HELIX LABS ABM

8123-25 Remmet Ave.
Canoga Park, CA 91304
(818) 710-0300
Outside California: 1-800-468-0004
A Company of Helix
Systems & Development Corporation

HELIX LABS

APPLE BUBBLE MEMORY BOARD

USER'S MANUAL

8123-25 Remmet Ave.
Canoga Park, CA 91304
(818) 710-0300
Outside California: 1-800-468-0004
A Company of Helix
Systems & Development Corporation

Comments concerning the Helix ABM board, software or documentation should be sent to:

8123-25 Remmet Ave.
Canoga Park, CA 91304
(818) 710-0300
Outside California: 1-800-468-0004

CONTENTS

HELIX APPLE BUBBLE MEMORY USER'S MANUAL

CHAPTER 1 : INTRODUCTION

1.1 PREFACE

The Helix Apple Bubble Memory (ABM) board is a valuable enhancement to your Apple II or IIe computer, providing highly reliable, non-volatile (data retained without power), solid-state mass memory with almost instant data access. The software included with this board will patch Apple DOS 3.3 and Apple Pascal 1.1 operating systems for ABM board emulation of a floppy disk drive including cold booting ability. Additional diagnostic software allows the user to ensure proper board operation or trouble shoot the board should that ever become necessary. Finally, interface and jump table documentation is provided which should enable the more sophisticated user to write his own ABM board routines.

1.2 HELIX ABM PACKAGE CONTENTS

- one ABM board

- one ABM MASTER DISK with the following files:

```
SIDE 1     DOS 3.3
  PATCHDOS3.3
  PATCHDOS3.3.OBJ
  DIAGNOSTICS
  DIAGNOSTICS.OBJ

SIDE 2     APPLE PASCAL 1.1
  BUBPASUTIL.TEXT
  BUBPASUTIL.CODE
```

- one Helix ABM User Manual

- one Intel "Primer on Magnetic Bubble Memories"

- one Intel "7220-1 Bubble Memory Controller" data sheet

- one Helix ABM board registration card

1.3 APPLE SYSTEM REQUIREMENTS

1)An Apple II or IIe computer with 48K and Applesoft basic.

2)One disk drive with Apple DOS 3.3 or Apple Pascal 1.1 operating system.

3)Monitor

4)Blank floppy disks

1.4 HELIX ABM BOARD DESCRIPTION

The Helix ABM board contains an Intel 7110-4 one megabit bubble memory and its support chips, the hardware interface to the Apple computer peripheral slot bus, and firmware drivers to operate the bubble memory. It provides 128K bytes of non-volatile user data storage with built-in error correction and power-fail protection. The bubble memory software drivers are located in an on-board 2716 EPROM so that only a small patch to the operating system is required for floppy disk emulation. This "bubble disk" operation offers speed (2 to 3 times faster read/write operation), portability (fits entirely inside the Apple computer), and reliability (all solid-state with no moving parts to wear out) advantages over a floppy disk while providing the protection of non-volatility and power-up booting capability advantages over the 128K RAM disk. A small jumper near the top of the board provides write-protection similar to the write-protect notch on the floppy disk. Immediately below this jumper is another jumper to allow the speaker click to be defeated.

1.5 BUBBLE MEMORY DESCRIPTION

The magnetic bubble memory is a form of mass storage in which the data is stored as small cylindrical magnetic domains, or "bubbles", in a thin single-crystal film of magnetic material. In other magnetic storage devices (magnetic tape or floppy/hard disks) the information is stored as magnetic domains which are fixed in the magnetic media. Therefore to read or write the domains, this media must be mechanically moved past read/write heads. In contrast the domains of the bubble memory can move freely in its stationary magnetic media or film. Deposited and patterned on the top of the magnetic film are metal conductor lines for control and patterns of permalloy to direct the bubbles along given paths. Actual bubble motion requires only an in-plane rotating magnetic field which is produced by phased current to x-y coils that enclose the bubble memory chip. Since this motion only depends on changes in the magnetic state of atoms in the film, it is a true solid state phenomenon with no mechanical wear involved.

A more detailed description of bubble memories and the specifics of the Intel bubble memory and its support chips is given in the Intel "A Primer on Bubble Memories" included with the ABM package.

CHAPTER 2 : INSTALLATION

2.1 ABM MASTER DISK BACKUP

Before proceeding, it is recommended that each side of the ABM MASTER DISK be copied to a separate blank floppy disk using the COPYA program from the master disk supplied with Apple DOS 3.3. The original ABM MASTER DISK should then be stored in a safe place, and the copies used for the following operations when the ABM MASTER DISK is referenced.

2.2 INSERTING THE ABM BOARD

Precautions:

1) The Helix ABM board comes in an antistatic pouch because it has components that could be damaged by the high static voltage that builds up on insulating surfaces. Be careful to handle the board only by the edges and do not touch the gold-fingered edge connector on the bottom edge of the board. When the ABM board is not inserted in the computer, it is suggested that the board be kept in the antistatic pouch.

