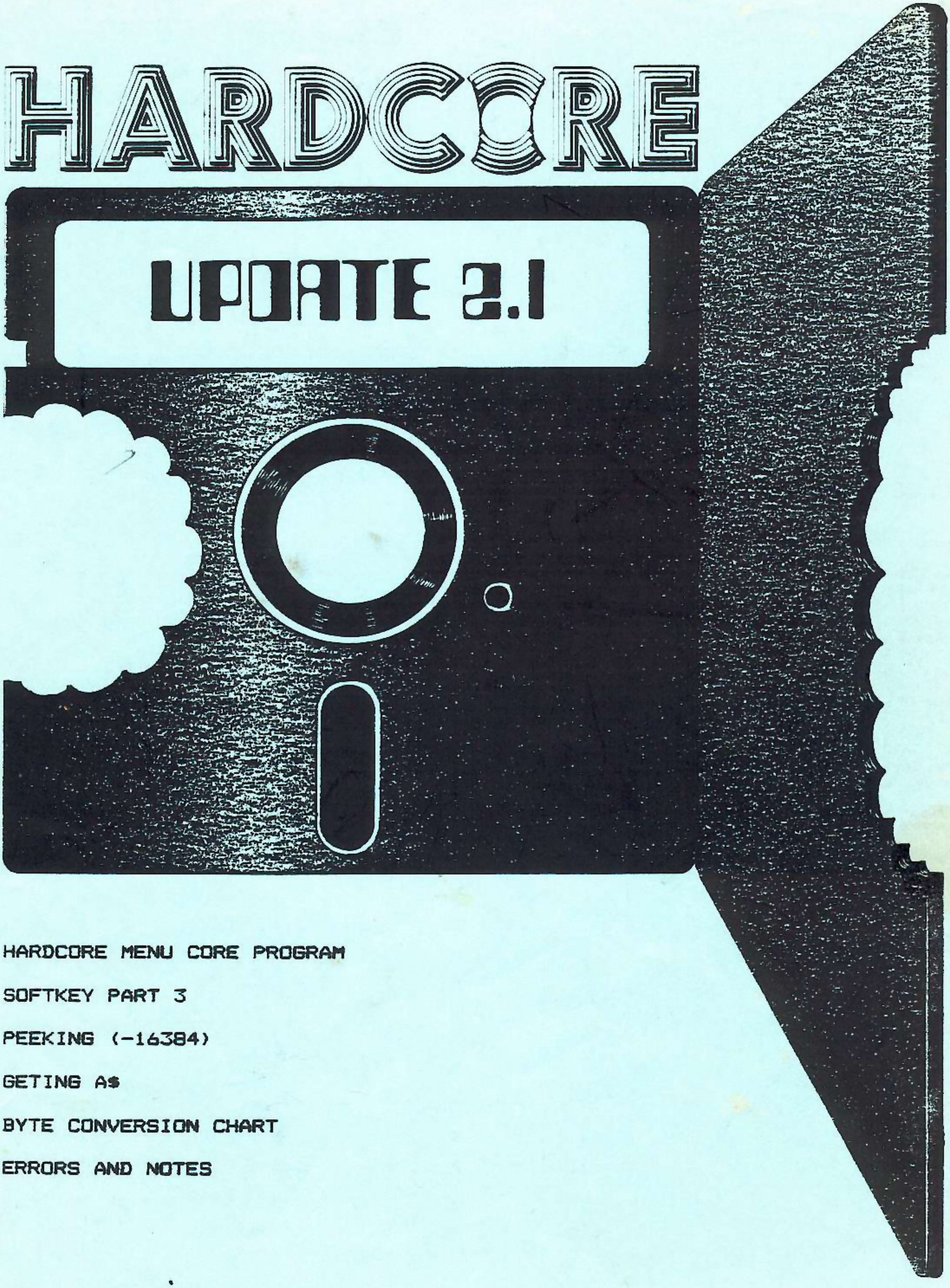
HARDCORE



UPDATE 2.1

HARDCORE MENU CORE PROGRAM

SOFTKEY PART 3

PEEKING (-16384)

