOLUME NO OF DRIVE 1 ?

1<cr>	SELECT volume to be Visicalc Drive 1
DRIVE 2 ?	
2<cr>	SELECT volume to be Visicalc Drive 2
CONNECT ZVX4 (Y/N)?	The following ZVX4 prompts appear if you have an SVA 8" disk on line
Y<cr>	TYPE Y if you want to save/load data from SVA 8"

Caution: Do not use the Visicalc INIT Commend on 8" volumes. Be sure that your 8" disks have already been 'INIT'ed under Dos 3.3

Note: You may select 1 volume per drive to be D1 and D2. Visicalc will save/load on 2 drives per slot. Each SVA 8" disk has 1 to 4 volumes.

VOLUME NO OF DRIVE 1 ?

Note: specifying a nonexistent volume here (e.g. '3' for a single sided, double density) can cause unpredictable results when in the Visicalc program

1<cr>	Type (1..4) for double density,double sided; (1..2) for double density, single sided; (1) for single density,single sided;
DRIVE 2 ?	Prompt for volume
2<cr>	Again, Type (1..4)
SLOT # OF MINI DRIVE W/Visicalc DISK ?	Enter the slot number containing your mini
6<cr>	drive (type '6' for most systems)
PUT PROGRAM DISC IN SLOT 6 THEN <RET>	Now insert your Visicalc diskette into your mini drive
<cr>	Type the return key when diskette is in place The screen goes blank while Visicalc is being loaded
Visicalc worksheet displayed	You are now ready to use Visicalc
Save/Load/Quit	Type /Storage,Save,FILENAME,S7,D1 /Storage,Quit,S7,D1

7.4 Misc VC256 Notes

***) Maximum Visicalc worksheet storage available:**

App-I-cache	Worksheet
64k	81k
128k	144k
192k	207k
256k	252k

*) **Addressing Disk Files from Visicalc** - Worksheets may be saved/loaded through any slot and drive (1 or 2) from within Visicalc by specifying the slot and drive number after the file name (e.g. MYFILE,S6,D1).

*) **Running/Quitting VC256 from 8" and Hard Disk Drives** - To run VC256 from SVA 8" or the AMS5000 hard disk drives, transfer the files VC256 and VC256.OBJ from the VC256 diskette to the desired drive, then run from the appropriate slot by typing RUN VC256,SN where N is the slot number of the drive.

To quit from Visicalc type the Visicalc command /Storage,Quit,SN where N is the slot number of the drive.

*) **Booting VC256 on an Integer Basic Machine** - If your Apple II doesn't have Applesoft Basic available you must first boot with a diskette containing this language. The HELLO program on your boot disk should then load Applesoft Basic into the language card portion of your App-I-cache (see section 5.2.2 (Boot on an Integer Machine) for details). This enables you to insert the VC256 diskette and run its HELLO program as described above.

8.0 Memory Management

8.1 Introduction

This section is devoted to addressing your App-I-cache as an extension of your Apple's main memory and is intended for advanced application users only. For most purposes, using your App-I-cache as a disk drive within the operating environments described above is by far the most straightforward way of addressing the additional ram.

All App-I-cache sources are provided on the appropriate distribution diskettes. A valuable insight into App-I-cache memory management can be had through the study of these sources (the rom listing is provided in text form herein). Also note the run-time distribution diskette **CHTEST:** not only can test the App-I-cache operation but is a valuable development tool (see chapter 5:Support Software).

Refer to the Apple II reference manual sections on peripheral board I/O for pertinent background information on this section.

8.2 Overview

Overview: The Ram - The App-I-cache contains up to 256k bytes of random access memory; these are broken into 64 4k byte banks numbered ($ for hexidecimal) $0 through $3F. All 64 of these banks can be addressed through the Apple's top 12k byte address space (shared with all 8 I/O slots and the Apple's on board rom). The top 12k are locations $D000 through $FFFF, these can be viewed as three 4k byte banks with locations $D000 - $DFFF, $E000 - $EFFF and $F000 - $FFFF.

The lowest most bank ($D000 - $DFFF) of the 12k can be thought of as a "window" through which any one of the 64 4k App-I-cache banks may be mapped (see figure 8.1). The upper two banks ($E000 - $EFFF & $F000 - $FFFF) are unique because the two highest App-I-cache banks ($3E & $3F) also are mapped into these locations regardless of the contents of the window bank. These can act like a pair of scratch pads or buffers for use with the window bank (you also can select $3E or $3F as the window bank thereby having this bank mapped into two I/O locations at once!).