2) The power to the Apple computer must be TURNED OFF before inserting or removing the ABM board or the disk controller card. Otherwise damage to these cards and the computer, as well as a shock hazard to the user, could result.

The ABM board is slot independent and thus operates under Apple DOS 3.3 in any slot except slot #0. For the purposes of pedagogical clarity only, the instructions in the following sections will refer to definite slot numbers for the ABM board and dis' controller card.

1) If the ABM board is brand-new and being inserted for the first time, it is suggested that the diagnostic program be run as described in the following section. This program will require the bootloop bitmap stored internally in the ABM to be compared with the bitmap on the label of the bubble memory package. Therefore, before inserting the ABM board, turn the board upside down and note that the large white label on the bubble memory package has 5 rows of 16 hexadecimal digits. Copy these rows just as they appear onto a piece of paper for convenient reference during the diagnostic program.

2) Ensure that the power to the Apple computer is OFF.

3) Insert the ABM board into slot #4, ensuring that the board is firmly seated.

4) Insert the floppy disk controller card into slot #6.

5) Load the floppy disk drive with a diskette that has a master or slave version of Apple DOS 3.3 (note that the ABM MASTER DISK does not have DOS on its first three tracks).

6) Turn on the power to the Apple computer and allow DOS to be booted.

7) Again, if the ABM board is brand-new, it is suggested that the diagnostics program be run at this time as described in the following section.

2.3 THE DIAGNOSTICS PROGRAM

DIAGNOSTICS is a program written in Applesoft basic that will check the hardware of the ABM board. This program uses the binary file, DIAGNOSTICS.OBJ, which must be on the disk with DIAGNOSTICS when it is run. The following shows the user entries and program responses for a successful diagnostics session. An explanation of the tests being run is found in Appendix A, and a listing of the error codes generated by this

program are given in Appendix B.

1)With Apple DOS 3.3 booted, insert side 1 of the ABM MASTER DISK and enter the following (note (CR) means press the RETURN key; user entries are shown in boldface):

```
]RUN DIAGNOSTICS (CR)

****************************************
*       HELIX LABORATORIES INC.        *
*   APPLE BUBBLE MEMORY DIAGNOSTICS    *
*            VERSION 1.20              *
****************************************

(ESC KEY TO ABORT)
ENTER SLOT# OF BUBBLE BOARD ) 4

TEST BUBBLE IN SLOT# 4 (Y OR N) ) Y

))TESTING THE INTERFACE TO BMC :GOOD

))TESTING THE BMC FIFO :GOOD

))TESTING THE FORMAT SENSE AMP :GOOD

))READING BOOTLOOP MAP :

6900- XX XX XX XX XX XX XX XX
6908- XX XX XX XX XX XX XX XX
6910- XX XX XX XX XX XX XX XX
6918- XX XX XX XX XX XX XX XX
6920- XX XX XX XX XX XX XX XX

COMPARE MAP TO LABEL ON BUBBLE PACKAGE
ARE THEY THE SAME (Y OR N)? : Y
```

(note: the above X's represent the hexadecimal digits of the bootloop read from the bubble memory. They should be compared to the bootloop map copied from the bubble memory label in the previous section)


```
CHECK SEED BUBBLES (Y OR N)? ) Y

))TESTING FOR SEED BUBBLES :

6A00- FF FF FF FF FF FF FF FF
6A08- FF FF FF FF FF FF FF FF
6A10- FF FF FF FF FF FF FF FF
6A18- FF FF FF FF FF FF FF FF
6A20- FF FF FF FF FF FF FF FF
6A28- FF FF FF FF FF FF FF FF
6A30- FF FF FF FF FF FF FF FF
6A38- FF FF FF FF FF FF FF FF
6A40- FF FF FF FF

))PASSED SEED CHECK TEST :

WARNING:THE FOLLOWING TEST WILL
        ERASE ALL DATA ON THE ABM.

WRITE/READ PATTERN TEST (Y OR N) ?Y

PATTERN   OPERATION
------------------
   FF       W/R
   AA       W/R
   55       W/R
   00       W/R

BUBBLE PASSED WRITE/READ PATTERN TEST

FIND EPROM BYTE SUM? (Y OR N) : Y

SUM OF EPROM BYTES = $XXXXXX
```

(note: the above X's represent a check sum that should match the number on top of the EPROM on the ABM board.)

```
DIAGNOSTICS COMPLETED
```

CHAPTER 3 : USING THE ABM WITH APPLE DOS 3.3

3.1 PATCHING APPLE DOS 3.3

Apple DOS 3.3 requires only a small 36-byte patch to recognize the ABM board as a floppy disk. The source code for this patch may be found in Appendix C. The Applesoft program, PATCHDOS3.3, which uses the binary file, PATCHDOS3.3.OBJ, will load this patch into sectors 6 and 7 of track 0 on either a master or slave Apple DOS 3.3 disk. The patch code is relocatable and can be moved with DOS to any desired location. Note that patching DOS on a disk will not destroy any of the user files, so your disks can be easily updated with patched DOS. However, beware that this patch will normally not work on copy-protected disks and is likely to make them unusable.