GETING A$

BYTE CONVERSION CHART

ERRORS AND NOTES

BYTE CONVERSION CHART

HARDCORE COMPUTING MAGAZINE P.O. BOX 44549 TACOMA, WA 98444 $20/YEAR

INVERSE				INVERSE				FLASHING				FLASHING				CONTROL				NORMAL				NORMAL							
BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC	BINARY	HEX	DEC	ASC
0000 0000	$00	0	@	0010 0000	$20	32		0100 0000	$40	64	@	0110 0000	$60	96		1000 0000	$80	128	@	1010 0000	$A0	160		1100 0000	$C0	192	@	1110 0000	$E0	224	
0000 0001	$01	1	A	0010 0001	$21	33	!	0100 0001	$41	65	A	0110 0001	$61	97	!	1000 0001	$81	129	A	1010 0001	$A1	161	!	1100 0001	$C1	193	A	1110 0001	$E1	225	!
0000 0010	$02	2	B	0010 0010	$22	34	"	0100 0010	$42	66	B	0110 0010	$62	98	"	1000 0001	$82	130	B	1010 0010	$A2	162	"	1100 0010	$C2	194	B	1110 0010	$E2	226	"
0000 0011	$03	3	C	0010 0011	$23	35	#	0100 0011	$43	67	C	0110 0011	$63	99	#	1000 0010	$83	131	C	1010 0011	$A3	163	#	1100 0011	$C3	195	C	1110 0011	$E3	227	#
0000 0100	$04	4	D	0010 0100	$24	36	$	0100 0100	$44	68	D	0110 0100	$64	100	$	1000 0011	$84	132	D	1010 0100	$A4	164	$	1100 0100	$C4	196	D	1110 0100	$E4	228	$
0000 0101	$05	5	E	0010 0101	$25	37	%	0100 0101	$45	69	E	0110 0101	$65	101	%	1000 0100	$85	133	E	1010 0101	$A5	165	%	1100 0101	$C5	197	E	1110 0101	$E5	229	%
0000 0110	$06	6	F	0010 0110	$26	38	&	0100 0110	$46	70	F	0110 0110	$66	102	&	1000 0101	$86	134	F	1010 0110	$A6	166	&	1100 0110	$C6	198	F	1110 0110	$E6	230	&
0000 0111	$07	7	G	0010 0111	$27	39	'	0100 0111	$47	71	G	0110 0111	$67	103	'	1000 0110	$87	135	G	1010 0111	$A7	167	'	1100 0111	$C7	199	G	1110 0111	$E7	231	'
0000 1000	$08	8	H	0010 1000	$28	40	(	0100 1000	$48	72	H	0110 1000	$68	104	(	1000 0111	$88	136	H	1010 1000	$A8	168	(	1100 1000	$C8	200	H	1110 1000	$E8	232	(
0000 1001	$09	9	I	0010 1001	$29	41	)	0100 1001	$49	73	I	0110 1001	$69	105	)	1000 1000	$89	137	I	1010 1001	$A9	169	)	1100 1001	$C9	201	I	1110 1001	$E9	233	)
0000 1010	$0A	10	J	0010 1010	$2A	42	*	0100 1010	$4A	74	J	0110 1010	$6A	106	*	1000 1001	$8A	138	J	1010 1010	$AA	170	*	1100 1010	$CA	202	J	1110 1010	$EA	234	*
0000 1011	$0B	11	K	0010 1011	$2B	43	+	0100 1011	$4B	75	K	0110 1011	$6B	107	+	1000 1010	$8B	139	K	1010 1011	$AB	171	+	1100 1011	$CB	203	K	1110 1011	$EB	235	+
0000 1100	$0C	12	L	0010 1100	$2C	44	,	0100 1100	$4C	76	L	0110 1100	$6C	108	,	1000 1011	$8C	140	L	1010 1100	$AC	172	,	1100 1100	$CC	204	L	1110 1100	$EC	236	,
0000 1101	$0D	13	M	0010 1101	$2D	45	-	0100 1101	$4D	77	M	0110 1101	$6D	109	-	1000 1100	$8D	141	M	1010 1101	$AD	173	-	1100 1101	$CD	205	M	1110 1101	$ED	237	-
0000 1110	$0E	14	N	0010 1110	$2E	46	.	0100 1110	$4E	78	N	0110 1110	$6E	110	.	1000 1101	$8E	142	N	1010 1110	$AE	174	.	1100 1110	$CE	206	N	1110 1110	$EE	238	.
0000 1111	$0F	15	O	0010 1111	$2F	47	/	0100 1111	$4F	79	O	0110 1111	$6F	111	/	1000 1111	$8F	143	O	1010 1111	$AF	175	/	1100 1111	$CF	207	O	1110 1111	$EF	239	/
0001 0000	$10	16	P	0011 0000	$30	48	Ø	0101 0000	$50	80	P	0111 0000	$70	112	Ø	1001 0000	$90	144	P	1011 0000	$B0	176	Ø	1101 0000	$D0	208	P	1111 0000	$F0	240	Ø
0001 0001	$11	17	Q	0011 0001	$31	49	1	0101 0001	$51	81	R	0111 0001	$71	113	1	1001 0001	$91	145	Q	1011 0001	$B1	177	1	1101 0001	$D1	209	Q	1111 0001	$F1	241	1
0001 0010	$12	18	R	0011 0010	$32	50	2	0101 0010	$52	82	Q	0111 0010	$72	114	2	1001 0010	$92	146	R	1011 0010	$B2	178	2	1101 0010	$D2	210	R	1111 0010	$F2	242	2
0001 0011	$13	19	S	0011 0011	$33	51	3	0101 0011	$53	83	S	0111 0011	$73	115	3	1001 0011	$93	147	S	1011 0011	$B3	179	3	1101 0011	$D3	211	S	1111 0011	$F3	243	3
0001 0100	$14	20	T	0011 0100	$34	52	4	0101 0100	$54	84	T	0111 0100	$74	116	4	1001 0100	$94	148	T	1011 0100	$B4	180	4	1101 0100	$D4	212	T	1111 0100	$F4	244	4
0001 0101	$15	21	U	0011 0101	$35	53	5	0101 0101	$55	85	U	0111 0101	$75	117	5	1001 0101	$95	149	U	1011 0101	$B5	181	5	1101 0101	$D5	213	U	1111 0101	$F5	245	5
0001 0110	$16	22	V	0011 0110	$36	54	6	0101 0110	$56	86	V	0111 0110	$76	118	6	1001 0110	$96	150	V	1011 0110	$B6	182	6	1101 0110	$D6	214	V	1111 0110	$F6	246	6
0001 0111	$17	23	W	0011 0111	$37	55	7	0101 0111	$57	87	W	0111 0111	$77	119	7	1001 0111	$97	151	W	1011 0111	$B7	183	7	1101 0111	$D7	215	W	1111 0111	$F7	247	7
0001 1000	$18	24	X	0011 1000	$38	56	8	0101 1000	$58	88	X	0111 1000	$78	120	8	1001 1000	$98	152	X	1011 1000	$B8	184	8	1101 1000	$D8	216	X	1111 1000	$F8	248	8
0001 1001	$19	25	Y	0011 1001	$39	57	9	0101 1001	$59	89	Y	0111 1001	$79	121	9	1001 1001	$99	153	Y	1011 1001	$B9	185	9	1101 1001	$D9	217	Y	1111 1001	$F9	249	9
0001 1010	$1A	26	Z	0011 1010	$3A	58	:	0101 1010	$5A	90	Z	0111 1010	$7A	122	:	1001 1010	$9A	154	Z	1011 1010	$BA	186	:	1101 1010	$DA	218	Z	1111 1010	$FA	250	:
0001 1011	$1B	27	[	0011 1011	$3B	59	;	0101 1011	$5B	91	[	0111 1011	$7B	123	;	1001 1011	$9B	155	[	1011 1011	$BB	187	;	1101 1011	$D8	219	[	1111 1011	$FB	251	;
0001 1100	$1C	28	\	0011 1100	$3C	60	<	0101 1100	$5C	92	\	0111 1100	$7C	124	<	1001 1100	$9C	156	\	1011 1100	$BC	188	<	1101 1100	$DC	220	\	1111 1100	$FC	252	<
0001 1101	$1D	29	]	0011 1101	$3D	61	=	0101 1101	$5D	93	]	0111 1101	$7D	125	=	1001 1101	$9D	157	]	1011 1101	$BD	189	=	1101 1101	$DD	221	]	1111 1101	$FD	253	=
0001 1110	$1E	30	^	0011 1110	$3E	62	>	0101 1110	$5E	94	^	0111 1110	$7E	126	>	1001 1110	$9E	158	^	1011 1110	$BE	190	>	1101 1110	$DE	222	^	1111 1110	$FE	254	>
0001 1111	$1F	31	_	0011 1111	$3F	63	?	0101 1111	$5F	95	_	0111 1111	$7F	127	?	1001 1111	$9F	159	-	1011 1111	$BF	191	?	1101 1111	$DF	223	_	1111 1111	$FF	255	?

