SOLID STATE DISK (SSD) INSTALLATION AND OPERATING GUIDE

May 1982

SYNETIX
15050 NE 95th Street
Redmond, WA 98052

(206) 881-8440
1-800-426-7412

01-09901-934

DOS

CP/M

PASCAL

SSD DOS 3.3 VERSION 3.00 RELEASE ENHANCEMENTS

1. The power on "START-UP" program has been improved and compiled to speed up disk copying to the SSD cards, provide better display of the SSD format, speed up conversion of Dual SSD cards to 32 Sector format, and allow copying of Dual SSD card data to a single Disk Drive in one step.

2. Volume Numbers and error messages now work properly with the SSD.

3. 32 Sector formatted SSD card data can now be copied to the standard Disk Drives using the track copy program START-UP" or "NO-INIT" (only "FID" worked with Version 2.2 to copy files from 32 Sector formatted SSD).

4. The program "NO-INIT" has been added which allows the SSD interface software to work with programs like Syneristic's PLE and Microsoft's TASC compiler (see "NO-INIT" program).

5. The start-up procedure no longer differentiates between single and dual SSD cards as stated on page 6 of the manual. This allows the user more freedom to specify input and output slots and drives for copying.

SSD DOS 3.3 VERSION 3.01 RELEASE ENHANCEMENTS

Previous versions of the SSD Dos 3.3 I/O do not allow creating a "TURN-KEY" system using Dos "EXEC" files. Version 3.01 allows the use of Dos "EXEC" files to provide an "AUTO-BOOT" system.

BOOTING UP

The booting up procedure is the same as described in the SSD manual except that there is only one file run after "HELLO". The program run after "HELLO" is either "START-UP" or "NO-INIT". The user can choose to run one or the other of these SSD interface programs (see "NO-INIT" program for details).

VOLUME NUMBERS AND THE SSD

Disk Drive Volume Numbers work normally with the SSD but the following precautions should be observed:

1. If a copy is to be made from the SSD to a diskette make sure the diskette has been initialized to the correct Volume Number (see the DOS 3.3 manual).

2. The programs "START-UP" and "NO-INIT" initialize all SSD's in the computer to Volume Number 254 when the computer is powered on and the program is stepping the user through copying diskettes to the SSD cards. The SSD cards assume the Volume Number of the diskettes copied into it.

16 TO 32 SECTOR CONVERSION

Apple 5¼" diskettes are formatted into 16 sectors per track and 35 tracks per diskette giving a total storage capacity of 560 sectors or 143,360 Bytes under DOS 3.3.

Some applications require larger file storage space than is available on one diskette. The SSD can solve some of these applications by converting the SSD Dual card (model 2202) to a single disk with the capacity of two diskettes (286,720 Bytes).

The method used is the 16 to 32 sector conversion. The SSD Dual card is converted from two 16 sector per track disks to a single 32 sector per track disk with 35 tracks. The DOS system can access the double sized disk with standard commands.

THE CONVERSION PROCESS

When the program "START-UP" or "NO-INIT" is run the user is asked if he wishes to copy diskettes into the SSD cards. When no more SSD's are found the program asks the user if he wishes the Dual SSD's in the computer to be converted from 16 to 32 sector format. If the answer is yes the program will convert each Dual SSD's in the computer into single disks in 32 sector format.

NOTE: Any single SSD (model 2201) will remain in 16 sector format.

CAUTION Make sure both halves of all Dual SSD's have been initialized or have had a 16 sector diskette copied to them ("D1" and "D2") before running the 16 to 32 sector conversion.

LOADING AND SAVING DATA USING 32 SECTOR SSD'S

Data can be copied to and from the SSD using the following methods:

1. Copies of 32 sector SSD data can be made by running "START-UP" or "NO-INIT" and copying both halves of the SSD ("D1" and "D2") to two diskettes.

   NOTE: Mark the copies made as "D1" and "D2" since they must be copied back to the SSD to the same half of the SSD.

   NOTE: When 32 sector diskettes are copied to the SSD using "START-UP" or "NO-INIT" the program will recognize the data as 32 sector data and configure the SSD card as such.

2. The program "FID" can be used to copy files from any disk to any disk.

   NOTE: If any file on a 32 sector formatted SSD is larger than can be stored on a diskette then method 1 above MUST be used to make copies of the data on the SSD.

NO-INIT PROGRAM

The binary program "NO-INIT" has been added which is a compiled Applesoft program and SSD I/O program. The SSD I/O program is transferred into the INIT command area of DOS creating a "hidden" interface.

USING "NO-INIT" INSTEAD OF "START-UP"

NOTE: Always remember when using "NO-INIT": New diskettes CANNOT be formatted.

To use "NO-INIT", simply BRUN "NO-INIT" after booting the computer. The program is the same as "START-UP" except that the SSD interface part of the program is installed inside DOS instead of beneath DOS (between DOS and the File Buffers).

"NO-INIT" can be run automatically on power up by substituting the name "NO-INIT" in place of "START-UP" in the program "HELLO" on the DOS 3.3 SSD I/O diskette.

ADVANTAGES USING "NO-INIT"

1. No memory space is lost to the SSD interface program (the program "START-UP uses 512 Bytes of computer memory).

2. "NO-INIT" allows the user to run the SSD with programs like PLE and TASC compiler which arbitrarily install themselves or use absolute memory locations in or near DOS.

3. Users who have a configuration they always use can initialize diskettes with the SSD imbedded in the DOS (the SSD cards must still be initialized after power on).

DISADVANTAGES USING "NO-INIT"

A strict procedure for initializing new diskettes MUST BE followed since new diskettes CANNOT be formatted when "NO-INIT" is installed in the computer (see INITIALIZING NEW DISKETTES WHILE USING "NO-INIT").

INITIALIZING DISKETTES WHILE USING "NO-INIT"

New diskettes CANNOT be formatted with "NO-INIT" installed in the computer.

To format new diskettes, boot up the computer with a standard DOS 3.3, initialize the new diskettes, then BRUN "NO-INIT" to re-install the SSD interface program.

NOTE: Any data on the SSD cards will remain intact throughout this procedure provided the computer is not powered off or new data copies to the SSD's.

Previously formatted diskettes may be re-initialized with "NO-INIT" installed using the following procedure:

1. Catalog a known good diskette.

2. Init the diskette to be used.

Caution

NOTE: At this point the diskette has on it a copy of the DOS 3.3 with the SSD interface program imbedded in it and can be used. -CAUTION- if the SSD configuration is changed the diskettes MUST BE re-initialized.

3. A standard DOS 3.3 can be installed on the diskette by running the program "MASTER CREATE" on the DOS 3.3 SYSTEM MASTER diskette supplied with your computer.