1)With DOS 3.3 booted, insert side 1 of the ABM MASTER DISK into the disk drive and run the patch program:

]RUN PATCHDOS3.3

2)The program will request that the ABM MASTER DISK (or the disk used to run PATCHDOS3.3) be replaced by the disk to be patched.

3)After finishing, the program will ask whether another disk is to be patched.

3.2 WRITE-PROTECTION AND SPEAKER-CLICK SWITCHES

Along the top edge of the board next to the LED is a blue jumper switch (J1). This is the write-protect switch which functions the same way the write-protect notch of a floppy disk does. By moving it over the "WP" on the circuit board, a flag is set that notifies software not to write to the ABM. This switch may be operated while the board is plugged in and the power is on, as long as care is taken not to displace the board from the slot socket. It is recommended that the top

edge of the board near the switch be pressed down with the fingertips while the switch position is changed.

Just below the write-protection jumper, is another jumper (J2) which will enable or disable the speaker click during ABM operation.

3.3 INITIALIZING THE ABM

To operate under the patched DOS 3.3, the ABM must first be initialized. Just as with a floppy disk, this is done with the INIT command as follows:

1)Boot the Apple computer using a disk with the patched DOS 3.3 created in the previous section. That is, insert the diskette with the patched version of DOS into the disk drive and either turn the power off and back on, or enter:

]PR#6 (CR)

3)Ensure that the write-protect switch of the ABM board is in the unprotected position.

2)Load an appropriate "greeting" program into memory. For instance, load the program, HELLO, from the ABM MASTER DISK. Now enter:

]INIT HELLO, S4, D1, V0 (CR)

where S4 = ABM board in slot 4

D1 = load DOS into the first three tracks
(D2 = do not load DOS)

V0 or V254 = important to specify default volume

During this operation, the LED on the ABM board will flicker and the Apple computer speaker will buzz.

3)If the initialization was successful it should be

possible to catalog the bubble memory by entering:

```
]CATALOG  (CR)
```

and the following should appear:

```
DISK VOLUME 254

 A 002 HELLO

]
```

4)Finally, the user should now be able to boot DOS from the ABM by entering:

```
]PR#4 (CR)
```

which will boot the patched DOS 3.3 and then run the greeting program that was loaded before the INIT command.

3.4 ABM OPERATON UNDER DOS 3.3

Once initialized, the ABM should respond just like a floppy disk to all the Apple DOS 3.3 commands. Note that the drive number is ignored except in the case of the INIT command as explained above. That is, if D1 is specified with the INIT command, the ABM is initialized with DOS loaded in the first 3 tracks. If D2 is specified, DOS is not loaded, freeing these tracks for user file storage; of course, the ABM will not be able to boot DOS in this case.

3.5 COPYING/TRANSFERRING FILES

It is often necessary to copy or transfer files from floppy disk to ABM or back. These functions can be done with the FID and COPYA programs that come with the Apple DOS 3.3 master disk and work with the ABM as if it were a floppy disk. However, care must be exercised in using the COPYA program to copy a floppy disk to the ABM.

3.5.1 USING FID

All functions of the FID (for FIle Developer) program work with the bubble memory. Again, the drive number is ignored in referencing the ABM board.

Files may be transferred between floppy disk and ABM with the COPY command function (1). Using the wild card, "=", in place of the filename, multiple files may be easily transferred, and although this is not as fast as the COPYA program, it does not have the complications as discussed below.

3.5.2 USING COPYA

An Apple DOS 3.3 floppy disk has 35 tracks and is normally organized with DOS on the first three tracks (0-2) and user storage and directory tracks on the rest (3-34). The COPYA program copies all 35 tracks from one storage media to another starting at track 0 and ending with track 34. This presents a problem when copying a floppy disk to the ABM because the ABM only has 32 tracks. If Apple DOS 3.3 allocated sectors for a file sequentially, as done in Pascal, this would not be such a problem, since only if the disk is almost full would data be lost when the last three tracks are not copied. But each DOS 3.3 file has a Track/Sector list that allows sectors of the file to be stored at any track/sector location. Thus, not copying the last three user file storage tracks can lead to unpredictable and disasterous results.

To deal with this situation, the ABM firmware for DOS 3.3 organizes the ABM so that logical tracks 0-2 and 3-5 both reference the first three physical tracks (see Fig. 1). For instance, logical tracks 0 and 3 both access the first bubble memory storage track. Thus the COPYA program will copy DOS from floppy tracks 0-2 to the ABM, then overwrite it with floppy tracks 3-5, eliminating DOS and allowing the ABM to store the full 32 tracks of user files from the floppy. Of course, the ABM will not be able to boot DOS when loaded this way.