In this manner you may address all 256k bytes of App-I-cache through a 4k window plus have an additional 8k of buffer area available, all mapped into the 12k bytes of APPLE I/O (please see figure 8.1).

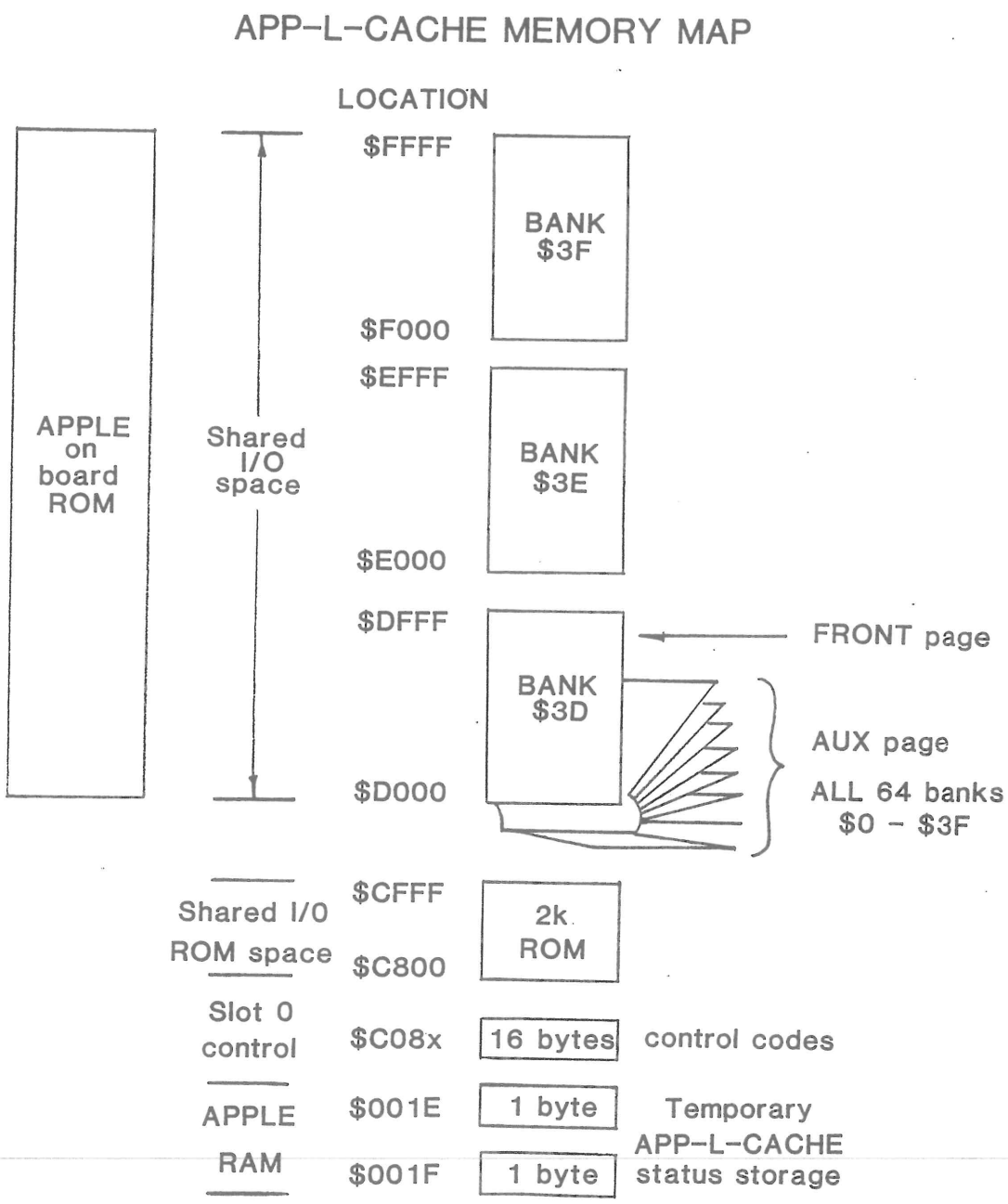

APP-L-CACHE MEMORY MAP
LOCATION
$FFFF
BANK $3F
$F000
$EFFF
BANK $3E
$E000
$DFFF
BANK $3D
$D000
FRONT page
AUX page
ALL 64 banks $0 - $3F
APPLE on board ROM
Shared I/O space
Shared I/O ROM space
$CFFF
2k ROM
$C800
Slot 0 control
$C08x
16 bytes
control codes
APPLE RAM
$001E
1 byte
Temporary APP-L-CACHE status storage
$001F
1 byte

Figure 8.1

Overview: The Rom - The APPLE computer contains a 2k byte "common area" that all 8 peripheral slots share. The 2k rom on your App-I-cache can be mapped into this space (see figure 8.1). It contains routines that **SAVE** and **RESTORE** the pertinent states of the ram and I/O slot.