NOTE: Do step 3 if your system will be re-configured or if you always want a standard DOS 3.3 system on your diskettes.

INTERRUPTS AND THE SSD

The design of the SSD allows an interrupt drived system to be created. Interrupts can be used by providing an interrupt source. The SSD Disk I/O can be run while interrupts are enabled allowing real time interrupt processing with disk I/O.

The SSD can be used to store data directly as extended memory with or without interrupts as in a printer buffer or communications buffer.

CAUTION

If the interrupt routine is designed to store data into the SSD you must ALWAYS save the "Y" register and restore it and write it to the low order address register on the SSD which was being accessed when the interrupt occurred before doing the "RTI" (Return From Interrupt) instruction.

The address where to store the "Y" register byte can be found at $9BAF if the program "START-UP" is running and at $BF5E if the program "NO-INIT" is running.

USING A SINGLE DISK DRIVE WITH CP/M AND THE SSD

A single disk drive may be used with the CP/M system and SSD cards.

The following procedure should be observed:

1. Boot up the standard CP/M System.

2. Copy the files "SSD.COM" and "SSDCCP.COM" onto your system diskette.

Anytime you initialize a new diskette you must also copy the CP/M system onto the diskette by invoking the "COPY" program and using the command "A:=A/S" (see your CP/M system manual for details).

PASCAL 1.1. SSD I/O VERSION 2.2D

The current version of Pascal 1.1. SSD I/O is changed to V.2.2D. The changes are:

1. The Utility MV CODE is replaced with U.CODE which allows greater flexibility manipulating files to and from the SSD. The instructions for the Utility are similar to MV.CODE and are printed on the screen when U.CODE is executed.

2. A "Clear to end of screen" character is used to provide greater compatibility when using external terminals with the SSD Pascal 1.1. System.

3. The date is updated on the boot diskette when the system is turned on.

4. A more extensive check is done to determine when a SSD card is plugged into the Apple. This should reduce the chance a card other than a SSD will be recognized as a SSD.

PASCAL 1.1 SSD I.O VERSION 2.2E

A problem has been corrected which did not allow the use of other drives when using the Pascal "system attach" procedures.

PASCAL 1.1 SSD I.O VERSION 3.2a

Corrected "language card" accessing to be compatable with the Apple IIe.

SSD Apple/UCSD support level #3.2a:

This system supports four SSD configurations

1) half card in slot 5
2) full card in slot 5
3) full card in slot 5 & half card in slot 4
4) full card in slot 5 & full card in slot 4

The SSD cards become a single volume mounted on I/O unit #4: - the boot volume. For those who find the SSD operation unacceptable, a turnkey tool kit is available from Synetix. It includes all SSD initializing and console communicating source, as well as, provisions for overriding much of the software configuration.

SSD as unit #4: or #11.
after the I/O unit number of the transient loader (128)
revise all displayed messages
control the file content of the SSD
chain to and from other startup systems
relocate SYSTEM PASCAL and SYSTEM LIBRARY

Different hardware configuration support is also available upon special request.

For those wishing to do some modification without purchasing the turnkey kit the following information is provided. Note: SYSTEM STARTUP only initializes the SSD drivers - not the SSD volume. That is the responsibility of SSD. FOLLOW.

SYSTEM.STARTUP will chain to SSD. FOLLOW with a chain value.
The first character will be:

'0' = No SSD card in slot 5.
'1' = Wrong level of Pascal (not 1.1 or creater)
'2' = SSD.DRIVERS not found
'3' = ATTACH.BIOS failed
'4' = I/O error on slot 6 drive 1
'5' = I/O error on slot 5
'6' = half card in slot 5
'7' = full card in slot 5
'8' = full card in slot 5. half card in slot 4
'9' = full cards in both slot 4 and 5

The second character reflects the volume status:

'x' = no device swap has occured. Slot 6, drive 1 is still the boot drive.
'O' = disk was old (matched supplied volume serial).
NOTE: vol ser has been changed to match diskette.
'N' = disk is new. The diskette has been copied to SSD-200
NOTE: it is an exact copy - no improvements.

On error ['2' .. '5'] the IoResults value - as string - is concatenated to the chain value

(*SP*)

Some of the software configuration can be overridden by this version of SYSTEM.STARTUP.

```
program Startup;
(*$5+*)
uses ChainStuff:
begin
   SetCval (', name of ssd.follow).,(vol ser of drivers).(name of SSD>');
   (* the default is '.*SSD.FOLLOW..#4,SSD') *)
   (* ssd.follow applies after slot 6 becomes units #11 & #12 *)
   (* vol set of drivers applies while slot 6 still is #4 & #5*)
   SetChain ('<new name of SSD driver initialization code>'),
end
```

Note: the commas are important.

An example of use: the SSD.FOLLOW has been removed from the SSD to free up critical disk space. System restart would cause SY TEM.STARTUP to look for it in the wrong place. The fix

```
SetCval (',#11:SSD.FOLLOW...'):
```

Now SYSTEM STARTUP will look on the diskette for the file.

Installation Guide Addendum for the SSD

NOTE: Refer to the SofTech USCD p-System IV.0 manuals for SSD installation.

This is an addendum to Chapter 3 of the "Installation Guide for the Apple II". Replace the DISKVECT.CODE routine with SSDVECT.CODE in order to generate the correct external references - source has been provided for additional customization. Please read the source for CONFIG (see Section V.3 of the Installation Guide) disk parameters. These parameters must be set in order to run correctly.

SSD200.CODE is the Synetix supplied driver. SSD200.CODE uses 20H thru 35H for page zero scratch. No external references.

Another (optional) program SSD.START.CODE has been provided to copy the boot volume to the SSD-200 and then make the SSD-200 the boot volume. The copy process is somewhat elaborate in that it tries to optimize the SSD-200 space utilization by not copying those files that are only used for the boot process. This program may be used as SYSTEM.STARTUP. The only liability with this is that some software (most notably ASE) appends non-system data to system files. In this case, the user will have to provide some sort of system crutch, probably using STARTUP to involk the filer to provide the correct file.

Multiple startup programs may be connected by STARTUP.CODE which merely involks an EXEC to do all the work. Source is provided.

The file SSD.SYSTM.SBIOS is a version of SYSTEM.SBIOS using the 40 column display, one SSD-200 card in slot 5, printer, and remote drivers. You may use it to verify that the system performs correctly. Be sure to use the CONFIG utility to change the number of tracks for #9 to 36 for a half card and 72 for a full card.

A revised version of SSD.START.CØDE (courtesy of SofTech Micro Systems) uses a file PRØFILE.TEXT to perform the SSD configuration. The documentation is included on the diskette.