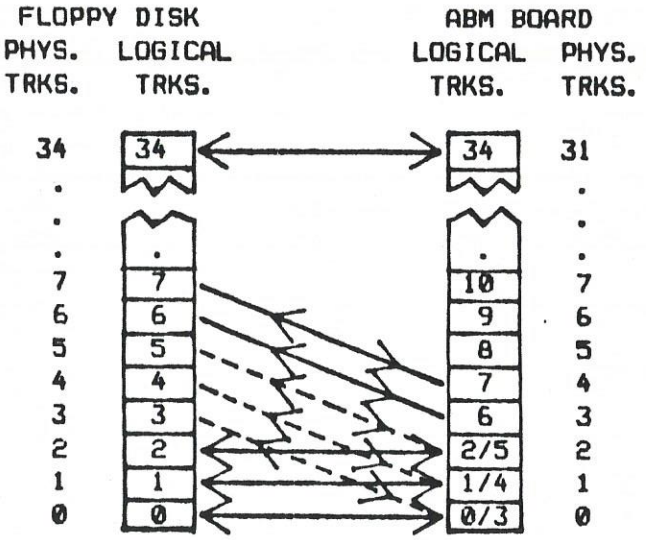

Fig. 1 : Relationship of Floppy/ABM tracks for DOS 3.3

On the other hand, using COPYA to copy the ABM to a floppy disk is straightforward. The first three physical ABM tracks will be copied to tracks 0-2 and tracks 3-5 of the floppy disk. ABM tracks 3-31 will be copied to floppy tracks 6-34. If the ABM had DOS on its first three tracks, then the floppy will also have DOS on its first three tracks giving it booting capability. In addition, the floppy's tracks 3-34 are now an exact copy, or backup, of the ABM's 32 tracks. Thus, when such an "backup" floppy disk is copied with COPYA back to the ABM and tracks 3-34 are mapped into the ABM's tracks 0-31 as explained above, the ABM data is fully restored.

3.5.3 COLD BOOTING FROM THE ABM BOARD

The ABM board or floppy disk card in the highest slot number becomes the device that the Apple Computer attempts to boot from after a power-up. Therefore, to cold boot from an ABM board which has DOS installed, it is only necessary to ensure that it is in a slot higher than any of the disk controller cards.

CHAPTER 4 : ABM OPERATION UNDER APPLE PASCAL 1.1

The ABM requires a small patch to the SYSTEM.APPLE file of Apple Pascal to emulate a floppy disk in this operating system. Additionally, a patch to the boot program stored in the first two blocks makes it possible to cold boot Pascal from the ABM if the ABM is loaded with the appropriate files. Thus, the ABM can be used as the root volume (unit 4 in slot #6) which is the most advantageous place to have a fast "disk" since this unit is constantly accessed. Note that in this operating system, the ABM board is only configured as drive 1, limiting it to units 4, 9, or 11 in slots 6, 4, and 5 respectively. Finally, the ABM board capacity is limited to 256 blocks instead of the 280 blocks on the floppy disk.

4.1 ABM PATCH TO SYSTEM.APPLE AND BOOT BLOCKS

1) Ensure the power to the Apple Computer is off.

2) Insert the disk drive into slot #6 and the ABM board into slot #4.

3) Insert the APPLE1: diskette into the floppy disk drive and turn on the power to the computer.

4) Once the system has booted, use the FILER command, TRANSFER, to transfer the file, BUBPASUTIL.CODE, from side 2 PASCAL of the ABM MASTER DISK to the APPLE1: disk. This is necessary because the BUBPASUTIL requires intrinsics on the APPLE1: disk.

6) Execute the program BUBPASUTIL. That is, enter:

Execute what file? BUBPASUTIL

8) Choose selection (1) and follow instructions to patch the file, SYSTEM.APPLE. Note that the date code of

this file will be changed to "30 MAY 83" which can be used to easily identify the SYSTEM.APPLE files that have been patched.

9) Choose selection (2) and follow instructions to patch the boot blocks.

10) Exit the program with selection (6).

11) From the command menu, select H(alt, or turn the power off then on, to reboot with the newly patched SYSTEM.APPLE.

4.2 INITIALIZING THE ABM FOR PASCAL OPERATION

Once SYSTEM.APPLE has been patched and the system has been rebooted, there are several ways to initialize the ABM for Pascal operation and two of these are listed below. Note that the APPLE3:FORMATTER program will not work with the ABM. For first-time operation, the method of $4.2.1 is recommended.

4.2.1 INITIALIZING WITH BUBPASUTIL

1) Start with the patched SYSTEM.APPLE booted and the ABM in slot#4 (unit#9). Check that the ABM write-protect switch is in the unprotected position.