hardcore

HardCore Updates are published between the quarterly issues of HardCore Computing and is copyrighted 1982 by:

SoftKey Publishing
P.O. Box 44549
Tacoma, WA 98444
(206) 531-5690

Non-commercial copying and distribution is encouraged. (make back-up copies!). However, copying for commercial gain is a violation of copyright laws and is therefore frowned upon by our editorial and legal staff.

Subscriptions (1 year):

U.S.A.	$20
Canada	$29
Mexico, Central America	$33
S. America	$38
Europe, Africa, others	$42

Sample Copies:

U.S. and Canada............	$ 5.00
Elsewhere..................	$ 8.00

U.S. FUNDS ONLY   NO CREDIT CARDS

Publisher....... Charles R. Haight
Editor............. Bev R. Haight
Comptroller..... Karen Fitzpatrick

Graphics.....................Ryuji

Columnists .... Bobby

TABLE OF CONTENTS

This chart will help you convert between binary, hex, decimal and ASCII codes. Save it for future reference.

When Bobby's Softkey "Muffins" fails to copy files on a protected disk, it may mean that you'll have to download it using what Bobby calls "Brute Force": a sector-by-sector copy. His Special I.O.B program will accomplish this for you. This article is for 16 sectors only, but he's working on a 13 sector version. You'll need Applesoft in ROM.

A versatile and short applesoft program that will become the core of a multi-use master catalog. This is for 48k only and has a unique machine language program that captures the entire catalog and,in this short version, PRINTs it in two columns. As a hello program, MENU will allow program selection by number. Enhanced versions will be published later.

For programmers who are tired of referring to the confusing charts in the applesoft manual, here are two handy charts:
PEEKing (-16384)
GETing A$

Soft Key

part 3
Special I.O.B.s (input-output blocks)

How To Back Up Copy-Protected Disks

Softkey III will deal with disks that change the address and data marks from track to track or use the standard DOS marks but change the sector numbers. None of the "MUFFIN's" will work on these disks. They require a special program to directly access the disk.

This program accomplishes the task by doing a "brute force", sector-by-sector copy of each track. On a 13 sector disk, it will cause the disk drive to recalibrate (the racheting sound) on any sector that doesn't contain data. This is nothing to worry about and does not mean the program is not working.

The program calls the RWTS (Read or Write a Track/Sector) directly and uses its own IOB (Input/Output Block). The procedure for this is very well documented in the DOS manual (page 94-98).