SofTech UCSD p-System IV.0 & IV.1 21/Feb/83

Installation Guide Addendum for the SSD

NOTE: Refer to the SofTech UCSD p-System IV.0 manuals for SSD installation.

This is an addendum to Chapter 3 of the "Installation Guide for the Apple][". Replace the DISKVECT.CODE routine with SSDVECT.CODE in order to generate the correct external references - source has been provided for additional customization. Please read the source for CONFIG (see Section V.3 of the Installation Guide) disk parameters. These parameters must be set in order to run correctly.

SSD200.CODE is the Synetix supplied driver. SSD200.CODE uses 20H thru 35H for page zero scratch. No external references.

Another (optional) program SSD.START.CODE has been provided to copy the boot volume to the SSD and then make the SSD the boot volume. The copy process is very customizable since it is driven by *PROFILE.TEXT. This program may be used as SYSTEM.STARTUP.

These BIOS extensions have been used in conjuction with the SVA8INCH support

---

This program is based on material provided by
Softech Micro Systems

This startup program reads the file *PROFILE.TEXT and interprets the commands lines in the file.

The file PROFILE.TEXT is looked for on the current root disk, it PROFILE.TEXT is found it's contents are interpreted as follows.

Syntex	Opertion
[*]VOLNAME	change root volume to VOLNAME
[.]VOLNAME	change prefix name to VOLNAME
[A]FILENAME	use FILENAME for SYSTEM.ASSMBLER
[C]FILENAME	use FILENAME for SYSTEM.COMPILER
[D]CONSOLE	read system date from console
[D]	not supported - used to read clock
[D]VOLNAME	write system date to VOLNAME
[E]FILENAME	use FILENAME for SYSTEM.EDITOR
[F]FILENAME	use FILENAME for SYSTEM.FILER
[G]FILENAME	goto FILENAME for more orders
[L]FILENAME	use FILENAME for SYSTEM.LINKER
[M]VOLNAME	not supported - used by corvus file server
[P]VOLNAME	get SYSTEM.PASCAL from VOLNAME
[Q]FILENAME	set ExecAll switch true if file not found
[Q]	negate ExecAll switch

Syntex	Operation
[R]	return to caller see [G]
[T]FILENAME1.FILENAME2	transfer FILENAME1 to FILENAME2
[V]VOLNAME	verifies that VOLNAME is online
[X]FILENAME	load FILENAME into the execution queue
[Z]VOLNAME,size,unit	zero I/O unit with VOLNAME and size

NOTES:

(1) don't add any spaces between anything,
(2) FILENAME may include standard volume specification, but no wildcards
(3) commands may be in any order
(4) moving a system file to ramdisk doesn't automatically make the operating system use the copy on ram disk, you must set that explicitly now.
(5) lower case commands will not execute if ExexAll switch is false. Set by [Q] command. Can be reset

(*$P*)

Sample PROFILE.TEXT contents (with comments in lower case):

```
[Q]SSD-200
[z]SSD-200,576,9
[t]#4:SYSTEM.PASCAL,SSD200:SYSTEM.PASCAL
[P]SSD-200
[t]#4:SYSTEM.LIBRARY.SSD-200.LIBRARY
[*]SSD-200
[t]#4:SYSTEM.FILER,SSD-200:SYSTEM.FILER
[v]P4AUX
[t]P4AUX:SYSTEM.EDITOR,SSD-200:SYSTEM.EDITOR
[v]P4BOOT
[:]SSD-200
[d]CONSOLE
[d]P4BOOT
[d]#9
[Q]PASCAL
[Q]
[t]PASCAL:SYSTEM COMPILER,SSD-200:SYSTEM COMPILER
[t]PASCAL:SYSTEM.SYNTAX,SSD-200 SYSTEM SYNTAX
[Q]ASM:
[Q]
[t]ASM:SYSTEM ASSMBLER,SSD-200 SYSTEM.ASSMBLER
[t]ASM:6500.OPCODES,SSD-200:6500.OPCODES
[t]ASM:6500.ERRORS.SSD-200:6500.ERRORS

-------------------------------------------------
[D]                                    read system date from clock
[D]RAMDISK                             store system date on ramdisk
[D]#4                                  store system date on bootdisk
[T]*SYSTEM.PASCAL.RAMDISK:SYSTEM PASCAL  move opsys to ramdisk
[P]RAMDISK:                            set opsys to load from ramdisk
```

```
[M]IV1:FILER SVOL                                mount corvus filer volume
[M]IV1:EDITOR SVOL                               mount corvus editor volume
[M]IV1:LINKER.SVOL                               mount corvus linker volume
[M]IV1:PASCALB.SVOL                              mount corvus compiler volume
[M]IV1:ASM8086.SVOL                              mount corvus assembler volume
[T]FILER:SYSTEM FILER.RAMDISK:SYSTEM FILER  move filer to ramdisk
[F]RAMDISK:SYSTEM FILER                          set filer to load from ramdisk
[E]EDITOR:SYSTEM.EDITOR                          set editor to load from corvus
[L]LINKER:LINK.CODE                              set linker to load from corvus
[C]PASCALB:SYSTEM.COMPILER                       set compiler to load from corvus
[A]ASM8086:ASM86.IV1.CODE                        set assmbler to load from corvus
[*]RAMDISK:                                      use RAMDISK: for next transfers
[·]ASM8086:                                      use ASM8086: for next transfers
[T]8086.OPCODES,*8086.OPCODES                    move assemblers opcodes
[T]8086.ERRORS,*8086.ERRORS                      move assemblers errors
[T]8087.FOPS.*8087.FOPS                          move assemblers fops
[X]#4:LOGON PI="GARY,"                           execute logon af-er this pgm
[:]RAMDISK:                                      make prefix RAMDISK:
[*]RAMDISK:                                      make root RAMDISK:
```