Overview: The Language Card - A standard 16k language card uses a scheme similar to the ram mapping described above, but instead of 64 banks mapped into the $D000 - $DFFF window it only has two banks. All four 4k banks of a standard 16k language card may be accessed indiscriminately by the operating system in use, switching the two lowest banks into the window as required.

You still may take advantage of the additional 60 bytes of ram even though these top 4 banks ($3F - $3C) are being used as a language card.

The App-I-cache 2k rom contains a routine that will **SAVE** the current status of the language card and associated control states. This allows you to select and interact with any of the 60 unused banks ($3B - $0).

Another specialized rom routine will **RESTORE** the former language card status thereby allowing the host operating system to resume operation.

8.3 Interaction

Interaction: The Ram

Control Codes - The Apple computer has 16 address locations to control each peripheral slot. Since the App-I-cache resides exclusively in slot 0, locations $C080 - $C08F are used as control codes and determine the state of the App-I-cache.

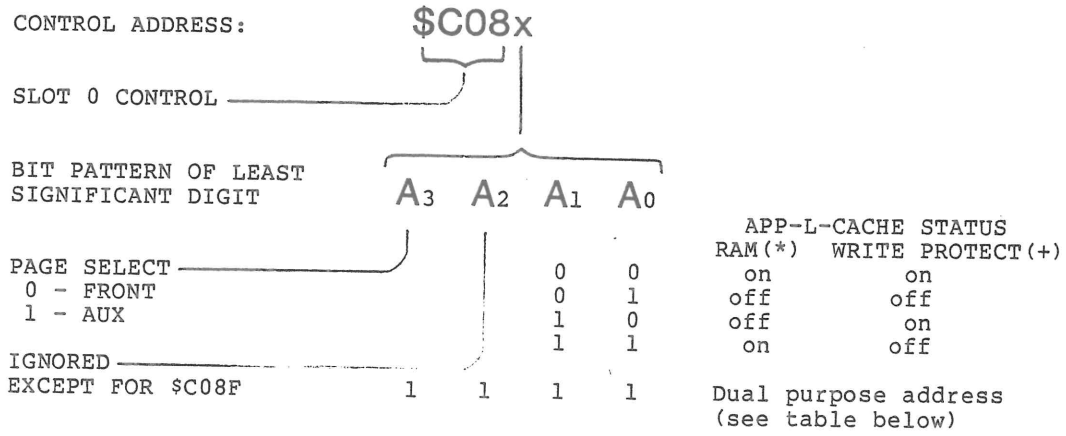
A breakdown of the bit pattern associated with this control code range is shown in figure 8.2.

Page Select - The window bank is mapped into I/O locations $D000 - $DFFF (affecting all 64 App-I-cache banks $3F - $0 as described above). We can break these banks into two "pages", FRONT and AUX, both of which use the I/O window bank.

When the FRONT page is selected bank $3D (and $3D only) always will be mapped into the window. However when the AUX page is selected any one of the 64 App-I-cache banks may be mapped into the window. This is done to allow emulation of a standard language card.

CONTROL CODE

Bit Pattern Breakdown



(*) RAM - 'on': APP-L-CACHE mapped to $D000 - $DFFF
'off': APPLE on board ROM mapped to $D000 - $FFFF

(+) WRITE PROTECT - Two sequential reads are required

Functional Table

$C080	0000	Selects RAM, FRONT page, and WRITE PROTECT
$C081	0001	Selects ROM, FRONT page, and WRITE ENABLE if accessed twice
$C082	0010	Selects ROM, FRONT page, and WRITE PROTECT
$C083	0011	Selects RAM, FRONT page, and WRITE ENABLE if accessed twice
$C088	1000	Selects RAM, AUX page, and WRITE PROTECT
$C089	1001	Selects ROM, AUX page, and WRITE ENABLE if accessed twice
$C08A	1010	Selects ROM, AUX page, and WRITE PROTECT
$C08B	1011	Selects RAM, AUX page, and WRITE ENABLE if accessed twice

$C08F 1111 READ - Switches on the 2k ROM and
READS card status with the format of:
BITS
0-5 bank selected
6 page (0-FRONT,1-AUX)
7 APPLE 12k ROM or APP-L-CACHE selected
(1-ROM,0-RAM)
WRITE - Selects the bank

$CFFF Switches off all peripheral card ROMs

Note: RAM - APP-L-CACHE RAM mapped into I/O addresses
ROM - APPLE on board ROM mapped into I/O addresses
WRITE ENABLE - if address is accessed twice consecutively