Basically, the IOB is a list of parameters that is used by the RWTS whenever a disk access is necessary. This list includes the Slot, Drive, Track, Sector and Volume number. Each call to the RWTS will, depending on the Command code, position the Read/Write head only, Read or Write a single sector or INITialize the disk.

requirements:
blank disk
IOB program (listed)
DiskView (or another
 nibbler program
DOS 3.3
48 K
Applesoft in ROM

SETTING UP THE CONTROLLER

The controller is the heart of the IOB program. It starts at line 1000 and defines the Track/Sector pairs that you wish to copy and the address and data mark values you will use to read or write them.

Two basic controllers are listed. The first assumes that only the sector numbers are changed. The second assumes that only the address and data marks are changed. They are presented as samples of what you can do with the program. The controller can be as simple or as complex as necessary.

Controller #1

This controller is set up to copy a disk that uses standard address and data marks but changes the sector numbers so that they step by even increments.

NOTE: Use the 4+4 Conversion Chart in Disklocks (issue 2) to convert the encoded sector bytes for each address mark to hex. Verify that the sector numbers step by even number increments (ie. 0,2,4,6,8,10 etc.)

```
1000 FOR TK = 3 TO 34
1010 DV = 1 : CD = RD : GOSUB 50 :
     GOSUB 85
1020 DV = 2 : CD = WR : GOSUB 50 :
     GOSUB 80
1030 NEXT
```

Often the checksum on the original disk will be altered. If you think this is the case then add line 200 to the IOB program.

```
 200  GOSUB 55
```

This change will cause DOS to look at the start of address and start of data marks and ignore the checksum and end marks.

Controller #2

This controller is set up to copy a disk that uses different address and data marks on each track. It assumes that all six bytes (3 address/3 data) are changed. If only one or two are changed then the controller can be altered to show this. (IE. If address mark A2 is unchanged then the 'READ A2' could be omitted from line 60 and the 'POKE 47455,A2' omitted from line 70.

Both controllers set up a FOR/NEXT loop for the track numbers and GOSUBs to the appropriate routine to READ/WRITE a track.

```
1000 FOR X = 3 TO 34
1010 DV = 1 : CD = RD : GOSUB 50 : GOSUB
      60 : GOSUB 80
1020 DV = 2 : CD = WR : GOSUB 50 : GOSUB
      90 : GOSUB 80
1030 NEXT
```

When typing the data for line 63050, be sure to maintain the proper order. The routine that READs the data assumes that the address mark information is first and the data mark information is second.

USING THE SOFTKEY I.O.B. PROGRAM

Perform the following steps:

1. RUN DiskView
2. Remove the DiskView disk and insert the back-up copy of your program disk.

- NEVER USE THE ORIGINAL DISK

3. READ track 3 and look for the start of address and the start of data marks. The normal values are D5 AA 96 (address) and D5 AA AD (data). Write down the values that you find.

NOTE: See Disklocks (issue 2) for more information on address and data marks. Practice on a normal DOS disk until you are familiar with the disk format and can easily find the address and data marks.

4. Repeat step 3 for each track (3 to 34).
5. Use the Byte Cross-reference Chart to convert the hex bytes to decimal.
6. Load the IOB program.
7. Make the appropriate changes in the controller (lines 1000 thru 2000) Insert the decimal values from step 5 into the data statement(s) at line 63050.
8. Run the program.

Listing

```
      >> set LOMEM above buffer <<

10  TEXT : HOME : LOMEM : 16385 : GOSUB
    63000 : GOTO 100

    >> print track and sector info <<

20  HOME : VTAB 12 : HTAB 12 : PRINT
    "TRACK "TK"  SECTOR "ST : RETURN

  >> center text on screen and print <<

30  HTAB 20 - ( LEN (A$) / 2) : PRINT A$
    ;: RETURN

   >> wait for key press then return <<

40  HOME : VTAB 12 : GOSUB 30 : VTAB 14
    : A$ = "PRESS ANY KEY TO CONTINUE "
    : GOSUB 30 : GET AN$ : RETURN

      >> poke variables into IOB <<

50  POKE BUF,32 : POKE CMD,CD : POKE
    TRK,TK : POKE SCT,ST : POKE DRV,DV :
    POKE VOL,VL : RETURN

  >> don't use checksum or end marks <<

55  POKE 47405,24 : POKE47406,96 :
    POKE 47497,24 : POKE47498,96

      >> get address and data marks <<

60  READ A1 : READ A2 : READ A3 :
    READ D1 : READ D2 : READ D3

          >> poke the info into DOS <<

70  POKE 47445,A1 : POKE 47455,A2 :
    POKE 47466,A3 : POKE 47335,D1 :
    POKE 47345,D2 : POKE 47356,D3 :
    RETURN

          >> read/write a track <<

80  FOR ST = 0 TO DOS : POKE SCT,ST :
    GOSUB 20 : CALL IO : POKE BUF, PEEK
    (BUF) + 1 : NEXT : RETURN

 >> read a track by even no. sectors <<

85  FOR ST = 0 TO DOS * 2 STEP 2 : POKE
    SCT,ST : GOSUB 20 : CALL IO : POKE
    BUF, PEEK (BUF) + 1 : NEXT : RETURN

       >> restore normal DOS values <<

90  POKE 47445,213 : POKE 47455,170 :
    POKE 47466,150 : POKE 47335,213 :
    POKE 47345,170 : POKE 47356,173 :
    RETURN

          >> get original disk <<

100  A$ =
    "INSERT ORIGINAL DISK IN DRIVE 1."
    : GOSUB 40

           >> read sector zero <<

110  CD = RD : DV = 1 : GOSUB 50 : CALL
     IO

     >> reset IOB variables for INIT <<

120  VL = PEEK (OVL) : DV = 2 : CD = IN
     : GOSUB 50

    >> INITialize destination disk <<

130  A$ =
     "INSERT BLANK DISK IN DRIVE 2. " :
     GOSUB 40 : CALL IO : VL = 0

  >> CONTROLLER portion of program <<

-------------------------------------
 INSERT APPROPRIATE CONTROLLER HERE.
-------------------------------------

              >> end routine <<

62990  A$ = "COPY COMPLETED" : GOSUB 40
       : END

         >> poke machine subroutine <<

63000  FOR X = 768 TO 796 : READ A :
       POKE X,A : NEXT
63010  DATA 169,3,160,8,32,217,3,96,1,96
       ,1,0,0,0,25,3,0,32,0,0,1,0,0,96,1
       ,0,1,239,216

           >> initialize variables <<

63020  TK = ST = VL = CD = DV
63030  TRK = 780 : SCT = 781 : CMD = 788
       : RD = 1 : WR = 2 : SLT = 777 :
       DRV = 778 : BUF = 785 : ERR = 789
       : VOL = 779 : IO = 768 : INIT =
       4 : OVL = 790
63035 DOS = 15
63040  RETURN

>> address and data mark information <<

63050  DATA 0
```