2) Select X(ecute from the command menu and execute the BUBPASUTIL program.

3) Choose selection (5) and follow the instructions, responding with "9" to the prompt "ENTER UNIT# OF ABM". If a Pascal directory already exists on the ABM, you will be asked if you want to destroy the volume. Otherwise, a new directory will be created with the name "BUB" followed by the unit number of the slot. In this case the volume name is "BUB9:".

4) The ABM can now be used as a Pascal volume. However if it is desired to cold boot from the ABM by adding

the necessary files from APPLE1:, then a patched version of the boot blocks must be transfered to the ABM. This is done by replacing the ABM MASTER DISK with the patched APPLE1: disk and choosing selection (4) from the BUBPASUTIL program. Follow the instructions by indicating that the boot is to be copied from UNIT#4 to UNIT#9.

4.2.2 INITIALIZING WITH F(ILER COMMANDS

1) With the same starting conditions as the last section, choose the F(iler from the command prompt line.

2) Choose Z for "zero directory" and answer the questions as follows:

```
Zero dir of ? #9
Are there 280 blks on the disk? N
# of blocks on the disk ? 256
New vol name ? BUB9: (example only)
BUB9: correct ? Y
BUB9: zeroed
```

3) If desired, the boot blocks can be added to be ABM as outlined in step 4 of the last section.

4.3 COLD BOOTING PASCAL FROM THE BUBBLE

To prepare the ABM board to cold boot Pascal, the above initialization techniques can be used followed by the transfer to the ABM of the additional appropriate files from the APPLE1: disk. But this is a slower process than the following method:

1) Start with the same initial condition conditions as the last two sections. Place the APPLE1: disk with the patched SYSTEM.APPLE and boot blocks in the disk drives.

2)Enter the F(iler and select the T(ransfer routine. Transfer the unit 4 to the unit 9, i.e., enter:

```
Transfer? #4,#9
Transfer 280 blocks? (Y/N) N
# of blocks to transfer? 256
APPLE1:                    -) #9
```

3)Q(uit the Filer and execute the program BUBPASUTIL. Change the directory of the ABM board to indicate only 256 blocks available using selection (3).

4)Turn off the power to the Apple computer and switch the slots of the ABM board and the disk driver card.

5)Turn on the power to the Apple computer and the ABM board should cold boot Pascal.

Note that the APPLE1: floppy disk should not be in the disk drive during the boot from the ABM or the operating system might become confused and use the floppy as the root volume.

APPENDIX A

TESTS PERFORMED BY DIAGNOSTICS

The terms used in the DIAGNOSTICS program are given only a cursory definition in this discussion. The interested user can find a more complete explanation in the Intel "Memory Components Handbook" for 1983. Error codes generated by this program can be found in Appendix B.

))TESTING THE INTERFACE TO THE BMC

The Bubble Memory Controller (BMC) is the large 40-pin chip on the ABM board. This test checks whether the Apple computer's microprocessor, the 6502, can communicate with the BMC by writing and then reading several test patterns to a register on the BMC (the User Register).

))TESTING THE BMC FIFO

The BMC has a first-in-first-out (FIFO) buffer that can hold 40 bytes of data. This test loads the FIFO with a pattern and then reads it back out, checking that the pattern read is the same as the pattern written.

))TESTING THE FSA

The Formatter/Sense Amplifier (FSA) is the 20-pin chip at the far end of the ABM board. It communicates serially with the BMC and it is this interface that is checked here. Basically, the 40 byte bootloop register in the FSA is written with a test pattern and read back (see "A Primer on Magnetic Bubble Memories" page 9).

))READING THE BOOTLOOP

In this test, the 40-byte bootloop bitmap stored in the bubble memory is transferred into the bootloop register of the FSA and then read into the RAM of the Apple computer starting at location hex 6900. This bitmap is then displayed and, ignoring the addresses and spaces, should match the 5 rows of 16 hexadecimal

numbers on the label of the bubble memory package. The bootloop bitmap is a record of which of the bubble memory storage loops are to be used and which are not to be used. That is, to increase manufacturing yields, the bubble memory chip contains more storage loops than required so that a few of them can be bad without affecting the operation or storage capacity of the entire chip. This record is stored on the bootloop in the bubble memory and can normally only be corrupted if the ABM board is inserted or removed from its slot with the power on or if there is a short circuit elsewhere in the computer that causes the 5V or 12V power to drop abruptly.

If the bootloop bitmap does not match the label, then it is possible to correct it by answering "N" to the question and following the instructions. However, normally if the bootloop was lost, it is also possible that one or more of the seed bubbles are lost. The seed bubbles should be checked before trying to correct the bubble memory bootloop.