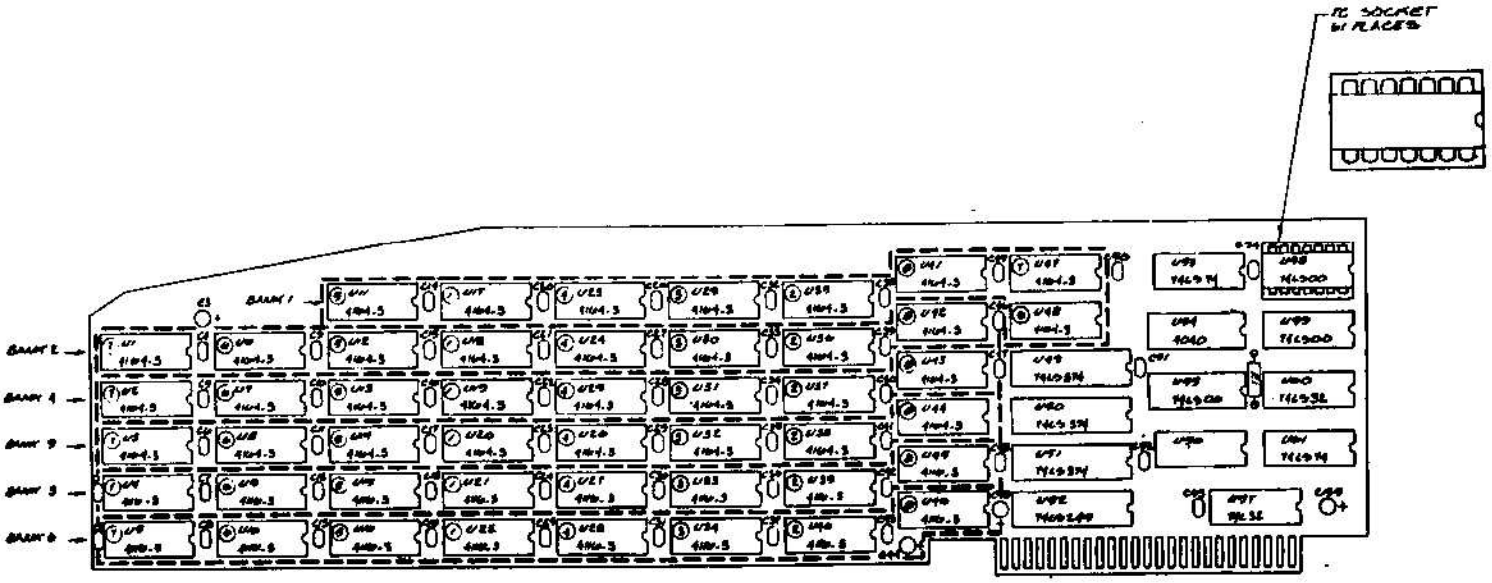
TABLE OF CONTENTS

I. INTRODUCTION

The Solid State Disk Emulator places the operation and memory capacity of disk drives within the Apple II* itself. It provides rapid solid state accessibility to data files and can be used to simulate greatly increased memory capacity. This increased speed and efficiency eliminates time spent waiting for the mechanical disk drive.

The SSD can be used with Apple DOS* 3.3, Apple PASCAL 1.1, and CP/M* (44K or 56K versions of CP/M 2.2, 2.20, or 2.2b) operating systems. With Apple DOS*, as many cards as the power supply will support can be used. The typical Apple* power supply supports three; the number also is dependent upon what other cards are in use. On the Apple PASCAL system (SSD I/O interface v.2.2b), and the CP/M* system (SSD I/O interface v.1.0), two cards are supported. Subsequent releases will support more cards.

The SSD card is shown below. Model 2202 has all banks of RAM installed. Model 2201 has banks 1,2 & 3 only.



II. INSTALLATION OF THE SSD CARD

TURN THE APPLE II's POWER SWITCH OFF. INSTALLING ANY PERIPHERAL CARD IN YOUR APPLE* WITH THE POWER ON CAN CAUSE SERIOUS DAMAGE BOTH TO THE CARD AND TO YOUR APPLE*.

For Apple DOS* 3.3, the SSD card may be installed in any slot from 1 to 7. The total number of cards the Apple II* can support is dependent only on its power supply.

The SSD Apple PASCAL 1.1 (V.2.2b) will support up to 2 Dual SSD cards, (see INTRODUCTION for card descriptions).

The SSD cards may be installed in the following configurations:

Configuration	Card(s)	Location
1	(1) Single (model 2201)	Slot 5
2	(1) Dual (model 2202)	Slot 5
3	(1) Dual & (1) Single	Dual-Slot 5,Single-Slot 4
4	(2) Dual	Slot 5,Slot 4

Two Singles (model 2201) configuration is not supported.

Apple* CP/M* requires at least one floppy disk drive in slot 6, and will support up to 6 drives in slots 6, 5, and 4. Each drive in each slot has a pre-assigned name:

Slot	Drive	CP/M Drive
6	1	A:
6	2	B:
5	1	C:
5	2	D:
4	1	E:
4	2	F:

Either single or dual SSD cards may be installed in either slot 4, slot 5, or both. SSD cards in slots other than 4 and 5 will not be recognized. There are no other slot dependencies; all possible combinations of single and dual SSD cards in slots 4 and/or 5 are supported. Each half of an SSD card will be named according to the table above (e.g., a dual card in slot 4 will be drives E: and F:).

III. GENERAL OPERATION

Once the SSD is in operation, the interface software treats it as a mechanical disk drive. This makes it accessible through the standard commands of the respective operating systems. Thus the user has the use of the mechanical disk drives and the additional disk drive capacity on the card. The SSD will operate with most non-protected software which allows booting to take place independent of the application software (i.e., software which runs after a normal APPLESOFT/INTEGER BASIC disk is booted).

For Apple II* users, the SSD answers these special needs:

The programmer's utility and development tool programs can be put into the SSD, leaving the mechanical disk drive free for the program in process. Also, many program copies can be produced quickly.

For the user handling database programs with very large data files, the SSD reduces data manipulation time to a minimum. Sorts and merges are accomplished quickly and easily, and larger volumes of data can be processed than is possible with the standard floppy disks.

It is assumed that the user is familiar with the operation of the Apple II* computer and floppy disk drives. These instructions show how to boot the SSD software for the three operating systems; Apple DOS* 3.3, Apple PASCAL, and CP/M*, and to make the operating systems think that the SSD card is a floppy disk. After booting, the respective operating system rules apply.

IV. APPLE DOS* 3.3

A. PROGRAM OPERATION

NOTE: MAKE A COPY OF THE SSD DISTRIBUTION DISK. Keep the original in a safe place as a backup. The following programs must be put onto the copy of the SSD distribution disk: HELLO, START-UP, APPLESOFT, INTBASIC, FPBASIC, and BOOT. Other programs can be put on the copy disk also. All programs will be automatically copied onto the SSD during the booting process.

1. BOOTING UP

a) Install the SSD card with the APPLE II* power off.

b) Insert the SSD BOOT diskette in the BOOT disk drive and power the APPLE* on.

c) The HELLO program on the BOOT disk will load the routines required for the SSD.

d) The program will initialize the first SSD card found, then stop and wait for the user to install disk(s) in drives 1 and 2 to be copied to the SSD. It also recognizes at this point whether there is a single or dual card in the slot. If it finds a dual card, the following display will appear:

```
                INPUT   OUTPUT

SLOT            X ==>   Y
DRIVE           1 ==>   1


SLOT            X ==>   Y
DRIVE           2 ==>   2
```

X - Slot booted or last active
Y - Slot SSD card found

If a single SSD card is found, one set only of slot and drive numbers will appear.