Figure 8.2

Selecting a Bank: An Example - This piece of machine code demonstrates general access to any one of the 64 App-I-cache 4k byte banks numbered $0 - $3F (recall, however, that banks $3D and $3F also are permanently mapped into locations $E000 - $EFFF and $F0000 - $FFFF). General access requires that the AUX page be selected (the FRONT page selects bank $3D only).

```
BIT  0C08B      Two consecutive calls to $C08B selects
BIT  0C08B      AUX page, & write enables the ram (the
                  low order bit pattern ($B) selects
                  the I/O (vs. Apple rom) & switches the
                  write protect off)

LDA  #10        Load accumulator with bank number desired
                  ($3F - $0 available through the AUX page)

STA  0C08F      Writing the bank number to this location
                  selects the bank
```

Now the App-I-cache bank $10 is mapped into 4k from $D000 to $DFFF

```
BIT  0C082      Write protects the ram and deselects the
                  App-I-cache (i.e. selects APPLE on board
                  rom)
```

Note: 64k, 128k & 256k Byte App-I-cache Users - The following table outlines the banks avaiable under each App-I-cache configuration.

App-I-cache	Banks Avail
64k	$3F - 30
128k	$3F - 20
192k	$3F - 10
256k	$3F - 0

Interaction: The Rom - The 2k rom on your App-I-cache was designed to aid an operating system in treating the App-I-cache as both a disk drive and a language card. You can take advantage of two primary functions contained within the rom to assist you in using the App-I-cache both under a high level operating system using the language card feature, and to simplify memory management applications.

This 2k rom is mapped into $C800 - $CEFF and shared by all 8 peripheral slots (see figure 8.1).

Note: - Any reference to the top 256 bytes ($CF00 - $CFFF) of this area will switch off the App-I-cache, switching on the APPLE's on board rom.

Use the following machine instructions to access this space:

BIT 0CFFF	Reading here switches off all peripheral roms
BIT 0C08F	Reading here switches on the 2k on board rom
JSR 0C80C	Save the current status of the APP-L-CACHE including bank number, page (FRONT or AUX) & wheather App-I-cache or Apple rom is mapped into $D000 to $FFFF

Now you may manipulate the App-I-cache through the I/O window. Be careful, however, not to write over locations $001E and $001F in the APPLE main memory because these locations now hold the previous status of the App-I-cache and will be used again by the rom to restore that configuration.

To return:

JSR 0C809	Restores App-I-cache to previous configuration

Interaction: The Language Card - When addressing the App-I-cache in the memory management mode described in this section, be aware that some operating systems (e.g. PASCAL) use the top 4k banks ($3C - $3F) indiscriminately. To have access to the additional banks (hopefully not configured as a disk drive!) you must first **SAVE** the current status of the language card portion of the App-I-cache, as described above, making certain to **RESTORE** the card when through.

Be careful not to modify the top 4 banks $3C through $3F as this will certainly cause the host system to crash.

9.0 Support Software

Included with the App-I-cache are four floppy disks (one for each of the four operating system environments and a diagnostic/development diskette). These contain special software that allows the App-I-cache to resemble a disk drive and source files for advanced application users. This section gives a brief description of each file.

9.1 ONE: Pascal Distribution Disk

ONE:	software release 800951 REV B
---CACHE1:---	empty file used as a marker
SYSTEM.STARTUP	determines if PASCAL 1.0 or 1.1 is in use. It then configures the App-I-cache and transfers files at boot time. This file must be present at the cold (power up) boot time.
RAMDRIVER	contains code that will support the App-I-cache as a volume. This file is used by SYSTEM.STARTUP at boot time.
STARTUP.OPTIONS	contains the fields SYSTEM PROMPT and UNIT NO that are used by SYSTEM.STARTUP to configure the App-I-cache.
---CACHE2:---	empty file used as a marker
SETOPTIONS.CODE	is used to create and/or modify the options present in the file STARTUP.OPTIONS.
BPATCH.CODE	is used to patch App-I-cache boot diskettes.
PBOOT.ALC	used as source for 8" Pascal boot code.
CACHEDEMO.CODE	code file that graphically demonstrates the relative speeds of the App-I-cache versus a mini drive using low level disk access routines with a minimum of processor overhead.
---SOURCES---	empty file used as a marker
RAMDRIVER2.TEXT	source file; assembled and used to patch the PASCAL operating system on startup.
SETOPT3.TEXT	source file for SETOPTION.CODE
STARTLIB2.TEXT	source file; assembled into machine code library file and linked into SYSTEM.STARTUP
STARTUPA5.TEXT	first part of source file for SYSTEM.STARTUP.