MENU

requirements:
DOS 3.3
48 K
Applesoft in ROM

Need a program which lists all those small programs in a double column format a page at a time?

Here is one that shows you how and can be moved from disk to disk without changes. If you don't find the program you want, slip in a new disk and press return, it takes care of itself.

In order to use the program, you will need an Apple II or Apple II+ with 48K and Applesoft in ROM. The program will only run under 16 sector DOS 3.3.

Coming in the next issue will be an expanded version which will accomodate 13 or 16 sector DOS and other sizes of memory.

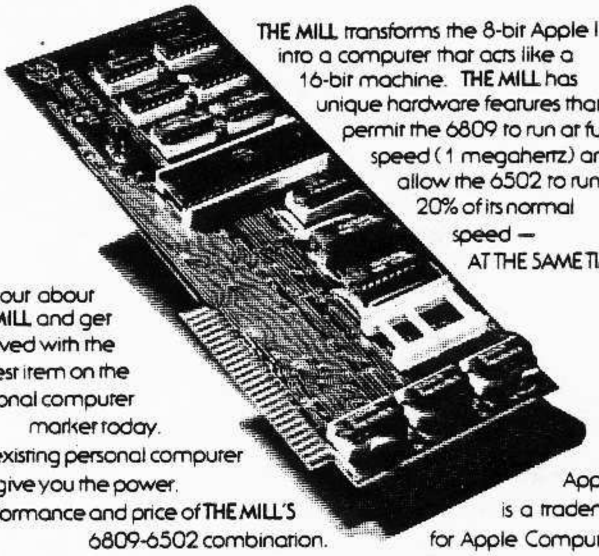
The beginning of a Master CATALOG HELLO.

```
10  TEXT : HOME : GOTO 63000

        >> print 40 dashes <<

20  FOR X = 1 TO 40: PRINT "=";: NEXT :
    RETURN

     >> print left column of names <<

30  PRINT X;: HTAB 5: PRINT  MID$
    (NA$(X),8,30): RETURN

    >> print right column of names <<

40  PRINT " "X"  ";: HTAB 6: PRINT  MID$
    (NA$(X),8,15): RETURN

>> print dashed lines / reset window <<

60  VTAB 1: GOSUB 20: VTAB 22: GOSUB 20:
    POKE 34,1: VTAB 2: RETURN

    >> change width of text window <<

80  POKE 33,21: POKE 32,19: VTAB 1: HTAB
    1: PRINT : RETURN
85  POKE 33,40: POKE 32,0: VTAB 23: HTAB
    1: PRINT : RETURN

     >> check if program is listed <<

90  GOSUB 85: VTAB 23: PRINT
    "IS YOUR PROGRAM LISTED HERE? Y/";:
    INVERSE : PRINT "N";: NORMAL : PRINT
    " ";: GET A$: PRINT

      >> set maximum file number <<

100  IF A$ = "Y" THEN MAX = N - 1:N =
     105: POP : GOTO 2000

    >> clear window and home cursor <<

110  VTAB 23: CALL  - 958: POKE 34,1:
     POKE 35,21: HOME : POKE 35,24:
     RETURN

    >> entry point to format screen <<

1000  GOSUB 60:X = 0
1010  FOR N = 1 TO 105
```

```
      >> check if last file name <<

1020  IF  LEFT$ (NA$(N),2) = "  " THEN
      MAX = N - 1:N = 105: GOTO 2000

 >> select appropriate print routine <<

1030  X = X + 1: IF X < 21 THEN  GOSUB
      30: GOTO 2000
1040  IF X = 21 OR X = 61 OR X = 101
      THEN  GOSUB 80
1050  IF X < 41 THEN  GOSUB 40: GOTO
      2000
1060  IF X = 41 OR X = 81 THEN  GOSUB 90
1070  IF X < 61 THEN  GOSUB 30: GOTO
      2000
1080  IF X < 81 THEN  GOSUB 40: GOTO
      2000
1090  IF X < 101 THEN  GOSUB 30: GOTO
      2000
1100  GOSUB 40
2000  NEXT

         >> reset text window <<

2005  GOSUB 85: POKE 34,22

         >> clear prompt area <<

2010  VTAB 23: CALL  - 958

       >> ask user to select a file <<

2020  INPUT "SELECT A FILE BY
      NUMBER -->";A$ : A = VAL (A$):
      IF A > MAX THEN 2010

>> if none selected, start over again <<

2025  IF A = 0 THEN  RUN

     >> get file type and file name <<

2030  X = ASC ( MID$ (NAME$(A),2,1)):
      B$ =  MID$ (NAME$(A),8,30)

      >> set up correct DOS command <<

2040  IF X = 194 THEN A$ = "BRUN "
2050  IF X = 212 THEN A$ = "EXEC "
2060  IF X = 193 OR X = 201 THEN A$ =
      "RUN "
     >> print command to screen <<

2065  VTAB 23: PRINT A$B$

       >> send command to DOS <<

2070  VTAB 23: PRINT D$A$B$
62999  END


     >> dimension NAME$ array <<

63000  DIM NAME$(105)

          >> B$ = 38 spaces <<

63010  FOR X = 1 TO 38:B$ = B$ + " ":
       NEXT
```

```
   >> fill array from top to bottom <<

63020  FOR X = 105 TO 1 STEP  -
       1:NAME$(X) = B$: NEXT
63030  NAME$(0) =  LEFT$ (B$,19)
63040  D$ =  CHR$ (4)

          >> keep user informed <<

63045  VTAB 12: HTAB 12: PRINT "READING
       CATALOG ": VTAB 12: HTAB 29

        >> poke machine subroutine <<

63050  FOR X = 768 TO 813: READ XZ: POKE
       X,XZ: NEXT X
63060  DATA 169,34,141,83,170,169,3,141
       ,84,170,169,45,141,85,170,169,3
       ,141,86,170,160,8,177,107,141,35
       ,3,200,177,107,141,36,3,96,141,0
       ,3,238,35,3,208,3,238,36,3,96
```

```
63065  ONERR  GOTO 63075


 >> call machine subroutine to fill <<
 >>     array with catalog names    <<

63070  CALL 768: PRINT D$"CATALOG"

         >> reset normal pointers <<

63075  POKE 216,0: PRINT D$"PR#0": PRINT
       D$"IN#0": PRINT

  >> check if no file names !ERROR! <<

63080  IF  LEFT$ (NA$(0),1) = " " THEN
       VTAB 12: CALL  - 868: PRINT
       "!!! ERROR - UNABLE TO READ
       DIRECTORY !!!": END
63100  HOME : GOTO 1000
```