Cursor is on the upper left position when display first appears. If desired, change slot and drive numbers or press arrow keys to move cursor from one field to the next.

If user wishes to leave a card initialized but not copy data onto it yet, he can skip over the copy step in the process by pressing the left arrow key when cursor is in the upper left location.

This causes the program to move past the copy step and return to step d), leaving space on the card free for copying at a later time.

With a dual card, if only one diskette of data is to be copied, "Ø" can be entered for the 2nd slot and drive.

e) Press RETURN. This will accept all displayed entry and begin the copying.

When copy is completed, the program will look for another card. If it finds one, it will initialize that one also, then will return to Step d) above and will repeat the process until no more SSD cards are found.

If it does not find another card, the program still allows further copying. User can place different floppy disks in disk drives to be copied onto the SSD also. Or, user can copy over in case of an error. Do this by entering appropriate number when prompt says:

"All the SSD cards have data on them.
Enter number of a slot you would like to copy to, or (RETURN)"

If user enters a number, the program will return to Step d) above and allow entry of slot and drive numbers for copy.

Press RETURN when finished. Prompt will then give the option of reformatting to 32 sector. If this is not desired, press "N" or RETURN to continue in 16 sector format.

2. USING THE SSD AS A HIGH SPEED DISK

Once data is copied into the SSD, the it can be used as a high speed disk. The data copied onto the card from each disk remains a distinct unit, just as from a disk drive. User "talks" to card by specifying slot and drive numbers. (See DOS 3.3 manual for more details.)

The SSD is treated as two separate disk drives allowing complete independent disk operation on the card (e.g., if there is a Single Disk SSD (Model 22Ø1-147K) it is "D1", if there is a Dual Disk (Model 22Ø2-294K) installed it is both "D1" and "D2" - see Apple II* DOS manual for more details).

To copy files back onto floppy disks, use"FID", or the APPLE* COPYA program.

B. 32 SECTOR OPTION

At times it is necessary to have all the data on the card as one disk instead of two, as in the handling of very large data files. For cases such as this, the 32 sector option incorporates the data from two disks into one large disk. A program called "SECTOR 16/32" is available to convert the SSD Dual Disk card into a single disk.

Single SSD cards will not be converted to 32 sector format. They will remain in standard 16 sector format.

To use this option, copy the disks onto the card as usual; when program prompts: "Would you like to re-format data to 32 sector? Y/N?", enter Y. This causes the files on the SSD card to be reformatted onto one large disk of 294K bytes. Using this option, it is assumed that the drive # is 1. There is no option to drive 2.

Once this is done, data files are accessible file by file only, not by disk drive unit. The files will be listed in one catalog; where file names repeat, (because of files of the same name from the two disks), the repeated name will be highlighted.

To copy files back onto the standard floppy disk, choose which files are to be copied from the catalog of files on screen. Copy those files onto the diskette in a mechanical disk drive. (Program "FID" may be used to copy and manipulate files.) Remember that files larger than 143K bytes will cause a disk full error on the floppy disk.

NOTE: For DOS users wanting to use the card directly, see the ADVANCED PROGRAMMING AND TECHNICAL DISCUSSION section at the end of this document.

V. APPLE* PASCAL 1.1

A. PROGRAM OPERATION

NOTE: MAKE A COPY OF THE SSD DISTRIBUTION DISK. Keep the original in a safe place as a backup. See FILE CONFIGURATION section for list of files that must be put on your copy of the SSD distribution disk.

1. BOOTING UP

a) Install the SSD with the Apple II* power off.

b) Insert the BOOT disk in the BOOT disk drive and power the Apple* on.

c) The boot process loads the routines required for the SSD. This initializes the cards and the data from the boot disk drive is copied onto it. The disk drive in slot 6, drive 1 becomes Volume #11 (see Theory of Operation section below).

The program then goes into the Apple PASCAL operating system.

RESET: Use of RESET does not cause any loss of operating or data files on the SSD cards. (The boot diskette, SSD1, must be in the boot disk drive, (slot 6, drive 1) during RESET.

2. COPYING DISKS AND FILES

Files - File copy may be done using the standard FILER commands or if between disk and SSD, the utility program MV.CODE may be used.

Disk to SSD - Do not use the disk copy commands between the SSD and a standard disk drive, as the number of blocks will be different. Use file copy only.

Disk to Disk - Disk to disk copy may be done normally between any two standard disk drives.

Single Disk Drive Users may use the SSD to store a complete diskette in a file for disk to disk copy (i.e., FILER command: t(ransfer #11:, #4:<filename> will copy a diskette onto the SSD card in a file. Use FILER command: t(ransfer #4:<filename>,#11: to copy diskette of data in "file name" from SSD card to formatted diskette in disk drive).

3. USING MV.CODE PROGRAM FOR FILE TRANSFER

Once the Apple PASCAL system is up, the utility program MV.CODE can be used for easy movement of files to and from the SSD.

Select X(ecute from the Apple PASCAL command mode menu; then enter MV. The screen then prompts these choices:

T(o) - causes the files selected to be copied from the disk drive (volume #11) to the SSD.
F(rom) - causes the files selected to be copied from the SSD to the disk drive (volume #11).

The following menu is then displayed to facilitate file selection:

Ctrl I(nvert) - deselects all files selected and selects all files not selected.
Ctrl C(lear) - deselects all files selected.
(cr) - moves cursor to next file name, if a file has been selected each (cr) will select the next file.
ESC - terminates the program, causes copying of any files selected from the source device to the destination device.

The file names on the source device are then loaded and displayed with a two-character sequence preceding each file name. To select a file, enter the two-character file identifier. A ">" will appear to the left of the file name indicating selection. After all selections are made, the program is exited by the ESC command which moves (copies) all selected files.

B. THEORY OF OPERATION

During the boot process (SYSTEM.STARTUP) the boot disk is copied into the SSD. The SSD is then assigned to Apple PASCAL unit #4-BOOT volume (normally assigned to slot 6, drive 1). Slot 6 is assigned the Apple PASCAL units #11 and #12 which are normally assigned to slot 5 (the disk drive which started the boot process becomes Apple PASCAL unit #11). Making the SSD emulate the BOOT volume and copying SYSTEM files into the SSD provides maximum system performance. The SSD I/O drivers use approximately 200 bytes of main memory storage. The SSD I/O drivers conform to the published ATTACH.BIOS specification and should work with all configurations and subsequent releases of Apple* UCSD PASCAL which also conform to the specification.