STARTUPB5.TEXT	second part of source file for SYSTEM.STARTUP.
BPATCH.TEXT	source file for BPATCH.CODE
PBOOT.ALC.TEXT	Source file for PBOOT.ALC.CODE
CACHEDEMO.TEXT	source file for CACHEDEMO.CODE

9.2 TWO: Dos Distribution Disk

TWO:	software release 800952 REV C
HELLO	file with App-I-cache sign-on message.
RAMINIT	Applesoft Basic file which contains the program and drivers required to configure the App-I-cache.
RAMINIT.OBJ	is the object file used by RAMINIT to discern the App-I-cache volume size at boot time.
A—— COPY ——A	empty file used as a marker.
IMAGE COPY	copy program host, calls FAST READ.E or FAST WRITE.A.
SAMPLE CACHE-LOAD HELLO	Applesoft program designed to demonstrate automatic disk image copying at boot time (see 5.3.1: The Image Copy).
FAST READ.A	read program, calls FAST READ.
FAST READ	binary file, does the read, cannot be BRUN.
FAST WRITE.A	write program; calls FAST WRITE.
FAST WRITE	binary file, does the write, cannot be BRUN.
A—— SOURCES ——A	empty file used as marker.
PRINT SOURCES	sends source files (.SRC) to printer.
FASTREAD.SRC	text file; sources for FAST READ.
FASTWRITE.SCR	text file; source for FAST WRITE.

Dos Source Listings - The assembly language sources for both FAST READ and FAST WRITE are available on disk "TWO:" in the files FASTREAD.SRC and FASTWRITE.SRC. These two files are Apple Dos sequential text files that can be read by most assemblers. In addition, the program PRINT SOURCES has been provided to print either of these files.

The following instructions for PRINT SOURCES assume that you have a printer interface card in slot 1:

COMPUTER	USER	COMMENTS
	RUN PRINT SOURCES	
<PRINT MENU> ENTER SELECTION:	1	choose FASTREAD.SRC
<source listing on printer>	form feed the printer	
<PRINT MENU> ENTER SELECTION:	2	choose FASTWRITE.SRC
<source listing on printer>	form feed the printer	
<PRINT MENU> ENTER SELECTION:	3	exit program

9.3 THREE: CP/M Distribution Diskette

THREE: software release 800953 REV B

DISKDEF LIB	CP/M library
Z80 LIB	CP/M library
RAMDRV ASM	source file for RAMDRV
RAMDRV PRN	assembled listing for RAMDRV
RAMDRV HEX	machine code for the App-I-cache CP/M drivers. Handles reads and writes and includes debug code for use during program modification.
RAMCON ASM	source file for RAMCON
RAMCON PRN	assembled listing for RAMCON
RAMCON HEX	machine code that loads in the driver (RAMDRV)
CACHE COM	App-I-cache initializing program; contains code from both RAMDRV and RAMCON

9.4 VC256: Visicalc Expansion Distribution Diskette

VC256:	software release 800960 REV C
HELLO	Applesoft program that runs VC256
VC256	Applesoft host program, checks & prompts for hard and/or 8" disks as storage locations and calls the binary file VC-256.OBJ
VC265.OBJ	Loads and patches VISICALC from the VISICALC disk

9.5 CHTEST: Diagnostic Distribution Diskette

CHTEST:	software release 800956 REV B
RTSTND.APPLE SYSTEM.PASCAL SYSTEM.LIBRARY SYSTEM.MISCINFO SYSTEM.CHARSET	code and data files to support this RUNTIME PASCAL Diskette.
----------	empty file used as a marker
CRDLIB.TEXT BNKLIB.TEXT ROMLIB.TEXT SWTLIB.TEXT RAMLIB.TEXT TSTLIB.TEST	source files for the machine level utilities used for various tests
CACHETEST.TEXT	Pascal source file for the host program
----------	empty file used as a marker
FIRMWARE.CODE	code file used as comparison (to verify the 2k on board rom)
----------	empty file used as a marker
SYSTEM.STARTUP	code file compried of the above machine level routines and the Pascal host file; used as diagnostic program for the App-I-cache.

Using CHTEST: - Boot the computer with CHTEST: in drive #1. Once booted your will see a prompt line beginning with the word "CHTEST:".

Now type 'R' to test the App-I-cache **ram.** Next type 'L' to test the on board switching **logic.** Next type 'E' to verify the on board **rom** (eprom).

Other commands are available for manipulating the App-I-cache banks. Follow the prompts for each option available to supplement the above basic tests.