PEEKing (-16384)

HARDCORE COMPUTING
P.O. BOX 44549
TACOMA, WA 98444

	ALONE		SHIFT		CONTROL		BOTH	
KEY	VALUE	CHR$	VALUE	CHR$	VALUE	CHR$	VALUE	CHR$
1 !	177	1	161	!	177	1	161	!
2 "	178	2	162	"	178	2	162	"
3 #	179	3	163	#	179	3	163	#
4 $	180	4	164	$	180	4	164	$
5 %	181	5	165	%	181	5	165	%
6 &	182	6	166	&	182	6	166	&
7 '	183	7	167	'	183	7	167	'
8 (	184	8	168	(	184	8	168	(
9)	185	9	169	)	185	9	169	)
0	176	0	176	0	176	0	176	0
: *	186	:	170	*	186	:	170	*
- =	173	-	189	=	173	-	189	=
; +	187	;	171	+	187	;	171	+
, <	172	,	188	<	172	,	188	<
. >	174	.	190	>	174	.	190	>
/ ?	175	/	191	?	175	/	191	?
<--	136		136		136		136	
-->	149		149		149		149	
SPACE	160		160		160		160	
RETURN	141		141		141		141	
ESC	155		155		155		155	
A	193	A	193	A	129	> <	129	> <
B	194	B	194	B	130	> <	130	> <
C	195	C	195	C	131	see a	131	see a
D	196	D	196	D	132	> <	132	> <
E	197	E	197	E	133	> <	133	> <
F	198	F	198	F	134	> <	134	> <
G	199	G	199	G	135	bell	135	bell
H	200	H	200	H	136	see b	136	see b
I	201	I	201	I	137	> <	137	> <
J	202	J	202	J	138	see c	138	see c
K	203	K	203	K	139	> <	139	> <
L	204	L	204	L	140	> <	140	> <
M	205	M	221	]	141	see d	157	>]<
N	206	N	222	^	142	> <	158	> <
O	207	O	207	O	143	> <	143	> <
P	208	P	192	@	144	> <	128	> <
Q	209	Q	209	Q	145	> <	145	> <
R	210	R	210	R	146	> <	146	> <
S	211	S	211	S	147	> <	147	> <
T	212	T	212	T	148	> <	148	> <
U	213	U	213	U	149	see e	149	see e
V	214	V	214	V	150	> <	150	> <
W	215	W	215	W	151	> <	151	> <
X	216	X	216	X	152	> <	152	> <
Y	217	Y	217	Y	153	> <	153	> <
Z	218	Z	218	Z	154	> <	154	> <