C. FILE CONFIGURATION

SYSTEM and other frequently used files should be placed on the boot disk which is copied automatically into the SSD for fast access. Other files which are to be used frequently should be copied into the SSD after the boot process is complete. (The program MV.CODE may be used to expedite file transfer from the BOOT disk drive to the SSD).

The following files are used to boot the SSD:

1 - SYSTEM.PASCAL Pascal 1.1 file.
2 - SYSTEM.LIBRARY Pascal library file.
3 - SYSTEM.STARTUP The SSD driver initializer
4 - SSD.FØLLØW.CØDE The SSD formatter
5 - SYSTEM.APPLE standard APPLE* Pascal file.
6 - SYSTEM.MISCINFO standard information file.
7 - SYSTEM.ATTATCH per APPLE*.
8 - ATTACH.DATA (see APPLE* ATTACH.BIOS document - Jan. 12, 1981). The procedure SSD is assigned to device 128 - no initialization is required.
9 - ATTACH.DRIVERS contain SSD procedure.
10 - SSD.4.DRIVERS The SSD driver routines

Files #3-#8 should be placed at the end of the BOOT disk since after the copy into the SSD is completed they are deleted.

NOTE: The system looks around for the five wandering files (SYSTEM.FILER, SYSTEM.COMPILER, SYSTEM.EDITOR, SYSTEM.ASSEMBLER, and SYSTEM.LINKER) and will find them on a disk drive before the SSD. To be sure they are not found on the BOOT disk they may be named SYSTUM instead of SYSTEM, the boot process will rename them if they are on the BOOT disk.

Also available is a turnkey kit which provides for alternate drivers and more control over configuration (system source is provided).

Call Synetix for more information.

D. UCSD p-System IV.Ø and IV.1

The SSD interface to the new version of PASCAL (available from Softech) will be released soon from Synetix.

The SSD will follow the Apple* installation guide standards which allow the SSD to go in any slot and be defined as any I/O unit including #4 (boot).

Call Synetix for more information.

VI. CP/M*

A. PROGRAM OPERATION

NOTE: MAKE A COPY OF THE SSD DISTRIBUTION DISK. Keep the original in a safe place as a backup. Use the standard CP/M* COPY.COM program to put CP/M* on your new SSD disk (the /S switch copies CP/M* itself to a new disk.)

1. BOOTING UP

a) Boot CP/M* from a system disk. The SSD distribution disk does not contain the CP/M* operating system, and will not boot.

b) Insert your copy of the distribution disk in a disk drive. The installation program will run from any drive.

WARNING: DO NOT USE STANDARD CP/M* FORMAT OR COPY UTILITY PROGRAMS WITH SSD. See SOFTWARE INTERACTIONS section below.

c) Run either SSD.COM or SSDCCP.COM (the difference is explained in the System Description section below). The driver installation program should identify itself, and let you know which disk drives (C: through F:) it found in which slots in your Apple*. If it fails to find all of your SSD drives, TURN YOUR APPLE* OFF, and try reseating the SSD cards in the slots. The SSD is a large, heavy card, and it's possible that vibration or a jolt could loosen it.

d) Check for correct operation by using CP/M*'s DIR command to list the contents of the SSD disk (it should be empty).

e) Each time you cold boot your Apple* (e.g., power up, or run BOOT.COM), you must re-install the SSD driver by running either SSD.COM or SSDCCP.COM. It is not necessary to re-install the driver after each warm boot (Control C.).

B. USING SSD.COM vs. SSDCCP.COM

A wide variety of different peripheral cards exist for your Apple*; many of them require special driver software to interface to the operating system. This diversity greatly increases the usefulness and power of your Apple* computer, but it brings problems, too. Sometimes the driver routines for two different cards may be incompatible. To help overcome this, two versions of the CP/M* software are provided with your SSD card. One or the other should be usable in almost any Apple*.

SSD.COM is the preferred version, because it does not reduce the memory space available to user programs. It may, however, interfere with other I/O drivers. SSDCCP.COM is compatible with nearly any other peripheral, but costs 2 1/4K of available data space.

SSD.COM: The Apple* CP/M* BIOS reserves three 128 byte long areas, starting at hex F200, for special purpose device drivers. SSD.COM stores the driver software in the first two of these areas (hex F200 to F2FF). These areas are normally reserved for the line printer (CP/M LST: device) and the punch/reader (PUN: and RDR:).

SSDCCP.COM: CP/M* reserves the first page (256 bytes) of memory for vectors and other data. The top of memory contains 6502 zero page, stack, screen, and I/O locations. Just below that is the Basic I/O System (the BIOS); below that is the Basic Disk Operating System (the BDOS); below that is the Console Command Processor (the CCP). Between hex 100 and the bottom of the CCP is the Transient Program Area (TPA). User programs (anything other than a built-in command such as all .COM programs) are loaded into the TPA; any space in the TPA not used for code may be used for data. Most of CP/M* is needed during the execution of transient programs. The CCP, however, is CP/M*'s interface to the user. Since the user's input is being processed by the transient program, the CCP isn't necessary. Many programs overlay the CCP to gain more data space; CP/M* provides a vector within the first 256 bytes pointing to the lowest address required by the rest of CP/M*. SSDCCP.COM stores the SSD driver code below the CCP, and changes this vector to point to the lowest address used by the driver. The advantage to this technique is compatibility with any standard Apple* CP/M* system; the disadvantage is that user programs can no longer use the CCP as data space. Including the SSD driver, this means that 2304 bytes (2 1/4K) are unavailable to transient programs.

One final note: every effort has been made to insure compatibility with other Apple II* peripherals. However, with the variety of cards that are available, there may be some that are incompatible with both SSD.COM and SSDCCP.COM.

C. SOFTWARE INTERACTIONS

The CP/M* SSD driver modifies certain locations in the CP/M* BIOS so that it can intercept disk I/O function calls at a low level. Since the software exactly simulates a normal Apple* disk drive, most user programs will work perfectly with the SSD. There are a very small number of programs, however, that use their own disk routines, (i.e., they do not call CP/M* to do disk I/O). THESE PROGRAMS WILL NOT WORK WITH THE SSD. Fortunately, there are very few such programs. Two standard CP/M* utilities fall into this category: FORMAT and COPY. It is never necessary to FORMAT an SSD disk; the driver installation program (SSD.COM or SSDCCP.COM) already does this. COPY performs two functions: 1) copying files from one disk to another, and 2) copying CP/M itself. The first function can also be done with PIP, although PIP is somewhat slower than COPY. The second function is unnecessary, since CP/M* always boots from drive A: (the first drive in slot 6). THESE PROGRAMS DO NOT WORK WITH A SOLID STATE DISK AS SOURCE

OR DESTINATION, AND MAY PRODUCE UNPREDICTABLE AND UNPLEASANT RESULTS, INCLUDING POSSIBLY WRITING RANDOM DATA TO YOUR FLOPPY DISK DRIVES. Both programs will work fine as long as neither the source nor the destination is an SSD. These are the only two programs known to the author to show this problem, however, there may be others. If in doubt, it wouldn't hurt to try a new .COM file on a scratch floppy disk before using it normally.