))CHECK FOR SEED BUBBLES

This test checks to see if all the seed bubbles are present. It requires a good bootloop bitmap to be present, so if the bootloop test above failed, it is necessary to manually enter the bitmap from the label. Basically, this routine saves the first page of the bubble memory (68 bytes without error correction), and then writes $FF's to the page. When the page is read back, if all $FF's are not returned, one or more of the seed bubbles is missing. This page of test data is displayed on the console for the user, and the original page of data is restored to the ABM.

If the ABM board fails this test, try it again making sure that the bootloop bitmap is correct. The ABM board must be returned to Helix Labs for repair if loss of a seed bubble is confirmed.

))WRITE/READ PATTERN TEST

This test writes a test pattern to all memory locations in the bubble memory and then reads it back again, using level 3 error detection so that even soft errors are detected. This operation is done with several patterns as shown on page 7. Note that this test will erase the entire ABM contents, and thus should be skipped if erasure is not desired.

))FIND EPROM BYTE SUM

For this test, all the bytes on the EPROM of the ABM board are summed and then displayed. The sum should match the number on the EPROM.

This completes the diagnotics program.

NOTES:

1) If the bootloop was lost and DIAGNOSTICS was used to restore it, it is very possible that some of the stored data was disturbed as well. Because of the error checking, I/O errors will occur when trying to access the data. In fact, because of the way the INIT command for the ABM operates, it might not be possible to INIT the ABM without an I/O error interruption. In this case, it is necessary to write over every data location. This should be done by using the WRITE/READ PATTERN TEST of the DIAGNOSTICS program.

APPENDIX B

DIAGNOSTICS PROGRAM ERROR CODES

The following are explanations of error codes that could occur during the DIAGNOSTICS program test of the ABM. The error code numbers not only specify what failed but indicate where in the program the error happened which is why several error codes can specify the same failure.

ERROR#	FAILED OPERATION OR ERROR
1	failure to properly read/access the internal ABM bus
2	cannot access the BMC user register
6	data written and then read from BMC FIFO not the same
8	ABORT command to BMC
9	MBM PURGE command to BMC
10	RESET FIFO command timeout or failure
11	writing to the BMC FIFO
13	WRITE BOOTLOOP REG command to BMC
15	RESET FIFO command to BMC
16	READ BOOTLOOP REG command to BMC
17	reading the BMC FIFO
18	data written and then read from bootloop register not the same
20	ABORT command to BMC
21	MBM PURGE command to BMC
22	RESET FIFO command timeout or failure
23	READ BOOTLOOP command to BMC
24	reading the BMC FIFO
25	ABORT command to BMC
26	MBM PURGE command to BMC
27	RESET FIFO command timeout or failure
29	WRITE BOOTLOOP REG command to BMC
30	writing to the BMC FIFO
31	WRITE BOOTLOOP command to BMC
33	data written and then read from the FIFO not the same
34	ABORT command to the BMC

ERROR#	FAILED OPERATION OR ERROR
35	MBM PURGE command to the BMC
36	RESET FIFO command to the BMC
37	writing to the BMC FIFO
38	WRITE BOOTLOOP REG command to the BMC
39	READ command to the BMC
40	WRITE command to the BMC
41	READ command to the BMC
42	WRITE command to the BMC
48	seedcheck test failed
80	ABORT command to BMC
81	MBM PURGE command to BMC
82	RESET FIFO command to BMC
83	writing to the BMC FIFO
84	WRITE BOOTLOOP REGISTER MASKED command to BMC
60,64,68,72	RESET FIFO command to BMC before write
62,66,70,74	RESET FIFO command to BMC before read
61,65,69,73	WRITE pattern to all ABM locations with error correction
63,67,71,75	READ pattern from all ABM locations with level 3 error correction