VALUE	CHR$	VALUE	CHR$
128	>@<	176	0
129	>A<	177	1
130	>B<	178	2
131	>C<	179	3
132	>D<	180	4
133	>E<	181	5
134	>F<	182	6
135	>G< bell	183	7
136	>H< <--	184	8
137	>I<	185	9
138	>J< line feed	186	:
139	>K<	187	;
140	>L<	188	<
141	>M< RETURN	189	=
142	>N<	190	>
143	>O<	191	?
144	>P<	192	@
145	>Q<	193	A
146	>R<	194	B
147	>S<	195	C
148	>T<	196	D
149	>U< -->	197	E
150	>V<	198	F
151	>W<	199	G
152	>X<	200	H
153	>Y<	201	I
154	>Z<	202	J
155	ESCAPE	203	K
156	>backslash<	204	L
157	>]<	205	M
158	>^<	206	N
159	>underscore<	207	O
160	SPACE	208	P
161	!	209	Q
162	"	210	R
163	#	211	S
164	$	212	T
165	%	213	U
166	&	214	V
167	'	215	W
168	(	216	X
169	)	217	Y
170	*	218	Z
171	+	219	***
172	,	220	***
173	-	221	]
174	.	222	^
175	/	223	***

a. PRINTing the CHR$() of CTRL C will end Applesoft Program.
b. CTRL H and Shift CTRL H are the same as the back space (<--).
c. CTRL J and Shift CTRL J are a line feed (drops down a line).
d. CTRL M is the same as RETURN.
e. CTRL U and Shift CTRL U are the same as -->.

***** cannot be directly keyed:**
219 left bracket
220 backslash
223 underscore

GETting A$

KEY	ALONE		SHIFT		CONTROL		BOTH	
===	==========		==========		==========		==========	
===	A$	ASC(A$)	A$	ASC(A$)	A$	ASC(A$)	A$	ASC(A$)
1 !	1	49	!	33	1	49	!	33
2 "	2	50	"	34	2	50	"	34
3 #	3	51	#	35	3	51	#	35
4 $	4	52	$	36	4	52	$	36
5 %	5	53	%	37	5	53	%	37
6 &	6	54	&	38	6	54	&	38
7 '	7	55	'	39	7	55	'	39
8 (	8	56	(	40	8	56	(	40
9)	9	57	)	41	9	57	)	41
0	0	48	0	48	0	48	0	48
: *	:	58	*	42	:	58	*	42
- =	-	45	=	61	-	45	=	61
; +	;	59	+	43	;	59	+	43
, <	,	44	<	60	,	44	<	8
. >	.	46	>	62	.	46	>	62
/ ?	/	47	?	63	/	47	?	63
<--		8		8		8		8
-->		21		21		21		21
SPACE		32		32		32		32
RETURN		13		13		13		13
ESC		27		27		27		27
A	A	65	A	65	> <	1	> <	1
B	B	66	B	66	> <	2	> <	2
C	C	67	C	67	> <	3	> <	3
D	D	68	D	68	> <	4	> <	4
E	E	69	E	69	> <	5	> <	5
F	F	70	F	70	> <	6	> <	6
G	G	71	G	71	>bell<	7	>bell<	7
H	H	72	H	72	see a	8	see a	8
I	I	73	I	73	> <	9	> <	9
J	J	74	J	74	see b	10	see b	10
K	K	75	K	75	> <	11	> <	11
L	L	76	L	76	> <	12	> <	12
M	M	77	]	93	see c	13	see c	13
N	N	78	^	94	> <	14	> <	14
O	O	79	O	79	> <	15	> <	15
P	P	80	@	80	> <	16	see d	16
Q	Q	81	Q	81	> <	17	> <	17
R	R	82	R	82	> <	18	> <	18
S	S	83	S	83	> <	19	> <	19
T	T	84	T	84	> <	20	> <	20
U	U	85	U	85	see e	21	see e	21
V	V	86	V	86	> <	22	> <	22
W	W	87	W	87	> <	23	> <	22
X	X	88	X	88	> <	24	> <	24
Y	Y	89	Y	89	> <	25	> <	25
Z	Z	90	Z	90	> <	26	> <	26

> < indicates that, when PRINTed to the screen, nothing appears, not even a space.

===	===	===	===	===	===
ASC	A$	ASC	A$	ASC	A$
===	===	===	===	===	===
1	>A<	33	!	65	A
2	>B<	34	"	66	B
3	>C<	35	#	67	C
4	>D<	36	$	68	D
5	>E<	37	%	69	E
6	>F<	38	&	70	F
7	>G<	39	'	71	G
8	see a	40	(	72	H
9	>I<	41	)	73	I
10	see b	42	*	74	J
11	>K<	43	+	75	K
12	>L<	44	,	76	L
13	see c	45	-	77	M
14	>N<	46	.	78	N
15	>O<	47	/	79	O
16	>P<	48	0	80	P
17	>Q<	49	1	81	Q
18	>R<	50	2	82	R
19	>S<	51	3	83	S
20	>T<	52	4	84	T
21	see d	53	5	85	U
22	>V<	54	6	86	V
23	>W<	55	7	87	W
24	>X<	56	8	88	X
25	>Y<	57	9	89	Y
26	>Z<	58	:	90	Z
27	ESCape	59	;	91	***
28	***	60	<	92	***
29	>]<	61	=	93	]
30	>^<	62	>	94	^
31	***	63	?	95	***
32	SPACE	64	@		