VII. ADVANCED PROGRAMMING AND TECHNICAL DISCUSSION

For DOS users wanting to use the card directly in a high speed data transfer application for moving blocks of data to and from the card, we recommend using the Read Write Track Sector routine. To use the RWTS, use procedures such as those documented in the April '82 issue of BYTE (Vol. 7, #4); articles beginning on pp. 455 and 458.

The SSD card may be used as a 147K/294K memory card for applications which require large amounts of on line memory. The SSD must be accessed through the RWTS or interface program.

SSD ACCESSING

The first two addresses on the card are WRITE only 8 bit resisters which together make up a 16 bit pointer.

The 16K BLOCKS are addressed slightly differently as only the least significant 7 bits of the two 8 bit resisters are active when using the 16K BLOCKS.

To access SSD data:

1) Write the 16 bit pointer to $C0X0 and $C0X1 corresponding to the address of the byte in question (i.e., WRITE $00 to $C0X0 and $00 to $C0X1 which points to the first byte.).

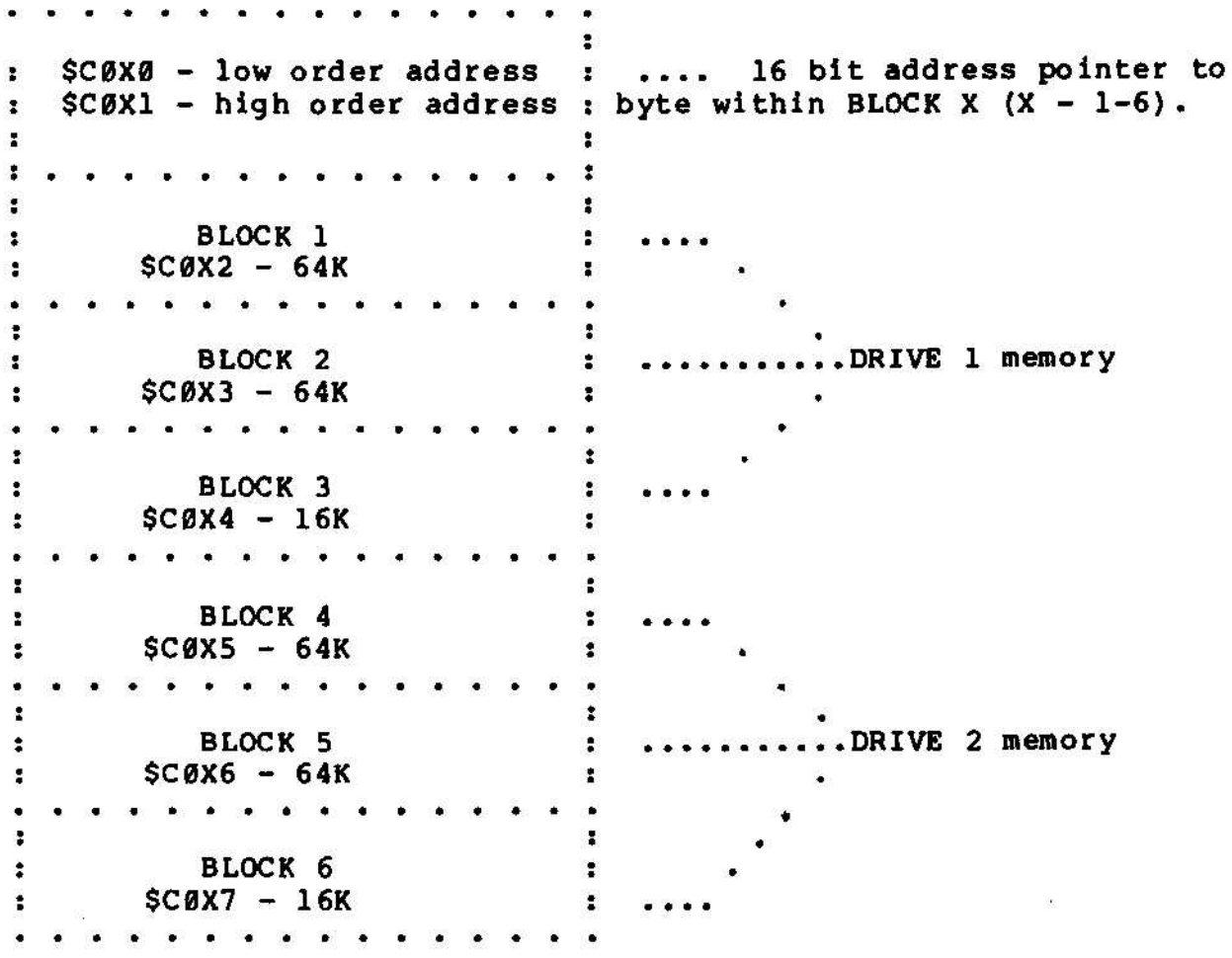
2) READ the byte from the BLOCK by READING the corresponding I/O address (i.e., if byte is in BLOCK 1, READ $C0X2).

3) WRITE the byte into the location by WRITING to the corresponding I/O address (see #2 above).

4) To access sequential bytes it is necessary to increment the 16 bit pointer by incrementing a CPU resister (or memory) and RE-WRITING the least significant 8 bits to $C0X0. The high order 8 bits need only be RE-WRITTEN if the low order byte increments back to $00 (see sample programs).

$C0X0-$C0X7 correspond to the I/O addresses for the card slot (see Apple* manuals for more details).

The SSD is organized into 6 blocks of memory.



$C0X0-$C0X7 correspond to the I/O addresses for the card slot (see APPLE* manuals for more details).