APPENDIX C

PATCH TO APPLE DOS 3.3

The following is the assembled source code for the patch to Apple DOS 3.3. This patch is installed in the unused area just ahead of the RWTS subroutine. Although the code is relocatable, it was assembled with the origin at $BCDF which is where it would end up when the patched DOS 3.3 is booted into an Apple computer with 48K of memory. After loading the Input/Output Block (IOB) address at a pointer location, the program proceeds to get the slot number (N) of the device to be accessed. It then reads the byte at $CN00. If this byte is $A6, the board is an ABM and the program executes a JSR to the ABM drivers starting at $C800. Note that even though a JSR is executed, the program does not return to the instruction following this JSR because this return address is POP'ed off the stack before an RTS instruction is executed, returning the program to the caller of the RWTS subroutine.

```
                134 ************************************
                135 * SEGMENT : ABM PATCH TO DOS 3.3
                136 *
                137 * MISC EQUATES
                138 *
                139 IOBSLT      EQU  $01
                140 SLTRMPT     EQU  $26
                141 BUBROMB1    EQU  $A6
                142 ISFLOP      EQU  $BD04
                143 IOBP        EQU  $48
                144 *
                145             ORG  $BCDF
                146 *
BCDF: 84 48     147 PATCH       STY  IOBP
BCE1: 85 49     148             STA  IOBP+1
                149 *
                150 * GET SLOT*16
                151 *
BCE3: A0 01     152             LDY  #IOBSLT
BCE5: B1 48     153             LDA  (IOBP),Y
BCE7: AA        154             TAX
                155 *
                156 * CREATE ADDR OF SLOT PROM ($CN00)
                157 *
BCE8: 4A        158             LSR
BCE9: 4A        159             LSR
BCEA: 4A        160             LSR
BCEB: 4A        161             LSR
BCEC: 09 C0     162             ORA  #$C0
BCEE: A0 00     163             LDY  #$00
BCF0: 84 26     164             STY  SLTRMPT
BCF2: 85 27     165             STA  SLTRMPT+1
                166 *
                167 * CLEAR $C800 ACCESS
                168 *
BCF4: AD FF CF  169             LDA  $CFFF
                170 *
                171 * CHECK FIRST BYTE OF PROM (BUB=$A6)
                172 *
BCF7: B1 26     173             LDA  (SLTRMPT),Y
BCF9: C9 A6     174             CMP  #BUBROMB1
BCFB: D0 07     175             BNE  ISFLOP
BCFD: 20 00 C8  176             JSR  $C800
                177 *
                178 * DOS 3.3 RWTS ENTRY
                179 *
BD00: 18        180 RWTSST      CLC
BD01: 90 DC     181             BCC  PATCH
BD03: 00        182             BRK
BD04:           183 ISFLOP      DS   1
                184 *
                185************************************
```

APPENDIX D

ABM BOARD TO APPLE I/O SLOT INTERFACE

The Apple computer interfaces with the ABM board through the Intel 7220-1 Bubble Memory Controller (BMC) and a 2716 EPROM. The EPROM holds the booting program in the $CN00 (N=slot#) address page and most of the driver routines for bubble memory operation in the $C800-CEFF address area.

The BMC is accessed through the 16 slot dependent I/O peripheral registers $C0MX (where M=8+slot# and X=$0 to $F) as follows:

ADDRESS	NEUMONIC	DESCRIPTION
$C0M0	DATAREG	Intel 7220-1 data register (A0=0)
$C0M1	CMDST	Intel 7220-1 command/status(A0=1)
$C0M2		a "read" returns XX111111B
to		
$C0M7		" "
$C0M8	LEDOFF	software switch to turn LED off
$C0M9	LEDON	software switch to turn LED on
$C0MA		a "read" returns XX111111B
to		
$C0MF		" "

The data sheets on the 7220-1 BMC are included with the ABM package. They describe the DATAREG and CMDST registers and how to use them. However a more detailed discussion can be found in the Intel "Memory Components Handbook" for 1983. The LEDON and LEDOFF are in the address locations for MOTORON and MOTOROFF of the floppy disk; they only operate the LED and have no other function than to indicate to the user that the ABM is being accessed. All the other address locations return a binary XX111111 where bit 7 is determined by the WRITE-PROTECT switch (enable = 0) and bit 6 is determined by the SPEAKER-CLICK switch (enable = 1).

The booting routine is on the EPROM and is accessed through the slot dependent addresses $CN00 to $CNFF. This routine duplicates for the ABM what the 256 byte PROM on the floppy driver card does for the floppy, i.e., reads TRK 0 SEC 0 into RAM starting at $800 and then jumps to $801. The physical address of this page in the EPROM is $700 to $7FF.

The driver routines and subroutines are in the physical EPROM addresses $000 to $6FF, and the Apple accesses them through the common 2K byte expansion EPROM area starting at $C800. Since the last page of the EPROM is the boot routine and is accessed through addresses $CN00 as described above, the last page of the expansion area, $CF00 to $CFFF, is hardwired as the expansion area disable software switch.

APPENDIX E

JUMP TABLE DESCRIPTION

At the beginning of the ABM boards $C800 expansion memory is the following jump table:

ADDRESS	INSTRUCTION
$C800	JMP BUBRWTS
$C803	JMP INITCHK
$C806	JMP CMDWAIT
$C809	JMP LDRDRAC
$C80C	JMP LDWRRAC
$C80F	JMP INITIAL
$C812	JMP WPTEST
$C815	JMP RDBUB
$C818	JMP WRBUB
$C81B	JMP INBITMAP
$C81E	JMP CLICK
$C821	JMP BUBPASC

These are all subroutines ending in an RTS instruction. On future updates to the ABM board EPROM the jump table and the function of the subroutines described will remain the same even though the actual address of the subroutines on the EPROM could change.