***** cannot be directly keyed:**
219 left bracket
220 backslash
223 underscore

a. CTRL H and Shift CTRL H are the same as the backspace key (⟵).

b. CTRL J and Shift CTRL J are line feeds (moves down one line).

c. CTRL M is the same as a RETURN (goes to beginning of next line).

d. Trying to get the ASC() of a Shift CTRL P will end up in an ERROR.

e. CTRL U and Shift CTRL U are the same as the ⟶ key.

GETting a CTRL C will not end the program.

E rrors

DiskView 1.Ø

There is an extra comma at the end of line 55Ø. Delete the comma from your listing.

DiskEdit 2.2 (listed in issue 2 as 2.1)

The variable TØ% came out looking like a TO%. "TO" is a reserved word in Apple soft. The correct variable name is TØ% (Tee Zero percent). It is used to hold the old track value in case you change your mind and press <ESC>. To avoid further confusion, change the variable name to a single letter "T%". The variable is used in line 18, 116 and 162. All of these lines must be corrected.

YOU'RE SENDING TOO MUCH MONEY.

If you ordered any programs from the HardCore Program Library and you sent money for more than one disk, then you sent too much.

All of the software in our library is on normal DOS and can be copied using FID or COPYA. It is sent out on one disk.

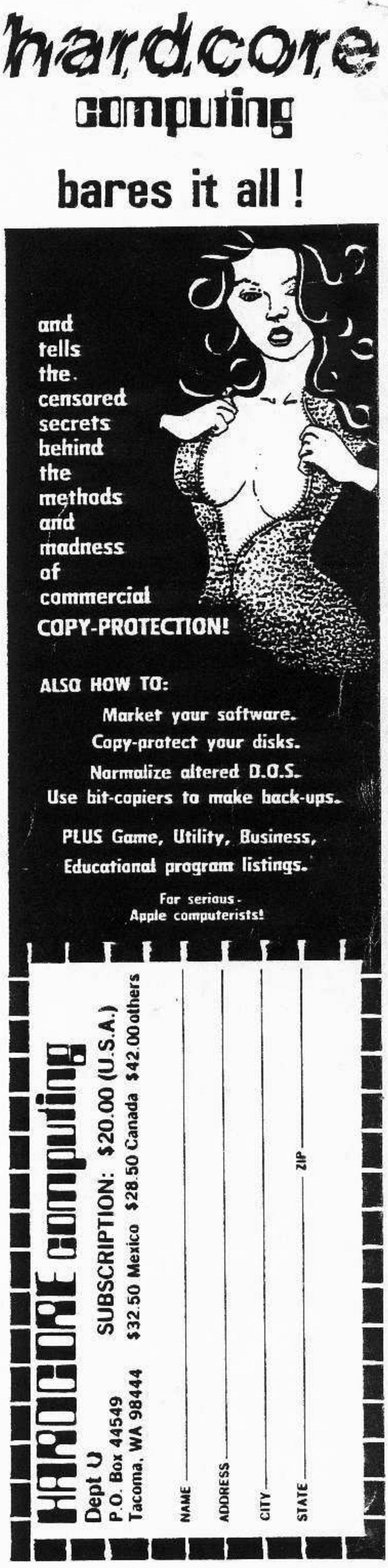
To order a program, add up the DRC (Direct Royalty Charge) and the cost for documentation (if any) and then add $6 for a single disk (postage and handling is included in the $6).

The UPDATE FEE is $2 (for postage, handling, etc). If you ordered a program and would like to receive the most recent version, return the original disk (with its label) in the original shipping package (the Floppy-Armor foam folder) and enclose the update fee of two ($2) dollars.

A WORD OR THREE ABOUT...
===== THE HARDCORE PROGRAM LIBRARY =====

Unlike other products, programs can be easily updated, renovated, improved, etc... and that is what happens to our programs. The authors are always making them better. Updated versions will be noted in the Program Library catalog even though the improved version will not be LISTed in any issue. Because of the delay between receipt of an article and its eventual appearance in the magazine, programs are often already out-of-date by the time they appear in print!

NIBBLE magazine rejected this ad, saying that it was "too suggestive", but added that it was not a rejection of the product, only the ad. We believe them, and we've redone the ad. If it is acceptable, it should appear in their MAY/JUNE issue.



hardcore computing

bares it all!

and tells the censored secrets behind the methods and madness of commercial COPY-PROTECTION!

ALSO HOW TO:

Market your software.
Copy-protect your disks.
Normalize altered D.O.S.
Use bit-copiers to make back-ups.

PLUS Game, Utility, Business, Educational program listings.

For serious Apple computerists!

HARDCORE computing
Dept U
P.O. Box 44549
Tacoma, WA 98444

SUBSCRIPTION: $20.00 (U.S.A.)
$32.50 Mexico $28.50 Canada $42.00 others

NAME
ADDRESS
CITY ZIP
STATE