A. SAMPLE ACCESS PROGRAM FOR 64K BLOCKS

```
STPNTR     = $FD     ; Address of high order starting address on
                       card
NDXFER     = $FC     ; Address of high order end of data address
BUFF       = $FE     ; Indirect buffer address for data
SSDPTR     = $C0D0   ; card is in slot 5 so addresses are
                        $C0D0-$C0D7
SS1PTR     = $C0D1   ; high order 8 bits of pointer
BLOCK 1    = $C0D2   ; address of bytes
in BLOCK 1
;
START      LDY       #0        ; SET Y=0
           STY       BUFF      ; SET UP LOW ORDER BYTE OF INDIRECT
                                 ADDRESS
           LDX       STPNTR    ; LOAD STARTING HIGH ORDER BYTE
           STX       SSIPTR    ; POINT TO $XX00 ON SSD CARD TO START
;
;  THE NEXT CODE GETS 256 BYTES
;  Y IS LOW ORDER ADDRESS POINTER
;
NXTBYT     STY       SSDPTR    ; STORE LOW ORDER 8 BITS OF POINTER
           LDA       BLOCK1    ; GET BYTE FROM CARD
           STA     (BUFF),Y    ; STORE BYTE IN BUFFER
           INY                 ; INCREMENT AND TEST FOR DONE
           BNE       NXTBYT    ; IF NOT 0 GO GET NEXT BYTE
           INC       BUFF+1    ; INCREMENT BUFFER POINTER
           LDA       BUFF+1    ; LOAD HIGH ORDER BYTE TO STORE ON
                                 CARD
           CMP       NDXFER    ; IS BYTE TRANSFER DONE?
           BEQ       DONE      ; YES, GOODBYE!
           INX

           STX       SS1PTR    ; NO, STORE HIGH ORDER 8 BITS OF
                                 POINTER-
           JMP       NXTBYT    ; LOW ORDER BYTE AND GET 256 BYTES
DONE------BACK TO CALLING ROUTINE------
```

B. ACCESSING 16K BLOCKS

The 16K BLOCKS require a 14 bit pointer consisting of the least significant 7 bits of the two 8 bit address pointers.

The access may be done by a routine which gets 128 bytes from the card, then increments the high order byte and gets the last 128 bytes to be compatible with a routine which gets 256 bytes from the 64K blocks.

SAMPLE PROGRAM FOR 256 BYTES FROM 16K BLOCKS

```
; USE ALL VARIABLES AS IN 64K PROGRAM ABOVE
;
HIBYTE    = $FB            ; SINCE HIGH ORDER BYTE ON CARD
                             MUST BE INC2 X BUFF+1
BLOCK3    = $C0D4          ; ADDRESS OF BYTES IN 16K BLOCK 3
;
START     LDX    #0        ; X IS POINTER FOR BYTE ON CARD - 0 to
                             127
          LDY    #0        ; Y IS POINTER FOR BYTE IN BUFFER 0 to
                             255
          STY    BUFF      ; SET UP INDIRECT POINTER
          LDA    STPNTR    ; LOAD HIGH ORDER ADDRESS POINTER
          STA    SS1PTR    ; STORE HIGH ORDER POINTER ON CARD
          STA    TEMP      ; TEMP=TEMP+2 FOR EACH 256 BYTES
NXTBYT    STY    SSDPTR    ; STORE LOW ORDER ADDRESS POINTER ON
                             CARD
          LDA    BLOCK3    ; LOAD BYTE FROM 16K BLOCK
          STA   (BUFF),Y   ; STORE BYTE IN BUFFER SPACE OFFSET BY
                             Y
          INY              ; INCREMENT Y, 256 BYTE POINTER
          BEQ    FINIS     ; 256 BYTES DONE?
          INX              ; CHECK IF 128 DONE
          BPL    NXTBYT    ; GO TO NXTBYT IF NOT 128
HIBYTE    INC    TEMP      ; 128 DONE SO INCREMENT HIGH ORDER
                             ADDRESS-
          LDA    TEMP      ; AND STORE ON CARD
          STA    SS1PTR    ; DO IT
          LDX    #0        ; SET X=0
          JMP    NXTBYT    ; GO GET 128 MORE
FINIS     INC    BUFF+1    ; 256 BYTES DONE SO SEE IF TRANSFER
                             DONE
          LDA    BUFF+1    ; GET TO COMPARE WITH END TRANSFER BYTE
          CMP    NDXFER    ; DONE?
          BNE    HIBYTE    ; IF NOT DONE, TEMP=TEMP+1 AND GET 256
                             BYTES
DONE -----BACK TO CALLING ROUTINE--------
```

VIII. IN CASE OF DIFFICULTY

If the SSD fails to operate properly, turn the Apple* off and insure the SSD is properly installed. Then re-boot. If the problem still exists, there is a program on the DOS diskette, "SSD TEST", which verifies proper operation of the SSD card.

- Turn the Apple* off.
- Install the SSD card in slot 5.
- Insert the DOS 3.3 SSD I/O diskette in the boot disk drive.
- Turn the Apple* on.
- After the Apple* is running, run the program "SSD TEST". The program will verify the operation of the SSD card, (type "Y" to the prompt if you have a single, model 2201).
- If any flashing numbers appear on the screen besides the flashing "TESTING", your card may have a defective component. Return the card with the problem description to the place of purchase along with your warranty information.
- The program will start beeping and display "TEST 1 COMPLETE" if the SSD card passes the test.

* - Apple, Apple II, and Apple DOS are registered trademarks of APPLE COMPUTERS INC.

* - CP/M is a registered trademark of Digital Research.

LIMITED WARRANTY FOR SYNETIX INDUSTRIES SOFTWARE

Computer software programs cannot replace your sound business judgement or make decisions for you. You, therefore, assume complete responsiblity for any decisions made or actions taken based on information obtained using Synetix Industries Inc. software programs and instructional materials.

Synetix Industries Inc. software and the attached instructional material are sold "AS IS", without warranty as to their performance. The entire risk as to the quality and performance of the computer is assumed by you.

However, to the original purchaser only, Synetix Industries Inc. warrants the software diskette to be free from defects in materials and faulty workmanship under normal use and service for period of one (1) year from the date of purchase. If, during this one year period, a defect in the diskette should occur, it may be returned to Synetix Industries Inc. or an authorized Synetix Industries Inc. dealer for replacement of the diskette without charge to you. Your sole and exclusive remedy in the event of a defect is expressly limited to replacement of the diskette as provided above.

If the failure of the diskette, in the judgement of Synetix Industries Inc. resulted from accident, abuse or misapplication of the diskette, then Synetix Industries Inc. shall have no responsibility to replace the diskette under the terms of this warranty. In such an event, replacement of the diskette is available to the original purchaser at a nominal charge.

The above warranties for goods are in lieu of all warranties, express, implied or statutory, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose, and of any other warranty obligation on the part of Synetix Industries Inc. In no event shall Synetix Industries Inc. or anyone else who has been involved in the creation and production of this product be liable for indirect, special or consequential damages, such as, but not limited to, loss of anticipated profits or benefits resulting from the use of this product, or arising out of any breach of this warranty. Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation may not apply to you.