BUBRWTS
This subroutine is called by the patch to DOS 3.3 (see Appendix B) and is the RWTS subroutine for the ABM. That is, it performs the RWTS task as set up in the RWTS parameter table (I/O block) on the bubble memory and returns an error code in the A register the same way a normal disk RWTS does. Note that it pops the JSR return address of the patch routine off the stack before an RTS to the caller.
INPUT: $48,$49 = pointer to RWTS parameter table
$3DC = subroutine to load A,Y registers with address of File Manager parameter table
X reg = slot*16
CALLS: all the subroutines below except BUBPASC

OUTPUT: error code in A reg
DESTROYS: A,Y registers

INITCHK
This subroutine checks to see if the 7220 BMC needs initializing. This should be done before attempting to perform any operations with the ABM. The ABM needs to be reinitialized after a power-up or a reset.
INPUT: X reg = slot*16
OUTPUT: carry = 1 if initialization is required
DESTROYS: A,Y registers

CMDWAIT
This subroutine waits for a command to the BMC to be competed.
INPUT: X reg = slot*16
OUTPUT: none
DESTROYS: A reg

LDRDRAC
This subroutine loads the BMC parametric registers for a read with error correction.
INPUT: Y reg = number of bubble pages to read
X reg = slot*16
$2E,$2F = starting bubble page to be read
$2C,$2D = address of the DATAREG of the BMC (use the subroutine LDDATREGP to load this address)
$2B = slot*16
OUTPUT: BMC parametric registers loaded for a read
DESTROYS: A reg

LDWRRAC
This subroutine loads the BMC parametric registers for a write with error correction.
INPUT: Y reg = number of bubble pages to write
X reg = slot*16
$2E,$2F = starting bubble page to be write
$2C,$2D = address of the DATAREG of the BMC (use the subroutine LDDATREGP to load this address)

```
          $2B = slot*16
OUTPUT:   BMC parametric registers loaded for a write
DESTROYS: A reg
```

LDDATRGP

This subroutine loads the zero page address $2C,$2D with the pointer to the DATAREG of the currently accessed BMC.

```
INPUT:    $2B = slot*16
OUTPUT:   $2C,$2D = address of DATAREG of BMC
DESTROYS: A reg
```

INITIAL

This subroutine initializes the BMC after a power-up or reset.

```
INPUT:    X reg = slot*16
OUTPUT:   carry = 0 if successful
CALLS:    LDDATRGP
          LDWRRAC
          CMDWAIT
DESTROYS: A,Y regs
```

WPTEST

This subroutine tests if the ABM board is write-protected (see Appendix D).

```
INPUT:    X reg = slot*16
OUTPUT:   carry = 1 if write-protected
DESTROYS: A reg
```

RDBUB

This subroutine reads one sector (256 bytes) from the ABM to a buffer in RAM.

```
INPUT:    X reg = slot*16
          $26,$27 = buffer address
          $2C = slot*16
OUTPUT:   buffer loaded from sector
          carry = 0 if successful
CALLS:    LDDATRGP
          LDRDRAC
DESTROYS: A,Y regs
```


WRBUB

This routine writes one sector from a buffer in RAM to the ABM.

```
INPUT:    X reg = slot*16
OUTPUT:   carry = 1 if write-protected
DESTROYS: A reg
```

INBITMAP

This subroutine changes the bitmap to protect trks 3-5 on the VTOC during a DOS 3.3 INIT command and would not be useful to external programs.

CLICK

This subroutine toggles the speaker if bit 6 of slot register C0MC (see Appendix D) is high.

```
INPUT:    X reg = slot*16
OUTPUT:   none
DESTROYS: A reg
```

BUBPASC

This subroutine is the RWTS routine called by the patch in Apple Pascal 1.1. It is specialized and not useful to external programs.

APPENDIX F

HELIX ABM BOARD SPECIFICATIONS

Bubble Memory: Intel 7110-4 One Megabit Bubble Memory

Storage Capacity: 128K bytes (equivalent to 32 16-sector tracks)

Case Operating Temperature: +10 to +55 C

Non-volatile Storage: -20 to +75 C

Average Access Time: 40 msec

Data Rate: 12.5K bytes/sec

Typical Current Requirements: 5V - 210 ma standby
215 ma read/write

12V - 40 ma standby
250 ma read/write

NOTES:

1) No more than two ABM boards should be inserted in the Apple computer at one time because the 5V current to the peripheral slots is limited to a total of 500 ma. There are power supplies available that have twice the 5V current output of the regular Apple computer supply and using one of these it should be possible to insert three or four ABM boards.

2) If power is interrupted during an ABM read/write operation, the power supply decay rates should not exceed 0.45 volts/msec for 5V or 1.1 volts/msec for 12V or else there is a possibility of loss of data, bootloop, or even one or more seed bubbles. Normally, AC power interruption will not cause these rates to be exceeded. However, if during ABM operation, there is a short circuit on the Apple motherboard or one o the peripheral boards that abruptly pulls the 5V or

12V supply low (or if the ABM board is pulled from the slot with the power on) the above problems are a possibility.